ultra fractal 5

Introduction

This manual contains complete documentation for Ultra Fractal 5 in printer-friendly format. All information in this manual is also accessible from the Help menu in Ultra Fractal. You can also get context-sensitive help in every dialog and tool window in Ultra Fractal. Click the **?** button in the title bar, and then click on a control to learn more about it.

The compiler reference is not included in this manual because it would make it unnecessary large. To access the compiler reference, click Contents on the Help menu in Ultra Fractal. In the Writing Formulas chapter, there is an additional Reference chapter that lists all built-in functions, operators, keywords, and so on. Alternatively, click on a symbol in the formula editor and then press F1 or click Help on the Help menu.

Table of Contents

Table of Contents

What's new in Ultra Fractal 5?

These are the major new features in Ultra Fractal 5:

- **Layer groups**
  With the new layer groups feature, you can organize layers in groups, which makes fractals with many layers easier to work with. Because a layer group has a separate merge mode, this also makes new creative merging effects possible. See Layer groups.
- **Select multiple layers**
  You can now select multiple layers and edit them together by Ctrl-clicking or Shift-clicking in the layers list in the Fractal Properties tool window. For example, you can easily move or delete multiple layers, apply a new fractal formula, or edit parameters that all selected layers share. Likewise, it is also possible to select multiple transformations in the Mapping tab of the Layer Properties tool window. This makes it much easier to work with fractals that have many layers or transformations. See Working with layers and Working with transformations.
- **Image import**
  Import PNG, JPEG or BMP images in your fractals with the new image import feature. Simply select a coloring algorithm that contains an image parameter, and you can select any image on your computer to use. The coloring algorithm determines how the information from the image is used. For example, the standard Image coloring algorithm just displays the entire image in the fractal window. See Using images.
- **Formulas with classes**
  Formulas in Ultra Fractal can now use separate sub-formulas, which are called classes. This allows formula authors to easily re-use common formula features in many formulas. A formula that use classes typically enables you to select those classes yourself, which is a very powerful way to extend the capabilities of a formula after it has been published. For more information, see About classes. If you are a formula author and you want to incorporate classes in your own formulas, see New compiler features.
- **Browser thumbnails**
  The browser now includes a thumbnail view for formulas, parameter sets, fractals and gradients that gives a great overview of the available items. See View style.
- **Formula ratings**
  Ultra Fractal now contains a rating system for formulas that makes it easy to see which formulas you should try first, and which are no longer recommended to use. See Formula ratings.
- **Improved support for dual-core processors**
  Ultra Fractal 5 uses dual-core or multi-processor systems more efficiently. Previously, on these systems, the fractal to calculate was split in vertical strips which made it harder to judge the calculation. Now, the image is split in as many strips as there are cores, so it appears to be calculated at once. If one strip finishes earlier than the other, the unfinished strip is subdivided again. This gives an impressive speed-up with most fractals and makes it even more worthwhile to have a dual-core system.
- **Improved formula compiler**
  Under the hood, Ultra Fractal's formula compiler uses a new and more efficient code generator. This means that your formulas will run faster than in Ultra Fractal 4. Selecting new formulas and opening parameter sets is faster, too.
- **Improved Timeline tool window**
  The Timeline tool window now lets you edit the properties of multiple selected ranges or keys together, just like the way this works with layers and transformations. Previously, you could only edit one item at a time.
- **Improved fractal window**
  The fractal window now enables you pan the fractal very precisely with the Ctrl+arrow keys. There is a new Copy Image command on the Edit menu so you can copy the parameters and the image of the fractal individually. Also, there is a new Gradient|Replace command on the pop-up menu that appears if you right-click in the fractal window.
- **Improved formula parameters list**
  The list of formula parameters in the Mapping, Formula, Inside and Outside tab of the Layer Properties tool window now enables you to horizontally resize the area reserved for

parameter labels. See Formula parameters.

- **Improved Explore window**
  The Explore window now shows the location of the original value of the parameter that is being explored.
- **Improved formula editor**
  The formula editor now supports editing class library files and contains an additional Check Class Syntax command in the File menu to easily check all classes in a file for errors.
- **Improved Windows Vista support**
  Ultra Fractal 5 supports Vista's Aero user interface even better with new file open/save dialogs, support for the Segoe UI font, and more.

Next: New compiler features

See Also
Tutorials
Writing formulas

New compiler features

Ultra Fractal 5 contains many compiler and formula language enhancements that make it easier to write large and powerful formulas:

- **Classes**
  You can now declare your own classes and use them as objects in a formula, like in Java or C#. Classes can be declared in formula files or in separate class library files (*.ulb). Classes can inherit from each other to create a class hierarchy. See Classes.
- **Functions**
  In classes, but also in formulas, you can declare and use functions. Functions can have an optional return value and any number of arguments. Arguments can be of any type, and can also be passed by reference or marked as constant. Static class functions are also supported. See Functions.
- **Class parameters**
  Formulas can have parameters of class type. These class parameter show up as the name of the class, with the option to select a different class that is derived from the base class of the parameter. Such a derived class can of course implement many different options, which is a very powerful way to expand the capabilities of a formula even after it has been published. The parameters of the selected class appear under the name of the class and they can be collapsed or expanded. See Class parameters.
- **Image parameters**
  Formulas, typically direct coloring algorithms, can now have image parameters. This is how images can be imported in Ultra Fractal 5. Inside the coloring algorithm, you have full access to the imported image data. The built-in Image class also enables you to create and manipulate stand-alone images. See Image parameters and Image class.
- **Dynamic arrays**
  The compiler now supports dynamic arrays, which can be resized at any time, unlike the static arrays that were available since Ultra Fractal 3. See Dynamic arrays.
- **New calculationPurpose predefined symbol**
  The new #calculationPurpose predefined symbol makes it possible to find out the purpose of a calculation from formula code: for a fractal window, browser preview, or a disk render.
- **New rating setting**
  The new rating setting lets you rate your own formulas, so users can quickly see which of your formulas are recommended and which are not (perhaps because you created a better version later). See rating setting and Formula ratings.
- **New isInf and isNaN functions**
  The new isInf and isNaN functions enable you to determine whether an invalid operation occurred on a floating-point number.
- **Miscellaneous**
  The compiler now always defines the symbol VER50. See Compiler directives.

See Also
What's new in Ultra Fractal 5?
Writing formulas

Getting help

Although Ultra Fractal has been carefully designed to be as easy to use as possible, you will probably need to refer to this help file from time to time, especially while you are learning to use the program.

Click the **Help** button in the toolbar, click Help on the Help menu, or press F1, to get help on the currently active document window.

The Help menu also provides links to other major chapters in the help file that you might want to explore. The help file is divided into three major sections:

- **Tutorials**
  If you are new to Ultra Fractal, or to its animation features, you should take some time to do the included tutorials. This is the easiest and most enjoyable way to get to know Ultra Fractal.
- **Reference chapters**
  After you have finished the tutorials, or any time you want to know something, the reference chapters provide in-depth information on any of Ultra Fractal's features, such as the workspace, fractal windows, gradients, formulas, and layers. You can access any of these from the table of contents in the help window.
- **Writing formulas**
  In case you are interested in writing your own fractal formulas, the Writing formulas chapter contains a short tutorial and a complete reference for the formula language used by Ultra Fractal. See Help for formula authors for more information.

Next: Context-sensitive help

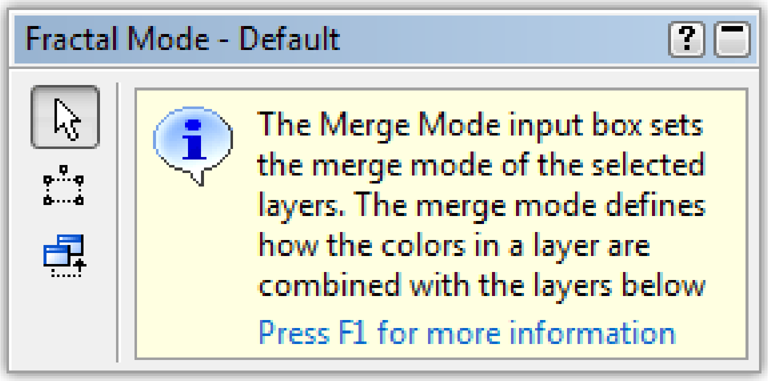
See Also
Tutorials
Workspace
Support

Context-sensitive help

To quickly get more information on a control or a window that you are working with, use the context-sensitive help in Ultra Fractal.

- Make sure the Fractal Mode tool window is open and just hover the mouse cursor over the control that you want to know more about.

The Fractal Mode tool window will now show help on the control.



The Merge Mode input box sets the merge mode of the selected layers. The merge mode defines how the colors in a layer are combined with the layers below

Press F1 for more information

Often the context-sensitive help contains a link to the main help file that you can follow by pressing the F1 key.

An alternative way to get context-sensitive help is to click the ? button in the title bar of a tool window or a dialog box, and then click the control you want to get help on. This displays the same information in a pop-up window.

You can also get help on most formula parameters in the same way.

Next: Help for formula authors

See Also
Tool windows
Workspace
Support

Help for formula authors

If you are interested in writing your own formulas, you will find all information you need in the Writing formula chapter of the help file. This part of the help file is divided into four sections:

- **Language**
  The Language section starts with a short tutorial and then discusses the various elements of the formula language and how to use them effectively.
- **Formulas**
  The Formulas section documents how to create fractal formulas, coloring algorithms, and transformations, and shows how to add help and how to use various other features.
- **Reference**
  The Reference section provides complete documentation for all built-in functions, operators, predefined symbols, compiler directives, settings, errors, and warnings.
- **Tips**
  Finally, the Tips section contains hints and guidelines that will help you to write and publish formulas efficiently.

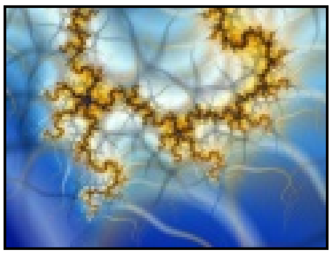
While you are working on a formula, you can easily access the documentation in the Reference section.

- Place the text cursor on the function, operator, predefined symbol, or setting that you want to know more about and press F1 or click **Help** on the Help menu.

This will open the help page for the symbol at the cursor position.

See Also
Writing formulas
Support

Tutorials



To help you to get up to speed quickly with Ultra Fractal, this chapter contains a complete set of tutorials. Starting with the basics, you will soon learn how to create your own fractals, change the colors, add layers, use masks, and create animations.

The tutorials are organized in such a way that beginners can follow the steps in the main text without stopping to learn more advanced features. Small boxes on the right contain Tips, Hints and Learn More links for more experienced users.

The following tutorials are available:

- Quick Start Tutorial
  In this first tutorial, you will learn how to create a new fractal from scratch. You will experiment with formula parameters, add a coloring algorithm, and learn how to save and re-open fractals.
- Learning basic skills
  This tutorial explains the basics of working with fractals: using Switch mode, zooming, and working with gradients to adjust colors.
- Working with layers
  Now that you know the basics, learn how to add new layers, work with opacity and merge modes, and create transparent gradients.
- Learning about transformations
  To make things even more interesting, this tutorial shows you how to add transformations to your fractals to create all kinds of different effects.
- Masking
  The Masking tutorial extends the skills that you have learned so far and shows how to use the masking feature. Along the way, you will create a wonderful image to decorate your Windows desktop with!
- Working with animations
  This final tutorial explains how to use the new animation features in Ultra Fractal 4. You will create a zoom movie, animate parameters, including gradient control points, learn how to use the Timeline tool window, and finally render your animation as an AVI movie.

All tutorials were written by Janet Parke (see Some final thoughts), except *Working with animations*.

See Also
New features
What are fractals?
Workspace

Creating a fractal image

When you first open Ultra Fractal, the default Mandelbrot set is shown. Since we're going to create a new fractal from scratch, let's close this fractal.

- Click **Close** on the File menu to close the fractal window.

Now, the workspace is empty except for a row of Tool Windows on the right side of the screen. These tool windows are where you will enter and edit the information which creates the fractal image on your screen. The tool windows are the information center of the program and it is advisable to keep them open at all times.

Learn more about...
Tool windows

The first step in creating a new fractal is to select a Fractal Formula which determines the structure of the fractal with which we will be working.

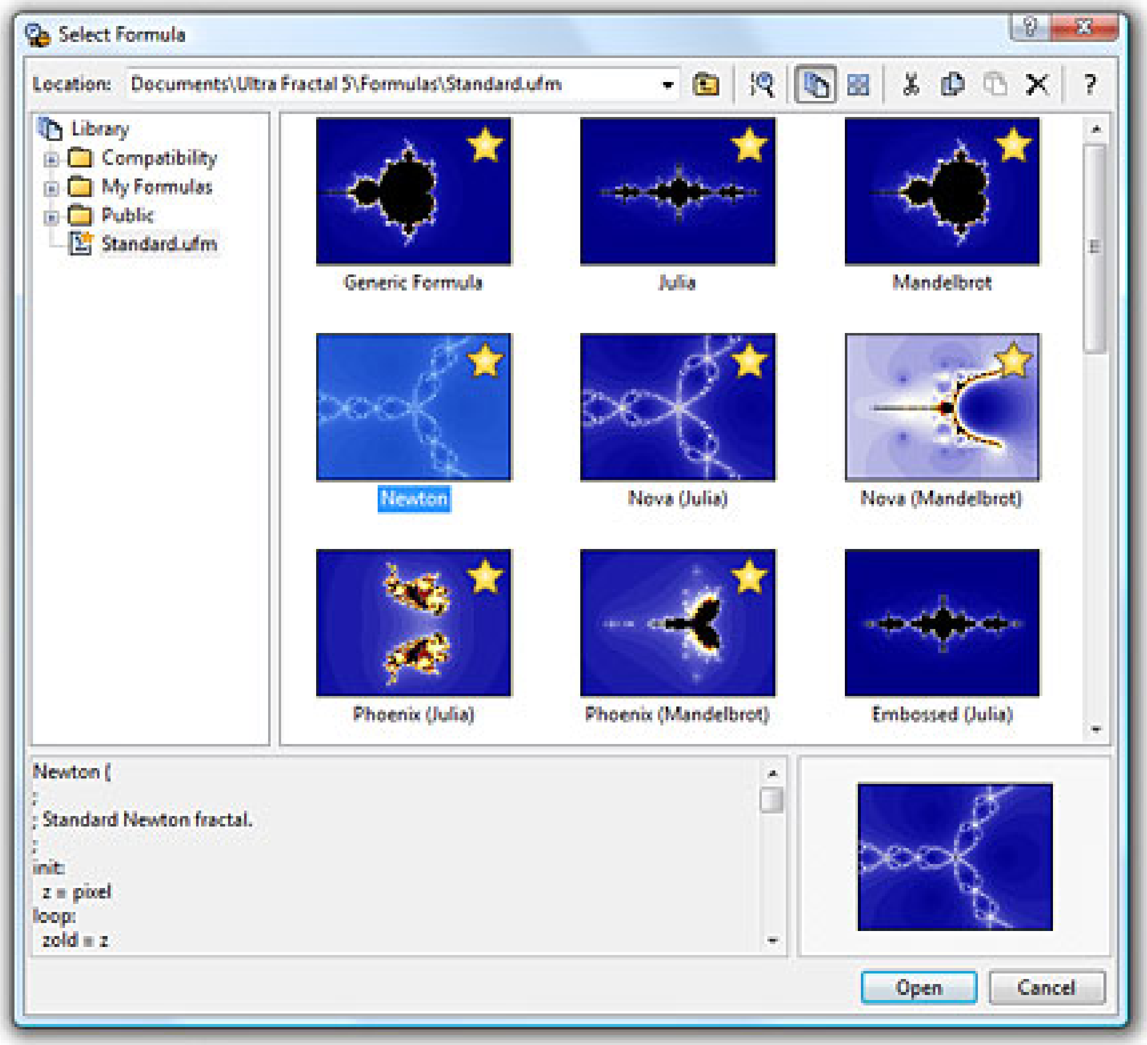
To open a new fractal window, click **New** on the **File** menu and select **Fractal**.

Learn more about...
Fractal formulas

Tip!
There are three additional ways to access most features and tools: Toolbar buttons, Right-click (mouse) menus, and Keyboard shortcuts.

This opens the "Select Fractal Formula" browser. The left pane shows three folders (Compatibility, My Formulas, and Public) and a file named "Standard.ufm".

When you click on **Standard.ufm**, its contents — a list of the formulas it contains, appears in the right pane of the browser window. Each formula appears as a thumbnail that shows a preview of the image that the formula will produce.



```
Newton {
;
; Standard Newton fractal.
;
init:
  z = pixel
loop:
  zold = z
```

- Click on the ***Newton*** formula and then on the **Open** button. *(Note: Make sure it's the **Newton** not the "Embossed Newton" formula!)*

This opens a new fractal window, loaded with the *Newton* formula, which looks like the preview in the browser window above.

Learn more about...
The *Newton* formula

Next: Changing formula parameters

Changing formula parameters

Look at the Tool Windows on the right side of your screen and click on the **Formula** tab of the topmost (**Layer Properties**) window.

Under the formula name at the top (in this case, *Newton*) you will see settings (Drawing Method, Periodicity Checking, Additional Precision, and Maximum Iterations) which appear for every fractal formula. The parameters specific to this particular formula are listed below the dividing line (in this case: Exponent (Re), Exponent (Im), Root (Re), Root (Im) and Bailout value).

Tip!

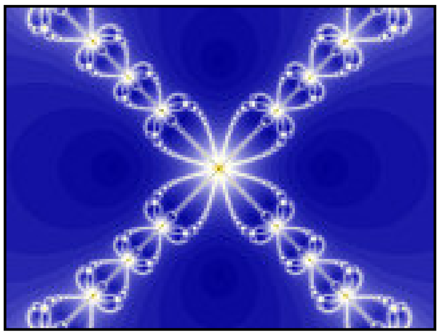
While Guessing, the default drawing method, is the fastest method of rendering fractals, it often introduces artifacts into the image. You may wish to choose either the Multi-Pass or One-Pass Linear options for accurate rendering.

The Exponent parameter determines the number of "arms" of the *Newton* fractal structure. The default value is "3" so the fractal has three arms. Try entering different values in the Exponent (Re) parameter.

- When you have finished experimenting, enter **4** in the Exponent (Re) parameter. Your fractal will look like this:

For fun...

Try entering negative numbers and other values (e.g. .5, 1.2, and .0385) in the various parameters and see how they twist and fracture the structure.



At this stage, you may want to change the size of the fractal window. To do this, click on the **Image** tab of the second (**Fractal Properties**) tool window.

- Make sure the "Maintain Aspect Ratio" option is checked and then enter a new value in the width field.

Tip!

You can customize the default fractal window size by clicking **Options** on the **Options** menu and selecting the **Defaults** tab.

Next: Applying a coloring algorithm

Applying a coloring algorithm

The next step in creating a fractal is to apply a **Coloring Algorithm** to the fractal structure. Here's where the real fun begins!

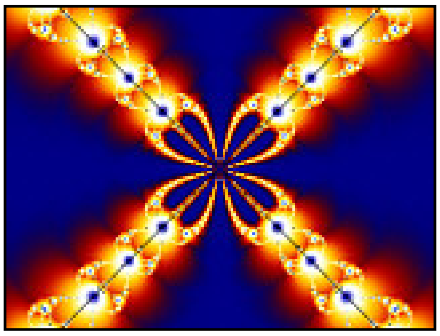
Learn more about...
Coloring Algorithms

- Click on the **Outside** tab of the **Layer Properties** tool window.

The default coloring algorithm is called "None" and it simply assigns a color to each pixel. Let's load an algorithm which will give us some more creative control over the image.

Click the **Browse** button on the **Outside** tab which brings up the "Select Outside Coloring Algorithm" browser.

- Click on the **Standard.ucl** file in the left pane and then on the ***Orbit Traps*** algorithm in the right pane.
- Click **Open** to apply this algorithm to your fractal.



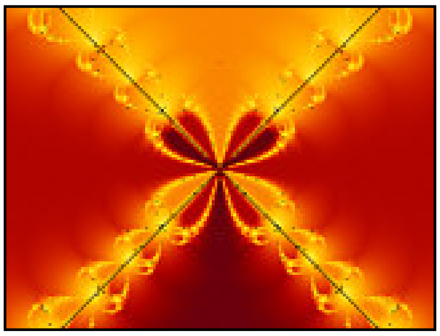
Learn more about...
The *Orbit Traps* Coloring Algorithm

Just as on the Formula tab, there are several settings on the Outside coloring tab (Color Density, Transfer Function, Solid Color, Gradient Offset, and Repeat Gradient) which appear regardless of the algorithm chosen. The parameters below the dividing line are specific to the *Orbit Traps* algorithm.

- Click on the arrow at the right of the **Transfer Function** setting and select **Log** from the drop-down list.
- Click on the arrow at the right of the **Trap Shape** parameter and select **Egg** from the drop-down list.

Tip!
To learn more about a particular parameter, make sure the Fractal Mode tool window is visible and hover over the parameter input box with the mouse cursor.

Your fractal will now look like this:



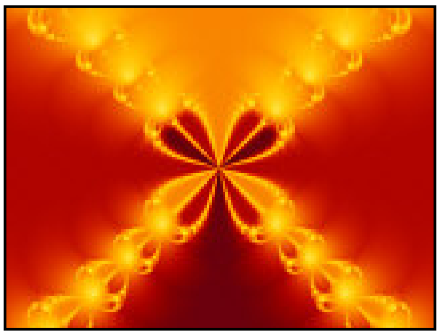
Tip!

There are so many options available with the *Orbit Trap* coloring algorithm. It would be well worth your time to work through them, observing what effect each change, and each combination of parameters, has on your image.

You may notice, now that we have selected this combination of parameters, the black lines that bisect each of the arms of the fractal. This is due to the precision of Ultra Fractal's calculations. We can circumvent this effect by making a small adjustment on the Location tab.

- Click on the **Location** tab of the **Layer Properties** tool window, and enter **.01** in the **Rotation Angle** setting.

This rotates the fractal imperceptibly and eliminates the black lines.



Next: Saving your fractal

Saving your fractal

This is not a very exciting fractal yet, but before we go further, let's save our work. There are various ways to save a fractal, and we will cover them one by one in these tutorials.

This time, we will save the image within Ultra Fractal in a text-only form called a **Parameter File**. This file takes up very little space on your hard drive and can be easily shared with other users via email and mailing lists.

Learn more about...
Sharing parameter files with other users

Choose **Save Parameters** on the **File** menu.

With the "Save Parameter Set" browser open, you can create a parameter file which will hold all the images we create in these tutorials. At the bottom of the browser, in the **File Name** field, append the entry with **tutorials.upr** so that the path reads:

My Documents\Ultra Fractal 5\Parameters\tutorials.upr

(This assumes that you have installed Ultra Fractal using the default document folders. On Windows Vista, the path will start with *Documents* instead of *My Documents*.)

- Next, in the **Title** field, enter **Newton 1** (the title can be any length and may contain spaces) and then click the **Save** button.

We will use this image again later in the tutorials so you can close the fractal window or even Ultra Fractal itself at this time.

Next: Opening your saved fractal

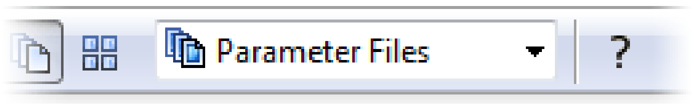
Opening your saved fractal

You can re-open the saved fractal at any time with the parameter browser.

Choose **Browse** on the File menu to open the browser.

Learn more about...
[Browsers]()

- Check to make sure that **Parameter Files** is selected on the toolbar.



- Now select **tutorials.upr** in the left pane of the browser window and **Newton 1** in the right pane. Double-clicking on the item in the right pane will open the fractal.

Note: *If you are brand new to fractals or Ultra Fractal, you might want to become more comfortable with these beginning steps before moving along to the next tutorial. Feel free to experiment with other fractal formulas and coloring algorithms in different combinations as you practice.*

Next tutorial: [Learning basic skills]()

Learning basic skills

In this tutorial, we are going to learn some basic skills for working with fractals — using Switch Mode, zooming, and working with gradients.

Let's create a new image using a different fractal formula.

Click **New** on the **File** menu, and then click **Fractal**.

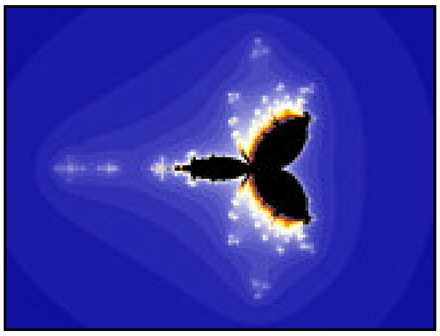
- Select the ***Phoenix (Mandelbrot)*** formula from the list in the right pane of the "Select Fractal Formula" browser window.
- Double-click on the name, or click **Open**.

Learn more about...

The *Phoenix* formulas

If you cannot find the formula, make sure that the file *Standard.ufm* is selected in the left pane first.

You should now see this image on your screen:



Next: Learning to use Switch Mode

Learning to use Switch Mode

Although we could work with this image as is, let's learn about using the Switch feature to open a corresponding *Julia* version of this *Phoenix (Mandelbrot)* fractal.

Using the Switch feature is often a good way to find interesting fractal structure.

Learn more about...
Switch Mode

Click **Switch Mode** on the **Fractal** menu.

- Place your mouse cursor anywhere in the active fractal window and look at the **Fractal Mode** tool window on the right-hand side of your screen.

Each point in the *Phoenix (Mandelbrot)* set corresponds to a separate *Julia*-type fractal. As you move your mouse around the fractal window, notice that a preview of that corresponding *Phoenix (Julia)* image is displayed in the Fractal Mode preview window.

- When you find an image in the preview window that appeals to you, click once and a new fractal window containing that image will open.
- This window (named *Fractal2*) is the one with which we will be working, so you can close the original window (*Fractal1*) without saving it.
- Click on the **Formula** tab of the **Layer Properties** tool window and notice that this fractal uses the *Phoenix (Julia)* calculation formula.

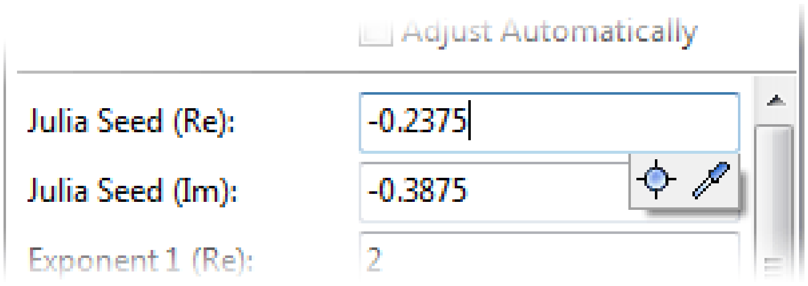
Just below the horizontal line in this same tool window are the Julia Seed parameters that you brought to this fractal when you switched from the *Phoenix (Mandelbrot)* to the *Phoenix (Julia)* formula.

Next: Using the Explore tool

Using the Explore tool

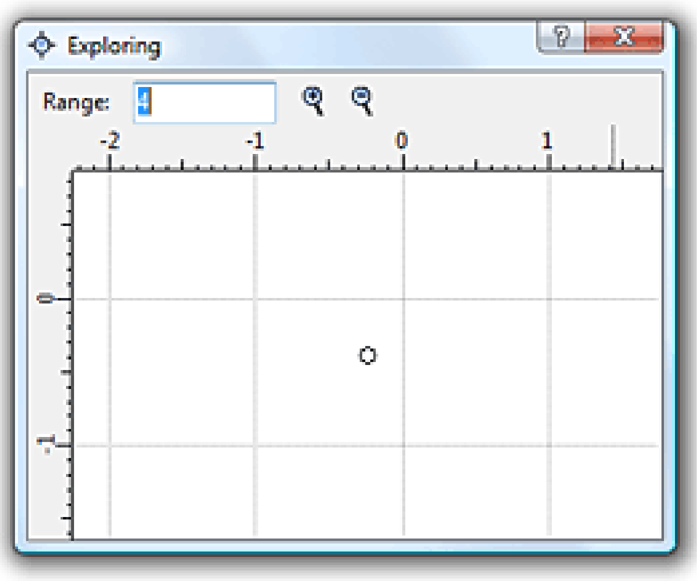
We can experiment with changing the Julia Seed values and other parameters to alter the image.

In the past, parameter values were often chosen at random because it was difficult to anticipate what changes the values would effect. To solve this problem, Ultra Fractal has the Explore tool that makes choosing parameter values much easier and more fun. When you click in any parameter field that takes a numerical entry, two icons appear directly below the field.



Julia Seed (Re): -0.2375
Julia Seed (Im): -0.3875
Exponent 1 (Re): 2

Click the **Explore** button to start exploring. This opens the Explore window with a rectangular coordinate grid.



Learn more about...
Explore

Tip!
You can also access both the Eyedropper and Explore tools by right-clicking in any numerical parameter field.

As you move the mouse cursor over the grid, the Fractal Mode tool window shows what the fractal will look like if the value under the cursor is selected.

You can zoom in and out to decrease or increase the range of potential values with the Zoom In and Zoom Out buttons, or by typing a new range value. Simply drag the rulers to pan the window.

Tip!

You can also zoom and pan in the Explore window by Shift-dragging and Ctrl-dragging, like in the fractal window.

Click to select a new value. Experiment with the Explore tool for a while until you are comfortable with it.

And remember that at any time, you can undo your changes by clicking **Undo** or **Redo** on the Edit menu.

Next: Synchronizing the Julia Seed

Synchronizing the Julia Seed

Because it is highly unlikely that you have chosen the exact Julia Seed values that we will be using in this tutorial, we will need to synchronize those values before we can proceed.

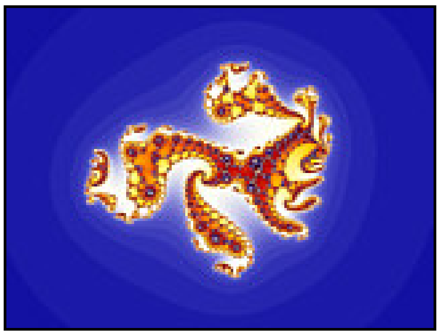
The Julia Seed parameters for this tutorial image are long, complicated numbers, but you will not need to type them in by hand.

- Just click on them below to copy them to the Clipboard.

-0.2589852008
-0.1395348837

- Switch to Ultra Fractal and right-click in either of the **Julia Seed** parameter fields on the **Formula** tab of the **Layer Properties** tool window. In the menu that appears, click **Paste Complex Value**.

Your image should now look like this:



Next: Zooming into the image

Zooming into the image

The most interesting fractal structure of this *Phoenix (Julia)* fractal exists in the red-orange-yellow areas so let's zoom in and explore the structure there.

Click **Select Mode** on the **Fractal** menu and notice that a rectangular box appears in your fractal window.

- Press **Enter**.

The fractal now re-draws on screen filling your image with the previously selected area.

Another way to invoke the selection box is to place your mouse cursor on an interesting part of your image. Click and hold the left button and drag the cursor until you have selected the area in which you wish to zoom.

Learn more about...
Select mode

Notice that by placing your mouse cursor over different parts of the selection box, you can also move, resize and rotate the box. The status bar at the bottom of the main Ultra Fractal window displays helpful hints while you move the mouse cursor around.

Try these options until you have framed an interesting section and then press **Enter** to zoom into this new area. To zoom out, press **Ctrl+Enter** instead.

You may want to spend some time zooming in and out to explore your fractal before we move on to the next part of the tutorial. Don't worry about where you travel, for we will synchronize locations in the next section.

Tip!
Instead of pressing **Enter** or **Ctrl+Enter**, you can also right-click in the fractal window and click **Zoom In** or **Zoom Out** on the menu that appears.

Next: Synchronizing the location

Synchronizing the location

Now that you have explored a bit, let's synchronize your fractal's location with that of the tutorial image so we can proceed together.

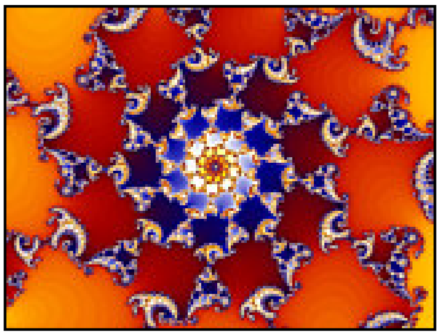
Click on the **Location** tab of the **Layer Properties** tool window. This time you are going to copy and paste all the necessary parameters in one operation.

- Click on the text below to copy it to the Clipboard.

```
BackgroundLocation {
location:
  center=-0.3979/0.28282 magn=1102.9412
}
```

Now switch to Ultra Fractal and click the **Paste Location** button on the **Location** tab of the **Layer Properties** tool window.

Your image should look like this:



Next: Adding outside coloring

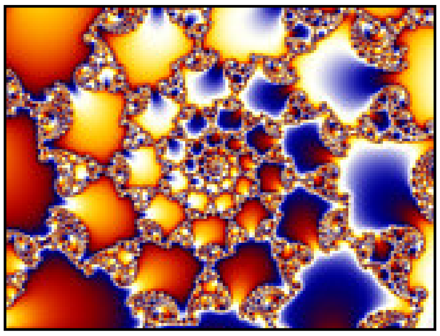
Adding outside coloring

Next, let's apply a coloring algorithm to this image.

Click on the **Outside** tab of the **Layer Properties** tool window and then click the **Browse** button.

- Select the ***Smooth (Mandelbrot)*** entry in the right pane of the "Select Outside Coloring Algorithm" browser and then click **Open**.

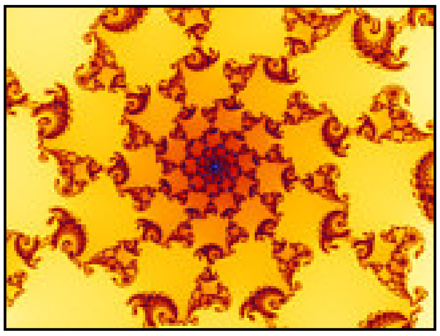
If you cannot find it, make sure *Standard.ucl* is selected in the left pane first.



Learn more about...
The *Smooth (Mandelbrot)* coloring algorithm

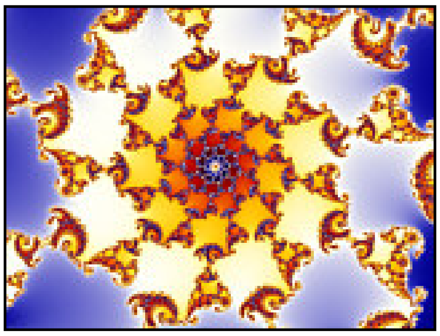
This is not very pretty but we can make some improvements by changing two of the coloring settings.

- Change the **Transfer Function** to **Sqrt**, which makes the image less busy...



Learn more about...
Coloring settings

- ... and change the **Color Density** to **5** to add in a few more colors.



Next: Working with the gradient

Working with the gradient

So far, the only adjustments we have made to the colors in our fractal have been through changes we have made to the coloring algorithm and its parameters. Since the coloring algorithm references colors found in the palette of the gradient, let's open and explore the **Gradient Editor**.

Learn more about...
How gradients work

Select **Gradient** from the **Fractal** menu to bring up the gradient editor for our fractal.

You will notice that the colors in the gradient correspond to the colors in your image. Both the gradient editor and your fractal contain blue, white, yellow, red, and black areas.

Learn more about...
Gradients

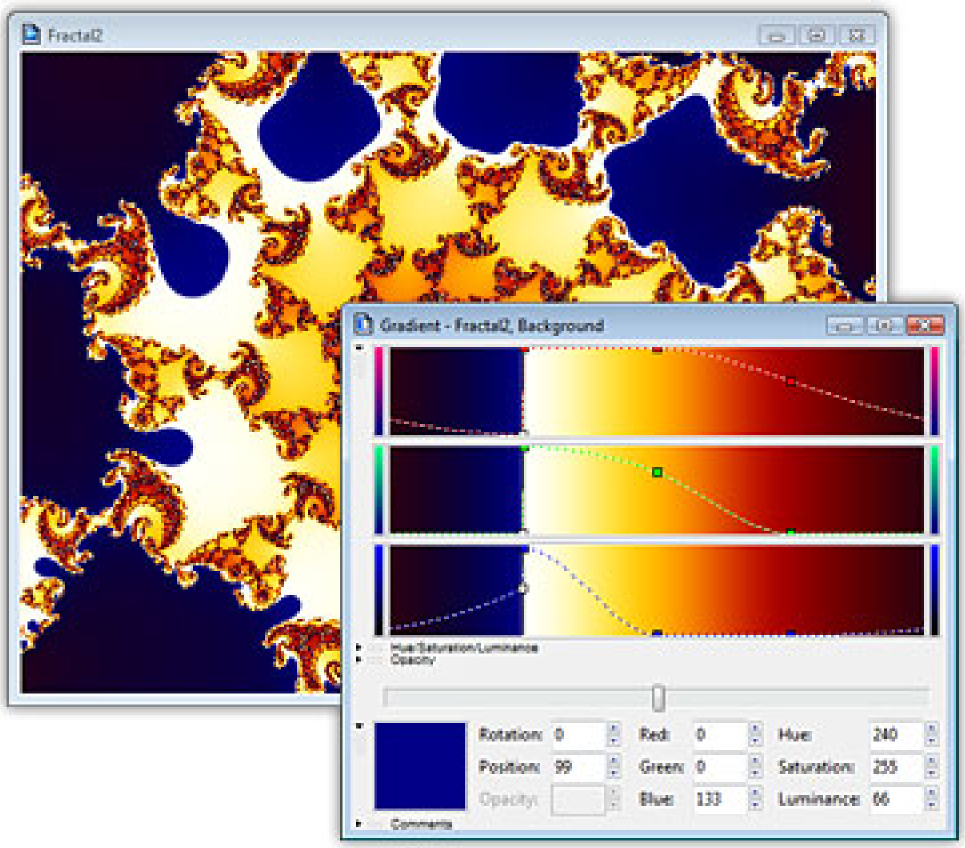
By clicking and dragging the horizontal slider on the Gradient editor to the right and left, you can rotate the gradient, and thus the colors in your image.

You can also move individual control points in the top 3 panels. Click on the top- and left-most control point and drag it to the right.

Learn more about...
Editing gradients

Notice that you can drag it horizontally past other control points and that doing so affects the shading between colors. Any changes made in the gradient editor are immediately reflected in your fractal onscreen.

If you drag a control point to a position immediately adjacent to another control point, this creates a sharp line between those two colors in the image rather than the smoother gradation that occurs when the control points are farther apart.



Rotation: 0
Position: 99
Opacity:
Red: 0
Green: 0
Blue: 133
Hue: 240
Saturation: 255
Luminance: 66

You can also drag each control point vertically within its panel of color to change the color of that point. Try moving the various control points up and down to see this.

Note: Unless you have experience with similar gradient editors in other graphics programs, you will want to spend time working with the gradient editor to learn how to manipulate the control points to make the colors you want.

While you can add, delete, move, and adjust the control points yourself — and you will eventually want to become very comfortable with these skills — an easy way to change colors is to use the **Randomize** feature.

Learn more about...
Adjusting colors and Random gradients

To generate a random gradient, click **Randomize Bright** (or **Randomize Misty**) on the **Gradient** menu. You can repeatedly press their respective keyboard shortcuts (**F6** and **F7**) until you find a set of colors you would like to use.

For fun...
You can create some really interesting gradients with the **Randomize Custom** editor.

Next: Synchronizing colors and Saving the image

Synchronizing colors and Saving the image

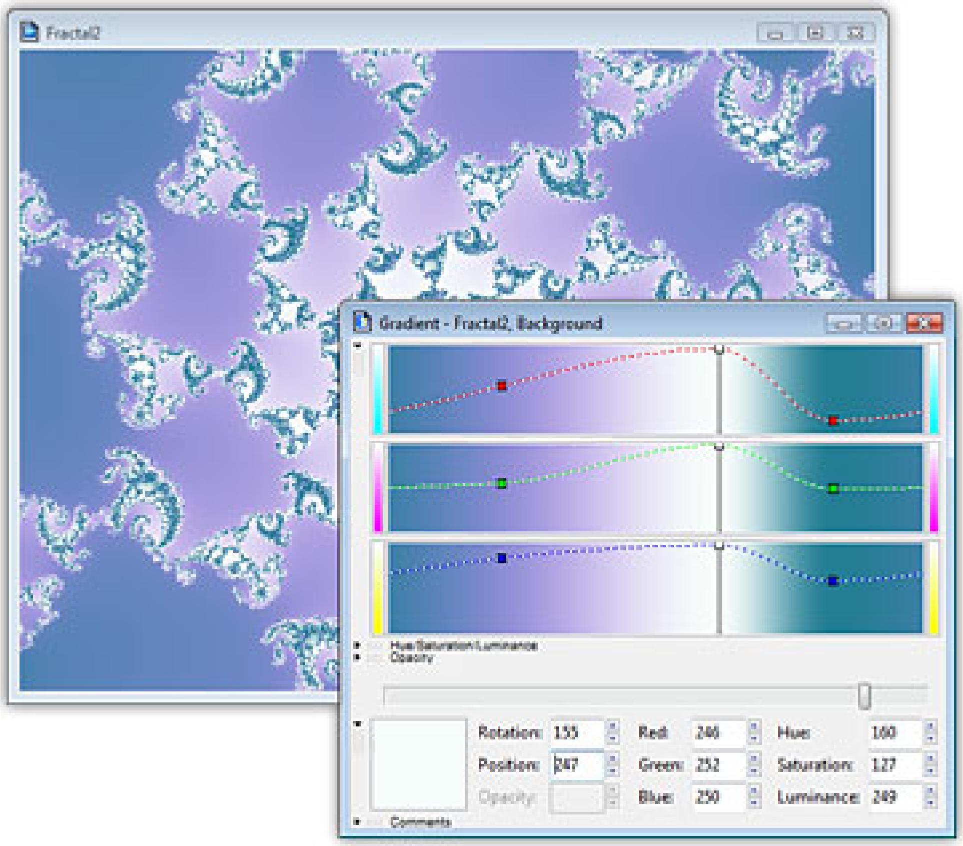
Now that you have become more familiar with gradients, let's synchronize the gradient of your image with the tutorial fractal.

- Click on the text below to copy it to the Clipboard.

```
Gradient-Fractal2,Background {
gradient:
  title="Gradient - Fractal2, Background" smooth=yes rotation=155 index=84
  color=13799050 index=247 color=16448758 index=332 color=9665827
opacity:
  smooth=yes
}
```

Now switch to Ultra Fractal. Right-click on the gradient editor and select **Paste**.

Your image and gradient editor should look like this:



Rotation: 155
Position: 247
Opacity:
Red: 246
Green: 252
Blue: 250
Hue: 160
Saturation: 127
Luminance: 249

Before we go on, let's save the parameters for this image.

- Click on the fractal image and then select **Save Parameters** on the **File** menu.
- Click on ***tutorials.upr*** in the left pane of the Save Parameter Set browser and then enter **Phoenix Julia 1** in the **Title** field at the bottom.

- Click **Save**.

Saving the parameters only saves the "recipe" for the image in text form. Ultra Fractal can also save the image in graphic form so that it can be re-opened and edited at a later time. Let's also save this image in this way as a **.ufr** file.

- Click **Save** on the **File** menu.

By default, Ultra Fractal puts all .ufr files in its "Fractals" folder. Since we have already named the image, its title appears in the **File Name** field and we can accept these defaults by clicking on the **Save** button.

Tip!

The **.ufr** format saves the rendered image so re-opening it does not require recalculation as parameter files do. This is very handy and time-saving for many-layered or slow-to-render images.

See also File types.

Next tutorial: Working with layers

Working with layers

Note: This tutorial assumes that you have worked through the Quick Start and Basic Skills tutorials and have saved the "Phoenix Julia 1" image from the Basic Skills tutorial.

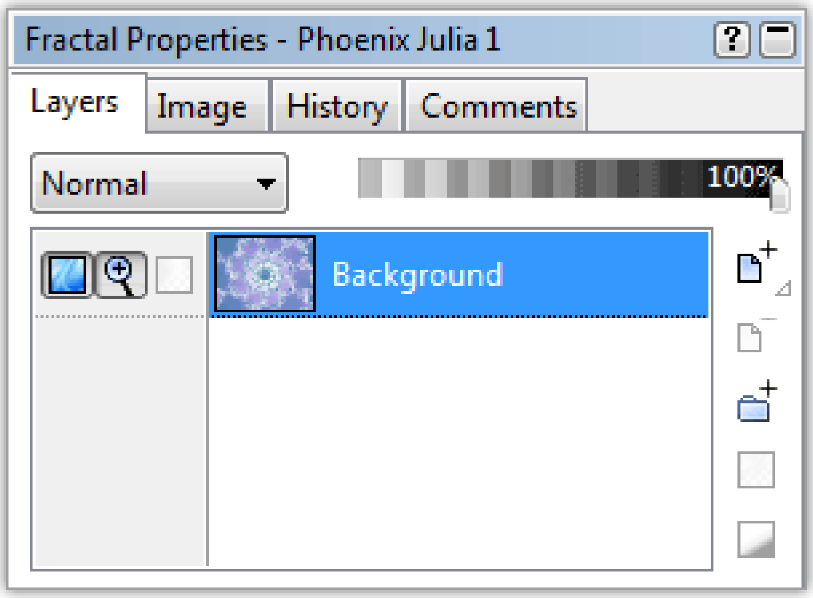
Now that we have learned some basic skills, it is time to explore one of Ultra Fractal's key features — layering.

Learn more about...
Layers

Let's open the image we created in the last tutorial. This time we will use the .ufr file, which will open the fully-rendered image on screen.

- Click **Open** on the **File** menu. Locate the .ufr file — remember that we saved it in the "Fractals" folder — click on the file name (***Phoenix Julia 1.ufr***) and click **Open**.

Now look at the **Fractal Properties** tool window. Click on the **Layers** tab and notice that there is a tiny copy of the image in a layer named Background.



To add a new layer, simply click the **Add Layer** button on the **Layers** tab.

Tip!
If you click and hold down the **Add Layer** button, a menu with predefined layers appears. Click a predefined layer to add it, or click **Define** to customize the menu.

Now you will see two layers — your original, still labeled "Background", and a new, identical layer labeled "Layer 1".

Next: Coloring the new layer

Coloring the new layer

Let's apply a different coloring algorithm to this new layer.

Click on the **Outside** tab of the **Layer Properties** tool window and then click the **Browse** button.

- Select the ***Triangle Inequality Average*** coloring from the browser's right pane and click **Open**.

You can immediately see the new coloring applied to the fractal. These colors are OK, but let's take this opportunity to learn more about gradients.

Learn more about...

The *Triangle Inequality Average* coloring algorithm

- First, open the gradient editor by selecting **Gradient** from the **Fractal** menu.

We could adjust the control points on this gradient, or generate a random gradient as we learned in the Basic Skills tutorial, but we can also load a pre-saved gradient.

- To load a pre-saved gradient, select **Replace** from the **File** menu.

The "Select Gradient" browser shows the pre-saved gradient files that come with Ultra Fractal and any gradients you have saved or imported.

Learn more about...

Opening and saving gradients

- Click on the ***Standard.ugr*** gradient file in the left pane of the browser window and ***Grayscale*** in the right pane. Click **Open**.

The pre-saved *Grayscale* gradient has now been loaded into the gradient editor and the active layer of your fractal.

Next: Editing the gradient

Editing the gradient

Before we edit the gradient, go to the **Outside** tab of the **Layer Properties** tool window.

Learn more about...
[Editing gradients]

- Change the Color Density setting to **1** and the Transfer Function setting to **Linear**.

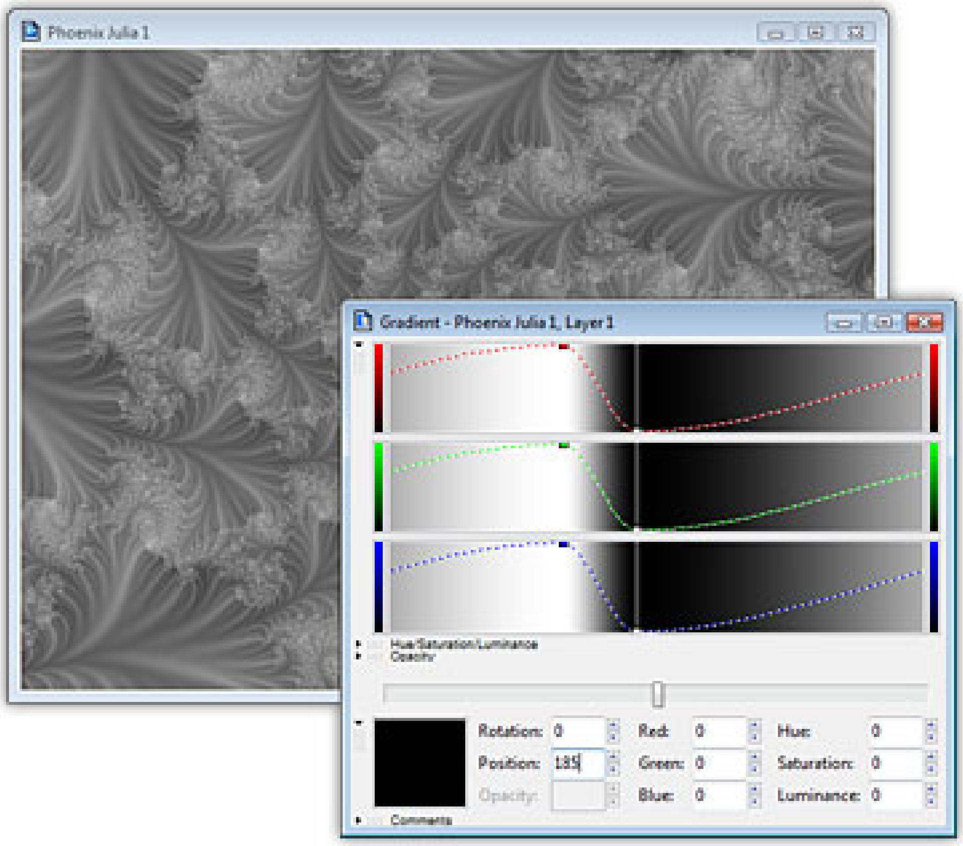
Now, looking again at the gradient editor, you will notice that there are only two sets of control points. Those in the left-most set are all at the bottom of their respective color panels — creating black, while those in the middle of the gradient editor are at the top of their panels — creating white.

- Let's move the control points around until we increase the contrast between the white and black areas. As you drag the control points, be sure to keep them pulled all the way down (for black) or all the way up (for white) in their color panels to prevent introducing color into the gradient.

After you have experimented with moving the control points, let's synchronize their positions.

- A nice contrast can be found with the set of white control points at Position setting **130** and the black control points at Position setting **185**. You may either drag the points to these locations, or click on the respective points and type their positions into the **Position** setting.

Your image and gradient should now look like this:



Rotation: 0
Position: 185
Opacity:
Red: 0
Green: 0
Blue: 0
Hue: 0
Saturation: 0
Luminance: 0

Next: [Learning about layer opacity]

Learning about layer opacity

If you will look again at the **Layers** tab of the **Fractal Properties** tool window, you will see both your layers and their tiny thumbnails listed.

Learn more about...
Working with layers

But when you look at your image, you only see the top, grayscale layer. This is because the top layer is 100% visible in "Normal" merge mode, which means that you will not see any layers underneath it.

To check that the other layer is still really there, click the **Visible** icon on the **Layer 1** layer.

This toggles this layer's visibility off and on. When the visibility is off, you will see the layer(s) underneath. When it is on, you will see the active layer.

Having multiple layers in our image is useless, though, unless we can see more than one at a time. There are many ways to do this, so let's work through them one at a time.

At the top of the **Layers** tab there are two controls. The one on the left is the Merge Mode setting (which currently says "Normal") and to its right is a horizontal slider (currently all the way to the right at 100%).

- Make sure the top layer of your fractal (*Layer 1*) is visible again. With the top layer highlighted in the layer list, click and drag the Opacity slider slowly to the left and watch what happens to your fractal.

As the top layer becomes more transparent, the bottom (*Background*) layer begins to show through. When the Opacity slider is all the way to the left, at 0%, the top layer is no longer visible at all.

Next: Learning about merge modes

Learning about merge modes

Another way to view more than one layer at a time is to change the way the layers interact with the **Merge mode** setting. This setting determines how each pixel combines with the pixel(s) directly underneath it in other layers.

Learn more about...
How layers are merged

- Make sure the opacity of the top layer is set at 100% and then click on **Normal** in the Merge mode drop-down box. Select the next option — **Multiply**.

Learn more about...
Merge modes

Notice that the colors from the *Background* layer are now visible (although they appear darker) along with the textures of *Layer 1*.

- Use the down arrow key on your keyboard to cycle through the list of merge modes. Remember that you can also change the opacity setting with any of the merge modes.
- You can also reverse the order of the layers by clicking on the title of one and dragging it up or down in the list. After reversing their order, try changing the merge modes of the *Background* (top) layer.
- When you are finished exploring all the options, make sure the *Background* layer is on the bottom. *Layer 1* should be on top with the **Hard Light** merge mode selected. Also make sure that both layers are set at **100%** opacity.
- Before we go on to the next step, save this image — either by saving the parameter file, or the fractal (as a .ufr). Keep the image name (*Phoenix Julia 1*) the same.

Next: Adding a third layer

Adding a third layer

Let's explore another way to work with layers.

Click on the *Layer 1* layer and then on the **Add Layer** button.

Note that a new layer appears at the top of the list and that it is identical to *Layer 1* in every respect except for its name, which is *Layer 2*.

These layer names start to get confusing, so let's give them more descriptive names.

- Right-click on the ***Background*** layer and choose **Rename** from the right-click menu. Type **Coloring** as its new name and press **Enter**.
- Rename the middle (*Layer 1*) layer **Texture**.
- Rename the top layer **Web**.

The *Texture* and *Web* layers still look the same, they just have different names.

With the *Web* layer highlighted, go to the **Outside** tab of the **Layer Properties** tool window and click on the **Browse** button.

- Select the ***Orbit Traps*** coloring algorithm from the right pane and click **Open**.

Tip!

One method of naming layers is to use the coloring algorithm applied to that layer, except that can get confusing if you use the same algorithm on more than one layer. Another idea is to give them functional names that relate to what effects they contribute to the overall image.

Note: There is no right or wrong way to name layers and indeed you may never want to rename them at all. You will eventually develop a system that is meaningful to you.

This creates a soft, gentle effect that you might be interested in pursuing at another time, so let's make a duplicate of the whole fractal and save it for later.

- Select **Duplicate** from the **File** menu.

Notice that the name of the duplicate is *Copy of Phoenix Julia 1*. Save the duplicate image either as a parameter or fractal file. You can choose to keep this name, or give it another, as you wish.

Next: Transparency in the gradient

Transparency in the gradient

Let's go back to the *Orbit Trap* coloring on the top (*Web*) layer and change some settings.

- Enter **5** in the **Color Density** setting, and select **rectangle** as the **Trap Shape** parameter.

We need to work with the gradient for this layer to better see the trap shapes, so open the gradient editor by selecting **Gradient** from the **Fractal** menu.

- Click and drag the set of **white** control points to the left to position **85**. Click and drag the set of **black** control points to position **86**.

This creates a sharp line between the white and black areas of your fractal.

- Now let's open up a new part of the gradient editor — the Opacity bar — by clicking on the little down arrow next to the word **Opacity**, just above the rotation slider. This opens a fourth horizontal band of color that looks right now very much like the other three.

Just as the opacity setting on the **Layers** tab controls the transparency of the entire layer, the opacity bar in the gradient editor allows us to assign different transparencies to the individual colors in our gradient.

Learn more about...
Transparent gradients

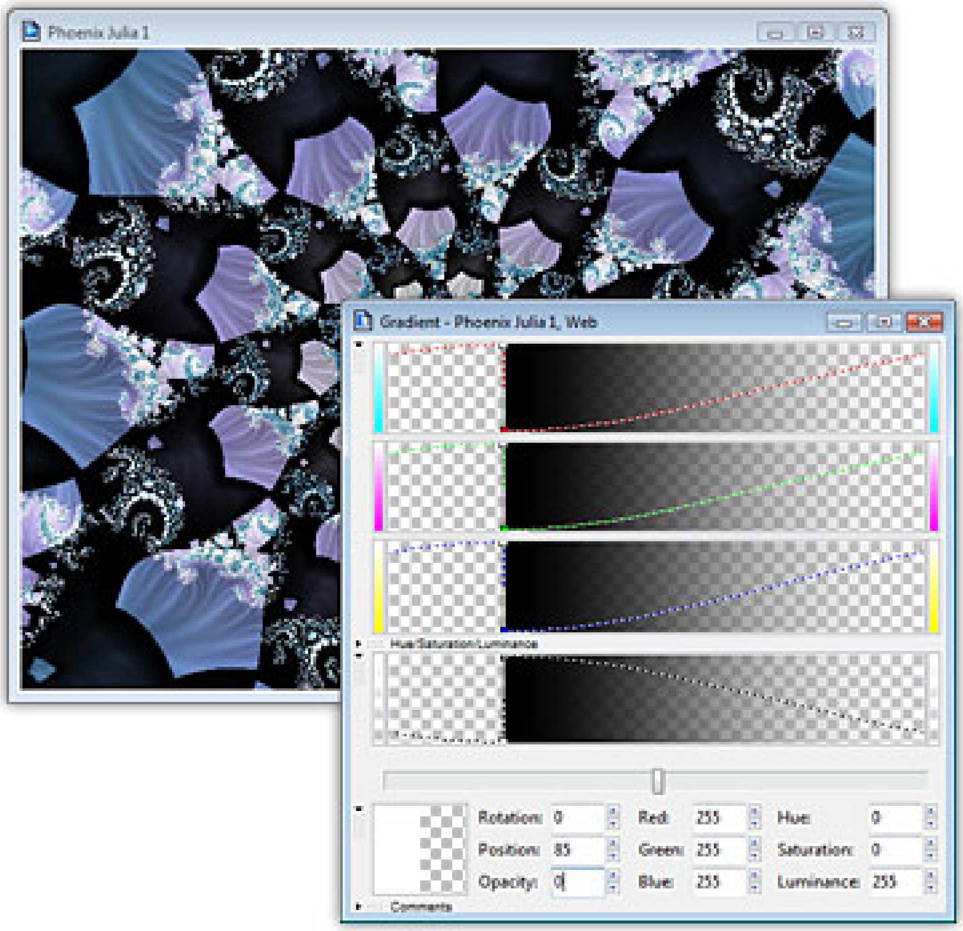
By default, the opacity bar is empty. Let's link it to the color bars so they both contain the same control points.

Click **Link Color and Opacity** on the **Gradient** menu.

- Click on the set of white control points — they are the ones at the tops of their respective color bands.
- Click and hold that same highlighted control point in the Opacity channel and drag it all the way to the bottom, keeping its position at 85.

Note: This can be a tricky maneuver with all the control points so close together. You can also enter the new opacity value (0) manually. Remember you can always ***Undo*** *an unwanted change.*

Your fractal and gradient should now look like this:



Rotation: 0
Position: 85
Opacity: 0
Red: 255
Green: 255
Blue: 255
Hue: 0
Saturation: 0
Luminance: 255

Note that the areas that were previously white are now completely transparent, allowing the *Coloring* and *Texture* layers to show through.

Next: Adding control points

Adding control points

Let's add a couple of more control points to our gradient to refine our web-like structure.

Right-click anywhere in the gradient editor and select **Insert** from the right-click menu.

- With the insert mouse pointer, click to the right of the set of black control points in any one of the horizontal color bands.
- Change the **Red**, **Green**, and **Blue** settings each to **0**.
- Change the **Position** setting to **115** and the **Opacity** setting to **255**.

This creates a black control point with 100% opacity.

- Add another control point with these settings:
  - **Red**, **Green**, and **Blue** set to **255**
  - **Position** set to **116**
  - **Opacity** set to **0**

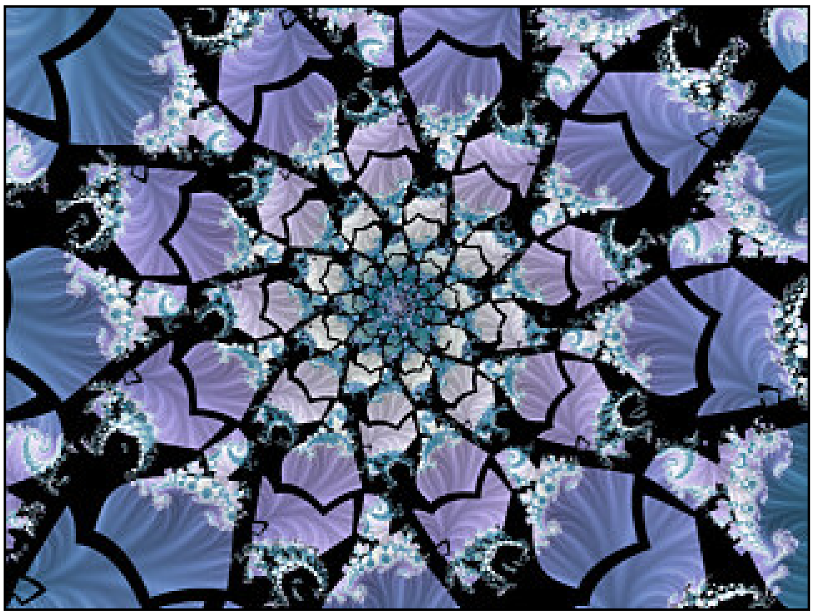
This creates a 4th set of control points — this time white, with 100% transparency.

- Save this image (either as a parameter or fractal file) as ***Phoenix Julia 2***.

Tip!

While you are working with the gradient, try to show and hide the various layers once in a while to see what is going on.

If you hide the *Texture* and *Coloring* layers to make only the *Web* layer visible, you can easily see and edit the transparent areas that the gradient creates.



For fun...

Try playing with the control points of the *Web* layer gradient and watch what happens in your image. Change their transparencies, move them around, give one or more of them color. And periodically, while you are playing around, try different merge modes and opacity settings on the **Layers** tab of the **Fractal Properties** tool window.

You can also experiment with changing the order of layers in the list to see how this affects the overall image.

Next tutorial: [Learning about transformations]()

Learning about transformations

Note: This tutorial assumes that you have completed the Quick Start, Basic Skills, and Layers tutorials. This includes: opening a new fractal, selecting fractal formulas and coloring algorithms, opening and working with the gradient editor, adding layers and changing merge modes.

In this tutorial we are going to use the ***Newton 1*** image we created in the Quick Start tutorial.

- Locate and open its parameter set. *(Hint: You will want to **Browse** the **tutorials.upr** file.)*

We are going to make a square image this time, so click on the **Image** tab of the Fractal Properties tool window.

- **Uncheck** the **Maintain aspect ratio** box and change the **Width** setting to match the Height setting.

In Ultra Fractal, **transformations** are formulas that apply special effects to the fractal. They are selected and applied to a layer on the **Mapping** tab of the **Layer Properties** tool window.

Learn more about...
Transformations

To add a transformation, switch to the Mapping tab and click the **Add** button.

- Select the ***Lake*** transformation in the **Standard.uxf** file of the "Select Transformation" browser. This applies a rippled water effect to this layer of your fractal.

Learn more about...
The *Lake* transformation

You will notice that the first two parameters for this transformation are *Water level (Re):* and *Water level (Im):*. This means that the coordinates that provide the location of the water level are a complex number. The default setting is **0, 0** and on our fractal, this places the water level at the middle of the fractal.

To change this, right-click in one of the ***Water level*** parameter fields and select **Eyedropper** from the menu that appears.

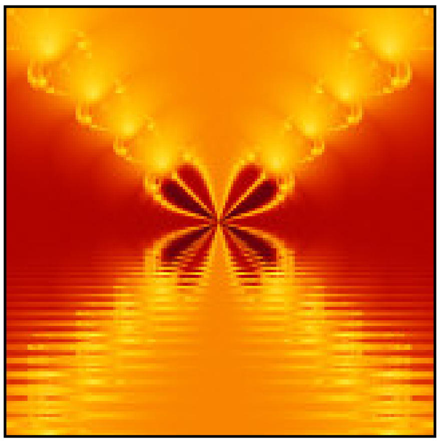
Now move your mouse cursor over the fractal image. You will notice that the cursor has become an eyedropper with crosshairs at the tip. The Fractal Mode tool window shows what the fractal would look like with the value currently under the mouse cursor.

- Choose a new placement for the water level with the crosshairs and click the left mouse button.

The fractal immediately redraws, placing the water level at these new coordinates. Try different locations.

- When you are finished playing with this nifty tool, **Check** the **Use screen center** option on the Mapping tab.

This will override any *Water level* settings and return the level to the center of your fractal so that your image looks like this:



Next: Using the Kaleidoscope transformation

Using the Kaleidoscope transformation

Before we add another layer, let's rename this layer in the **Layers** tab of the **Fractal Properties** tool window.

- Rename the current layer **Lake**.
- Add another layer and rename it **Sky**.

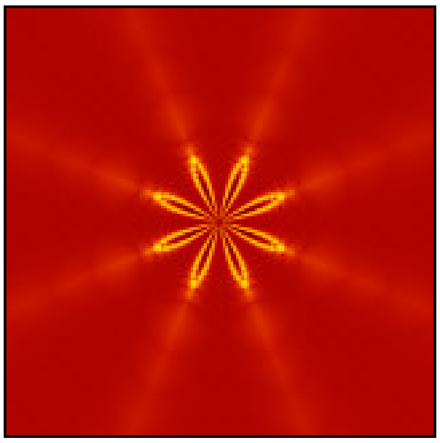
Go ahead and leave the **Normal** merge mode and **100%** opacity settings as they are. This means we will not be seeing the bottom layer for a bit.

On the **Mapping** tab of the **Layer Properties** tool window, delete the *Lake* transform by clicking the **Delete** button.

The layer again looks like our *Newton 1* image from the Quick Start tutorial.

- Add a new transformation — the ***Kaleidoscope*** transformation in **Standard.uxf**.

We are going to keep the default settings for this layer, but this transformation has a lot of interesting effects you may want to pursue at a later time.



Learn more about...
The *Kaleidoscope* Transformation

Next: Using 3D Mapping

Using 3D Mapping

It is possible to add more than one transformation to the same fractal layer, so let's do just that.

- **Add** the **3D Mapping** transformation on top of *Kaleidoscope*.

Learn more about...
[Multiple transformations]

Learn more about...
The *3D Mapping* transformation

The idea of the 3D Mapping transformation is to map the fractal layer onto a three-dimensional surface. Let's first use the default, *Plane* shape.

As you can see, our kaleidoscopic flower shape has been mapped onto the lower half of the fractal and the top half of the image is now black. This gives the effect of a floor extending toward the horizon.

We can make this mapping shift to the top half of the image, to become more of a sky-effect, by changing one of the transformation parameters. It is not immediately clear which of the parameters effects this change, so now is a good time to learn another way to get **Help**.

- Make sure the Fractal Mode tool window is visible, and move the mouse cursor over the different parameters for the 3D Mapping transformation.

The tool window will display a help text for the parameter currently under the mouse cursor.

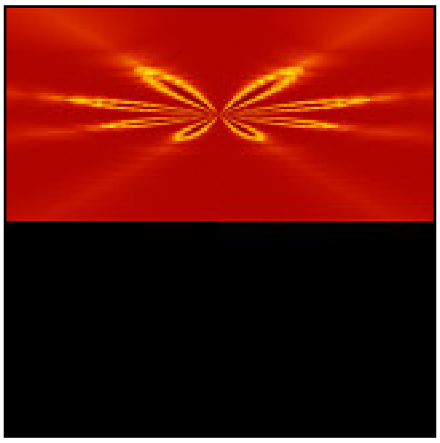
In this case, a little investigation tells us that the parameter we need to change in order to make our fractal appear in the "sky" is the **Y Translation** parameter. It is currently set to **-0.5**.

- If we change that to **0.5**, the fractal moves into the upper half of the image.

Tip!

You can also click on the **?** button in the title bar (in the upper right corner) of the **Layer Properties** tool window and place your cursor over any parameter field or transformation setting. When you left-click, a helpful hint for that particular parameter or setting will pop up on the screen.

Conscientious formula writers will provide helpful hints for their formulas' parameters.



Now... what is this big black area doing in our image? Some transformations, like *Lake* and *Kaleidoscope* warp the fractal but still fill the entire image.

Learn more about...
Solid color in transformations

Others, like the *3D Mapping* transform, map the fractal onto an object or create a mask that does not fill the entire image. The leftover area that is not fractal is filled with a solid color, which is designated on the Mapping tab.

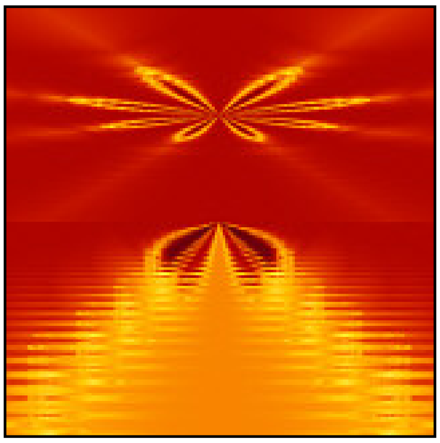
- To see this, click on the black area next to the **Solid Color** setting. This opens a Select Color dialog.

To change the solid color from the default Black to another color, adjust the sliders. You will see the changes immediately on your image. But what we really want to do in this image is to make the black area transparent.

- Drag the **Opacity** slider all the way to the left.

Now we can see both the *Lake* effect from the bottom layer and the *Kaleidoscopic* flower in the sky plane of our top layer.

- Click **OK** to close the "Select Color" dialog.



Next: Twist transformation

Twist transformation

- Add a third layer and rename it **Sphere**.
- Delete the *Kaleidoscope* and *3D Mapping* transformations from the Mapping tab.
- Add a new transformation — ***Twist***.

This is a fun transform, because you can use the eyedropper to select the center of the twist. Remember to right-click in one of the **Twist Center** parameters to select and activate the **Eyedropper**, or click the eyedropper button that pops up when you click on the parameter.

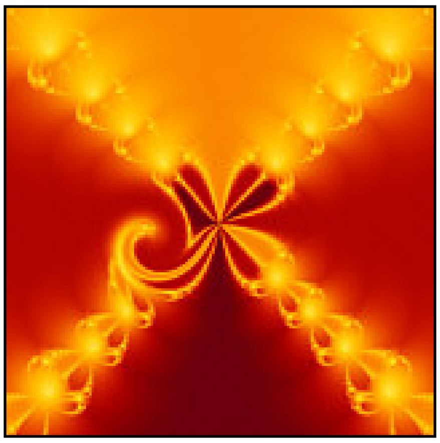
Learn more about...
The *Twist* transformation

The **Strength** and **Decay Factor** parameters affect the tightness and shape of the spiral. Try making them smaller and larger. Try using a negative number in the **Strength** parameter. Remember that you can use the Explore tool as well.

When you are done playing with these parameters, click on the complex number below to copy it to the Clipboard.

-0.65 / -0.18125

- Paste it into the **Twist Center** parameter by selecting **Paste Complex Value** on the right-click menu.
- Enter **4** in the **Strength** setting and **10** as the **Decay Factor**.



Next: Mapping a sphere

Mapping a sphere

Let's add another mapping transformation to this layer.

- Load ***3D Mapping*** again and this time select the **Sphere** shape.
- First, let's change the solid color opacity to **0**.
- Next, to center the sphere in the image, change the **Y Translation** setting to **0**.

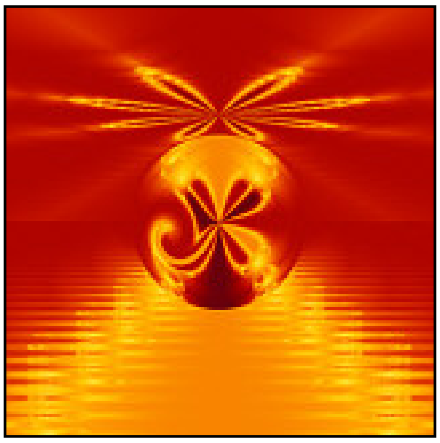
Tip!

You can resize the list of transformations by clicking on the dividing line above the parameters and dragging downwards.

The help hint for the **Z Translation** setting says that increasing the value will move the sphere farther away (thus making it smaller).

- Let's change that setting to **2.5**.

Now the sphere is positioned just below the Kaleidoscopic flower.



For fun...

Transformations are applied in a particular order, starting with the bottom one on the list and working upward. So it matters, sometimes, in which order they are placed in the list.

For fun, reverse the order of the *Twist* and *3D Mapping (Sphere)* transforms by dragging one above or below the other.

As you can see, the sphere is now mapped before *Twist* is applied resulting in an unusual effect.

Next: Adding a frame

Adding a frame

In this last part of this tutorial, we are going to add a simple frame to our image. One way to do this is to create a solid color layer (which will become the frame) and then clip out a transparent area in the middle of the image through which the other layers below may be seen.

- Add a new, fourth layer, name it **Frame**, and remove the transformations on the Mapping tab.
- Click on the **Formula** tab of the **Layer Properties** tool window and replace the current (*Newton*) formula with the ***Mandelbrot*** formula in *Standard.ufm*.

Before we go on, notice the black area on the inside of the Mandelbrot figure. Up until now, in the images with which we have worked, we have only dealt with "Outside" points and "Outside coloring." In this layer, we are going to work with the **Inside** points.

Learn more about...

Inside and Outside areas

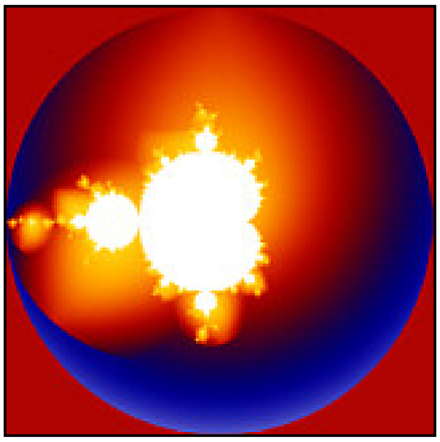
The coloring of inside points is controlled on the **Inside** tab of the **Layer Properties** tool window. Switch to the Inside tab and notice the **Transfer Function** setting. When the Transfer Function is set to *None*, the coloring algorithm is ignored and the **Solid Color** setting takes effect.

Learn more about...

Solid Color

To see how this works, let's change the solid color to something other than black.

- Click on the **Solid Color** swatch on the **Inside** tab. Click and drag the **Luminance** slider to **255**, and note that the inside of the Mandelbrot figure becomes solid white.



Next: Zooming with multiple layers

Zooming with multiple layers

We will use the solid white area in the center of the *Frame* layer as the basis of our frame so we are going to zoom into it without changing the location of any of the layers below.

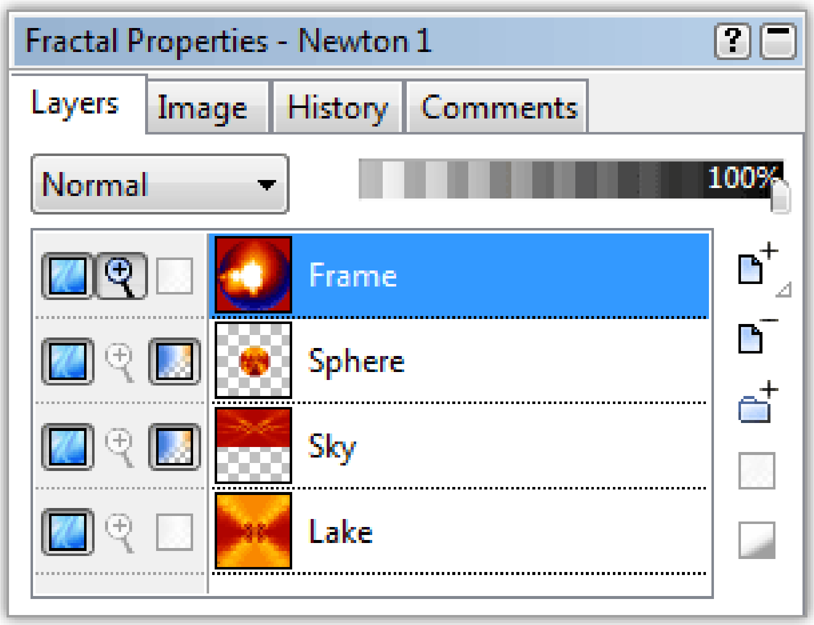
To do this, click on the **Layers** tab of the **Fractal Properties** tool window.

Locate the **Editable** icon on the top (*Frame*) layer. While holding down the **Shift** key, **click** on this icon.

This disables the editability of all the other layers.

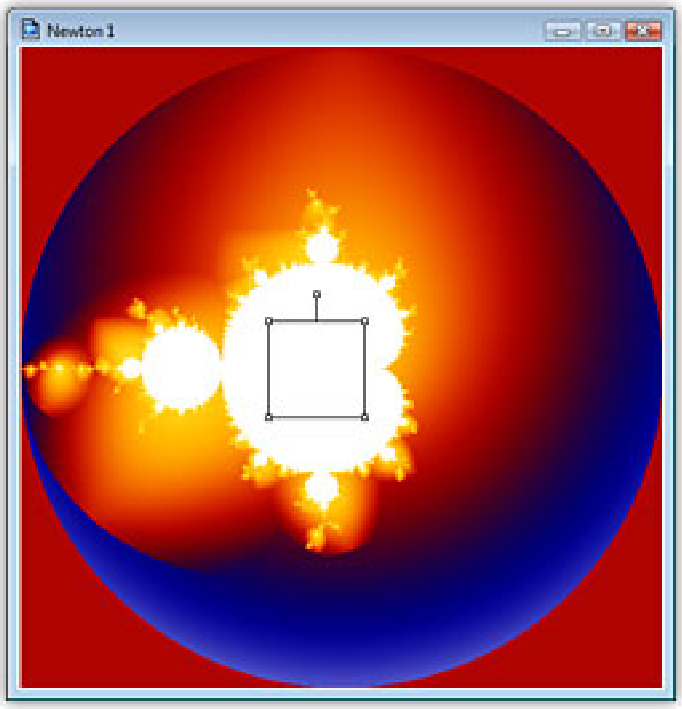
Tip!

By holding down Shift while clicking on the Visible, Editable, or Transparent icons, you toggle all other layers instead. This also works with the Enable icon on the list of transformations on the Mapping tab.



Having the Editable icon enabled for just this layer means that any location changes we make will only affect this layer. The layers for which the Editable icon is grayed out will remain unchanged.

- Now choose **Select Mode** from the **Fractal** menu to activate the selection box. Click and drag on one of the sides to make the selection box small enough to fit entirely into the solid white area, something like this:



- Select **Zoom In** from the **Fractal** menu.

Your top layer should now be solid white. You can verify this by looking at your **Layers** list.

And, by **Shift-clicking** the **Visible** icon on each layer in the list, you can see that none of the other layers has changed location.

*Note: If something went awry, you can always use the **Undo** option on the **Edit** menu to recover.*

Next: Using the Clipping transform

Using the Clipping transformation

The last transformation we will explore in this tutorial is ***Clipping***. This handy transform has many uses, but we will use it in this image to clip out the inside of our frame.

Learn more about...

The *Clipping* transformation

- Add the ***Clipping*** transformation to the *Frame* layer.

First, we will center the frame on the screen.

- Switch to the **Location** tab, right-click on one of the **Center** settings and select **Copy Complex Value**.
- Then switch to the **Mapping** tab and right-click on either of the **Clipping Center** parameters and select **Paste Complex Value** from the right-click menu.

This ensures that the clipping center is set to the center of the screen.

- On the **Mapping** tab, find the **Region** parameter and change it to **inside** — because we will be clipping out the inside area.

The image will become black — the designated solid color for this transformation.

- To select our frame width, right-click in one of the ***Right Edge*** parameter fields and select **Eyedropper** from the menu. Place the eyedropper (your mouse cursor) near the right edge of the image and **left-click**.

You can repeat this until your frame is the desired width. You should now have a white frame with a black square inside.

- Click on the **Solid Color** swatch on the **Mapping** tab and change the **Opacity** to **0**.

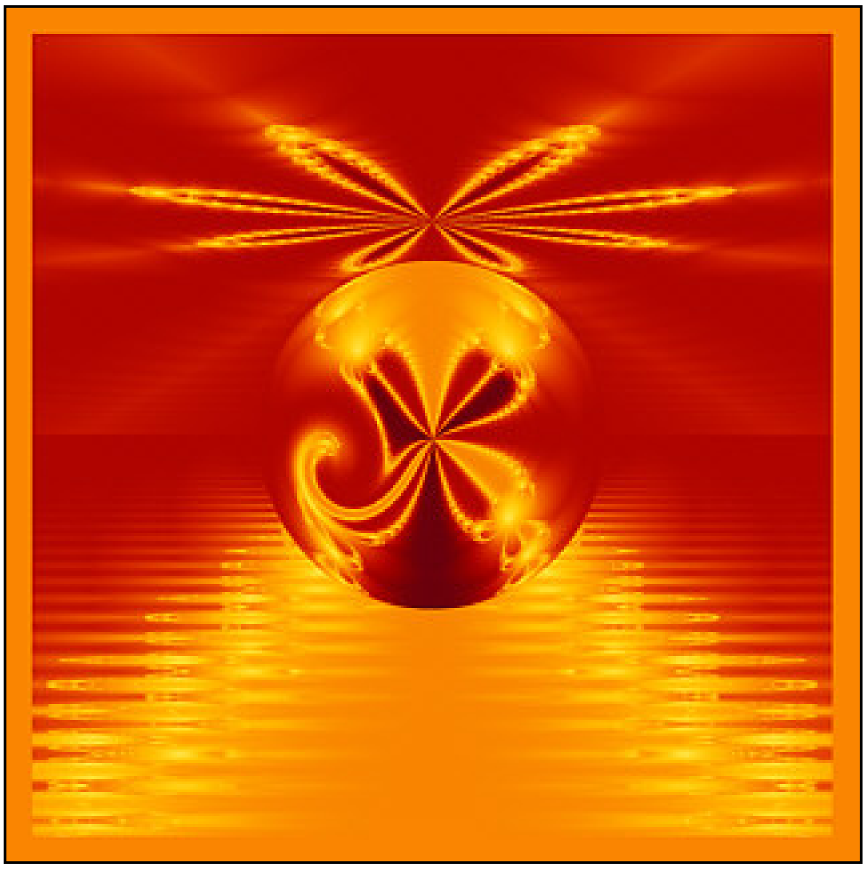
You should now be able to see the underneath layers surrounded by a white frame. As one last finishing touch, let's change the color of the frame to coordinate with the fractal by using the eyedropper tool.

- Switch to the **Inside** tab and right-click on the white-colored **Solid Color** swatch. Select **Eyedropper**.

As you move the eyedropper over the image, the color underneath the center of the crosshairs is shown in the **Solid Color** swatch.

- When you have found a color you like, **left-click**.

This changes the white frame to a color that coordinates with your image. With the exception that your frame may be a different color and/or width, your image should now look like this:



Next: Exporting the image

Exporting the image

Let's save this image in both parameter and fractal file form. Save it with the name **Newton World**.

We can also export images to graphic file formats that can be used outside of Ultra Fractal. This is useful when you want to print an image or post it on the web.

- To export the image, click **Export Image** on the **File** menu.

By default, the image will be saved in Ultra Fractals' **Export** folder with the name we have given it (*Newton World*).

- Ultra Fractal supports several file formats so let's select **JPEG** from the **Save as type:** drop down list.
- When you click **Save**, Ultra Fractal will ask you to select the export quality for the JPEG image. Move the slider to 95%. This will allow for some compression (which makes the file smaller) without too much loss of quality. Click **OK**.

Learn more about...

Exporting

Learn more about...

File Formats

Now you may open the image up in another graphics program, email it to a friend, or post it on a web page.

***Note**: All exported and rendered images made with an evaluation copy of Ultra Fractal will be marked with* Evaluation Copy *text. Please purchase your copy of the software!*

Next tutorial: Masking

Introduction to Masking

One of the most exciting features of Ultra Fractal is the ability to create layers that serve as masks for other layers. These layers contain areas of both transparency and opacity that allow only designated areas of the linked layer to be visible. This ability opens up a whole range of artistic possibilities that have never before been available in fractal software.

Before we get to the actual masking concept, though, let's create a new image using some of the skills we have learned thus far.

- Create a **New Fractal** using the ***Julia*** formula.
- Click on the complex number below to copy it to the Clipboard.

-0.815 / 0.235

- Right-click on the **Julia Seed** parameter on the **Formula** tab and click **Paste Complex Value.**
- Apply the ***Triangle Inequality Average*** coloring algorithm on the **Outside** tab.

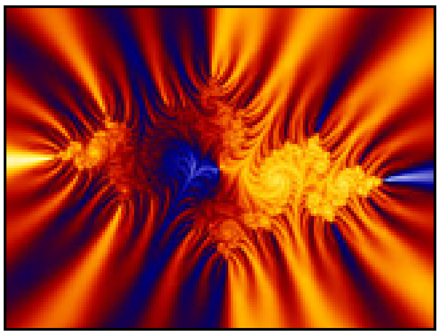
Note that the *Julia* calculation formula and the *Triangle Inequality Average* coloring algorithm each have a **Bailout** parameter. This tells Ultra Fractal how many times to iterate the formula before designating a point "inside" or "outside."

Learn more about...
Inside and outside points

In this case, the bailout setting for the *Julia* formula is **4** and the bailout for the *Triangle Inequality Average* coloring is **1e20** — a much higher number (100 sextillion) that, for our purposes, approximates infinity.

- This coloring algorithm is intended to work best when the formula and coloring have matching bailout values, so let's change the **Bailout value** on the **Formula** tab to **1e20** to match the higher value on the Outside tab.
- Next, open the Gradient Editor and rotate the rotation slider to the left until the **Rotation** setting is **-137**.

Our first layer should look like this:



- In this image, we are going to name our layers by the coloring algorithm used, so **Rename** this layer to **TIA**.

Next: Layer 2 - Waves Trap

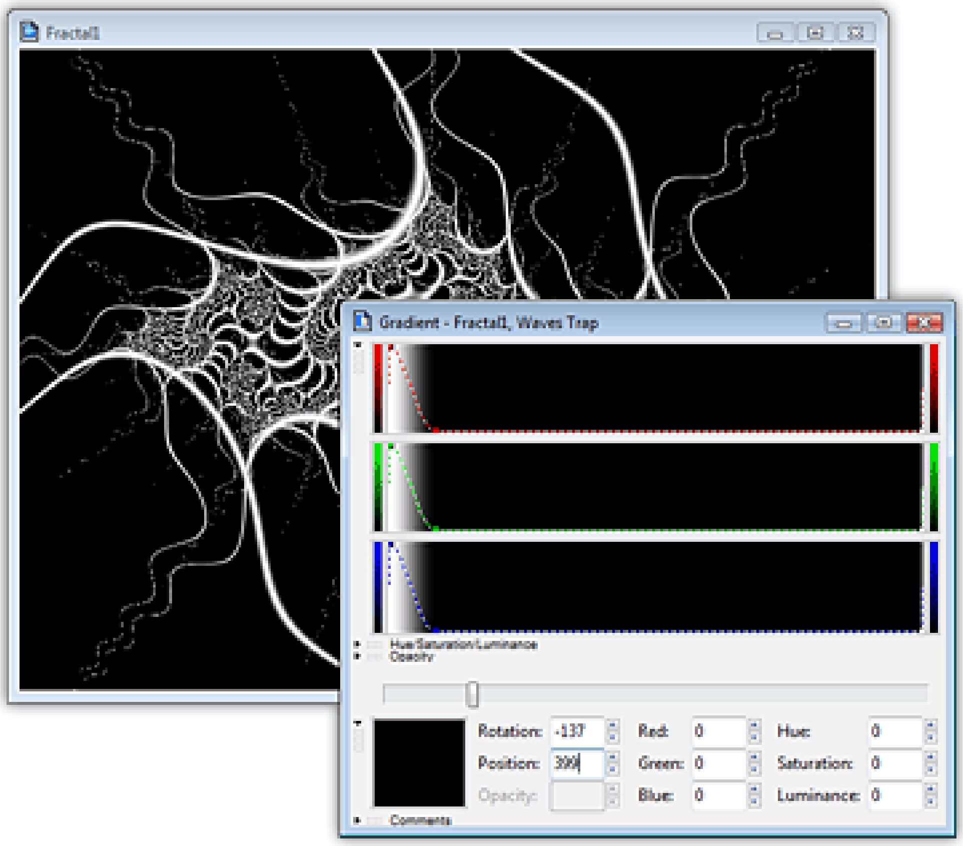
Layer 2 - Waves Trap

- **Add** a new layer and **Rename** it **Waves Trap**.
- Replace the current **Outside** coloring with ***Orbit Traps*** and make the following setting and parameter changes:
- Change the **Transfer Function** to **Log**
- **Uncheck** the **Repeat Gradient** box
- Change the **Trap Shape** to **Waves**

Now, open the gradient editor to edit the gradient for this layer to meet the following conditions:

- Three control points — *You can delete unneeded control points with the right-click menu*
- **Position** the first (left-most) control point at **0** and color it **White**
- **Position** the second control point at **35** and color it **Black**
- **Position** the third control point at **399**, also colored **Black**

Your image and gradient editor should look like this:

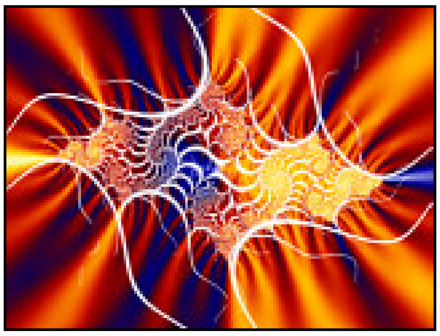


Rotation: -137
Position: 399
Red: 0
Green: 0
Blue: 0
Hue: 0
Saturation: 0
Luminance: 0

- Change the **Merge mode** on the **Layers** tab of the **Fractal Properties** tool window to **Screen**.

The *TIA* layer now shows through the white filaments of the wave trap.

Learn more about...
[Merge modes]



Next: Layer 3 - Box Trap

Layer 3 - Box Trap

Let's do some fun things with gradient transparency.

Learn more about...
Transparent gradients

- **Add** a new layer and **Rename** it **Box Trap**.
- Change the **Trap Shape** parameter on the **Outside** tab to **Box**.

Switch back to the **Layers** tab and **Shift-click** the **Visibility** icon on the *Box Trap* layer to toggle the other two layers off. This will help us to better see what we are doing with this layer.

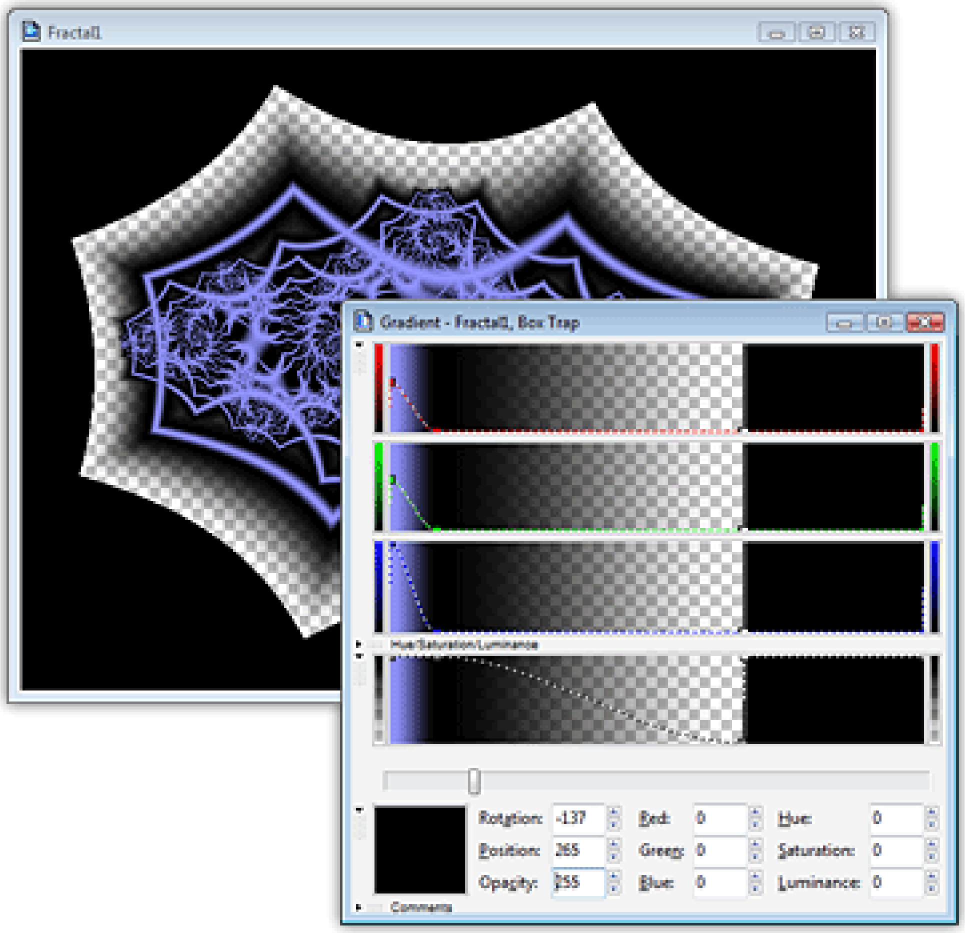
Since the Waves trap tendrils are white, let's make the Box trap shape a different color.

- In the gradient editor, edit the set of white control points so that the **Red**, **Green**, and **Blue** settings are **145**, **147**, and **253**, respectively.

Select **Link Color and Opacity** from the **Gradient** menu. This allows us to move and edit the color and opacity curves simultaneously. Also, make sure that the opacity part of the gradient editor is visible — click the small opacity button to expand it if necessary.

- **Insert** a new point in the gradient editor. Set the **Opacity** of this control point to **0**. Also set the **Red**, **Green**, and **Blue** settings to **0** to create Black.
- Now **Insert** another control point to the right of that one. Change its **Opacity** to **255** and make it black, also. Click and drag this point until it is positioned just to the right of the transparent point.

Although the exact location of the transparent areas may differ slightly, your image and gradient editor will look similar to this:



Rotation: -137
Position: 265
Opacity: 255
Red: 0
Green: 0
Blue: 0
Hue: 0
Saturation: 0
Luminance: 0

The gray and white checkerboard on both the gradient and the fractal layer indicates areas of transparency.

- To see how this works, change the **Merge Mode** of the *Box Trap* layer to **Normal** and then **Shift-Click** its **Visible** icon to toggle the other layers on and off.

Notice that the underneath layers are only visible in the checkerboarded area of the top layer.

Next: Fine-tuning the gradient

Fine-tuning the gradient

We now have five control points in our gradient. The two on the left (bluish-purple and black) control the box trap structure in the center of our image. The next two, one transparent and one opaque, create a sharp scalloped line outside of the box trap structure. And the fifth (black/opaque) control point is positioned at the far right of the gradient editor.

Tip!

You can resize the gradient editor to make it easier to edit control points that are closely spaced together.

Insert a new control point somewhere between the second and third points. Color it black, with **0** opacity.

- Click and drag this new point slowly to the left and watch how the spaces inside the box trap structure become transparent. Place this point just to the right of the second (black) control point.
- Hold down the **Ctrl** key and click on the second (black/opaque) and third (black/transparent) control points. This selects both control points, allowing them to be moved together.
- Since we want to maintain their color and opacity values as they are moved, hold the **Shift** key down and drag the two points left and right. Notice that moving the two points to the right makes the shapes fatter, and to the left, thinner.
- Find a location for these two points that appeals to you, then click elsewhere on the gradient to deselect them.

Learn more about...

Keyboard shortcuts for gradient editors

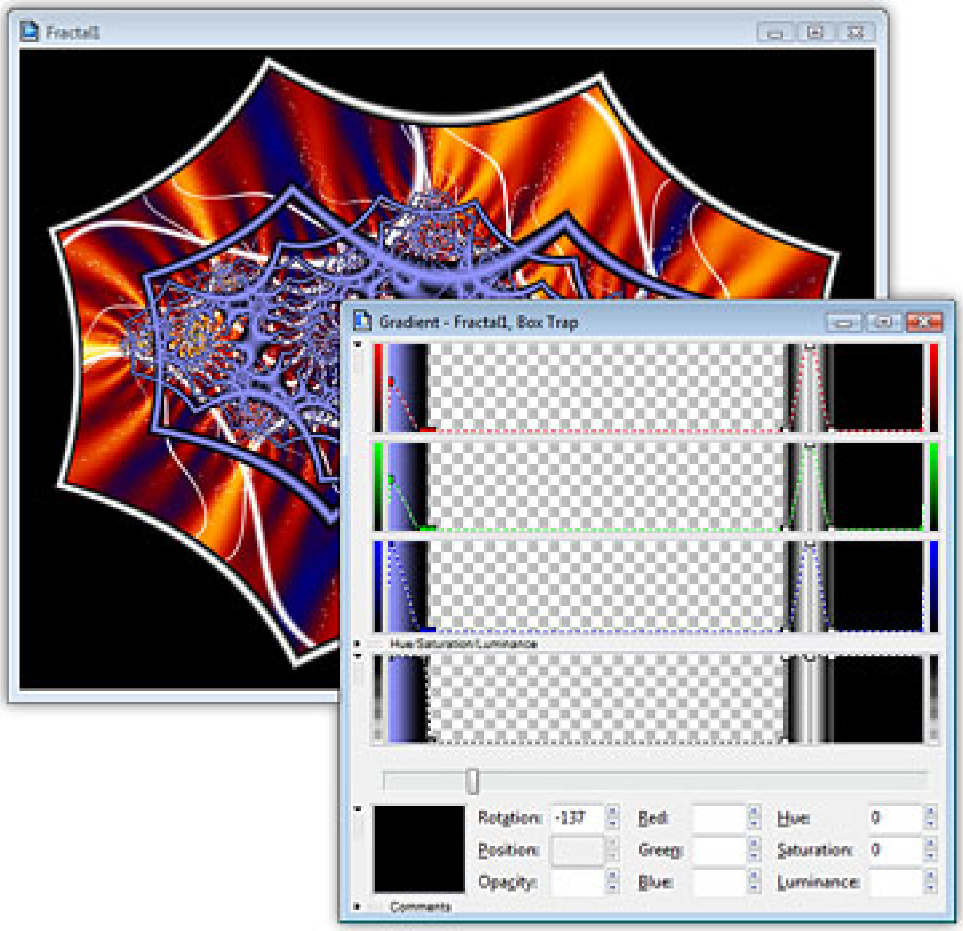
Now, let's work more with the next two control points — those that control the outer, scalloped edge. We are going to add a little sculpted edge to the scalloped frame.

- To the right of the fourth (black/transparent) and fifth (black/opaque) points, **Insert** a new control point. Make it white and set its **Opacity** to **255**. Move it close to the fourth and fifth points.
- **Insert** one more point to the right of the white one. Make this one black and fully opaque, also. Move it close to the white point.
- Experiment with the spacing of these two newest points -- dragging them a little to the right and left to see how they affect the width of the scalloped edging. Find an arrangement that appeals to you.
- Now select and move the grouping of our four scalloped edge control points, maintaining their spacing and coloring. **Ctrl-click** each to select and add them to the group. Hold the **Shift** key as you drag them right or left to position the edging. Find a position that appeals to you.
- **Shift-click** this layer's **Visible** icon to toggle all the layers on.

Learn more about...

Editing gradients

Your image should look something like this:



Rotation: -137
Position:
Opacity:
Red:
Green:
Blue:
Hue: 0
Saturation: 0
Luminance:

Save your image as a parameter and/or a fractal file. Name it ***Masked Julia***.

Next: Layer 4 - Gaussian Integer

Layer 4 - Gaussian Integer

We are almost ready to learn about masking, but first we need to add one more layer.

In the Layers tab of the Fractal Properties tool window, **Add** a new layer and **Rename** it ***Gaussian Integer***.

- Replace the current **Outside** coloring with ***Gaussian Integer***.

No changes to the parameters are needed but we do need to work with the gradient a bit. Many of the control points we added in the last layer are not needed here. We want to keep the first two control points on the left side of the gradient. They are, respectively, bluish-purple/opaque and black/opaque.

Learn more about...
The *Gaussian Integer* coloring algorithm

We will not need the third (black/transparent) point, so click on it, **right-click** in the gradient editor, and select **Delete** from the menu.

We also want to keep the black/opaque point at the very right of the gradient editor. But we can delete the group of four points that control the scalloped frame.

- **Ctrl-click** to select each of them, **right-click** in the gradient editor and select **Delete** from the menu.
- Now, to make the little dots in the image a little bigger, click and drag the second (black) control point to the right, somewhere around the **Position** of **40**.

Tip!
Most right-click commands are also available in the Gradient pull-down menu and on the toolbar.

Next: Adding a mask layer

Adding a mask layer

We now have little bluish-purple dots covering the entire image. We could make the black areas transparent or change the merge mode to allow us to see the underneath layers, but wouldn't it be great if we could have the dots appear only in the solid black areas outside the scalloped frame?

Editing the transparency of the gradient will not accomplish this, nor will changing the merge mode or layer transparency. What we need to do is create a **Mask** for this layer that has the same shape as the scalloped edge in the *Box Trap* layer.

Learn more about...
Masks

Go to the **Layers** tab of the **Fractal Properties** tool window.

Click on the ***Box Trap*** layer and then click the **Add layer** button.

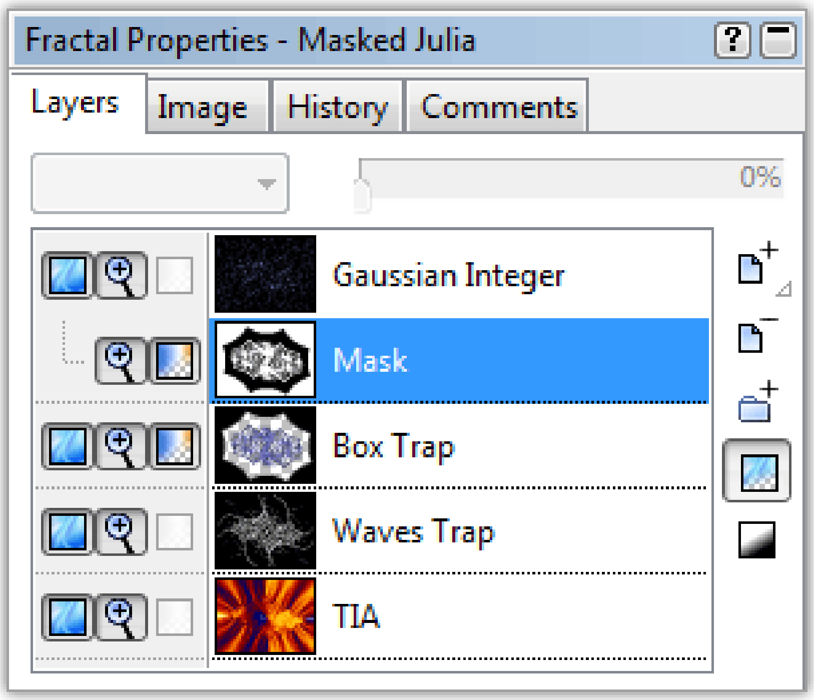
This adds the new layer between the *Box Trap* and *Gaussian Integer* layer.

- **Rename** this layer ***Mask***.

But it is not a mask layer yet until we associate it with the *Gaussian Integer* layer.

To turn the layer into a mask, click the **Use as Mask** button.

Look at your image and notice how the dots no longer appear inside the scalloped frame (except on the box trap structure inside, which we will fix in a minute).



Also notice on the layer list that the *Gaussian Integer* and *Mask* layers now share a **Visible** icon. If you **shift-click** this icon to toggle the other layers off, you will clearly see which areas are visible and which are made transparent by the mask layer.

Next: Editing the mask

Editing the mask

Now let's edit the mask itself. To make this easier, we need to make the mask layer visible on its own temporarily.

Learn more about...
[Working with masks]

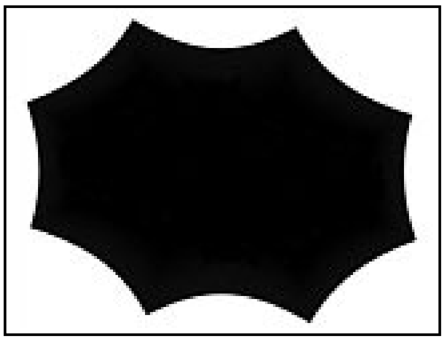
Click on the ***Mask*** layer and then on the **Show Mask Only** button.

Masks are always shown in black and white — never any colors. White represents the areas that are transparent and black represents the opaque, masked areas.

Looking at our *Mask* layer and its gradient, can you see what we need to do in order to clear out the center of the scalloped frame? (Make sure the opacity part of the gradient is visible before you continue.)

Since the first two control points on the left are white, and they correspond with the inner structure on the *Box Trap* layer, they are the ones we need to edit.

- Drag the first two points downward, making them black, so that the mask looks like this:



- Click the **Show Mask Only** button again (so it is no longer down) and make sure that the bottom three layers are now visible.

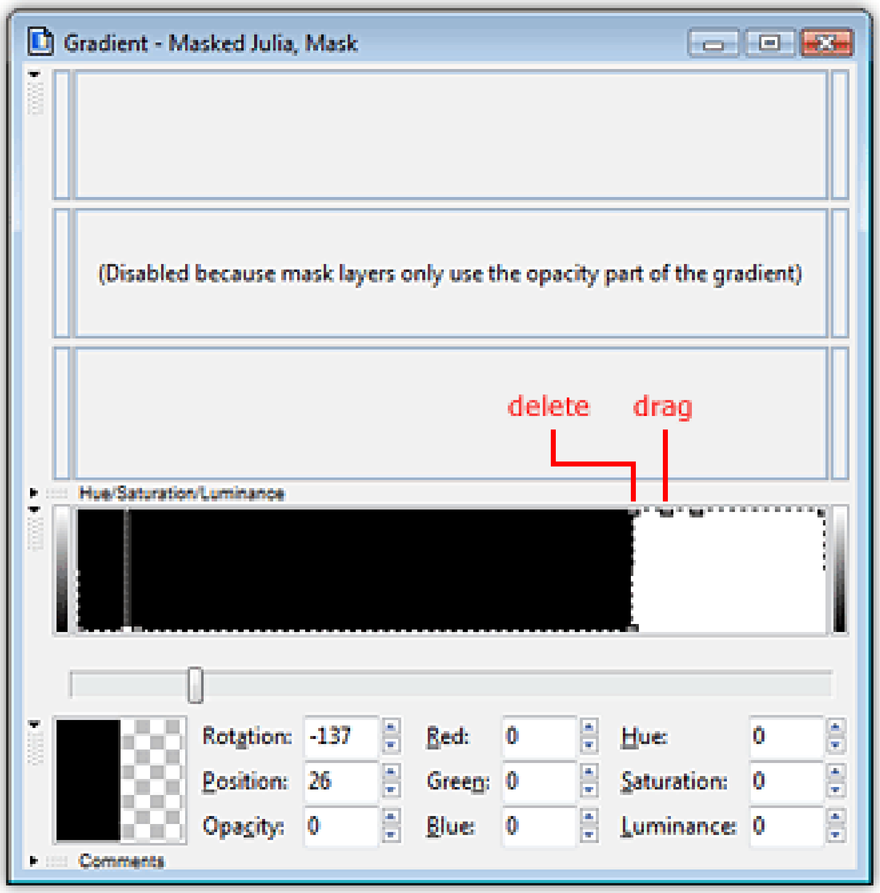
You should see the rippling of the *TIA* layer, the white tendrils of the *Wave Trap* layer, the bluish-purple structure of the *Box Trap* layer, and the dots of the *Gaussian Integer* layer — masked to only appear outside of the scalloped edge.

Tip!
When working with masks, you will often toggle the **Show Mask Only** button on and off to alternatively work on the mask and judge its effect on the final image.

What is missing, though, is the little white edging we created around the scalloped frame.

With all the layers still visible, click again on the ***Mask*** layer. There are still some control points on its gradient that are preventing us from seeing the white edging.

Locate the two white points indicated in the screenshot below:

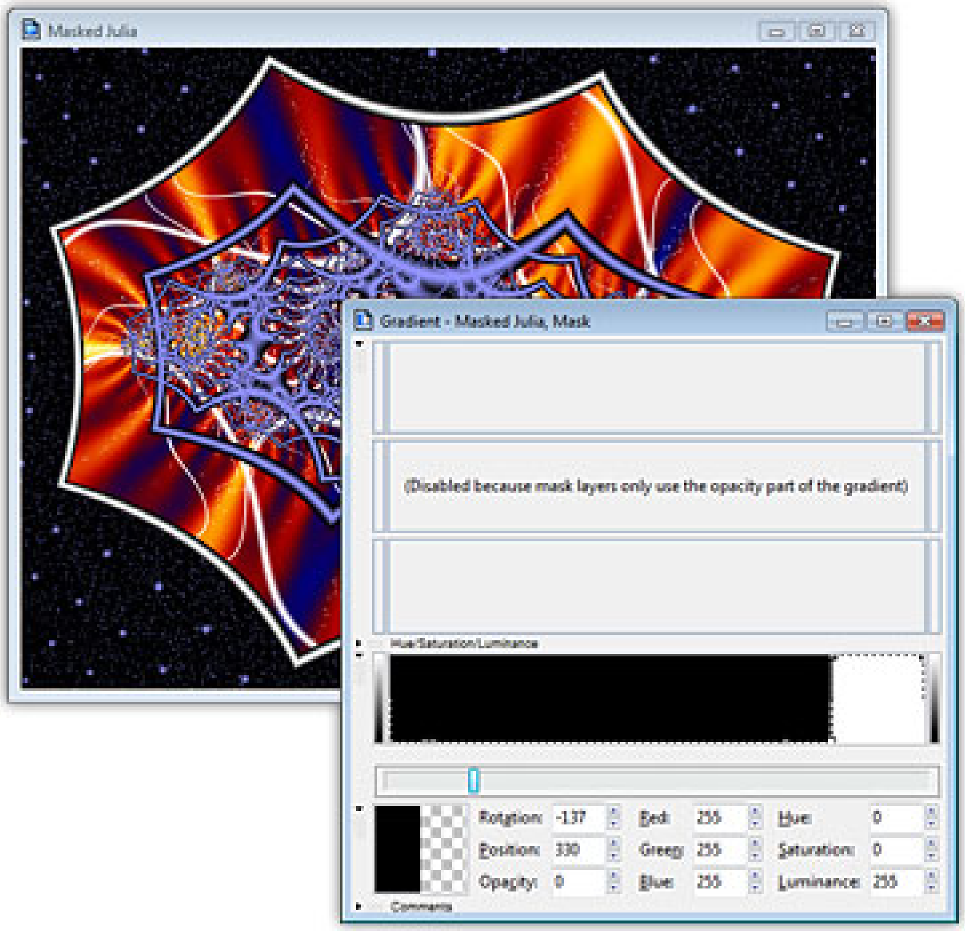


(Disabled because mask layers only use the opacity part of the gradient)

Rotation: -137
Position: 26
Opacity: 0
Red: 0
Green: 0
Blue: 0
Hue: 0
Saturation: 0
Luminance: 0

- Click on the first (left) of these two points and **delete** it.
- Click on the second point and drag it downward (making it black) and to the right, just next to the white control point.

You should now see the white edging from the *Box Trap* layer along the scalloped frame.



(Disabled because mask layers only use the opacity part of the gradient)

Rotation:	-137	Red:	255	Hue:	0
Position:	330	Green:	255	Saturation:	0
Opacity:	0	Blue:	255	Luminance:	255

Next: [Rendering the image]

Rendering the image

Now that our image is complete, save it again, in either the parameter or fractal file format you chose earlier.

There is one other method of saving images that you will want to use on occasion. If you want to make a larger render than is practical onscreen or render the best-quality image, you will want to use Ultra Fractal's **Render to Disk** feature.

Learn more about...
Rendering images

For fun, let's make a render of this image that you can use as your desktop wallpaper.

To start the disk render, select **Render to Disk** from the **Fractal** menu.

In the **Destination File** field, Ultra Fractal will suggest a file name for the rendered image. Write it down so you will know where the image is going to be saved.

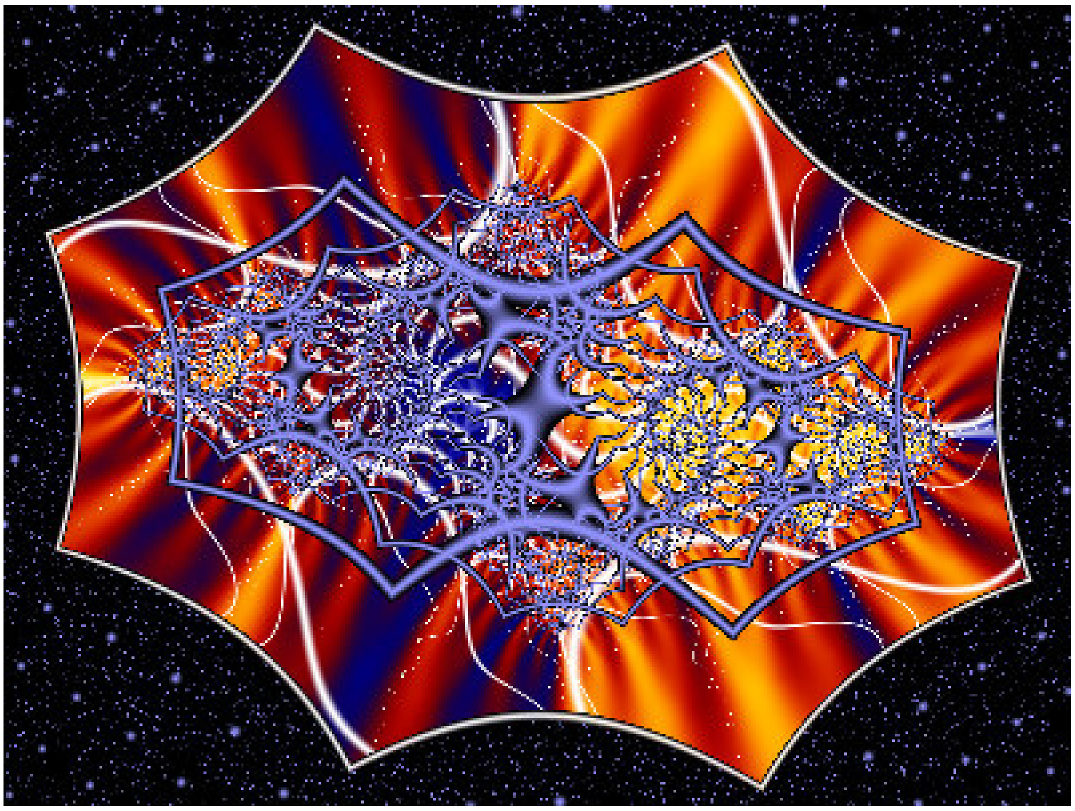
- Make sure that **Bitmap image (*.bmp)** is the selected **File Type**.
- In the **Size** field, enter the **Width** and **Height** of your Windows desktop (for instance, **1024** and **768**).

*If you do not know your desktop setting, minimize Ultra Fractal and **right-click** on your Windows desktop. Select **Properties**. Click on the **Settings** tab and look for the **Screen resolution** that is selected.*

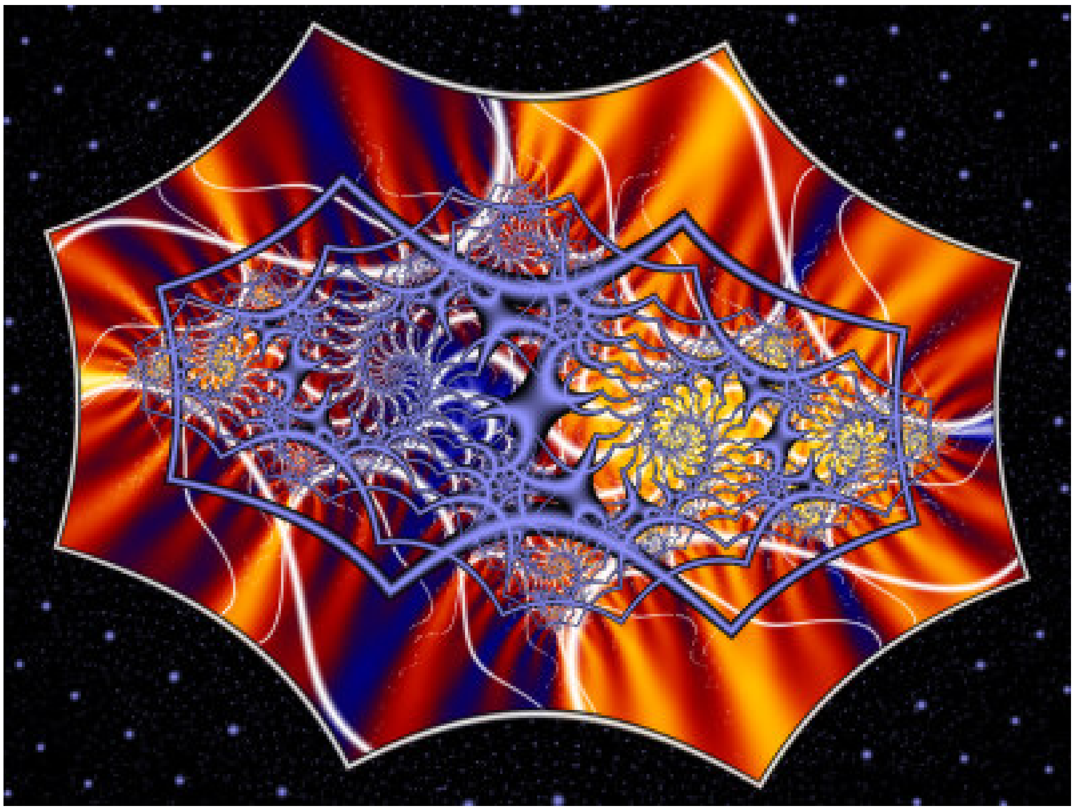
In most cases, the **Anti-aliasing** setting of **Normal** is sufficient.

Learn more about...
Anti-aliasing

The difference between an exported and an anti-aliased render of this image is demonstrated in the two examples below:



Exported



Rendered to disk with anti-aliasing

Notice how much smoother and cleaner the second image is. Fine filaments, like the white wave tendrils, are much nicer when the image is anti-aliased.

***Note**: Remember that all exported and rendered images made with an evaluation copy of Ultra Fractal will be marked with* Evaluation Copy *text. Please purchase your copy of the software!*

- To make sure the fine dots and filaments are rendered correctly, enable the **Force Linear drawing method** option. This forces Ultra Fractal to use the One-pass Linear drawing method on all layers which produces more accurate results than the default Guessing drawing method.
- You can accept all the other default settings in the Render to Disk tool window so click **OK** to start the render.

As the render starts, you will notice that the **Render to Disk** tool window on the right side of your screen opens. This window monitors and shows the progress of the render.

Learn more about...
Managing render jobs

- When the render is complete, right-click on the Windows desktop and select **Properties**. On the Desktop tab, click the Browse button and locate the rendered image. (Usually, it will have been saved as *My Documents\Masked Julia.bmp*). Click **OK**.

Next: Some final thoughts

Some final thoughts

These tutorials are intended to introduce many of the features of Ultra Fractal. They, by no means, cover all of the program's capabilities. The creative possibilities, particularly in the use of masking layers, are endless.

You are encouraged to work through the tutorials more than once, to become familiar with the program's user interface and the various concepts and skills introduced in them. As you become more comfortable, try experimenting with different parameter values and settings; move layers around and alter their gradients; use different merge modes and opacities. If you will take the time, at first, to explore what is possible using just these tutorial images, you will have a much better idea of how to achieve the effects you desire when you strike out on your own.

And lastly, many people are loathe to read Help files, often because the software is poorly documented and the Help is not very helpful. Ultra Fractal is quite different in this regard and you will find its extensive Help files to be easy to read and full of information.

All tutorials except Working with animations were written by Janet Parke. Living in Tennessee, USA, Janet is a widely respected fractal artist and also a ballet teacher. She has been working with Ultra Fractal since the first beta releases in 1998. For some examples of her work, visit her online galleries at www.parkenet.org/jp/galleries.html. You will also find a lot of Ultra Fractal resources and information on her web site.

Working with animations

Note: You will only be able to follow the steps in this tutorial if you are using Ultra Fractal Animation Edition. In addition, if you are not already familiar with Ultra Fractal, you should work through the Quick Start, Basic Skills, and Working with layers tutorials first.

One of the major features in Ultra Fractal is the ability to turn any fractal into an animation and make a movie out of it. In this tutorial, we will start with a simple fractal and gradually turn it into a more and more complex animation using the various new animation features. Finally, we will render it as a movie.

First, close any open fractal windows to begin with an empty workspace.

Click **New** on the File menu, and then click **Fractal**. In the Select Fractal Formula browser, select **Phoenix (Julia)** in Standard.ufm and click **Open**.

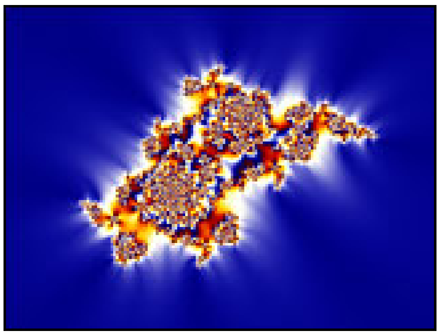
This opens a new fractal window with the Phoenix (Julia) fractal. Let's change the Julia Seed parameter of this fractal formula to make it a little more interesting.

- Click on the complex number below to copy it to the Clipboard.
  -0.41 / -0.53
- Right-click on the **Julia Seed** parameter on the Formula tab and click **Paste Complex Value**. This should fill the input boxes with -0.41 and -0.53, respectively.

Go to the Outside tab, and click the **Browse** button to choose another coloring algorithm. Select **Orbit Traps** in Standard.ucl and click **Open**.

- Change the **Trap Shape** parameter to **pinch**, and set **Trap Mode** to **inverted sum squared** (at the end of the list).

The fractal should now look like this:



Next: Making a zoom movie

Making a zoom movie

Let's animate this fractal to make a simple zoom movie. Make sure the animation bar at the bottom of the main Ultra Fractal window is visible (click Options|Animation Bar if it is not).

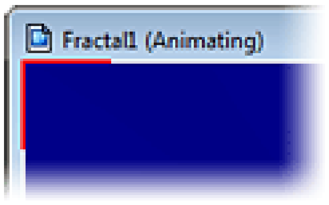
Learn more about...

[Animation bar]

[Animate mode]

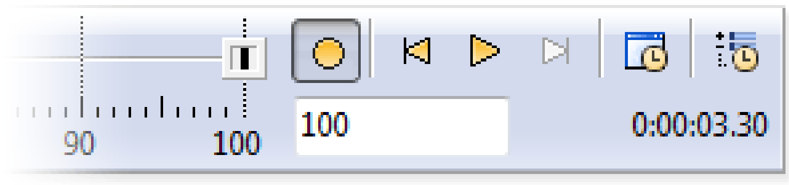
In Ultra Fractal, you create an animation by making changes while **Animate mode** is on.

Click the **Animate** button on the animation bar to turn Animate mode on. The fractal window will now show red marks at the corners and the text *(Animating)* in the title bar.



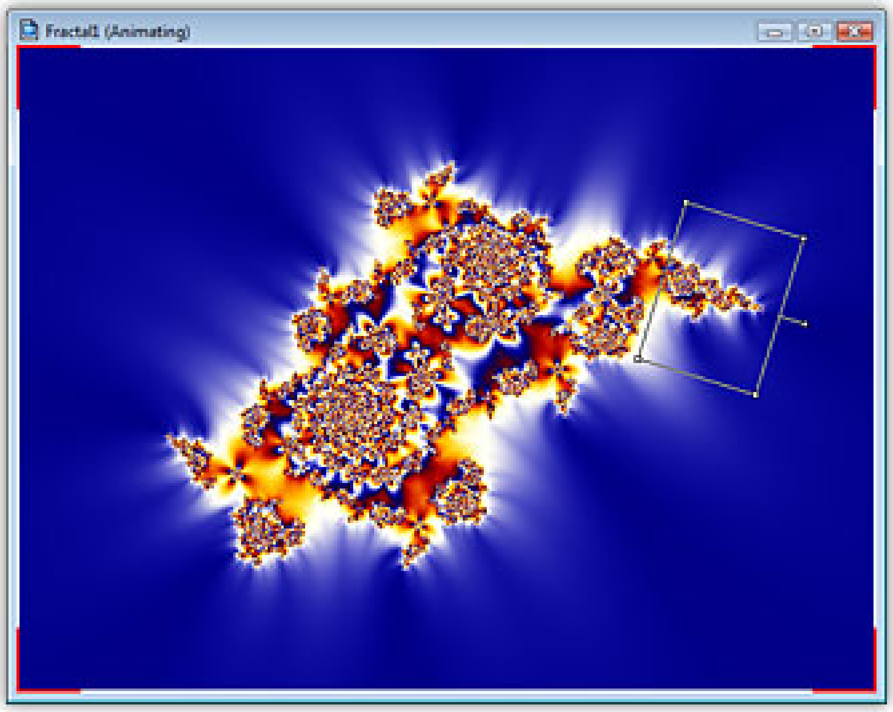
Look at the animation bar and note the **time slider**, going from frame 1 to 100. Every fractal starts with 100 frames by default. The time slider sets the current frame, which is still frame 1 at this point.

- Move the time slider to frame 100, because we are going to animate a zoom from frame 1 up to frame 100.



- Click and drag inside the fractal window to enter [Select mode]. Move and resize the selection box to frame an interesting portion of the fractal. Add some rotation for a better zoom effect.

The exact location does not matter much for this tutorial, but if you want to recreate the final movie, position the selection box like this:



- Right-click in the fractal window and click **Zoom In** to perform the zoom.

We are done recording the first part of our animation, so click the **Animate** button again to turn Animate mode off. You should make it a habit to turn Animate mode off as soon as possible to avoid making unwanted changes.

Next: Playing the movie

Playing the movie

Congratulations! You have just created a zoom movie. Let's have a look at it.

- Move the **time slider** slowly back to frame 1.

Note that the fractal window immediately recalculates as soon as you move the slider to show how the zoom is interpolated from frame 1 to frame 100. Dragging the time slider back and forth is a convenient way of previewing the animation quickly or precisely.

To view a real-time preview of the animation, click the **Play** button on the animation bar. The animation will keep looping until you click somewhere or hit a key.

The level of detail of the preview depends on the speed of your computer — as long as you are using the Guessing drawing method. To make the preview faster, you can reduce the size of the fractal window on the Image tab of the Fractal Properties tool window.

Tip!

Most commands on the animation bar are also available on the Animation menu, with keyboard shortcuts.

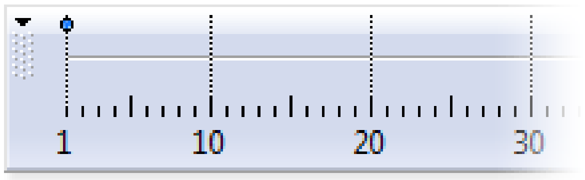
Tip!

The preview is always played with a fixed frame rate. To change the preview frame rate, click Options on the Options menu to open the Options dialog, and click the Fractal tab. The Animation preview speed setting is in the Advanced calculation options area.

See also Calculation details.

Any fractal in Ultra Fractal is potentially an animation. As you have seen, you can turn any still fractal into an animation by enabling Animate mode and changing the fractal in some way.

Look at the time slider. Two blue dots have appeared, one at frame 1, and one at frame 100. If you set the slider to exactly frame 1 or 100, the corresponding dot turns into a yellow marker to show that it is at the current frame.



When you make changes to a fractal while Animate mode is on, Ultra Fractal records **animation keys** at the current frame, and at frame 1 if there are no animation keys yet. The blue dots show at which frames the animation keys are located. When you click on a dot, the time slider jumps to the frame at which the key was recorded.

Learn more about...

Animation keys

Next: Experimenting with Animate mode

Experimenting with Animate mode

The Animate mode toggle controls how Ultra Fractal responds to changes that you make to a fractal. The basic rule is that if Animate mode is on, your changes are only applied to the current frame. Otherwise, your changes are applied to the entire animation.

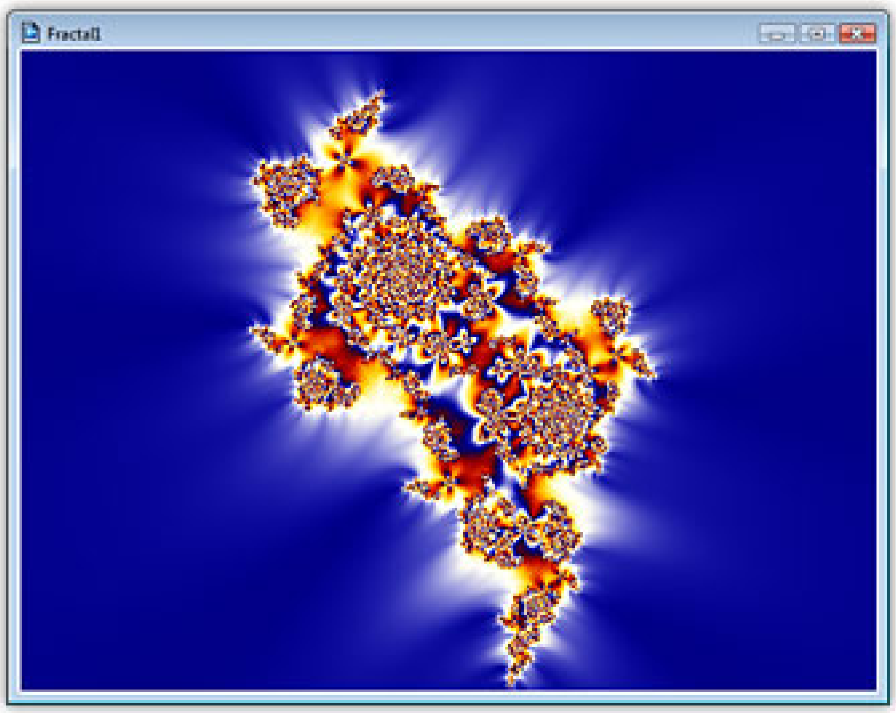
Learn more about...
Animate mode

Let's do some experiments with the rotation of the fractal to get a feel for how this works in practice.

- Move the **time slider** to frame **1** and verify that the **Rotation Angle** setting in the Location tab is set to **0**. Also note the yellow marker before the input box. This shows that this parameter is animated and that there is a key at the current frame.
- Move the **time slider** to frame **100**. The Rotation Angle value will now probably change, depending on how you rotated the selection box when zooming in earlier.
- Set **Animate Mode** to **on**, and change the **Rotation Angle** value to **60**.
- Set **Animate Mode** to **off**, and move the **time slider** to frame **1** again.

Note that at frame 1, the Rotation Angle setting is still 0. That is because Animate mode was on when you changed it at frame 100. When Animate mode is on, changes only apply to the current frame. Let's try changing the rotation with Animate mode off to see how that works.

- Make sure Animate mode is still off and the time slider is at frame 1. Change the **Rotation Angle** setting to **90**, which will rotate the fractal clockwise.



- If you now move the **time slider** (slowly) to frame **100**, you will see that the entire animation has been rotated. That is because Animate mode was off when you changed the rotation. At frame 100, you will find that the Rotation Angle setting is now 150 instead of 60. It has been increased by 90 as well.

You can use this technique with any parameter if you need to adjust the entire animation. For example, you can move the animation, globally change the color density, and so on.

Let's now restore the animation to its previous state, this time by changing the keys at different frames individually.

- Make sure the time slider is still at frame 100 and enable **Animate mode**. Hold down the **Alt** key and click and drag inside the fractal window to rotate the zoomed-in fractal at frame 100 back to its original state.
- With Animate mode still enabled, move the **time slider** to frame **1**. Enter **0** in the **Rotation Angle** input box. Set **Animate mode** to **off** again.

Note that it does not matter what tools you use to make changes — the selection box, Alt-dragging, or manually entering values. They all work together with the current state of the Animate mode toggle.

Also, note that you can easily change the value that was recorded for an animation key simply by going back to the frame where the key is located and adjusting the parameter while Animate mode is enabled.

Next: Extending the animation

Extending the animation

Now that we have created a simple zoom movie and have learned about the Animate mode toggle, it is time to make this animation more interesting.

- Make sure that Animate mode is off. Go to the Outside tab and set the **Threshold** parameter to **0.05**.

The fractal looks much 'thinner' now. Let's animate the fractal from a really thin look to its previous look, so it appears to grow, and then zoom in. First, we have to make the animation a bit longer to insert a new part at the beginning.

Click the **Time Settings** button on the animation bar to open the Time Settings dialog. This dialog enables you to change the length and the frame rate of the animation.

Learn more about...
Time settings

- Enter **200** in the **Frames** input box. In the **Existing keys** area, select the **Keep at last frame** option. Click OK.

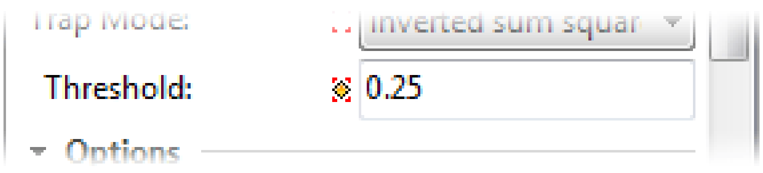
You have now doubled the length of the animation, with the existing zoom movie at the end of the animation. In the animation bar, the keys that were previously located at frame **1** and **100** have now moved to frame **101** and **200**, respectively. We can now insert our 'growing' animation at the beginning.

- Make sure the **time slider** is at frame **1** and set **Animate mode** to **on**.
- Set the **Threshold** parameter to **0.01** so the fractal almost disappears.
- Move the **time slider** to frame **120**.
- Set the **Threshold** parameter to **0.25**. Set **Animate mode** to **off** again.

Learn more about...
Animating parameters

Note that while Animate mode is on, red animation indicators appear before every parameter that can be animated. There are also other possible indicators here:

- A **blue dot** means that the parameter is animated. It has one or more animation keys, so it changes somewhere during the animation.
- A **yellow marker** means that the parameter is animated and has an animation key at the current frame. The parameter changes to a specific value at the current frame.


Trap Mode: inverted sum squar
Threshold: 0.25

- Use the **Play** button or drag the **time slider** back and forth to preview the animation at this point.

Next: Using the Timeline tool window

Using the Timeline tool window

Animating the Threshold parameter is a nice addition, but the animation now falls apart into two separate parts. It would be better to start the zoom earlier, but how do we do this?

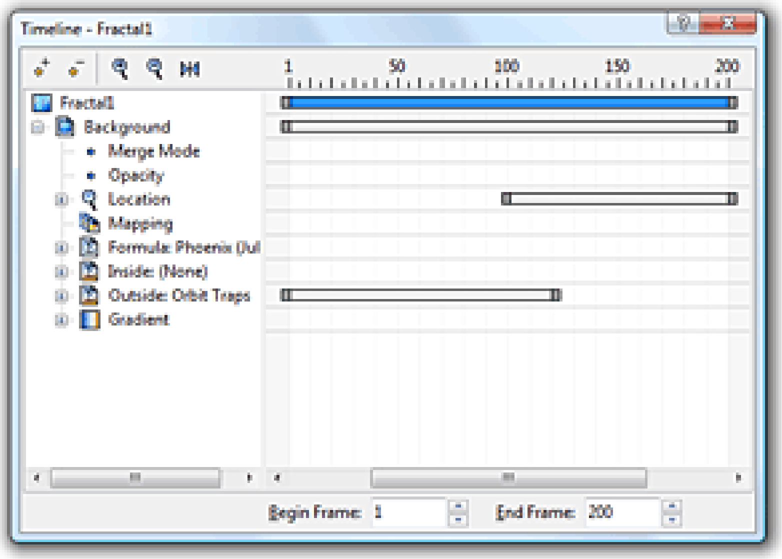
The most powerful way to edit your animations is the Timeline tool window. It shows all parameters that can be animated, together with an overview of the range of frames over which they are animated.

Click the **Timeline** button on the animation bar to open the Timeline tool window.

Learn more about...
The Timeline tool window

Click the **Reset View** button in the toolbar to make sure the complete animation fits in the window.

On the left side, the Timeline tool window shows a tree view of all parameters in the fractal, grouped by layer and category. On the right, the animated range of each category and parameter is shown.



In this case, the animated range for both the entire fractal and the Background layer ranges from frame 1 to 200. The location is animated from frame 101 to frame 200, and the outside coloring algorithm is animated from frame 1 to 120. Click on a category in the tree or on a range bar to see the exact begin and end frames.

- Expand the **Location** and **Outside** categories to see the individual parameters that form the animated range of each category. Note how you can select an animation key to edit it individually.
- Move the mouse cursor over the left-hand end of the range bar of the **Location** category until it changes into a resize cursor. Drag the left end to frame **30**. (You can also enter 30 into the Begin Frame input box at the bottom while the Location category is selected.)

Close the Timeline tool window and preview the animation to see the effect of this change. Note how the fractal already begins zooming at frame 30. This is good, but maybe it would be even better if the zoom would start slower. We can accomplish this by inserting additional animation keys somewhere between frame 30 and frame 200.

- Move the **time slider** to frame **100** and set **Animate mode** to **on**.
- Zoom out the fractal, and position it such that it is only slightly zoomed in and rotated relative to the initial location. (Tip: Use Shift-dragging, Ctrl-dragging, and Alt-dragging to achieve this. See also Normal mode.) Move the time slider back and forth while you are working to judge the smoothness of the animation, but make sure you are only making changes while the slider is at frame 100.
- When you are finished, set **Animate mode** to **off** again.

If you have not saved the fractal already, click **Save Parameters** on the **File** menu and save the fractal as **Animated Phoenix** in **tutorials.upr**.

Next: Adding gradient animation

Adding gradient animation

Like any parameter in Ultra Fractal, you can also animate the gradient. Let's add some gradient animation to let the fractal fade in from a dark grayscale gradient to its current bright colors. First, we create the desired new gradient.

Click **Gradient** on the Fractal menu to open the gradient editor.

Learn more about...
Animating gradients

The gradient editor works together with the Animate mode toggle like all other tools in Ultra Fractal.

Set **Animate mode** to **on**, and move the **time slider** to frame **20**.

Click **Adjust Colors** on the Gradient menu.

- On the HSL tab, set **Saturation** to **-100**, which removes all color from the gradient. Set **Luminance** to **-40** to make the gradient darker. Click OK.

Drag the time slider back and forth to preview the result and observe that we have achieved just the opposite what we wanted. Let's fix that.

- Move the **time slider** to frame **1**. Make sure the gradient editor is active and Animate mode is still on.

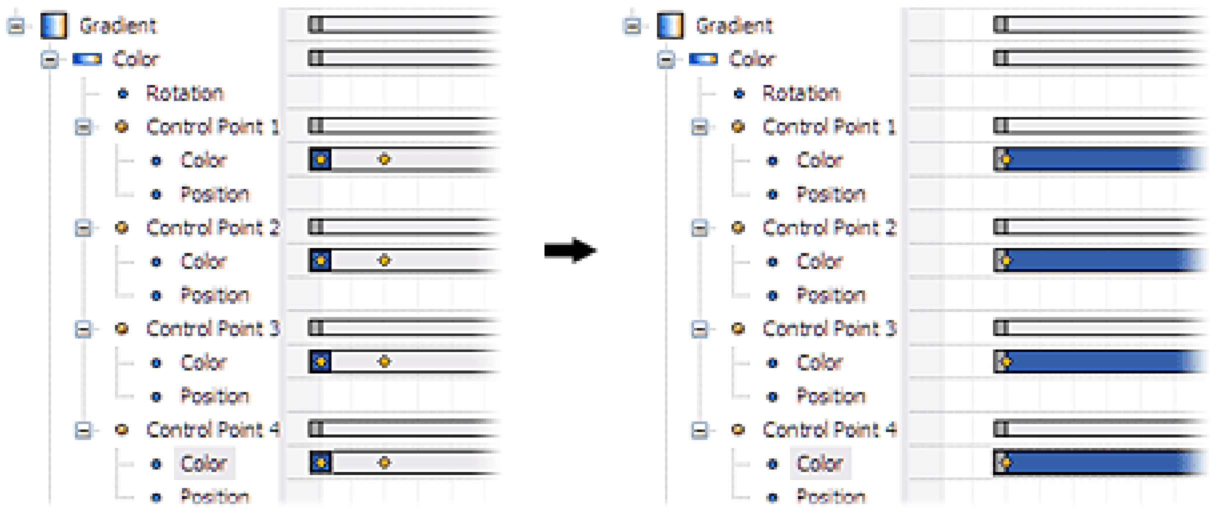
Click **Copy** on the Edit menu to copy the gradient to the Clipboard, as it is at the current frame. It is important that Animate mode is on while you copy the gradient, otherwise you would have copied all animation keys as well. We just want the gradient as it is at frame 1.

Move the **time slider** to frame **140**, and click **Paste** on the Edit menu. Again, Animate mode must be on, because we want the paste operation to affect the current frame only. Otherwise, the gradient for the entire animation would have been replaced by the pasted gradient.

Turn off Animate mode and drag the time slider back and forth again to view the effect. This is almost what we want, except that the dark gradient should be at the start of the animation. We can fix that with the Timeline tool window.

Click **Timeline** on the animation bar to open the Timeline tool window. Scroll down and expand the Gradient category, its Color category, and all its four control points.

- Click on the **leftmost** animation key of the **Color** parameter of **Control Point 1** to select it. Hold down the **Ctrl** key and click on the **leftmost** keys of **Control Point 2** to **4** as well to add them to the selection. The timeline should look like the left picture below.



Click the **Delete Selection** button to delete these four animation keys. The timeline should now look like the right picture above.

- Finally, click on the range bar for the **Gradient** category to select it, and then **drag** its **left edge** to frame **1**. The Gradient bar should now range from frame **1** to frame **140**.

Drag the time slider back and forth to view the effect in the fractal window. This is what we wanted, but it turns out that frame 140 is too late — almost the entire animation is now quite dark and gray. Let's fix this as well.

- In the Timeline tool window, drag the **right edge** of the range bar for the **Gradient** category to frame **60**.

Again, examine the effect with the time slider and experiment until you are satisfied.

Next: Adding a new layer

Adding a new layer

As a final step, we are going to extend the animation once again, this time at the end, to add a final fade to a new layer. First, we will create a new layer.

Click the **Add** button on the Layers tab of the Fractal Properties tool window to duplicate the Background layer.

Learn more about...
[Animating layers]

The new layer is a complete duplicate of the background layer, with all its animation keys. Because the new layer is going to be hidden most of the time, a static layer is good enough and it saves valuable calculation time. Therefore, it would be better to remove the animation keys.

- Make sure the **time slider** is at frame **200** and open the **Timeline** tool window.

Click on the **range bar** for the **Layer 1** category, and click the **Delete Selection** button to delete all animation keys for Layer 1. You can close the Timeline tool window now.

Layer 1 is no longer animated now. Because the time slider was at frame 200, all previously animated parameters have been set to the value they had at that frame.

Click the **Browse** button on the Outside tab of the Layer Properties tool window to choose a new coloring algorithm. Select **Triangle Inequality Average** in Standard.ucl and click **Open**.

- On the Layers tab of the Fractal Properties tool window, select **Hard Light** as the merge mode for **Layer 1**.

Now, the idea is to extend the animation to 250 frames, and let this new layer fade in, beginning at frame 200.

Click the **Time Settings** button on the animation bar to open the Time Settings dialog. Set **Frames** to **250**, and select **Keep at first frame**. Click OK.

- First, set the **opacity slider** for **Layer 1** to **0%**, as this is the initial value.

If we would enable Animate mode and change the opacity at frame 250, the opacity would be animated from frame 1 to frame 250. We could then rescale that to the range 200-250 using the Timeline tool window, but there is an easier way.

Ensure the **time slider** is at frame **200**. Right-click on the **opacity slider** and click **Insert Key** on the menu that pops up. This inserts a new key for the opacity parameter at frame 200, with the current value (0%).

Learn more about...
Editing animations

- Move the **time slider** to frame **250** and set **Animate mode** to **on**. Change the **opacity slider** to **100%**. Set **Animate mode** to **off** again.

Examine the results by dragging the time slider. Note that the opacity animates from 0% at frame 200 to 100% at frame 250, just as we intended.

Next: Rendering the animation

Rendering the animation

Now that the animation is finished, you can preview it with the Play button. However, you will probably want to view it as an AVI file with much higher quality. For that, you have to render the animation first.

Click **Render to Disk** on the Fractal menu to open the Render to Disk dialog.

Learn more about...
Rendering animations

The Render to Disk dialog looks a bit different than you might be used to, because it contains options specific to animations that are normally hidden.

- From the File Format drop-down list, choose AVI movie. The Export Options dialog will pop up so you can set additional options for the AVI format.
- The list of compressors depends on what is installed on your computer. Some are better suited to fractal movies than others — it is best to experiment. For now, choose **Cinepak Codec by Radius**, set **Quality** to **100%** and **Key frame** to once in every **15** frames. Click OK.
- If you prefer a different destination file than what Ultra Fractal automatically suggests, enter it into the **Destination File** input box, or click its Browse button.
- In the Animation area, set **Frame Range** to **Entire Animation**. You can use this option to render only a selected range of frames, or just the current frame.
- Set **Motion Blur** to **Normal**. This applies a motion blur effect to animated zooms, which makes them smoother and more natural-looking.

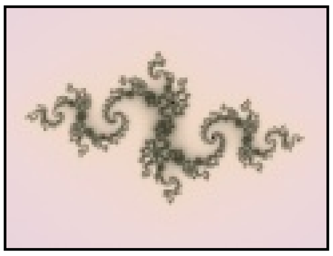
The other options are the same as when rendering still images. You can set them as you prefer, but make sure to enable anti-aliasing for the best results. If you check the **Open when finished** option, Ultra Fractal will automatically open the resulting AVI file in the default player, for example Windows Media Player.

- Click **OK** to start rendering. The Render to Disk tool window opens to display the rendering progress. Since animations are much more complex to calculate than still fractals, the render can easily take an hour or more, depending on the width and height that you selected for the final movie.

Congratulations! You have made it to the end of this tutorial. We have used most of the animation features of Ultra Fractal to make this animation, and you probably have many ideas for your own animations now.

For more information, please refer to the Animation chapter of the help file. Enjoy!

What are fractals?



Ultra Fractal creates images of fractals. Fractal images are created by repeatedly calculating a fractal formula. Although these formulas are purely mathematical, the resulting pictures are often very beautiful and complex.

Ultra Fractal goes a long way toward hiding the mathematical stuff. Instead, you focus on the fractals themselves, the way they are combined, and how they are colored. This enables you to turn your fractals into true works of art.

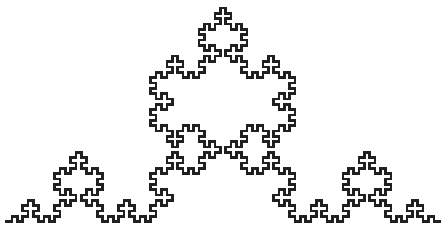
This chapter will explain the basics of fractals, and why they are so interesting.

Next: Self-similarity

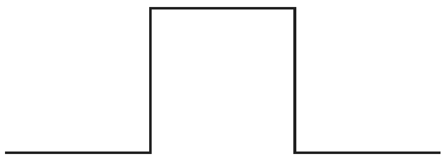
See Also
Tutorials
Workspace
Fractal windows

Self-similarity

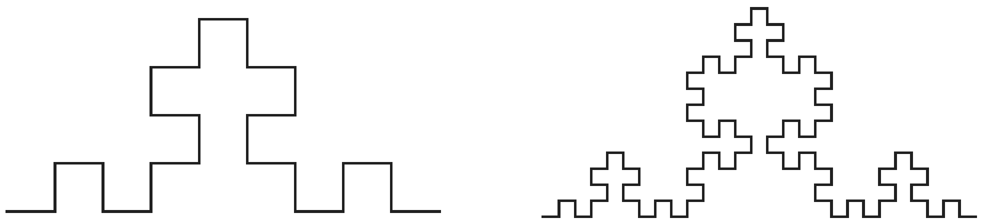
There are various definitions of what a fractal is. One of the easiest is that a fractal is usually self-similar. That means that it repeats itself. For an example, look at the following fractal.



This is a Van Koch fractal. It is based on a very simple shape.



To create the fractal, the flat lines are replaced by the entire shape itself.



This process is repeated again and again to create an infinitely complicated fractal. Still, every part of the fractal contains the original shape. We say that the fractal is self-similar. Most fractals in Ultra Fractal are calculated differently, but the principle of self-similarity still applies.

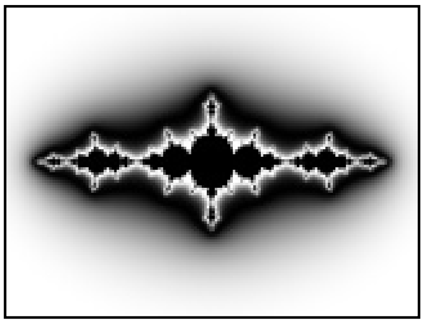
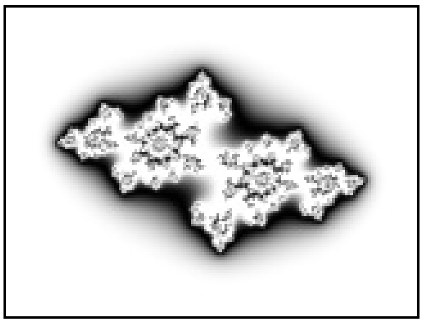
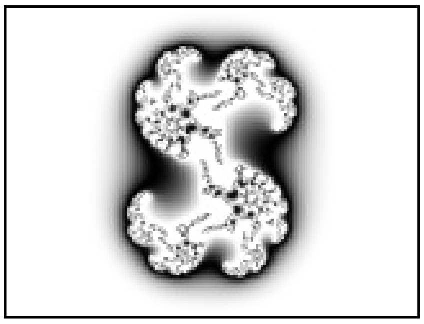
This is also the reason that it is so popular to zoom into fractals: there are always more details to be discovered, no matter how far you zoom.

Next: Julia sets

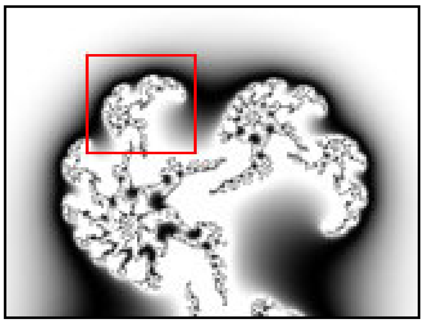
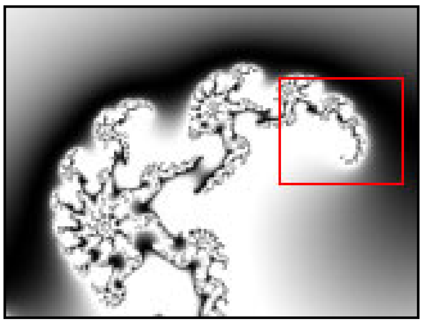
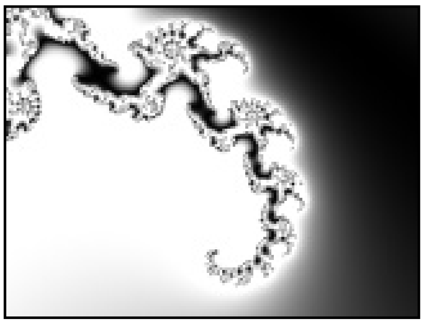
See Also
What are fractals?

Julia sets

One of the most basic fractal types is the family of Julia sets, discovered by the French mathematician Gaston Julia during the first World War. Julia sets are created by a simple formula with one complex parameter called **c** or **seed**. This parameter can be varied to create many variations. Here are a few examples.

Julia sets are also self-similar, as illustrated by the following zooms into the last image above. The first zoomed image shows the top of the original. Further zooms are illustrated by the small red rectangles in the images.

The same spiral-like shape is repeated over and over again.

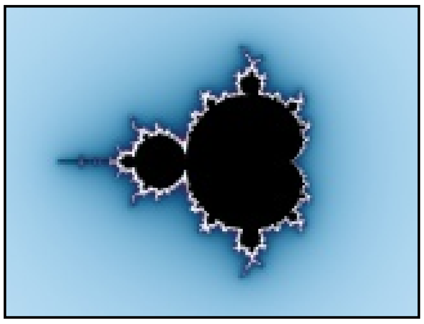
It can be difficult to find good values of the **c** parameter. Fortunately, the Mandelbrot set, which is discussed next, can help you with that.

Next: The Mandelbrot set

See Also
What are fractals?
Julia

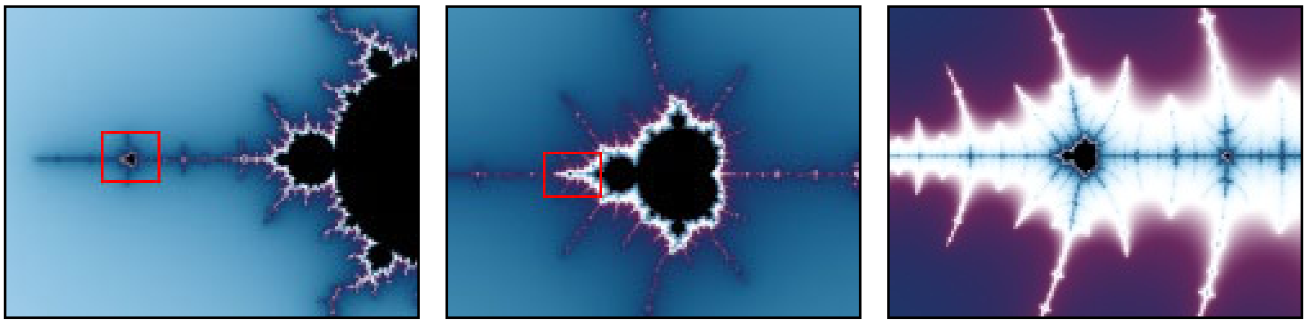
The Mandelbrot set

The Mandelbrot set, discovered in 1980 by Benoit Mandelbrot, is probably the most famous fractal. Like Julia sets, it is generated by a very simple formula, but it is incredibly complex.



The Mandelbrot set is loosely self-similar: parts of the original fractal appear again when zooming in, but often deformed and with different ornaments. This is what makes it so rewarding to zoom into this fractal: you never know what you will see next.

This is illustrated by the following short zoom, starting at the very left of the Mandelbrot set shown above. As you zoom in, you see copies of the original Mandelbrot set, but with different surroundings.



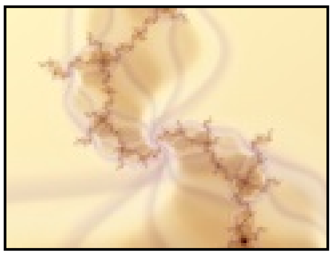
Another interesting aspect of the Mandelbrot set is that it is actually a map of all Julia sets. Each point corresponds to a Julia set. Points inside the Mandelbrot set (here shown as black) are **connected** Julia sets; points outside the Mandelbrot set tend to give more disorganized Julia sets.

With the switch feature in Ultra Fractal, you can easily pick a point of a Mandelbrot fractal to see the corresponding Julia set. This is the best way to discover interesting Julia sets.

Next: Fractals today

See Also
What are fractals?
Mandelbrot
Julia sets

Fractals today



While Ultra Fractal is well suited to exploring the classic fractal types discussed so far, it can do much more than that. There are many more fractal types to choose from, and you can even write your own fractal formulas (or use formulas written by other people). Most fractal types are variations on the Mandelbrot and Julia sets.

Each fractal type can be combined with various coloring algorithms, each capable of coloring the fractal in a different way. Transformations can be added to distort the shape of the fractal. Colors are easy to change and tweak with the gradient editor. On top of that, you can use multiple layers to combine different fractals or different coloring methods to form the final image.

Because of these changes, fractals have grown from a mathematical curiosity to a respected form of art. There are fractal exhibitions in museums and galleries all over the world. There is a large number of online galleries on the web, where you can purchase prints and posters from various fractal artists.

Next: Where to start

See Also
What are fractals?
Fractal windows

Where to start

Now that you know a bit more about fractals, you are probably wondering how to produce these with Ultra Fractal. By default, Ultra Fractal opens with a standard Mandelbrot fractal, so the easiest way is to take this fractal and start zooming.

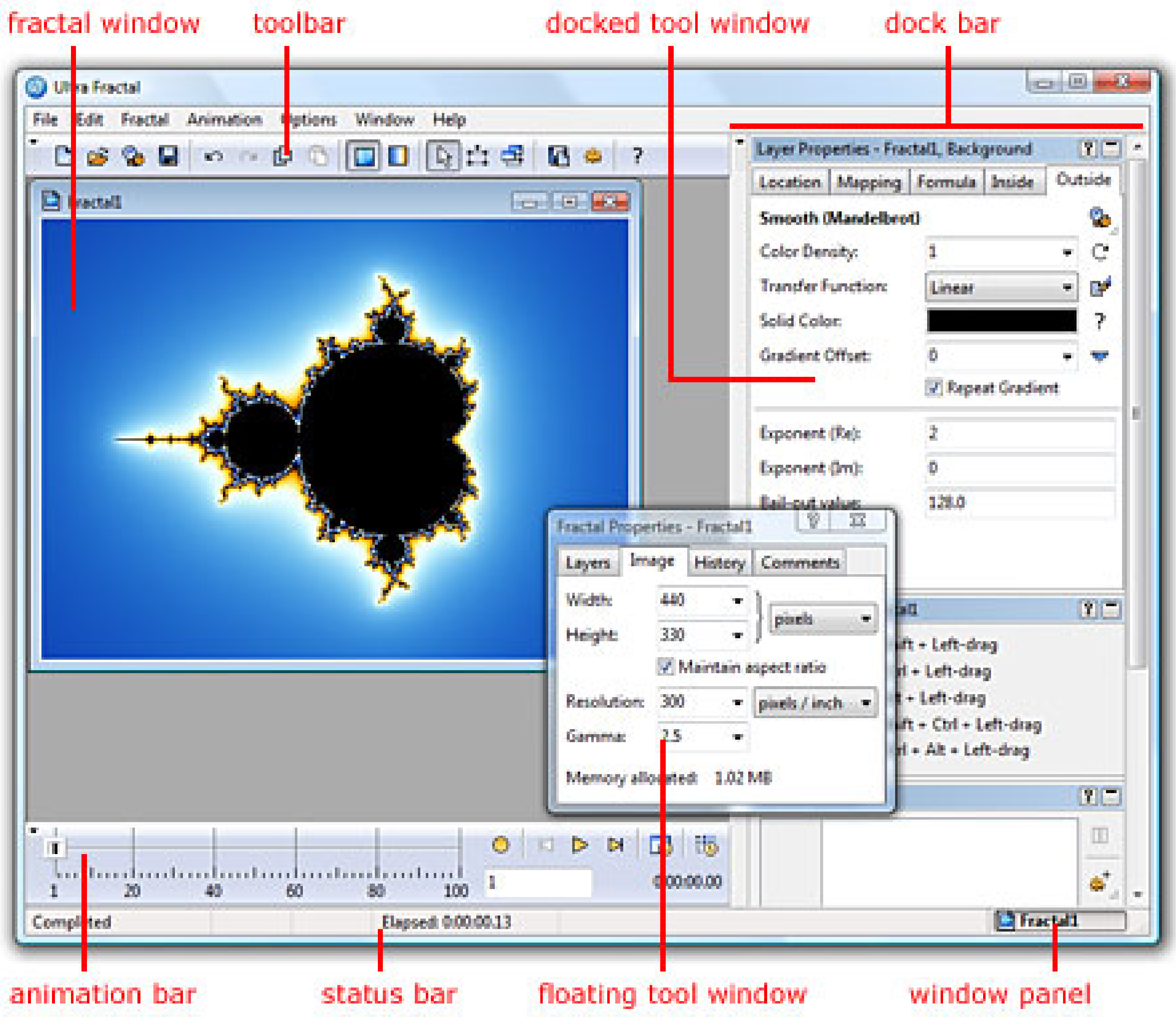
Click and drag inside the fractal window to open the selection box. Drag and resize it, and then double-click inside the selection box to zoom in.

Ultra Fractal has many more possibilities, but it is a good idea to start with simple zooming to get a feeling for what fractals are and how Ultra Fractal works. Also, be sure to work through the tutorials to learn more.

See also
Tutorials
Workspace

Workspace overview

Ultra Fractal has one main application window that contains all open documents, such as fractals, gradients, and formula files. Secondary windows, called **tool windows**, edit the properties of fractals and provide access to other functionality.


fractal window
toolbar
docked tool window
dock bar
Ultra Fractal
File Edit Fractal Animation Options Window Help
Fractal1
Layer Properties - Fractal1, Background
Location Mapping Formula Inside Outside
Fractal Properties - Fractal1
Layers Image History Comments
Completed
Fractal1
animation bar
status bar
floating tool window
window panel

Smooth (Mandelbrot)
Color Density: 1
Transfer Function: Linear
Solid Color:
Gradient Offset: 0
Repeat Gradient
Exponent (Re): 2
Exponent (Im): 0
Bail-out value: 128.0

Width: 440
Height: 330
pixels
Maintain aspect ratio
Resolution: 300 pixels / inch
Gamma: 2.5
Memory allocated: 1.02 MB

Elapsed: 0:00:00.13

The workspace contains the following elements:

- **Document windows** contain the documents that you work on, such as fractals and gradients. In the screen shot above, the fractal window is an example of a document window.
- The buttons on the **toolbar** provide access to frequently-used commands. Usually, these commands can also be accessed through the **pull-down menu** directly above the toolbar. The toolbar can be hidden and restored by clicking Toolbar on the Options menu.
- The **dock bar** is a place to store tool windows, to keep them from using too much screen space. Tool windows can be dragged into and out of the dock bar at any time. Tool windows in the dock bar are called **docked tool windows**. They can also be collapsed so you only see the title bar. To hide and restore all tool windows and the dock bar, click Tool Windows on the Options menu or press F12.
- **Floating tool windows** are tool windows that float freely over the screen, instead of being in the dock bar. This is useful if you use a tool window a lot, or to place some tool windows on a secondary monitor if you have one. The Timeline tool window works better when floating than when docked.
- The **animation bar** contains animation controls for the active fractal window. See Animation bar. *(Ultra Fractal Animation Edition only)*

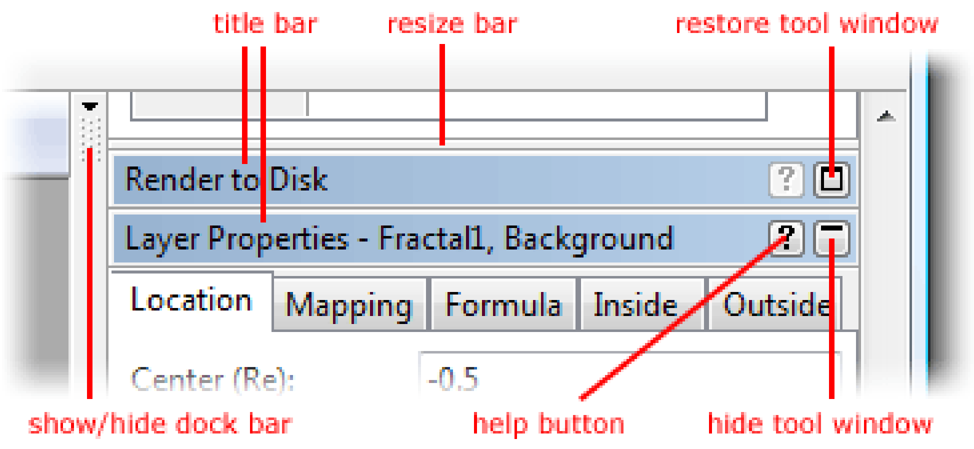
- The **status bar** provides additional information about the active document window, such as the elapsed calculation time for a fractal. To hide and restore the status bar, click Status Bar on the Options menu.
- The **window panel** is an area in the status bar that lists all open document windows, much like the Windows task bar. To bring a document window to the foreground, click on its button in the window panel. To hide and restore the window panel, click Window Panel on the Options menu.

Next: Working with tool windows

See Also
Tutorials
Fractal windows
Gradients

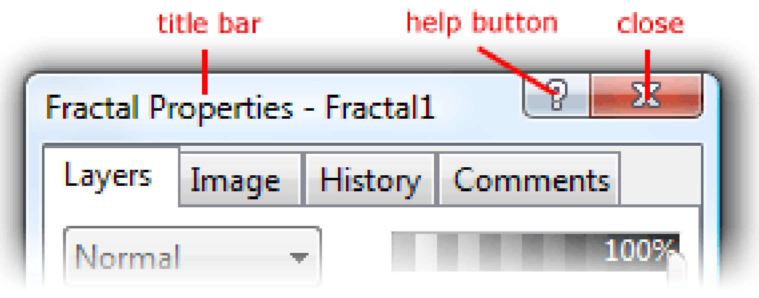
Working with tool windows

Most of the functionality in Ultra Fractal is accessed through the various tool windows. All tool windows are resizeable. They can be put in the dock bar to save screen space, or they can float on the screen for easy access.



With docked tool windows:

- Use the **title bar** to drag a tool window out of the dock bar to change it into a floating tool window.
- Drag the **resize bar** up and down to change the height of a tool window.
- Click the **hide tool window** button to temporarily hide a tool window, to make more space for the other windows. The tool window collapses to show only the title bar.
- Click the **restore tool window** button to restore a hidden tool window.
- Click the **help button** and then click a control inside the tool window to get context-sensitive help.
- Click the **show/hide dock bar** button to hide the entire dock bar temporarily. Drag the vertical bar to the left and right to change the width of the dock bar.



With floating tool windows:

- Use the **title bar** to drag the tool window around. Drop it on the dock bar to change it into a docked tool window.
- Click the **help button** and then click a control inside the tool window to get context-sensitive help.
- Click the **close button** to hide the tool window. To show it again, click Tool Windows on the Window menu, and then click the name of the tool window.
- Right-click on the title bar and click Decrease Opacity to make the tool window transparent, so you can see the windows beneath it.

In general:

- Click Tool Windows on the Window menu for a menu listing all tool windows. Click the name of a tool window to show and hide it. Right-clicking the dock bar will also show this menu.
- Click Tool Windows on the Options menu or press F12 to hide and restore all tool windows.

Next: Tool windows overview

See Also
Workspace overview
General keyboard shortcuts

Tool windows overview

There are two categories of tool windows in Ultra Fractal. Most tool windows work with the active fractal document, but there are also stand-alone tool windows that work indepently.

Tool windows that work with the active fractal document:

- The Layer Properties tool window edits the active layer of the fractal. This is where you do most of the work, for example changing fractal types or experimenting with parameters.
- The Fractal Properties tool window edits the properties that apply to the entire fractal, such as the image size and the layers that it contains.
- The Fractal Mode tool window controls what happens and provides feedback when you click and drag inside the fractal window.
- The Timeline tool window shows all parameters in the active fractal window, grouped by category, and enables you to edit their animation properties. *(Ultra Fractal Animation Edition only.)*
- The Statistics tool window shows additional information about the fractal and the calculation process.
- The Color Cycling tool window animates the colors in the fractal.

Stand-alone tool windows:

- The Network tool window manages the computers that are connected to Ultra Fractal for distributed fractal calculations. *(Ultra Fractal Animation Edition only.)*
- The Render to Disk tool window manages background calculations of disk render jobs.
- The Compiler Messages tool window collects warnings and errors generated by the formula compiler when you select or reload a formula.

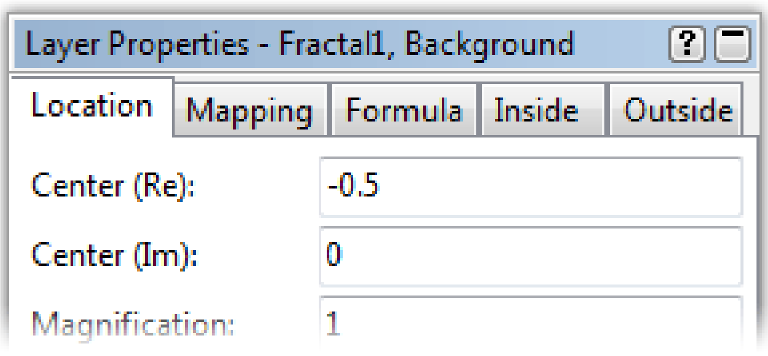
Next: Options dialog

See Also
Working with tool windows
Workspace overview

Layer Properties tool window

The Layer Properties tool window edits the selected layers in the active fractal window. Layers are selected in the Layers tab of the Fractal Properties tool window. The title bar shows which layers are currently being edited.

To open the Layer Properties tool window if it is hidden, click Tool Windows on the Window menu, and then click Layer Properties.



Center (Re): -0.5
Center (Im): 0
Magnification: 1

The Layer Properties tool window contains five tabs:

- The **Location** tab specifies the coordinates of the layer. The coordinates define which portion of the fractal is visible. The coordinates are shown in two different forms: as the center coordinate with magnification, and as the coordinates of the corners of the layer. Although it is possible to enter coordinates directly here, you will usually use the zooming, panning and rotating capabilities of the fractal window instead. The Copy and Paste buttons are useful for copying locations from one layer or fractal to another. The Reset button resets the location to the default for the currently selected fractal formula.
- The **Mapping** tab contains a list of geometric transformations that are applied to the layer. These are used to transform the shape of a fractal.
- The **Formula** tab specifies the fractal formula (fractal type) that is used by the layer. The fractal formula defines the shape of the fractal.
- The **Inside** and **Outside** tabs specify how the data from the fractal calculations is interpreted to obtain the final coloring of the layer. By selecting different coloring algorithms here and adjusting the parameters, many different images can be created with the same fractal formula. See also Inside and outside.

To understand how the controls on the tabs work together, it helps to remember that in a way, the calculation "flows" through the tabs from left to right, starting with the location, and ending with the coloring algorithms.

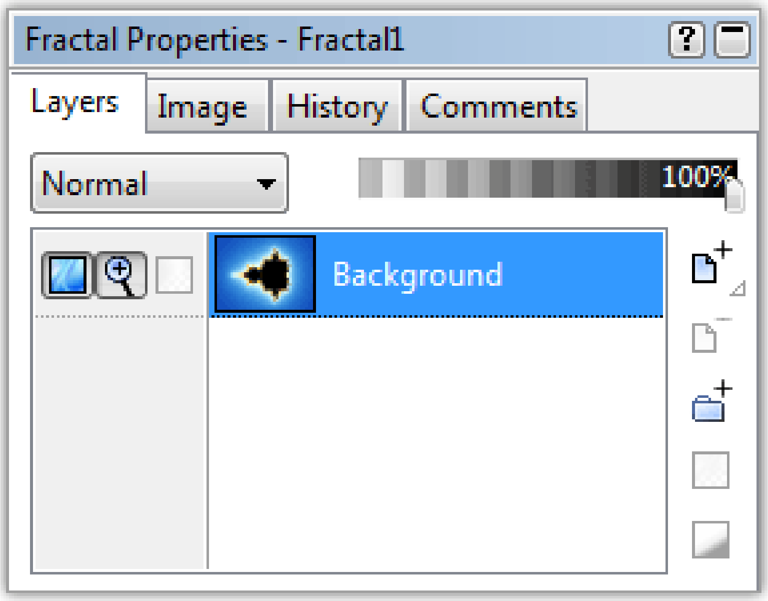
For more information on a specific control, click the help button in the title bar of the tool window, and then click the control, or move the mouse over the control while the Fractal Mode tool window is open.

See Also
Keyboard shortcuts for the Layer Properties tool window
Tool windows

Fractal Properties tool window

The Fractal Properties tool window edits global properties of the active fractal window, such as the size of the image, the list of layers, history, and comments.

To open the Fractal Properties tool window if it is hidden, click Tool Windows on the Window menu, and then click Fractal Properties.



The Fractal Properties tool window contains four tabs:

- The **Layers** tab manages the layers in the fractal. Here, you can control how the layers are merged together to produce the final image. The properties of individual layers are edited by the Layer Properties tool window.
- The **Image** tab specifies the dimensions of the fractal window. It is important to remember that the fractal window is really only a preview. Use the Render to Disk feature to create final images at any size, independent of the size of the fractal window. See also Resolution.
- The **History** tab shows the previous states of the fractal. It allows you to go back any number of steps in time, without recalculations. See Fractal history list.
- The **Comments** tab provides a space for you to type comments on the fractal, such as copyright information. It also contains the credits list that automatically tracks the artists that have worked on the fractal, so everyone receives proper credit.

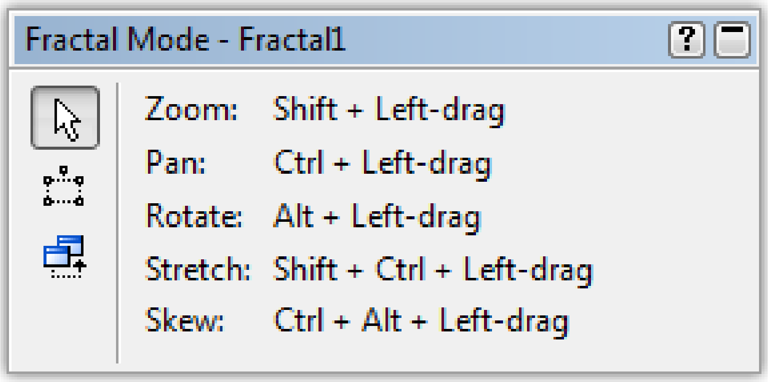
For more information on a specific control, click the help button in the title bar of the tool window, and then click the control, or move the mouse over the control while the Fractal Mode tool window is open.

See Also
What are fractals?
Keyboard shortcuts for the Fractal Properties tool window
Tool windows

Fractal Mode tool window

The Fractal Mode tool window controls how mouse operations are interpreted by the active fractal window, shows context-sensitive help, and it provides live previews for the Explore and Eyedropper features.

To open the Fractal Mode tool window if it is hidden, click Tool Windows on the Window menu, and then click Fractal Mode. The Fractal Mode tool window will open automatically when it is needed to show previews.


Fractal Mode - Fractal1

Zoom:	Shift + Left-drag
Pan:	Ctrl + Left-drag
Rotate:	Alt + Left-drag
Stretch:	Shift + Ctrl + Left-drag
Skew:	Ctrl + Alt + Left-drag

Use the buttons on the left to select the active mouse mode for fractal windows:

- In **Normal mode**, click and drag while holding down the Shift, Ctrl, or Alt keys to zoom, pan, rotate, skew, and stretch the fractal. Clicking and dragging without holding down a shift key enters Select mode by default. Double-click to zoom in twice. All mouse bindings are customizable in the Mouse tab of the Options dialog. The Fractal Mode tool window shows the current bindings.
- In **Select mode**, a selection box is used to zoom. The area inside the box is expanded to fill the entire fractal window when you zoom in. The Fractal Mode tool window shows a preview of the new fractal, and contains additional options.
- **Switch mode** is used to switch from Mandelbrot-like fractals to their Julia counterparts. This is possible because the Mandelbrot set is actually a map of Julia sets. Move the mouse cursor over the fractal and the Fractal Mode tool window will show a preview of the Julia set that corresponds to the point under the cursor. Click to open a new fractal with this Julia set.

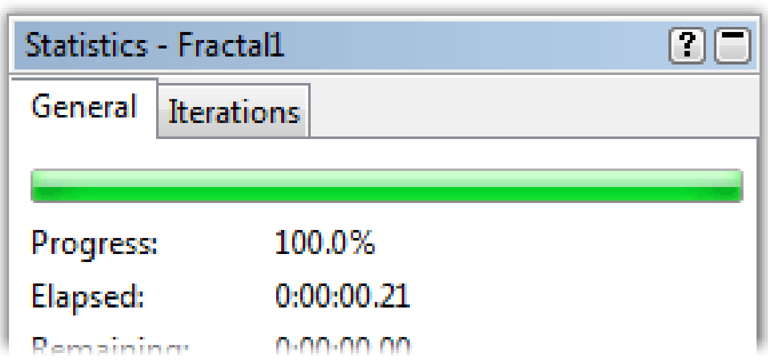
If the Fractal Mode tool window is open and Normal mode is selected, it also shows context-sensitive help for the control that is currently under the mouse cursor. This also works for formula parameters. See Getting help.

For more information on a specific control, click the help button in the title bar of the tool window, and then click the control.

See Also
Normal mode
Select mode
Switch mode
Tool windows

Statistics tool window

The Statistics tool window shows additional information on the active fractal window. To open the Statistics tool window if it is hidden, click Tool Windows on the Window menu, and then click Statistics.



Progress: 100.0%
Elapsed: 0:00:00.21

The **General** tab shows the calculation progress of the fractal at the top. At the bottom, several statistics on the active layer are shown, such as the precision of the calculations used, the percentage of pixels that were actually calculated (not guessed), and the iteration limits of the pixels calculated so far.

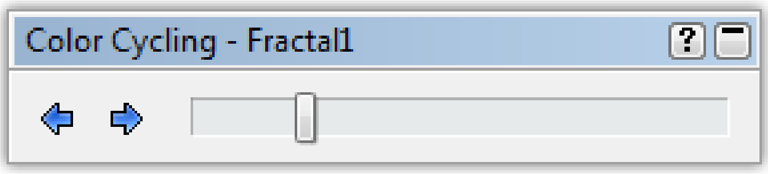
The **Iterations** tab shows a histogram with detailed information on how the iterations values are distributed. You can use this information to estimate a good value for the Maximum Iterations setting in the Formula tab of the Layer Properties tool.

For more information on a specific control, click the help button in the title bar of the tool window, and then click the control, or move the mouse over the control while the Fractal Mode tool window is open.

See Also
Arbitrary precision
Maximum iterations
Tool windows

Color Cycling tool window

The Color Cycling tool window rotates the colors of the layers in the active fractal. To open the Color Cycling tool window if it is hidden, click Tool Windows on the Window menu, and then click Color Cycling.



Click on one of the buttons to start cycling the colors. Move the slider to change the cycling speed.

What color cycling does is repeatedly moving the Rotation slider of the gradients of the editable layers in the fractal. This rotates the gradients in the layers, reproducing the "palette animation" effect that is well-known from older 256-color fractal programs.

Color cycling is also possible without the Color Cycling tool window. Right-click inside the fractal window to open a pop-up menu, click the Gradient submenu and then click Cycle Colors Forward or Cycle Colors Backward. These commands are available in Full-screen mode as well.

For more information on a specific control, click the help button in the title bar of the tool window, and then click the control, or move the mouse over the control while the Fractal Mode tool window is open.

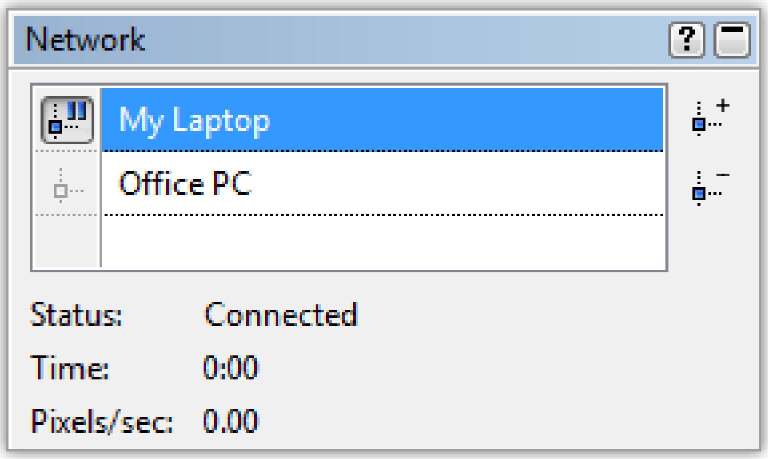
See Also
Gradients
Tool windows

Network tool window

Note: You need Ultra Fractal Animation Edition for network calculations.

The Network tool window manages connections to other computers on the network. Ultra Fractal can then use these computers to distribute fractal calculations, so they are performed much faster.

To open the Network tool window if it is hidden, click Tool Windows on the Window menu, and then click Network.



Status: Connected
Time: 0:00
Pixels/sec: 0.00

With the list of connections in the tool window, you can add, rename, edit, and delete connections. Connections can also be enabled and disabled. By clicking on a connection, you can see its status, how long it has been connected, and how many pixels per second are calculated by the connected computer on average.

It is important to understand that the network tool window shows and edits the connections, but it does not "own" them. So, even when you hide or close the tool window, the connected computers will still continue to be used.

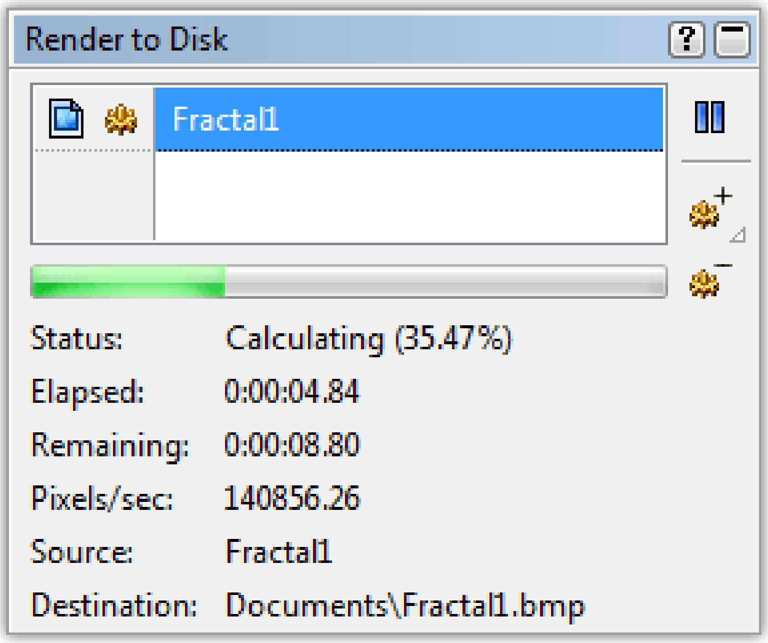
For more information on a specific control, click the help button in the title bar of the tool window, and then click the control, or move the mouse over the control while the Fractal Mode tool window is open.

See Also
Network calculations
Connections
Tool windows

Render to Disk tool window

The Render to Disk tool window manages render jobs. Fractals and animations can be rendered to disk to create high-resolution images and fractal movies with better quality than is possible in the fractal window. Each render command creates a render job that is subsequently performed in the background. The Render to Disk tool window shows the list of render jobs that are still to be finished.

To open the Render to Disk tool window if it is hidden, click Tool Windows on the Window menu, and then click Render to Disk.



Status:	Calculating (35.47%)
Elapsed:	0:00:04.84
Remaining:	0:00:08.80
Pixels/sec:	140856.26
Source:	Fractal1
Destination:	Documents\Fractal1.bmp

Render jobs are calculated from the top of the list to the bottom, with new jobs added at the bottom. You can add, delete, pause, and resume render jobs. Multiple jobs can be calculated simultaneously.

It is important to understand that the Render to Disk tool window shows and edits the render jobs, but it does not "own" them. So, even when you hide or close the tool window, the jobs will still continue to be calculated normally.

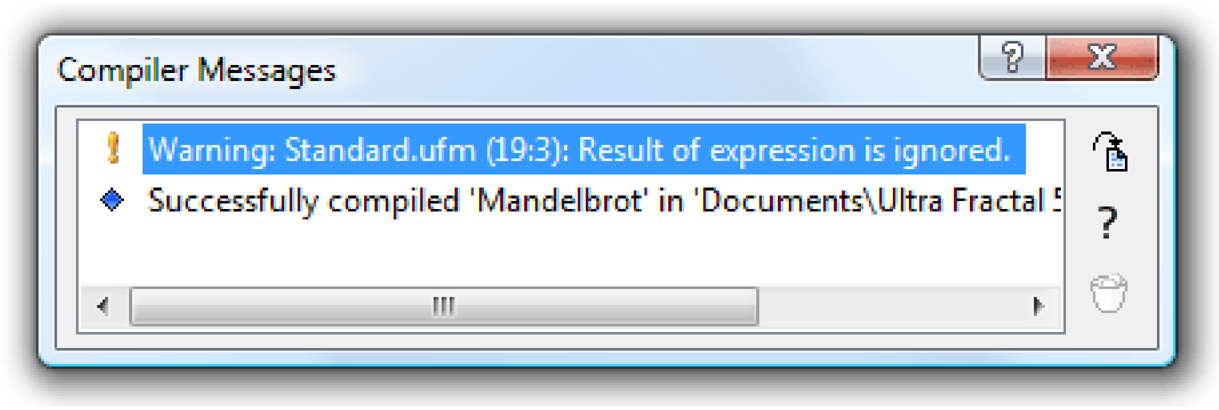
For more information on a specific control, click the help button in the title bar of the tool window, and then click the control, or move the mouse over the control while the Fractal Mode tool window is open.

See Also
Exporting and rendering
Render jobs
Tool windows

Compiler Messages tool window

The Compiler Messages tool window collects warning and errors generated by the compiler when you select or reload a formula. These messages are intended for formula authors. If you encounter errors in a formula that you did not write yourself, it is best to contact the author of the formula.

To open the Compiler Messages tool window if it is hidden, click Tool Windows on the Window menu, and then click Compiler Messages.



Warning: Standard.ufm (19:3): Result of expression is ignored.
Successfully compiled 'Mandelbrot' in 'Documents\Ultra Fractal 5

The tool window attaches itself to the most recently compiled formula. It also shows any run-time messages generated by the attached formula when it is used for fractal calculations. Run-time messages can be used for debugging purposes.

Double-click a message to open the line of code that corresponds to the message in the formula editor, so you can inspect the code and correct the error. Click the Help on Error button to get help about the selected message.

For more information on a specific control, click the help button in the title bar of the tool window, and then click the control, or move the mouse over the control while the Fractal Mode tool window is open.

See Also
Writing formulas
Debugging
Tool windows

Options dialog

The Options dialog provides a single place to customize Ultra Fractal according to your personal preferences.

Click **Options** on the Options menu to open the Options dialog.

The options are grouped by several tabs:

- The **Mouse** tab enables you to customize the mouse actions for the fractal window to zoom, pan, rotate, and so on. It also contains options for double-clicking and zooming, and for the Switch feature.
- The **Fractal** tab contains options for fractal windows. See also Calculation details and Playing animations.
- The **Defaults** tab specifies the default settings for fractal windows. See also Default fractal.
- The **Gradient** tab sets an optional default gradient.
- The **Editor** tab contains options for formula editors.
- The **Syntax** tab enables you to customize the syntax highlighting colors for various language elements in the formula editor.
- The **Browser** tab contains options for browsers, such as template parameter sets for previews and thumbnail cache settings.
- The **Environment** tab contains general workspace options.
- The **Folders** tab enables you to change the location of the various document folders used by Ultra Fractal.

To get help on individual settings, click the ? button in the title bar of the Options dialog, and then click the setting that you want help for.

The Options menu also provides commands to show and hide various user interface elements, and to update your collection of public formulas.

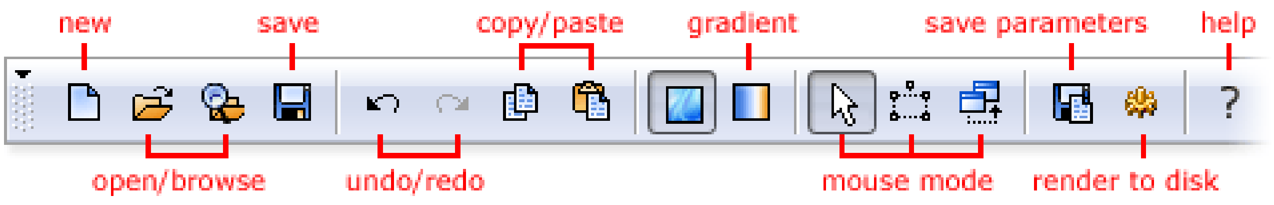
See Also
Workspace overview

Fractal windows

Fractal windows contain the fractals that you work on in Ultra Fractal. While you edit the fractal using the Fractal Properties and Layer Properties tool windows, the fractal window is continually updated to show the result of your changes.

Note: Although fractal windows are resizeable, this does not change the size of the fractal itself. Use the Image tab in the Fractal Properties tool window to resize the fractal.

The toolbar contains commands to edit and save the fractal:



- The **New** button creates a new fractal from scratch. To duplicate the current fractal instead, click Duplicate on the File menu.
- The **Open** and **Browse** buttons open files from disk.
- The **Save** button saves the fractal to disk. See Opening and saving fractals.
- The **Undo** and **Redo** buttons can undo and redo your previous actions. See Fractal history list.
- The **Copy** and **Paste** buttons copy fractal parameters to and from the Clipboard. See Copying and pasting fractals.
- The **Gradient** button opens the gradient editor associated with the fractal window to edit the colors of the fractal.
- The **mouse mode** buttons show and select the active mouse mode. The mouse mode determines what happens when you click and drag inside the fractal window. There are three mouse modes:
  - Normal mode
  - Select mode
  - Switch mode
- The **Save Parameters** button saves the fractal to a parameter set. See Parameter files.
- The **Render to Disk** button starts rendering the fractal or animation to disk, creating a high-resolution image or a fractal movie with better quality than possible in the fractal window. See Rendering images.

The commands on the toolbar are duplicated on the File, Edit, and Fractal pull-down menus. Frequently used commands are also on the menu that pops up when you right-click inside the fractal window.

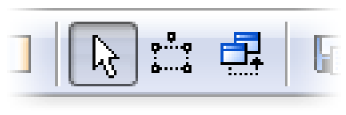
Next: Normal mode

See Also
Keyboard shortcuts for fractal windows
Animation
Exporting and rendering
Workspace

Normal mode

The mouse mode determines what happens when you click and drag inside the fractal window. By default, a fractal window is in Normal mode.

To put the fractal window in Normal mode, click **Normal Mode** on the Fractal menu, or make sure the Normal mode button in the toolbar is down.



In Normal mode, you can zoom, pan, rotate, stretch, and skew the fractal simply by clicking and dragging inside the fractal window. Before you click, hold down one of the Ctrl, Shift, or Alt keys to indicate what you want to do.

To:	**Do this:**
Zoom	Hold down Shift, click and drag
Pan	Hold down Ctrl, click and drag
Rotate	Hold down Alt, click and drag
Stretch	Hold down Shift and Ctrl, click and drag
Skew	Hold down Ctrl and Alt, click and drag
Enter Select mode	Click and drag

While still holding down the left mouse button, move the mouse around to make adjustments. The fractal window continually shows a preview of the result. In the status bar, additional information is shown, such as the current rotation angle.

The status bar also shows extra keys that you can hold down for fine adjustments or to constrain rotation to 45° increments, for example. To use them, first release the key that you have been holding down (still holding down the left mouse button) and then press and hold down the appropriate key.

When you are finished, release the mouse button and the fractal will recalculate to apply your changes. To cancel the operation while you are still holding down the left mouse button, briefly click the right mouse button. If you have already released the mouse button, click **Undo** on the Edit menu to undo the operation.

Notes

- The table above displays the default set of actions. Use the Mouse tab in the Options dialog to customize them. The Fractal Mode tool window always shows the current key bindings.
- If the fractal contains multiple layers, only the editable layers are affected.
- To move the fractal very precisely with the keyboard, hold down Ctrl and use the arrow keys to pan in steps of one pixel. Hold down Shift as well to increase the step to ten pixels. See also Keyboard shortcuts.

Next: Select mode

See Also

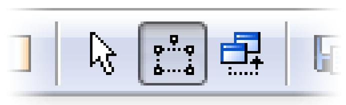
Animating locations

Fractal windows

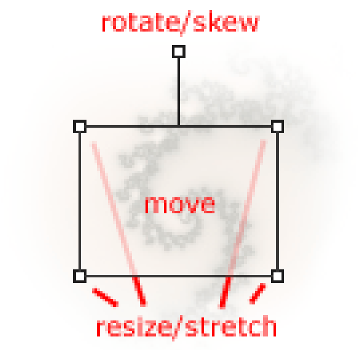
Select mode

The mouse mode determines what happens when you click and drag inside the fractal window. In Select mode, a selection box is used for zooming, panning, rotating, stretching, and skewing.

To enter Select mode from Normal mode, simply click and drag inside the fractal window. Alternatively, click **Select Mode** on the Fractal menu, or make sure the Select mode button in the toolbar is down.



In the fractal window, a selection box appears.



The area inside the selection box will be magnified to fill the fractal window when you zoom in. You can manipulate the selection box by dragging the handles and edges.

To:	**Do this:**
Move the selection box	Click inside the box and drag
Resize the selection box	Drag the edges or the handles at the corners
Rotate the selection box	Drag the top handle
Stretch the selection box	Hold down Ctrl and drag the edges or the handles at the corners
Skew the selection box	Hold down Ctrl and drag the top handle
Cancel	Click outside the selection box

When you are finished, double-click inside the selection box to zoom in. To zoom out, hold down Ctrl while double-clicking. Alternatively, right-click inside the fractal window to open a menu with the following commands:

Zoom In	Magnifies the area inside the selection box to fill the fractal window
Zoom Out	Shrinks the area of the fractal window to fill the selection box
Crop	Resizes the fractal window to fill the selection box
Reset	Restores the default position of the selection box
Resize Fractal	Resizes the fractal window to match the aspect ratio of the selection box

Stretch Fractal	Stretches the fractal window to fit the selection box

While you are working with the selection box, the Fractal Mode tool window shows a preview of the resulting fractal. The buttons in the tool window select what you want to do: zoom in, zoom out, or crop. They duplicate the menu commands listed above, except that they are not applied until you click the Apply button in the tool window.

Notes

- While the selection box is visible, the status bar provides additional information, such as the current magnification and rotation angle.
- If the fractal contains multiple layers, only the editable layers are affected.

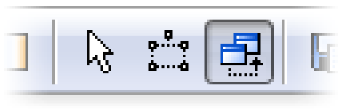
Next: Switch mode

See Also
Tutorial: Learning basic skills
Keyboard shortcuts in Select mode
Animating locations
Fractal windows

Switch mode

The mouse mode determines what happens when you click inside the fractal window. In Switch mode, a mouse click switches between related fractal types.

To enter Switch mode, click **Switch Mode** on the Fractal menu, or make sure the Switch Mode button on the toolbar is down.



Switching is typically used with Mandelbrot and Julia-like fractals. The Mandelbrot set is actually a map of Julia sets. Each point in the Mandelbrot set corresponds to a unique Julia set. The shape of this Julia set reminds of the immediate surroundings of the corresponding point in the Mandelbrot set.

If you enter Switch mode while looking at a Mandelbrot set and move the mouse cursor over the fractal window, the Fractal Mode tool window shows a preview of the corresponding Julia set. Click to open a new fractal window with this Julia set, so you can further explore it.

Notes

- The Explore and Eyedropper features are similar to the switch feature, but they work with all parameters and are much more flexible.
- Sometimes the preview remains static while you are moving the mouse cursor. In this case, the fractal is not a map of the fractal to switch to, for example when you are working with a Julia set. Just click anywhere to switch to the corresponding Mandelbrot set.
- If the fractal has multiple layers, Ultra Fractal uses the active layer for switching.
- The Mouse tab in the Options dialog contains several options for Switch mode, such as the option to open the Julia set in the same fractal window, to copy the coloring of the original fractal, and so on.
- The fractal formula contains switch settings that control how Switch mode works. Refer to the compiler documentation if you want to add switching support to your own formulas.

Next: Opening and saving fractals

See Also
Tutorial: Learning basic skills
Explore
Eyedropper
Fractal windows

Opening and saving fractals

Fractals are saved to fractal files (*.ufr). A fractal file contains one fractal, complete with the calculated pixels and all the information required to restore it. Because the calculated pixels are saved as well, fractal files can become quite large. However, the fractal does not have to be recalculated when opening the file.

To save a fractal to a fractal file, click **Save** on the File menu. If the fractal has not been saved before, a file dialog will pop up, where you can type a name for the fractal.

To open a previously saved fractal file, click **Open** on the File menu. In the file dialog, set the File type input box to Fractal files and select the fractal you want to open. The fractal will be opened in a new fractal window.

At the bottom of the File menu, there is a list of recently opened files. Simply click the name of a file to open it.

Fractal files are a good choice if you want to save fractals for your own reference, for example to a hard disk or CD. However, if you want to share your fractals with other Ultra Fractal users, it is better to use parameter files or copying and pasting.

Notes

- Fractal files are saved in a proprietary format. If you want to import your fractals in graphics software such as Adobe Photoshop, you need to export or render the fractal first.
- You can change the number of recently opened files displayed at the bottom of the File menu in the Environment tab of the Options dialog.

Next: Parameter files

See Also
Tutorial: Learning basic skills
Browsers
Fractal windows

Parameter files

Fractals can be saved as parameter sets as well as in fractal files. Parameter sets are much smaller than fractal files because they do not contain the calculated pixels. That means that the fractal has to be recalculated when a parameter set is opened. Parameter sets are ideal for sharing fractals with other users on the Internet.

Parameter sets are stored in parameter files. A parameter file (*.upr) can contain any number of parameter sets. This makes it easy to store and organize collections of parameter sets. Parameter files are stored as plain text and can be opened in text editors such as Notepad or the built-in formula editor.

To save a parameter set, click **Save Parameters** on the File menu. The Save Parameters browser will open. You can save the parameter set in an existing parameter file or in a new file. Type the name of the file and the title of the parameter set and click Save.

To open a previously saved parameter set, click **Browse** on the File menu. This opens a modeless browser. Select the parameter file that contains the parameter set that you want to open, and then double-click the parameter set inside the file.

You use browsers to organize and manage your parameter sets and parameter files, as well as other files that contain multiple entries, such as formula files. See Browsers.

Notes

- By checking the **Save Formulas** checkbox when saving a parameter set, the formulas used by the fractal are embedded in the parameter file. When opening the parameter set, they will be installed if they are not already present in the formulas folder. Check this option when you are going to send the parameter file to another user (but not on the mailing list).
- Parameter sets larger than 2 KB are saved in a compressed format. To save them without compression, open the Options dialog and uncheck the *Compress parameter sets larger than 2 KB* option in the Fractal tab. This is necessary if you want to edit the parameter files manually.
- Ultra Fractal can also import most Fractint parameter sets in PAR files (*.par). They can be opened just like other parameter files.

Next: Copying and pasting fractals

See Also
Quick Start Tutorial
Fractal windows

Copying and pasting fractals

Fractals can easily be shared between fractal windows and even between Ultra Fractal users by copying them to the Windows Clipboard.

To copy a fractal to the Clipboard, click **Copy** on the Edit menu. The Clipboard now contains a parameter set describing the fractal in plain text format. You can paste it into another fractal window, but you can also share it with other Ultra Fractal users just by pasting it into an email message (for example in Outlook).

To paste a fractal on the Clipboard into an open fractal window, click **Paste** on the Edit menu.

To open a fractal that you have received via email, select the entire parameter set in the email message:

```
MyFractal {
fractal:
...
...
}
```

and then copy it to the Clipboard (in most email software, such as Outlook, press Ctrl+C). Now open a new fractal window in Ultra Fractal (click **New** > **Fractal** on the File menu), and click **Paste** on the Edit menu to paste the parameter set into the fractal window.

Notes

- Click **Copy Formulas** on the Edit menu to copy a fractal to the Clipboard, including the formulas that it uses. This enables other users to open it regardless of the formulas that they have installed on their computer. (Do not use this option on the mailing list, though.)
- Click **Copy Image** on the Edit menu to copy the image of the fractal to the Clipboard so you can paste it into a document or a graphics editor like Paint. However, you can create larger, higher-quality images by rendering them.

Next: Fractal history list

See Also
Ultra Fractal mailing list
Parameter files
Fractal windows
Editing animations

Fractal history list

Every fractal has a history list. The history list stores previous states of the fractal, so you can easily undo and redo your changes. Because the calculated pixels are saved as well, the fractal does not have to be recalculated when undoing changes. This gives you the freedom to explore in the knowledge that you can always effortlessly go back to a previous state, without having to wait.

To go back to the previous state of the fractal, click **Undo** on the Edit menu.

To cancel the last Undo operation, click **Redo** on the Edit menu.

The History tab in the Fractal Properties tool shows a list of previous states of the fractal, complete with previews and descriptions. To go back to a previous state, simply click it.

Next: Full screen mode

See Also
Fractal windows

Full screen mode

A fractal window can be maximized to full screen mode so you can see and explore the fractal without being distracted by other windows.

To enter full screen mode, click **Full Screen** on the Fractal menu. The fractal will now appear full screen. To go back to the normal fractal window, right-click to open a pop-up menu and click Full Screen again.

In full screen mode, a limited number of operations are available through the pop-up menu. You can fully use Normal mode, Select mode and Switch mode to explore the fractal, although without the help of the Fractal Mode tool window. The menu also offers undo and redo commands.

The Gradient submenu provides some additional commands to alter the colors of the fractal. Although this submenu is also available in the normal fractal window, it is especially useful in full screen mode, where you cannot access the gradient editor.

Most fractal window keyboard shortcuts, such as the commands on the Animation menu, also function in full screen mode.

Randomize	Randomizes the colors of the gradient. There are four different options.
Adjust Colors	Opens a dialog to adjust the colors of the gradient. See Adjusting gradients.
Cycle Colors	Cycles the colors of the gradient forward or backward. See Color Cycling.

Next: Default fractal

See Also
Keyboard shortcuts for fractal windows
Fractal windows

Default fractal

When you start Ultra Fractal, a new fractal window is opened automatically with the default fractal. You can use this fractal as a base to create your own fractals.

However, you may wish to modify some of the default settings. For example, you might want to have a different gradient, a different fractal formula, a larger number of frames, and so on.

To change the default fractal, simply modify it as you wish. Then, save it as a parameter set with a new name. For example, you can save it as 'My Default Fractal' in the file 'My Fractals.upr'.

Click **Options** on the Options menu to open the Options dialog. On the Defaults tab, select the parameter set you just saved as the Default parameter set.

From now on, this parameter set will be opened when you start Ultra Fractal.

Notes

- On the Defaults tab of the Options dialog, you can also set the default size of the fractal window, and specify that this should override any parameter sets that you open. It is also possible to set up default copyright and comments notices.
- On the Environment tab, you can also select a different action on startup than loading the default parameter set.
- It is not recommended to save anything to the Examples.upr file that is installed with Ultra Fractal. Use the My Fractals.upr file or other files instead.

Next: Copyright and tweaking

See Also
Parameter files
Fractal windows

Copyright and tweaking

If you want to create your own fractals to perhaps display them on a web site or to sell prints, it is important to respect copyright issues and make sure you own your fractals completely. Fortunately, this is quite straightforward.

The best way to be sure that you are creating an original new fractal is by starting from scratch. You can either start from the default fractal that is opened when starting Ultra Fractal, or by clicking New on the File menu, and then Fractal. You can freely use all formulas, transformations, and coloring algorithms that come with Ultra Fractal or that are in the public formula database.

If you start a fractal by modifying someone else's fractal, creating something original is less easy. The general consensus is that if you make enough modifications so that your fractal does not look like the original fractal anymore, the copyright from the original fractal no longer applies. This could include selecting different fractal formulas or coloring algorithm, zooming in or out, modifying parameters, and so on. The distinction between making enough modifications and barely changing the original fractal is subjective, of course. Therefore, you should always contact the creator of the original fractal to ask for permission before considering your derived fractal to be yours only.

As mentioned above, formulas are free to use. However, modifying formulas is another issue. Generally, you can modify formulas for your own use, but you may not distribute them. If you want to distribute a modified formula, you must ask the original formula author for permission first.

Next: Calculation details

See Also
Fractal windows
Public formulas
Mailing list
Writing formulas

Calculation details

Tip: This slightly technical topic explains how multi-threading in Ultra Fractal works and how its settings can be adjusted. However, you don't need to know this in order to work with Ultra Fractal.

Ultra Fractal is fully multi-threaded [1] and can take advantage of multi-processor computers to speed up fractal calculations. This is done by splitting the fractal into multiple parts so the processors can work on each part in parallel. You can even distribute calculations to other computers with the network calculations feature.

This works well with most Mandelbrot-type fractals, where each pixel of the fractal can be calculated independently. However, with fractal types such as IFS and Flame Fractals [2], the entire image is calculated in one step and it cannot be subdivided. Therefore, multiple processors and network calculations will not speed up these fractal types. In Ultra Fractal 3, rendering these fractal types was not recommended, but that limitation does not apply anymore. (Formula authors: see the render setting.)

If you have a processor with HyperThreading, Ultra Fractal will recognize it as a dual processor and split up its calculations accordingly. In most cases, this will result in a modest speed improvement. Otherwise, you can force Ultra Fractal to use just one processor.

Open the Options dialog and go to the Fractal tab. In the Advanced calculation options area, the **Minimum number of threads** option sets the minimum number of threads that Ultra Fractal will use for a single fractal window. If this is set to 1, calculations will not be subdivided. Typically, this should be set to the number of processors in your computer.

Ultra Fractal will typically use one thread per layer. If the number of layers is low, extra threads will be added (subdividing one or more layers) to reach the minimum number of threads setting, which ensures that all processors are used. However, if there are many layers, this would create a large number of threads, which would saturate the system.

The **Maximum number of threads** option limits the number of threads for a fractal window, making sure that complex fractals will not start an unlimited number of threads. By default, this is set to four times the number of processors in your computer. If you would like more layers to be calculated simultaneously, you can increase this setting.

[1] A thread is an independent part of a program that can run on a processor. Each thread can run on a separate processor. By splitting calculations into multiple threads, Ultra Fractal ensures that all processors in your computer are used efficiently.

[2] To use IFS and Flame Fractals, you first need to download the set of public formulas from the online formula database. Select **Pixel** in **mt.ufm** as the fractal formula and either **Iterated Function Systems** in **mt.ucl**, or **Flame Fractals** in **enr.ucl** as the outside coloring algorithm.

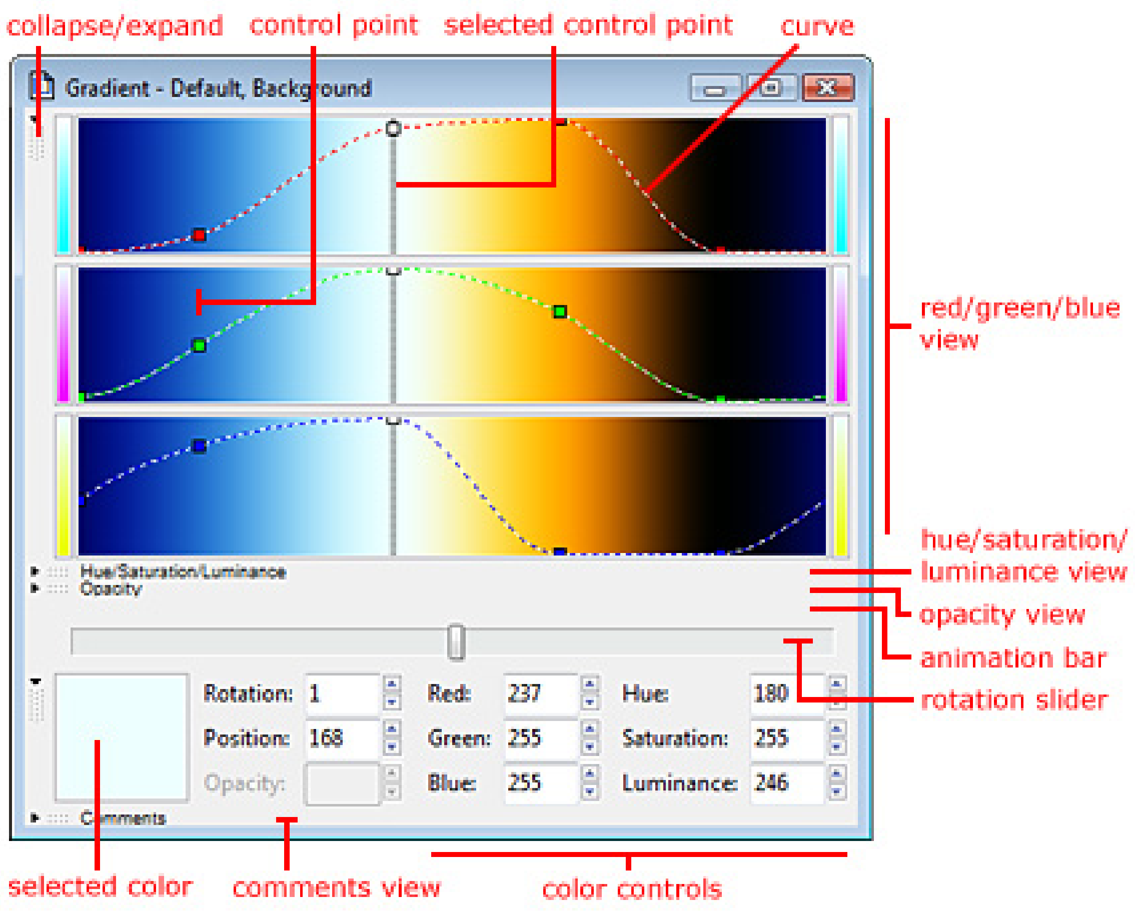
See Also
Fractal windows
Playing animations

Gradients

Gradients contain coloring information for fractals. Each layer in a fractal has its own gradient. Gradients can also be edited and saved independently with a stand-alone gradient editor.

To open the gradient editor associated with a fractal window, click **Gradient** on the Fractal menu. This gradient editor can be recognized because it shows the name of the fractal and the active layer in the title bar. When the gradient is edited, the fractal window immediately redraws itself to show the new colors.

To open a stand-alone gradient editor, click New on the File menu, and then click Gradient.



Rotation: 1
Position: 168
Opacity:
Red: 237
Green: 255
Blue: 255
Hue: 180
Saturation: 255
Luminance: 246

The gradient editor provides various views on the gradient. Each view can be collapsed and expanded by clicking the button on the left of it. There are five views:

Red/Green/Blue	Edits the gradient in the RGB color model
Hue/Saturation/Luminance	Edits the gradient in the HSL color model
Opacity	Edits the transparency of the gradient
Controls	Allows you to fine-tune the selected control point by entering values manually
Comments	Provides a place to type comments

The **rotation slider** is placed outside the collapsible views, so it is always visible. It rotates the gradient to change the way the colors are mapped onto the fractal.

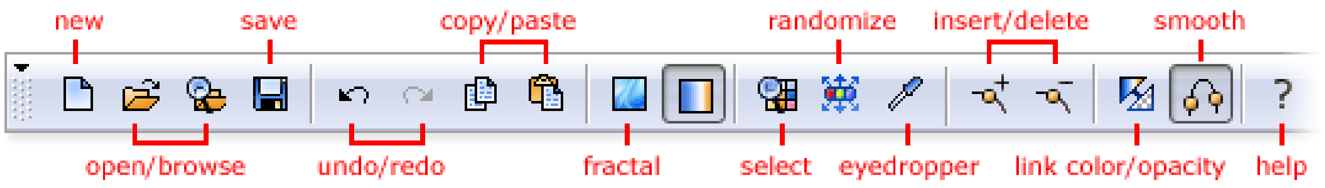
Just above the rotation slider is the **animation bar**, which is normally empty. It shows the animation keys for animated control points in the gradient. See Animating gradients.

Next: Gradient toolbar

See Also
Tutorial: Learning basic skills
How gradients work
Transparent gradients
Fractal windows

Gradient toolbar

The toolbar for the gradient editor contains commands to edit and save the gradient:



- The **New** button creates a new fractal from scratch. To create a new gradient, click New on the File menu and then click Gradient. To duplicate the existing gradient, click Duplicate on the File menu.
- The **Open** and **Browse** buttons open files from disk.
- The **Save** button saves the gradient to disk. See Opening and saving gradients.
- The **Undo** and **Redo** buttons can undo and redo changes to the gradient.
- The **Copy** and **Paste** buttons copy gradients to and from the Clipboard. This is useful for copying gradients between layers or between fractals.
- The **Fractal** button activates the fractal window that owns the gradient editor. This button is not available with stand-alone gradient editors. Together with the **Gradient** button next to it, you use these buttons to switch back and forth between the fractal window and the gradient editor.
- The **Select Color**, **Randomize Color**, and **Eyedropper** buttons change the color of the selected control point. See Editing gradients.
- The **Insert** button adds a new control point. The **Delete** button removes the selected control point.
- The **Link Color and Opacity** button links and unlinks the color and opacity parts of the gradient. See Transparent gradients.
- The **Smooth Curves** button controls how curves between control points are interpolated: linearly or smoothly.

The commands on the toolbar are duplicated on the File, Edit and Gradient pull-down menus. Frequently used commands are also on the menu that pops up when you right-click inside the gradient editor.

Next: How gradients work

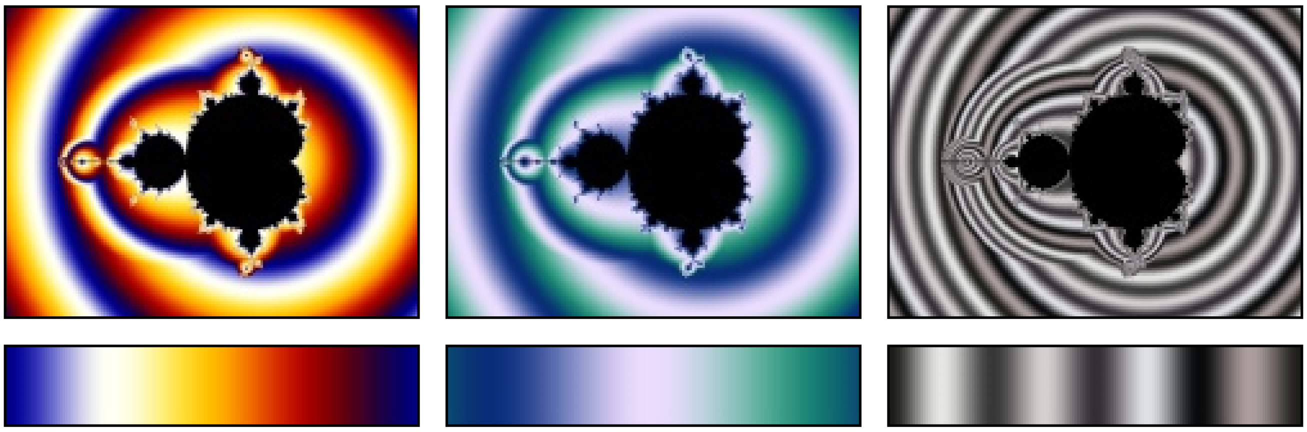
See Also
Keyboard shortcuts for gradient editors
Gradients

How gradients work

When Ultra Fractal calculates a fractal, it does not immediately calculate a color for each pixel. Instead, it calculates an intermediate **index value**. The index value is a single floating-point number that is returned by the selected coloring algorithm.

The gradient translates index values to colors. Since only the index values are stored, the colors can be changed without having to recalculate the fractal. To put it another way, the coloring algorithm defines the distribution of the colors, the gradient defines the colors themselves.

Here is an example. This is the same fractal with three different gradients, shown below each fractal.



The fractal contains only colors from the gradient. In addition, you can also recognize the color transitions from the gradient in the fractal. This is because most coloring algorithms, such as the one used here, create smooth ranges of index values. Note how the gradient wraps around at the endpoints to create smoothly colored images.

Note that with direct coloring algorithms, the coloring algorithm directly calculates the color of a pixel, and the gradient is used differently. A direct coloring algorithm can use colors from the gradient, but it is not limited to the gradient alone.

Next: Editing gradients

See Also
Gradients

Editing gradients

The colors in the gradient are edited by dragging the control points. You can use either the Red/Green/Blue view or the Hue/Saturation/Luminance view. They both edit the same control points, but with different color models. You can resize the gradient editor for more accurate positioning of the control points.

Click a control point to select it. Hold down Shift or Ctrl and click to select multiple control points. Click on the curve background to deselect all control points. To draw a rectangle around control points to select them, click on the curve background and drag.

Drag a control point to change its color and position. To change only the position or only the color, hold down Shift while dragging. In the Controls view, you can manually enter the color and position of the control point for fine-tuning.

Click **Insert** on the Edit menu, and then click somewhere in the gradient editor to insert a new control point there. You can also hold down Ctrl and click on the curve background where you want to insert a new control point.

Click **Delete** on the Edit menu to delete the selected control point. You can also hold down Ctrl and click on the selected control point, if no other control points are also selected.

Right-click in the gradient editor and click **Select Color** to open a dialog box to change the color of the selected control point. This is an alternative to dragging the control point itself.

Right-click in the gradient editor and click **Randomize Color** to set the color of the selected control point to a random value.

Right-click in the gradient editor and click **Eyedropper** to pick the color of the selected control point from any open gradient editor or fractal window. Click Eyedropper again to cancel.

Click **Smooth Curves** on the Gradient menu to toggle between linear interpolation and smooth interpolation for the curves. This affects how the gradient is colored between control points.

Click **Undo** on the Edit menu to undo the changes you have made. The history list of the gradient is independent from the fractal history list, but it is reset when undoing changes to the fractal.

Use the **Copy** and **Paste** commands on the Edit menu to copy the gradient to the Clipboard and paste the gradient on the Clipboard into the editor, replacing the current gradient. This enables you to copy gradients to other layers, other fractal windows, and to and from stand-alone gradient editors.

Next: Transparent gradients

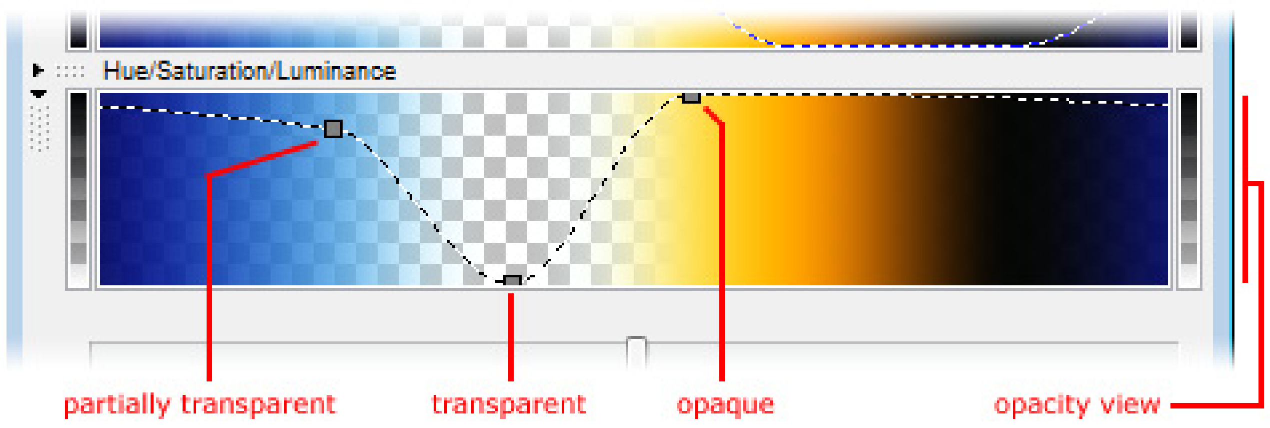
See Also

Tutorial: Learning basic skills
Tutorial: Working with layers
Keyboard shortcuts for gradient editors
Gradients
Animating gradients

Transparent gradients

The gradient defines not only the colors of a layer, but also the transparency. An opacity value is associated with every color to make it more or less transparent. By default, all opacity values are set to 255, which makes them completely opaque.

Use the Opacity view to edit the opacity of the gradient. The opacity curve can be edited independently from the color curves (they do not necessarily share the same control points). To edit the curve, click on the Opacity view to activate it, and drag the control points, just like when you edit the color curves. Drag a control point up to make it opaque, drag it down to make it transparent.



The pattern of blocks shows the transparency of the gradient. This pattern is also visible in the fractal window unless there is a layer below the current layer that is completely opaque, in which case the underlying layer is shown.

Many gradient commands work only on the active curve or curves. For example, when the opacity curve is active, the Smooth Curves command (click Smooth Curves on the Gradient menu) will adjust the curvature of the opacity curve instead of the color curves. Only the active curve shows control points.

You can link the color curves and the opacity curve so they share the same control points. To link them, click **Link Color and Opacity** on the Gradient menu. This will adjust the opacity curve to give it the same control points as the color curves. You can now edit the opacity and color curves simultaneously.

Next: Adjusting gradients

See Also
Tutorial: Working with layers
Tutorial: Masking
Layers
Masks
Gradients

Adjusting gradients

To edit the gradient, you usually manipulate individual control points. However, there are also several commands that adjust the entire gradient. These commands will work on the active curves (color or opacity), or both when they are linked together (see Transparent gradients).

Click **Adjust Colors** on the Gradient menu to open the Adjust dialog. This dialog allows you to change the color balance, hue, saturation, brightness, and contrast of the entire gradient. For example, by moving the Saturation slider in the HSL tab completely to the left, you can create a grayscale version of the gradient.

Click **Randomize** on the Gradient menu to randomize the gradient. This fills the gradient with a random number of control points, all with random colors.

Click **Randomize Bright** or **Randomize Misty** for a different selection of colors.

Click **Randomize Custom** to open a dialog with a variety of options for randomizing the gradient. Here, you can randomize for example only the positions of the control points, or choose from specific ranges of values for the colors.

Click **Reverse** on the Gradient menu to reverse (mirror) the positions of the control points, so the leftmost point will appear on the right, and the rightmost point on the left.

Click **Invert** on the Gradient menu to invert the colors or opacity values of the control points. This is often useful with the opacity curve, to invert what is transparent and what is opaque.

Next: Opening and saving gradients

See Also
Animating gradients
Editing gradients
Gradients

Opening and saving gradients

Gradients are saved in gradient files (*.ugr). A gradient file is a plain text file that can contain any number of gradients, so you can store and organize sets of gradients.

To save a gradient, click **Save** on the File menu. The Save Gradient browser will open. You can save the gradient in an existing gradient file or in a new file. Type the name of the file and the title of the gradient and click Save.

To open a previously saved gradient, click **Browse** on the File menu. This opens a modeless browser. Make sure **Gradient Files** is selected in the toolbar. Select the gradient file that contains the gradient that you want to open, and then double-click the gradient inside the file. The gradient will be opened in a new stand-alone gradient editor.

To open a gradient in the active gradient editor, click **Replace** on the File menu instead. This is useful if you want to use the saved gradient in a fractal.

Notes

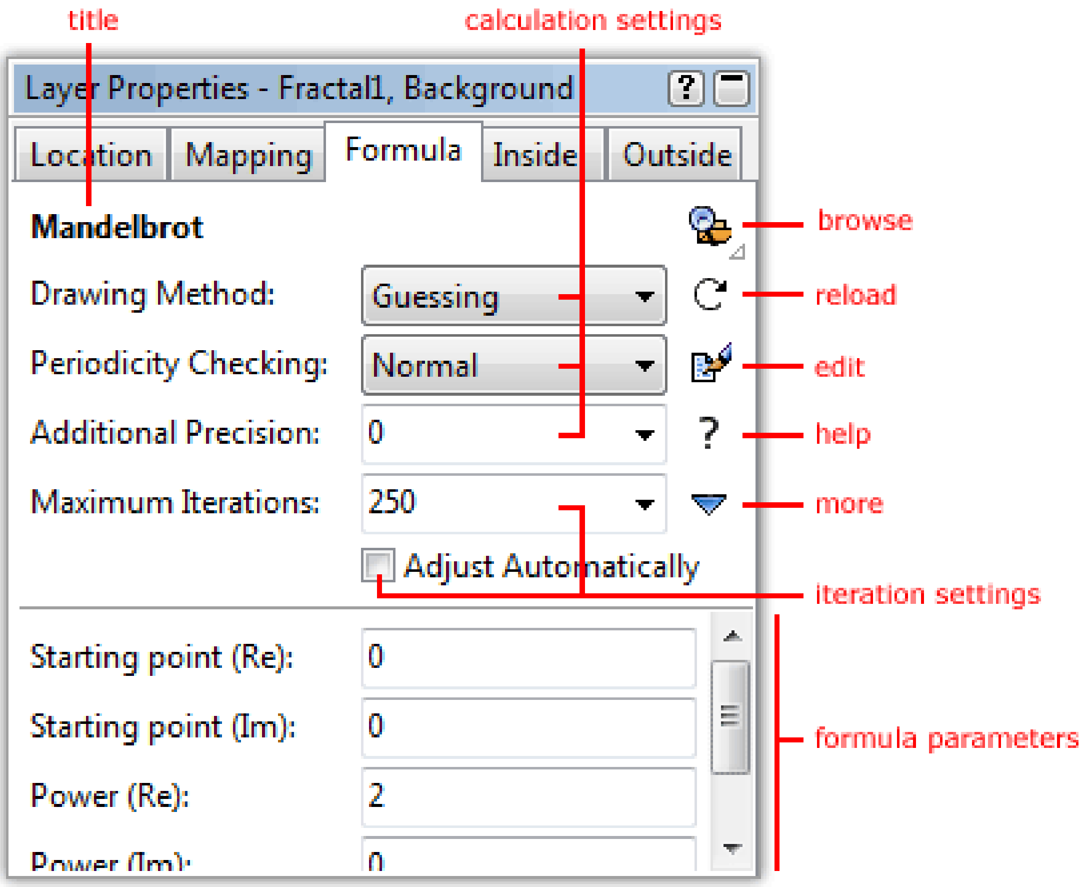
- When saving a gradient, you can choose to save only the color or opacity parts of the gradient with the **Save Color** and **Save Opacity** checkboxes.
- Another way to open gradients is to click **Open** on the File menu and select a gradient file. A modal browser window will open, showing the gradients in the file. Double-click a gradient to open it.
- Palette files in Fractint's MAP format (*.map) can be opened just like other gradient files.

See Also
Browsers
Parameter files
Gradients

Fractal formulas

The fractal formula creates the basic shape and form of a fractal. Ultra Fractal comes with a number of standard formulas that you can use. You can also download additional formulas from the Internet and even write your own formulas.

Fractal formulas are managed in the Formula tab of the Layer Properties tool window.



Mandelbrot
Drawing Method: Guessing
Periodicity Checking: Normal
Additional Precision: 0
Maximum Iterations: 250
Adjust Automatically
Starting point (Re): 0
Starting point (Im): 0
Power (Re): 2
Power (Im): 0

- At the top, the **title** of the fractal formula is shown. Hold the mouse cursor over the title to see the entry identifier and the file name of the formula.
- The **Browse** button opens a modal browser to select another fractal formula.
- The **Reload** button reloads the fractal formula from disk and recalculates the layer.
- The **Edit** button opens the fractal formula in the formula editor.
- The **Help** button opens the help file for the formula, if one exists.
- The **More** button shows a menu with additional commands.
- The **calculation settings** specify how the fractal should be calculated. See Working with fractal formulas.
- The **iteration settings** specify how many iterations should be used. See Maximum iterations.
- The **formula parameters** are additional parameters specific to the selected fractal formula. See Formula parameters.

Next: Working with fractal formulas

See Also
Quick Start Tutorial
Coloring algorithms

Transformations
What are fractals?

Working with fractal formulas

You work with fractal formulas in the Formula tab of the Layer Properties tool window. This tab also contains global calculation settings for the layer, since these are closely related to the selected fractal formula.

Fractal formulas are stored in fractal formula files (*.ufm). Each file can contain multiple formulas.

To select a fractal formula, click the **Browse** button. This opens a modal browser that shows the formula files and formulas on your computer. Double-click on a formula to select it.

Hold down the Browse button to open a menu with fractal formula presets. See Presets.

Some formulas contain additional help. Click the **Help** button to open it.

Click the **More** button to access commands to **copy** and **paste** the settings and parameters on the Formula tab, and to **reset** all parameters to the default values.

The Formula tab is divided into two panes. The top pane contains global calculation settings.

Drawing Method	Selects how the pixels are calculated. • **Guessing** starts with a low-resolution preview, and then gradually increases the resolution while trying to guess pixels instead of calculating them. This is the fastest option, but also the least accurate. When you are working with animations, use this drawing method for fast animation previews. • **Multi-pass Linear** also starts with a low-resolution preview, but it calculates all pixels instead of guessing them. • **One-pass Linear** calculates all pixels from top to bottom. For maximum accuracy, use one of the linear drawing methods.
Periodicity Checking	Specifies the amount of periodicity checking used. Periodicity checking can greatly enhance the speed at which inside areas are calculated. **Rough** is the fastest option, but also the least accurate. **Off** turns off periodicity checking completely for the highest accuracy. Some fractal formulas do not work well with periodicity checking, in which case you should turn it off. If odd lines or dots appear on the top of the fractal, this may be the case.
Additional Precision	Specifies how many extra digits of precision should be used for calculations. See Arbitrary Precision.

The top pane also contains the iteration settings. The bottom pane contains the formula parameters. These parameters are specific to the selected fractal formula.

Next: Maximum iterations

See Also

Fractal formulas

Standard fractal formulas

Maximum iterations

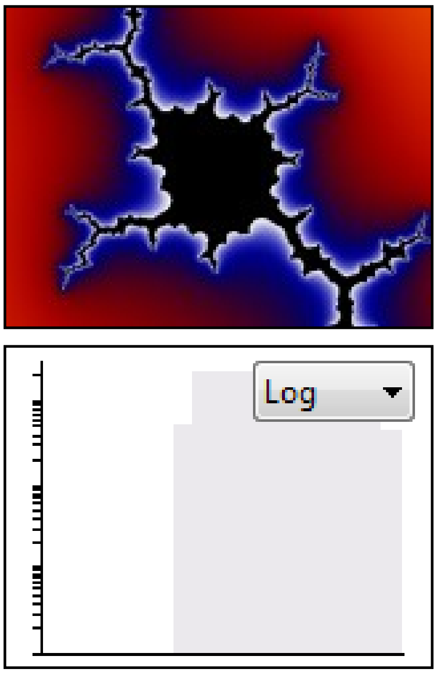
To calculate a pixel in a fractal, Ultra Fractal iterates the selected fractal formula. It executes it multiple times, each time using the result from the previous calculation as input.

The formula is iterated until the maximum iteration count is reached, or until the bail-out condition (specified by the fractal formula) is met. If the bail-out condition is met, the pixel is colored as an outside pixel. Otherwise, it is colored as an inside pixel.

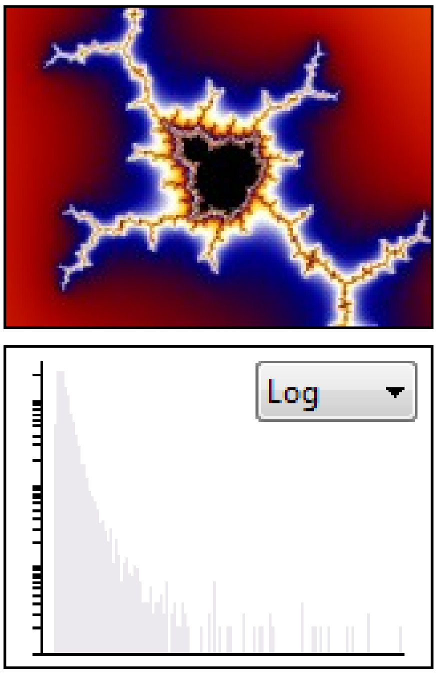
Sometimes, many iterations are necessary to reach the point where the bail-out condition is satisfied. If the maximum iteration count is too small, the pixel will be incorrectly colored as an inside pixel because the bail-out point is not reached. On the other hand, if the iteration count is too large, many iterations will be performed for the pixels that are inside, and the fractal will be calculated slowly.

The **Maximum Iterations** setting on the Formula tab of the Layer Properties tool window specifies the maximum iteration count. To help you find a good value, the Statistics tool window shows a histogram of the iteration values on its Iterations tab.

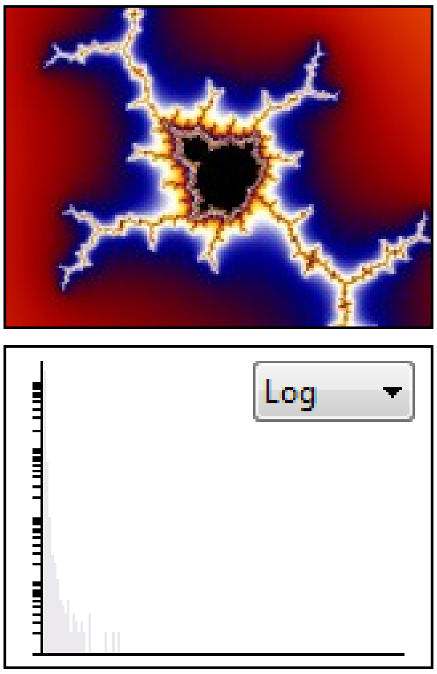
This example illustrates the influence of the maximum iterations setting:



Maximum iterations: **20**



Maximum iterations: **200**



Maximum iterations: **2000**

We see the same image three times, with three different values for the maximum number of iterations. Below each image, the iterations histogram from the Statistics tool window is shown.

The first image clearly suffers from a low value for the maximum iterations setting. By increasing it, we obtain the second image, which looks much better. Further increasing the value does not change the image much, so we conclude that 200 is a good value in this case.

Notes

- The histogram can aid you in deciding whether or not you have to change the maximum iterations value. If most of the iterations end up on the left side (as with the middle image), you are probably safe. If they are all at the far left side, the value is probably too high (which makes the fractal slower to calculate). If there are many iterations on the right side, the value is too low. Move the mouse pointer over the tool window to see the corresponding iteration values.

- Experiment with the maximum iterations setting to learn how to use it. Sometimes, a (too) low value can also be artistically pleasing.
- By checking the **Adjust Automatically** checkbox on the Formula tab, Ultra Fractal automatically adjusts the maximum iterations value when zooming in or out. This can be helpful, but it is not fool-proof: you may need to adjust the value manually every once in a while.

Next: Formula parameters

See Also

Fractal formulas

Working with fractal formulas

Inside and outside

Formula parameters

The bottom pane on various tabs of the Layer Properties tool window contains parameters that are specific to the selected transformation, fractal formula, or coloring algorithm. There are eight types of parameters:

- **Complex** parameters consist of a real and an imaginary value. They appear as two input boxes labeled with "(Re)" and "(Im)" after the name of the parameter. Right-click one of the input boxes to open a menu with additional options.
- **Floating-point** parameters specify a single floating-point value (such as -0.2 or 3.14).
- **Integer** parameters specify an integer value (such as -2 or 3).
- **Enumerated** parameters appear as a drop-down box. They are typically used to select one of various behaviors, or to choose between a number of options.
- **Color** parameters specify a color and are typically used only by direct coloring algorithms. They appear as a color swatch. Click it to adjust the color. Right-click it to open a menu with additional options.
- **Boolean** parameters appear as a checkbox. They are used to turn options on or off.
- **Class** parameters provide a way to extend a formula with additional features. See About classes.
- **Image** parameters, typically used in coloring algorithms, enable importing external images. See Using images.

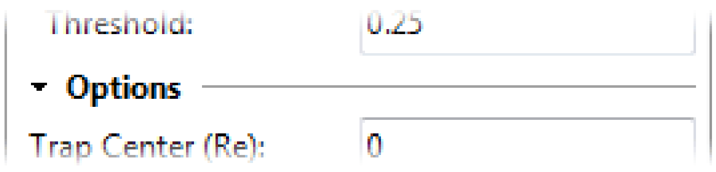
Many formulas require a bit of experimentation with the parameters to learn how to use them. For new users, it is best to stick to the standard formulas and learn how to use these first.

Some formulas (such as the standard formulas) contain additional help. To access it, click the **Help** button.

To copy the formula settings (including parameter values) between layers, click the **More** button, and then click **Copy** or **Paste**.

To reset all parameters to their default values, click the **More** button, and then click **Reset Parameters**.

In some formulas, such as Orbit Traps, parameters are divided into groups by **headings**.


Threshold: 0.25
Options
Trap Center (Re): 0

You can **collapse** and **expand** the heading and the parameters below it by clicking on the arrow button. Right-click the heading for a menu with commands to collapse and expand all headings.

Notes

- You can horizontally resize the area reserved for parameter labels, for example to make more space available for long labels. Simply position the mouse cursor just left of a parameter input box so it turns into a horizontal resize cursor, and drag this edge to where you want it to be. Double-click here to reset the label area to the default size.

- Many formulas contain help for individual parameters. Make sure the Fractal Mode tool window is visible and hover the mouse cursor over a parameter to see its help text, if one exists.
- The Explore feature makes it easy to try new parameter values.
- If multiple layers or transformations are selected that do not use the same formula, the list of parameters remains empty.
- You can easily copy complex values from one parameter to another. Right-click a complex parameter and click **Copy Complex Value** or **Paste Complex Value**. You can also copy coordinates between the Location tab and a complex parameter in this way. To copy a color parameter, right-click it and click **Copy** or **Paste**.

Next: Explore

See Also

Animating parameters

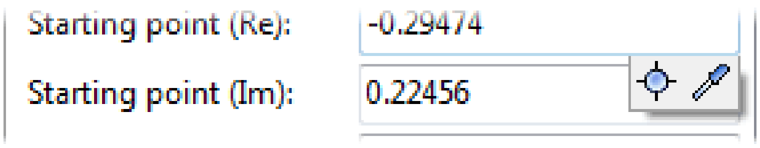
Fractal formulas

Working with fractal formulas

Explore

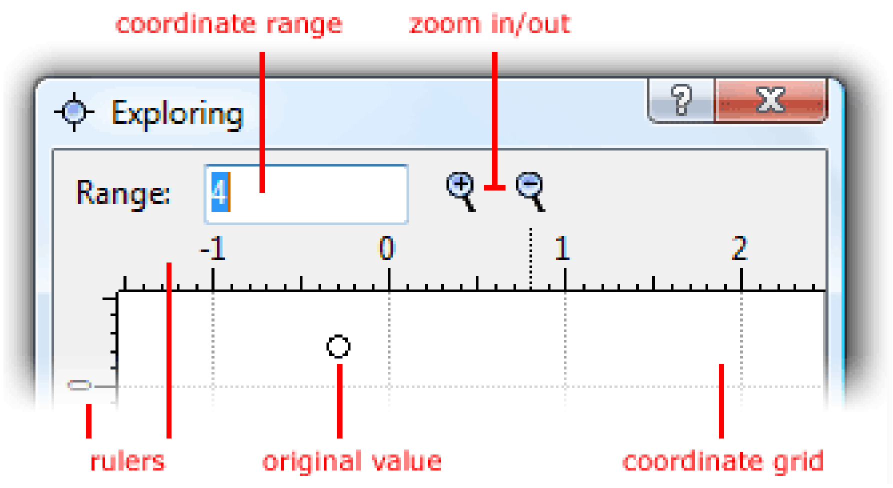
You can try new parameter values by typing one after the other, but it is much better to use the Explore feature. The Explore feature makes experimenting with parameters easier and more fun.

If you click on a parameter to give it the keyboard focus, a small window will pop up below the input box with two buttons. These invoke the Explore and Eyedropper features.



Click the **Explore** button to start exploring the parameter.

The Explore window will pop up. It contains a coordinate grid, rulers, and zooming controls.



- Move the mouse cursor over the **coordinate grid** to try different parameter values. Look at the Fractal Mode tool window to see a live preview of the active fractal layer with the current parameter value.
- Enter a new value in the **range** input box, or use the **Zoom In** and **Zoom Out** buttons to decrease or increase the coordinate range.
- The **rulers** show the current coordinates. Drag them around to pan the coordinate grid.

Click inside the coordinate grid to select a new value. The Explore window will close and the new parameter value will be applied.

The Explore feature works with complex, floating-point, and integer parameters. The picture above shows the Explore window in complex mode. When exploring floating-point and integer parameters, the vertical ruler is not included.

Notes

- You can also pan or zoom by dragging the coordinate grid while holding the Ctrl or Shift key, just like in the fractal window.

- A small circle in the Explore window shows the original value of the parameter that is being explored.
- To cancel without selecting a new parameter value, either close the Explore window, or click on the explore icon that has appeared in the input box for the parameter that is being explored.
- Another way to start exploring a parameter is by right-clicking it and selecting **Explore** from the menu that pops up.

Next: Eyedropper

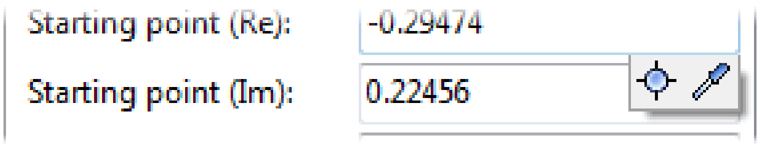
See Also
Formula parameters
Fractal formulas
Working with fractal formulas

Eyedropper

Another tool to choose new parameter values is the Eyedropper feature. It is similar to the Explore feature, but it selects parameter values from the coordinates in the fractal window, instead of using a separate coordinate grid. This is useful if the fractal coordinates have a relationship with the parameter that you are working with.

For example, to choose new values for the Julia Seed parameter of a Julia set, open a separate fractal window with the Mandelbrot set, and use the Eyedropper feature to pick values from the Mandelbrot fractal, as an alternative for Switch mode.

If you click on a parameter to give it the keyboard focus, a small window will pop up below the input box with two buttons. These invoke the Explore and Eyedropper features.


Starting point (Re): -0.29474
Starting point (Im): 0.22456

Click the **Eyedropper** button to start eyedropper mode.

As you move the mouse cursor over a fractal window, you see the fractal coordinates for the point under the mouse cursor in the input box for the parameter that is in eyedropper mode. You can select coordinates from any fractal window, not just the active window.

Look at the Fractal Mode tool window for a live preview of the active fractal layer with the current parameter value. Simply click inside the fractal window to select the new parameter value.

The Eyedropper feature works with complex, floating-point, integer, and color parameters. With floating-point and integer parameters, only the real part of the fractal coordinates is used.

To use the eyedropper with color parameters, right-click the color parameter and click **Eyedropper**. You can select colors from any fractal window or gradient editor.

Notes

- To cancel without selecting a new parameter value, click on the eyedropper icon that has appeared in the input box for the parameter that is in eyedropper mode.
- Another way to start the eyedropper is by right-clicking a parameter and selecting **Eyedropper** from the menu that pops up.

Next: Presets

See Also
Explore
Formula parameters
Fractal formulas
Working with fractal formulas

Presets

As you work with formulas in Ultra Fractal, you will often use the same combinations of formulas and parameters to achieve certain effects. You can save the combination of a formula and parameters as a preset for quick access in the future.

To open a saved formula preset, click and hold down the **Browse** button until a menu with presets appears. Simply click a preset to load it.

To save and organize presets, click **Define** on the presets menu. This opens the Edit Presets dialog.

- Click **Add Current** to save the current settings as a new preset.
- Click **Delete** to delete the selected preset.
- Click **Rename** to rename the selected preset. You can also click a preset in the list twice, or press F2.
- Use the **Move Up** and **Move Down** buttons to reorder presets, or just drag them around in the list.

Click **OK** to apply your changes.

Besides fractal formulas, there are also presets for coloring algorithms, transformations, and layers. To access the transformation presets, click and hold down the **Add** button in the Mapping tab. To access the layer presets, click and hold down the **Add** button in the Layers tab of the Fractal Properties tool window.

Presets are saved in files with the .ups file extension in the Ultra Fractal system folder, which is displayed on the Folder tab of the Options dialog. You might want to include these files when you make a backup of your fractal documents.

Next: Arbitrary precision

See Also
Fractal formulas
Working with fractal formulas
Parameter files

Arbitrary precision

Ultra Fractal can perform fractal calculations with almost any desired precision. This enables you to zoom as deep as you want, without hitting a precision limit. This is called arbitrary precision or deep zooming.

Three types of precision are available:

- **Double** is the fastest and the least precise method. It supports magnifications up to about 10^{10} (1E10, or 10 billion). It has a precision of 15-16 decimals.
- **Extended** is slightly more precise and a little slower, supporting magnifications up to about 10^{16}. It has a precision of 19-20 decimals.
- **Arbitrary** is much slower, but it supports magnifications up to 10^{4000}. Its precision can be scaled from 20 to 10,000 decimals.

Ultra Fractal automatically selects the best precision type, depending on the current magnification and the selected fractal formula, transformations, and coloring algorithms. It calculates the number of decimals required and selects the fastest precision type that can support it.

You can verify the number of decimals required and the selected precision type in the General tab of the Statistics tool window.

Sometimes, you may want to adjust the number of decimals to force Ultra Fractal to use a different precision type, or to change the number of decimals used with the Arbitrary precision type. The **Additional Precision** input box on the Formula tab of the Layer Properties tool window allows you to do this.

The value in the Additional Precision input box is added to the default number of decimals required. Positive values will increase the precision; negative values will decrease it. Keep an eye on the Statistics tool window to see the effect.

Notes

- It is not recommended to try and use the additional precision to achieve artistic effects, since they would rely on artefacts in the current implementation, and might be broken by future versions.
- Some older formulas rely on the Extended precision that was always used by Ultra Fractal 2. In this case, you can adjust the additional precision (looking at the Statistics tool window) until the Extended precision type is used.
- To force Ultra Fractal to always or never use Arbitrary precision, select the desired option in the Use arbitrary precision area on the Fractal tab of the Options dialog.

Next: Public formulas

See Also
Fractal formulas

Public formulas

Ultra Fractal comes with a set of standard fractal formulas, transformations, coloring algorithms and classes. They are easy to use and well documented. However, when you get more experienced, you will want to work with more different formulas.

Most of the custom formulas and classes written for Ultra Fractal are available through a public formula database on the Internet (formulas.ultrafractal.com). Here, you can download a complete set of formulas, browse the collection, and add your own formulas.

You can download formulas and classes from the database from within Ultra Fractal. This will automatically download new and updated files and install them appropriately.

- To start updating your collection of public formulas, click **Update Public Formulas** on the Options menu. You can select what to download: the weekly or monthly update, or the full collection. Ultra Fractal will automatically select the most appropriate option depending on how long ago you last updated the formulas.

The downloaded files will be installed in the Public formulas folder.

Notes

- You are free to organize the public formulas in subfolders in the Public formulas folder, for example to put files that you rarely use in a separate folder. The files will still be updated correctly.
- The formula database also contains text files with documentation and parameter files with examples. These are copied to the Public formulas folder as well.
- If you have a dial-up connection, make sure you are connected to the Internet before updating your formulas. Otherwise, Ultra Fractal might not be able to connect to the formula database.
- When opening a parameter set that uses formulas that cannot be found, you can directly download the file from the formula database.

How are the public formulas installed?

The formulas will be installed according to the following rules:

- If a formula file already exists in the Public formulas folder or in a subfolder, it will be overwritten. If the existing file is newer than the downloaded file, you will be prompted for confirmation.
- If a formula file does not exist yet, it will be created in the Public formulas folder.
- If a formula file already exists outside the Public formulas folder, it will not be updated or overwritten. These files are listed as skipped. If you are a formula author, you can thus avoid overwriting your own files when updating the formulas by placing them in the Formulas\My Formulas folder instead of in Formulas\Public.

When all files are downloaded and installed, a short summary of the changes is shown. Details can be found in the Update.log file created in the Public formulas folder.

By default, the location of the Public formulas folder is Documents\Ultra Fractal 5\Formulas\Public. It is always located under the main Formulas folder. You can change its location in the Folder tab of the Options dialog.

Next: Standard formulas

See Also

Formula ratings

Transformations

Fractal formulas

Coloring algorithms

Standard fractal formulas

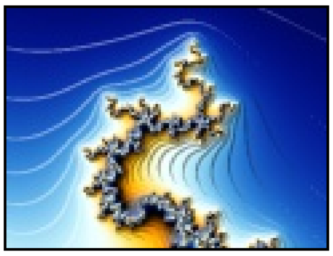
Ultra Fractal comes with a set of standard fractal formulas. They are located in the file Standard.ufm in the Formulas folder. It contains the following formulas:

- Embossed (Julia, Mandelbrot, Newton)
- Generic Formula
- Julia
- Julia (Built-in)
- Lambda (Julia, Mandelbrot)
- Magnet 1 and 2 (Julia, Mandelbrot)
- Mandelbrot
- Mandelbrot (Built-in)
- Newton
- Nova (Julia, Mandelbrot)
- Phoenix (Julia, Mandelbrot)
- Pixel
- Slope (Julia, Mandelbrot, Newton)

All standard fractal formulas are also implemented as fractal formula classes in Standard.ulb. This makes it possible to use them with Generic Formula or in other formulas that accept fractal formula classes. See About classes and Example 1 - Formula classes for more information.

See Also
Standard transformations
Standard coloring algorithms
Fractal formulas
Public formulas

Embossed (Julia, Mandelbrot, Newton)



The Embossed fractal formulas are modifications of the classic Mandelbrot, Julia, and Newton fractals that create 3D bevel effects with contour lines. They are available as fractal formulas in Standard.ufm and as a fractal formula class in Standard.ulb.

In class form, Embossed uses a class parameter to select which fractal formula to use, so it is not just limited to Mandelbrot, Julia or Newton fractal types. See Example 1 - Formula classes for more information.

Embossed should be combined with the Emboss coloring algorithm. This coloring algorithm correctly translates the results from the Embossed fractal formulas to colors in the gradient.

For best results, use a black-to-white gradient such as **Emboss** in Standard.ugr. This will create a grayscale image with shaded contour lines. You can then combine this with other layers to add colors while retaining the 3D effect. For the merge mode of the layer with the Embossed formula, try Soft Light or Hard Light.

The following parameters are provided:

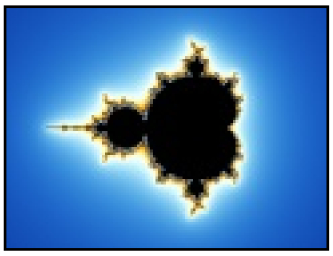
Fractal Formula	Selects the fractal formula to use for the embossing effect. (Class version of Embossed only.)
Emboss Type	Specifies what kind of information from the fractal calculations is used to create the embossing effect. This changes the shape and place of the contour lines.
Light Angle	This is the angle of the apparent light source, in degrees. The default value 0 corresponds to light from above. Positive values rotate the light source in clockwise direction.
Contour Size	Specifies the relative size of the contour lines. When zooming in, the size in pixels of the contour lines appears to stay the same.

For the non-class version of the Embossed fractal formulas, the other parameters are described in the topics for the regular (non-Embossed) formulas.

Note: When rendering embossed fractals, use non-adaptive anti-aliasing to ensure that the contour lines are anti-aliased correctly, and make sure the Force Linear drawing method option is checked.

See Also
Slope (Julia, Mandelbrot, Newton)
Standard formulas

Generic Formula



Generic Formula is a skeleton fractal formula that lets a fractal formula class do the actual work. All fractal formulas in Standard.ufm are also implemented as fractal formula classes. You can select any of these as the fractal formula class to be used together with Generic Formula.

The **Fractal Formula** class parameter sets the fractal formula class to be used. By default, the Mandelbrot class from Standard.ulb is selected.

An example of using Generic Formula is given in Example 1 - Formula classes.

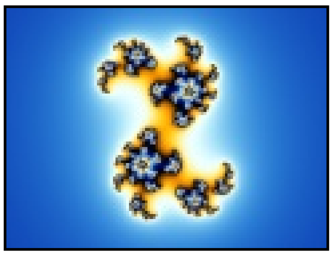
See Also

About classes

Standard fractal formulas

Standard classes

Julia



Julia sets are closely related to the well-known Mandelbrot set. In fact, the Mandelbrot set is a map of Julia sets. For each point in the Mandelbrot set, there exists a unique Julia set.

Use the Switch feature to select a Julia set by moving the mouse cursor over a Mandelbrot fractal. The most interesting Julia sets are found at points close to the edge, where the colors change quickly.

Julia sets are strictly self-similar and less complex than the Mandelbrot set. Still, they can be strikingly beautiful, and they are certainly very interesting to explore.

Julia is available as a fractal formula in Standard.ufm and as a fractal formula class in Standard.ulb.

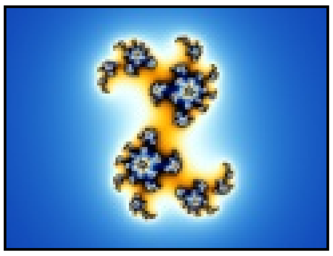
The following parameters are provided:

Julia seed	This parameter specifies the point in the Mandelbrot set that corresponds to the current Julia set. It defines the shape and behavior of the Julia set. Use the Switch feature to select good values.
Power	Specifies the exponent. The default value is (2, 0), resulting in the classic equation. **$z = z^2 + c$** Try (3, 0) and (4, 0) and so on to increase the symmetry order. Non-integer values for the real part of the exponent or non-zero values for the imaginary part will distort the fractal.
Bailout value	Specifies the magnitude of z that will cause the formula to stop iterating. To obtain "true" Julia sets, this should be set to 4 or larger. Larger values tend to smooth the outside areas. Some coloring algorithms require specific bail-out values for good results.

Note: The Julia formula is also available as a more efficient built-in formula with fewer options. See Julia (Built-in).

See Also
Lambda (Julia, Mandelbrot)
Julia sets
Standard formulas

Julia (Built-in)



This is a built-in and faster version of the Julia formula. Julia sets are closely related to the well-known Mandelbrot set. In fact, the Mandelbrot set is a map of Julia sets. For each point in the Mandelbrot set, there exists a unique Julia set.

Use the Switch feature to select a Julia set by moving the mouse cursor over a Mandelbrot fractal. The most interesting Julia sets are found at points close to the edge, where the colors change quickly.

Julia sets are strictly self-similar and less complex than the Mandelbrot set. Still, they can be strikingly beautiful, and they are certainly very interesting to explore.

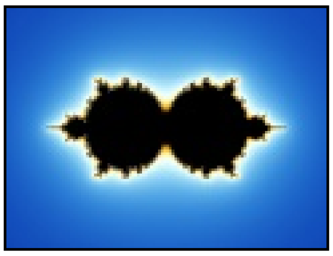
The formula provides the following parameters:

Julia seed	This parameter specifies the point in the Mandelbrot set that corresponds to the current Julia set. It defines the shape and behavior of the Julia set. Use the Switch feature to select good values.
Bailout value	Specifies the magnitude of z that will cause the formula to stop iterating. To obtain "true" Julia sets, this should be set to 4 or larger. Larger values tend to smooth the outside areas. Some coloring algorithms require specific bail-out values for good results.

Note: The Julia formula is also available as a normal formula that is less efficient, but offers more options. See Julia.

See Also
Julia sets
Standard formulas

Lambda (Julia, Mandelbrot)



The Lambda formula is an alternative version of the equation for Julia fractals. While it is capable of creating the same Julia sets, the corresponding Mandelbrot version looks different.

Both Mandelbrot and Julia versions of the Lambda fractal are available as fractal formulas in Standard.ufm and as fractal formula classes in Standard.ulb.

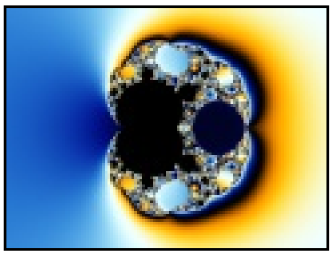
Because the Mandelbrot version is a map of Julia sets, this allows you to find Julia sets with the Switch feature in a different way than with the usual Mandelbrot set. It is easier to find good spirals and other interesting Julia sets.

The formulas provide the following parameters:

Start Value (Mandelbrot only)	For the standard Lambda Mandelbrot set, this should be set to (0.5, 0). Other values create distorted shapes that can be interesting, but they are usually not as well-formed as the standard set. For well-formed sets, the real value should be set to 1 divided by the real value of the exponent. For example, use (0.25, 0) if Exponent is set to (4, 0).
Julia Seed (Julia only)	This parameter specifies the point in the Mandelbrot version that corresponds to the current Julia set. It defines the shape and behavior of the Julia set. Use the Switch feature to select good values.
Exponent	Specifies the exponent. The default value is (2, 0), resulting in the classic equation. **c * z * (1 - z)** Try (3, 0) and (4, 0) and so on to increase the complexity of the fractal. Non-integer values for the real part of the exponent will interpolate between these well-formed sets. If the imaginary part is not zero, the fractal will be further distorted.
Bailout	Specifies the magnitude of z that will cause the formula to stop iterating. To obtain well-formed fractals, this should be set to 4 or larger. Larger values tend to smooth the outside areas. Some coloring algorithms require specific bail-out values for good results.

See Also
Julia sets
Standard formulas

Magnet 1 and 2 (Julia, Mandelbrot)



The magnetic fractal types are created by a formula that models the way magnets behave under high temperatures. This leads to fractal pictures that are similar to the classic Mandelbrot and Julia sets, but with more complex patterns and numerous Mandelbrot set miniatures.

There are two common magnetic fractal types. Type 2 is more complex than type 1. Both types are available as Mandelbrot and Julia versions, so you can use the Mandelbrot version as a map to switch to the corresponding Julia fractals.

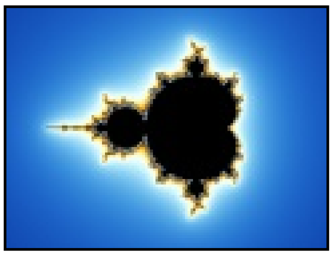
Both Mandelbrot and Julia versions of the Magnet fractals are available as fractal formulas in Standard.ufm and as fractal formula classes in Standard.ulb.

The formulas provide the following parameters:

Perturbation (Mandelbrot only)	For the standard fractals, this should be set to (0, 0). Other values create distorted shapes that can be interesting, but they are usually not as well-formed.
Parameter (Julia only)	This parameter specifies the point in the Mandelbrot version that corresponds to the current Julia set. It defines the shape and behavior of the Julia set. Use the Switch feature to select good values.
Bailout value	Specifies the magnitude of z that will cause the formula to stop iterating. Use a value above 30 to obtain well-formed fractals.
Convergent bailout value	Specifies the minimum difference between subsequent z values at which the formula will stop iterating. Smaller values give more accurate results at the expense of calculation time.

See Also
Standard formulas

Mandelbrot



The Mandelbrot set is the most well-known fractal type. Although it is calculated by a simple formula, it is incredibly complex. As you zoom in, more and more ever-changing detail becomes visible, such as little "baby" Mandelbrot sets and all kinds of spirals.

Because the Mandelbrot set lends itself well to basic zooming and exploring, it is a good starting point if you are new to fractals. It is available as a fractal formula in Standard.ufm and as a fractal formula class in Standard.ulb.

The formula provides the following parameters:

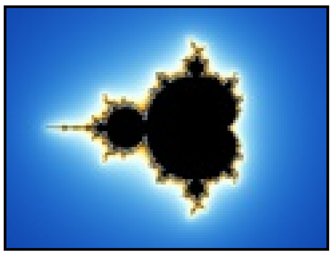
Starting point	For the standard Mandelbrot set, this should be set to (0, 0). Other values create distorted shapes that can be interesting, but they are usually not as well-formed as the standard set. Try (0, -0.6), for example.
Power	Specifies the exponent. The default value is (2, 0), resulting in the classic equation. $\mathbf{z = z^2 + c}$ Try (3, 0) and (4, 0) and so on to increase the number of main "buds". Non-integer values for the real part of the exponent will interpolate between these well-formed sets. If the imaginary part is not zero, the fractal will be further distorted.
Bailout value	Specifies the magnitude of z that will cause the formula to stop iterating. To obtain the "true" Mandelbrot set, this should be set to 4 or larger. Larger values tend to smooth the outside areas. With the Basic coloring algorithm and the Color Density set to 4, try the bail-out values 4 and then 16 to see the difference. Some coloring algorithms require specific bail-out values for good results.

Notes

- The Mandelbrot set is also available as a more efficient built-in formula with fewer options. See Mandelbrot (Built-in).
- The Mandelbrot set also acts as a map of Julia sets. Use Switch mode to switch to related Julia sets.

See Also
The Mandelbrot set
Standard formulas

Mandelbrot (Built-in)



This is a built-in and faster version of the standard Mandelbrot set: the most well-known fractal type. Although it is calculated by a simple formula, it is incredibly complex. As you zoom in, more and more ever-changing detail becomes visible, such as little "baby" Mandelbrot sets and all kinds of spirals.

Because the Mandelbrot set lends itself well to basic zooming and exploring, it is a good starting point if you are new to fractals.

The formula provides the following parameters:

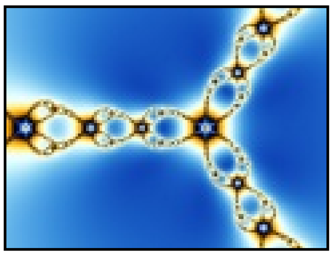
Starting point	For the standard Mandelbrot set, this should be set to (0, 0). Other values create distorted shapes that can be interesting, but they are usually not as well-formed as the standard set. Try (0, -0.6), for example.
Bailout value	Specifies the magnitude of z that will cause the formula to stop iterating. To obtain the "true" Mandelbrot set, this should be set to 4 or larger. Larger values tend to smooth the outside areas. With the Basic coloring algorithm and the Color Density set to 4, try the bail-out values 4 and then 16 to see the difference. Some coloring algorithms require specific bail-out values for good results.

Notes

- There is also a version of the Mandelbrot set as a normal formula (not built-in). Although it is less efficient, it offers more options, and it is better at handling very large bail-out values. See Mandelbrot.
- The Mandelbrot set also acts as a map of Julia sets. Use Switch mode to switch to related Julia sets.

See Also
The Mandelbrot set
Standard formulas

Newton



The Newton fractal is generated by Newton's method for solving polynomial equations. Different equations are available by changing the parameters. It is available as a fractal formula in Standard.ufm and as a fractal formula class in Standard.ulb.

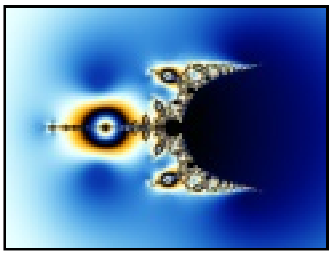
This is a simple and attractive fractal type. Newton fractals are strictly self-similar, so they are not very interesting zooming subjects. Instead, try a few different coloring algorithms to decorate them in various ways.

The formula provides the following parameters:

Exponent	Specifies the exponent of the equation to be solved. The default value is (3, 0), resulting in the equation: **z^3 + Root** Try (4, 0), (5, 0) and so on to increase the symmetry order. Non-integer values for the real part of the exponent will interpolate between these. If the imaginary part is not zero, the fractal will be further distorted.
Root	Specifies the root of the equation. This tends to rotate and magnify the fractal.
Bailout value	Specifies the minimum difference between subsequent z values at which the formula will stop iterating. Smaller values give more accurate results at the expense of calculation time.

See Also
Nova (Julia)
Standard formulas

Nova (Julia, Mandelbrot)



The Nova fractal is a modified Newton fractal. The Julia version can be used as a normal Newton fractal, but there are all kinds of other possibilities with intriguing spirals.

Both Mandelbrot and Julia versions of the Nova fractal are available as fractal formulas in Standard.ufm and as fractal formula classes in Standard.ulb.

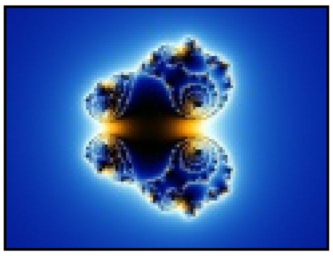
Use the Nova (Mandelbrot) formula to switch to interesting Nova (Julia) sets. The standard Newton fractals can be found in the empty circle to the right.

The formulas provide the following parameters:

Start Value (Mandelbrot only)	For well-formed fractals, this should be set to (1, 0). Other values create distorted shapes that can be interesting, but they are usually not as well-formed.
Julia Seed (Julia only)	This parameter specifies the point in the Mandelbrot version that corresponds to the current Julia version. It defines the shape and behavior of the fractal. Use the Switch feature to select good values.
Exponent	Increase the real value of the exponent to create more complex fractals. Non-integer real values and non-zero imaginary values create irregular fractals.
Bailout	Specifies the magnitude of z at which the formula will stop iterating. Since z converges to a fixed value, smaller values will give more detailed images.
Relaxation	This can be used to influence the convergence of the fractal. Changing this parameter will twist and transform the fractal.

See Also
Standard formulas

Phoenix (Julia, Mandelbrot)



The Phoenix fractal is a modification of the classic Mandelbrot and Julia sets. The Phoenix (Julia) type is particularly interesting, with beautiful shapes and lots of spirals.

Use the Phoenix (Mandelbrot) formula to switch to interesting Phoenix (Julia) sets. Both versions are available as fractal formulas in Standard.ufm and as fractal formula classes in Standard.ulb.

The formulas provide the following parameters:

Start Value (Mandelbrot only)	For well-formed fractals, this should be set to (0, 0). Other values create distorted shapes that can be interesting, but they are usually not as well-formed.
Julia Seed (Julia only)	This parameter specifies the point in the Mandelbrot version that corresponds to the current Julia version. It defines the shape and behavior of the fractal. Use the Switch feature to select good values.
Exponent 1	Increase the real value of the exponent to create more complex fractals with more symmetry. Non-integer real values and non-zero imaginary values create irregular fractals.
Exponent 2	By default, this is set to (0, 0). Use other values to create more complex, twisted fractals.
Distortion	Sets how strong the effect of the previous iteration is upon the current iteration. Set this to (0, 0) to obtain standard Mandelbrot and Julia sets.
Bailout	Specifies the magnitude of z at which the formula will stop iterating. Higher values will give smoother images with more detail.

See Also
Standard formulas

Pixel

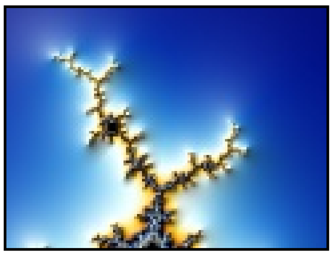
The Pixel fractal formula is a simple stub formula that marks all pixels in the layer as outside. This gives exclusive control over the layer to the coloring algorithm selected on the Outside tab of the Layer Properties tool window.

Combined with a direct coloring algorithm such as the standard Image coloring algorithm, this makes it possible to import images, as shown here. See Using images for more information.

Pixel is available as a fractal formula in Standard.ufm and as a fractal formula class in Standard.ulb.

See Also
Standard formulas
Image coloring algorithm
Using images

Slope (Julia, Mandelbrot, Newton)



The Slope formulas are modifications of the classic Mandelbrot, Julia, and Newton fractals that can create various 3D lighting effects. They are available as fractal formulas in Standard.ufm and as a fractal formula class in Standard.ulb.

In class form, Slope uses a class parameter to select which fractal formula to use, so it is not just limited to Mandelbrot, Julia or Newton fractal types. See Example 1 - Formula classes for more information.

Slope should be combined with the Lighting coloring algorithm. For each pixel, the Slope formula calculates a "height" value that is passed to Lighting, which performes the final lighting calculations.

For best results, use a black-to-white gradient such as **Lighting** in Standard.ugr. This will create a grayscale image with highlights and shadows. You can then combine this with other layers to add colors while retaining the 3D effect. For the merge mode of the layer with the Slope formula, try Soft Light or Hard Light.

The following parameters are provided:

Fractal Formula	Selects the fractal formula to use for the lighting effect. (Class version of Slope only.)
Orbit Separation	To determine proper lighting for a particular point, the Slope formulas test two orbits that are close together. This parameter specifies how close they should be. Smaller values give better results, especially for zoomed-in images. Avoid to use values that are too small for the current precision range.
Height Value	Specifies how the apparent height of each pixel will be calculated. Smooth images can be obtained with **potential** and **distance estimator**. The other options will produce images with sharper edges.
Height Transfer	This function will be applied to the height value before calculating the slope. It can be used to reduce (**log**) or exaggerate (**exp**) certain ranges of height values. The default **linear** option will not change the height value.
Height Pre-Scale	Scales the height value before it is processed by the transfer function.
Height Post-Scale	Scales the height value further after it has been processed by the transfer function. When zooming in, you should reduce this to make sure the highlights and shadows do not become too large.
Every Iteration	If selected, the height value is calculated every iteration, which is much slower. This is only necessary if you are combining the Slope formula with a coloring algorithm that processes every iteration, such as Orbit Traps. The normal Lighting algorithm does not need this option.

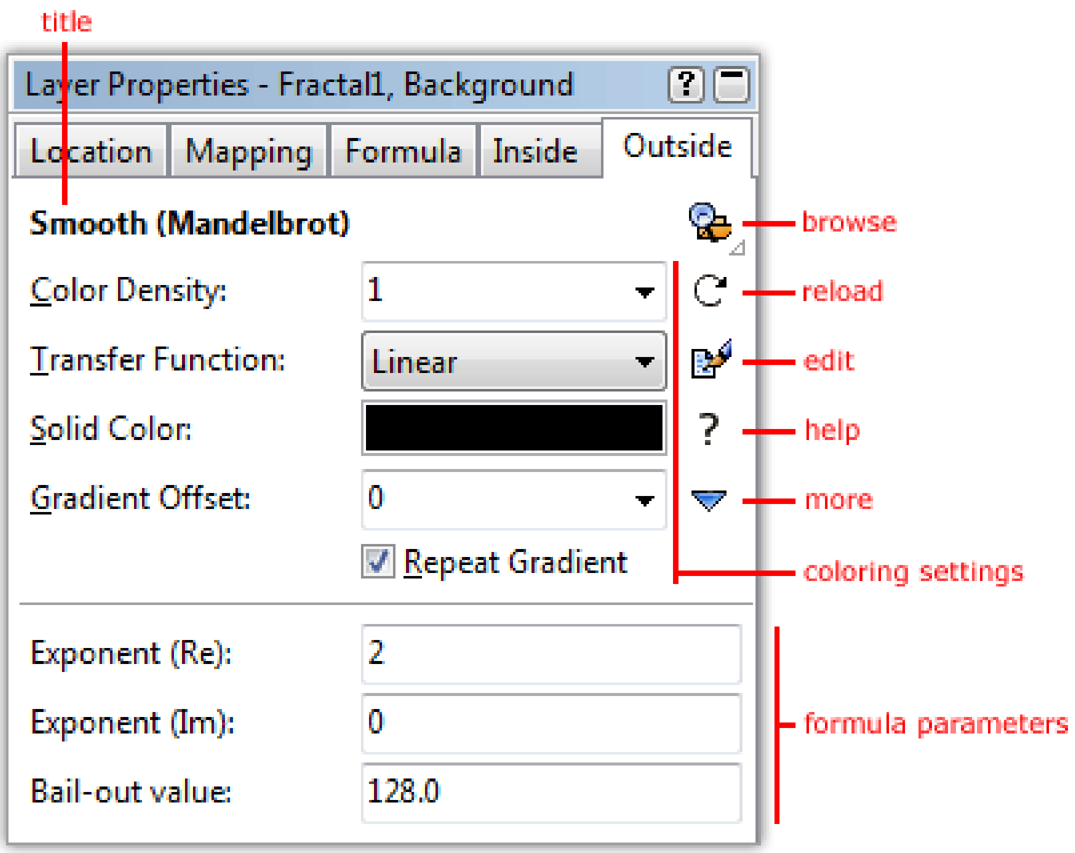
For the non-class version of the Slope fractal formulas, the other parameters are described in the topics for the regular (non-Slope) formulas.

See Also
Embossed (Julia, Mandelbrot, Newton)
Standard formulas

Coloring algorithms

Coloring algorithms define how fractals are colored. The fractal formula creates the basic shape of the fractal, and coloring algorithms provide ways to color that shape. This gives you the flexibility to freely combine coloring algorithms with any fractal formula.

Coloring algorithms are managed in the Inside and Outside tabs of the Layer Properties tool window.



Layer Properties - Fractal1, Background

Location | Mapping | Formula | Inside | Outside

Smooth (Mandelbrot)

Color Density: 1

Transfer Function: Linear

Solid Color:

Gradient Offset: 0

☑ Repeat Gradient

Exponent (Re): 2

Exponent (Im): 0

Bail-out value: 128.0

- At the top, the **title** of the coloring algorithm is shown. Hold the mouse cursor over the title to see the entry identifier and the file name of the coloring algorithm.
- The **Browse** button opens a modal browser to select another coloring algorithm.
- The **Reload** button reloads the coloring algorithm from disk and recalculates the layer.
- The **Edit** button opens the coloring algorithm in the formula editor.
- The **Help** button opens the help file for the coloring algorithm, if one exists.
- The **More** button shows a menu with additional commands.
- The **coloring settings** specify how the information from the coloring algorithm must be interpreted to color the fractal. See Coloring settings.
- The **formula parameters** are additional parameters specific to the selected coloring algorithm. See Formula parameters.

Next: Inside and outside

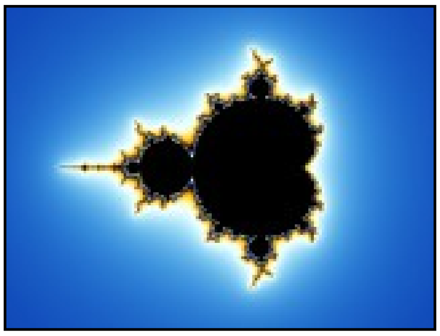
See Also
Quick Start Tutorial
Using images
Standard coloring algorithms
Fractal formulas
Transformations

Inside and outside

To calculate a pixel in a fractal, Ultra Fractal iterates the fractal formula selected in the Formula tab of the Layer Properties tool window. The formula is executed multiple times, each time using the result from the previous calculation as input.

The formula is iterated until the maximum iteration count (set in the Formula tab) is reached, or until the bail-out condition (specified by the fractal formula) is met. If the bail-out condition is met, the pixel is colored as an outside pixel. Otherwise, it is colored as an inside pixel.

Most classic fractal types, such as the Mandelbrot set, are actually a set of points. A pixel can either be inside or outside the set. If a pixel is inside, it belongs to the Mandelbrot set, for example.



In this image of the Mandelbrot set, the inside area is black. The outside area is colored according to the number of iterations required to meet the bail-out condition.

By iterating the fractal formula, Ultra Fractal decides whether a pixel is inside or outside the set. The pixels that are inside are colored according to the settings in the Inside tab of the Layer Properties tool window. The pixels that are outside are colored according to the settings in the Outside tab.

The Inside and Outside tabs are identical and provide the same options and settings. Since the outside area is usually the most interesting area, you will probably use the Outside tab more often.

Notes

- Some coloring algorithms can only be used in the Inside tab, or only in the Outside tab.
- Some fractal formulas can change the meaning of inside and outside areas. For example, they can be intended to work together with a special outside coloring algorithm, and ignore the settings for inside areas. This is usually noted in the comments at the top of the formula.

Next: Working with coloring algorithms

See Also
Maximum iterations
Coloring algorithms

Working with coloring algorithms

You work with coloring algorithms in the Inside and Outside tabs of the Layer Properties tool window. These tabs select the inside and outside coloring algorithms and contain additional coloring settings.

Coloring algorithms are stored in coloring algorithm files (*.ucl). Each file can contain multiple coloring algorithms.

To select a coloring algorithm, click the **Browse** button. This opens a modal browser that shows the coloring algorithm files on your computer and the coloring algorithms that they contain. Double-click on a coloring algorithm to select it.

Hold down the Browse button to open a menu with coloring algorithms presets. See Presets.

Some coloring algorithms contain additional help. Click the **Help** button to open it.

Click the **More** button to access commands to **copy** and **paste** the settings and parameters on the Inside or Outside tab, and to **reset** all parameters to the default values.

Coloring algorithms interpret the calculations performed by the fractal formula selected in the Formula tab and visualize parts of these calculations. Each coloring algorithm uses the information from the calculations in a different way.

Coloring algorithms do not directly calculate colors (except for direct coloring algorithms). Instead, they produce a floating-point **index value** that is converted to a color by the gradient.

Typically, the index value 0 produces the left-most color in the gradient, and 0.5 produces the color in the middle. The index value usually wraps around, so 1 produces the left-most color again. The coloring settings in the Inside and Outside tabs can be used to tweak this.

Next: Using images

See Also
Coloring algorithms
Standard coloring algorithms
How gradients work

Using images

Image import in Ultra Fractal works via image parameters in formulas, typically coloring algorithms. This lets you do much more with imported images than just adding them as a separate layer. However, it also might make it a little less obvious how to use them.

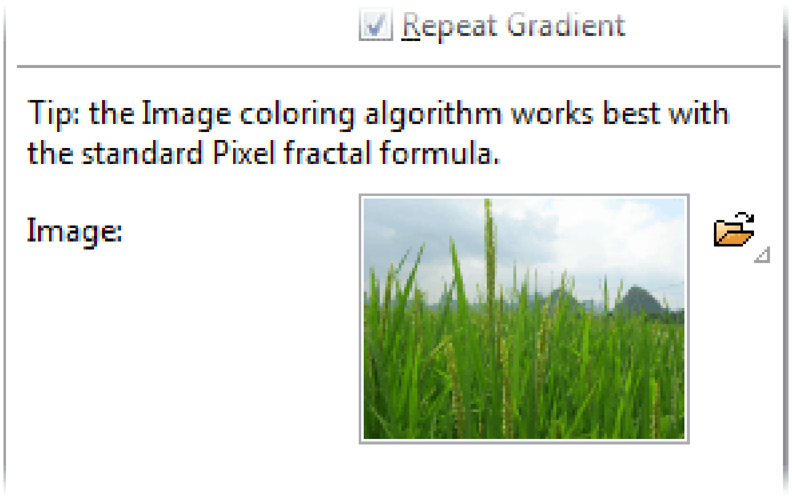
As a starting point, let's load an image such that it fills the entire layer.

- First, use the browser to open the Default parameter set in Examples.upr.
- Click the **Browse** button in the Formula tab of the Layer Properties tool window to select the Pixel formula in Standard.ufm.
- Go to the Outside tab in the Layer Properties tool window, and click the **Browse** button here to select the Image coloring algorithm in Standard.ucl.

The Image coloring algorithm contains a single image parameter. It fills the entire layer with the selected image. Currently, no image is selected, so the image parameter shows up as a rectangular box with the text *(Empty)*.

Click the **Open** button to select an image on your computer, for example the Rice.jpg image that comes with Ultra Fractal. Hold down the Open button to open a menu with the option to clear the image.

The image parameter now shows a preview of the selected image:



Tip: the Image coloring algorithm works best with the standard Pixel fractal formula.

Image:

In the fractal window, the image is now shown, but it is moved slightly to the right. This occurs because the location of the layer is still set to the default Mandelbrot location.

On the Location tab of the Layer Properties tool window, click the Reset Location button to reset the location to the default for the currently loaded Pixel formula.

Next: Combining fractals with images

See Also
Coloring algorithms

Image coloring algorithm

Combining fractals with images

Tip: This example assumes that you are familiar with formulas that use classes, as explained in the Classes chapter.

As explained in Using images, you can do more with image import in Ultra Fractal than just adding images as a separate layer. Let's try a different way of importing images.

- Again, use the browser to open the Default parameter set in Examples.upr.
- This time, we're going to use a Julia fractal. Click the **Browse** button in the Formula tab of the Layer Properties tool window to select Julia in Standard.ufm.
- To make the fractal more interesting, locate the **Julia seed** parameter and enter **-0.6** for the (Re) part, and **0.7** for the (Im) part. (Tip: Normally you would use Switch mode to find values such as these.)
- On the Location tab of the Layer Properties tool window, click the Reset Location button to reset the location to the default for the currently loaded Julia formula.
- Go to the Outside tab in the Layer Properties tool window, and click the **Browse** button here to select Generic Coloring (Direct) in Standard.ucl.

The Generic Coloring (Direct) coloring algorithm contains the Direct Orbit Traps class by default, which is perfect for our purpose.

- Locate the **Color Trap** class parameter (currently set to Trap Shape Wrapper) and click the Browse button next to it. Navigate to the common.ulb file in the Public folder and select the **Image Trap** class.
- Observe that the Image Trap class contains an image parameter, which is currently empty. Click the **Open** button next to it, and select the *Ultra Fractal.png* image that comes with Ultra Fractal.

The fractal now fills itself with the *Ultra Fractal* text from this image, but the image is rotated and transformed according to the structure of the fractal! Try zooming in to see that the text is repeated infinitely. It's a true fractal, but using texture from an image that you choose. This works very well if the image contains transparent areas, which is possible with the PNG image format.

This is just a simple example, but the parameters for the Direct Orbit Traps algorithm let you take this much further, so feel free to experiment. Also, in the public formula library, you will find more coloring algorithms that let you use images to color the fractal structure.

Notes

- Ultra Fractal imports JPEG, PNG, and BMP images. Only 24-bit or 32-bit BMP images are supported. Interlaced PNG files are not supported. The alpha channel from PNG files or 32-bit BMP files will be treated as transparency information for the image.
- When selecting an image, by default Ultra Fractal starts in the Images folder inside the main Ultra Fractal documents folder. It is recommended to keep the images that you use with your fractals here. You can change the location of the Images folder in the Folders tab of the Options dialog.
- When saving fractals or parameter sets, Ultra Fractal stores only a reference to the image used. When you re-open the fractal or parameter set, the Images folder is automatically searched for the image that is needed. Only if the image cannot be found, Ultra Fractal will ask you to locate the image.

- You can write your own formulas that use imported images. See Image parameters.

Next: Coloring settings

See Also
Coloring algorithms
Example 2 - Orbit trap classes

Coloring settings

The Inside and Outside tabs of the Layer Properties tool window select the inside and outside coloring algorithms. They also contain additional coloring settings.

These coloring settings specify how the **index value** returned by the selected coloring algorithm is interpreted by the gradient.

Color Density	Specifies how quickly the colors in the gradient follow each other. Values larger than 1 increase the color density. Values below 1 decrease the color density. The color density must be larger than 0. The index value is multiplied by the color density.
Transfer Function	Selects a transfer function that translates the index value, multiplied by the color density value, to an entry in the gradient. • **None** returns the solid color, ignoring the coloring algorithm. • **Linear** directly returns the index value. • **Sqr** squares the index value. When the index value increases, the color density appears to increase as well. • **Sqrt** returns the square root of the index value. This decreases the apparent color density as the index value increases. • **Cube** cubes the index value. The color density increases faster than when using Sqr. • **CubeRoot** returns the cubed root of the index value. The color density decreases faster than when using Sqrt. • **Log** returns the natural logarithm of the index value. The color density decreases even faster than when using CubeRoot. • **Exp** calculates e^{index}. The color density increases even faster than when using Cube. • **Sin** returns the sine of the index value. The result never exceeds -1...1 and it repeats itself when the index value increases. • **ArcTan** returns the inverse tangent of the index value. The result approaches ½ pi when the index value becomes very high
Solid Color	Specifies the solid color, which can be used for special purposes. See Solid color.
Gradient Offset	Specifies an optional offset in the gradient. This value is added to the index value after applying the Transfer function. Since the gradient contains 400 entries, the offset value can range from 0 to 399. You can achieve the same effect by rotating the gradient, but the offset can be specified for Inside and Outside coloring algorithms separately.

Repeat Gradient	Specifies if the gradient must be repeated. When checked, the index value will wrap around when it reaches 1. So, index values of 0, 1, 2, and 3 all map to the same gradient index. Otherwise, the index value is limited to 1, so all colors in the gradient are only used once.

Next: Solid color

See Also
Coloring algorithms
Working with coloring algorithms
Gradients

Solid color

In the Inside and Outside tabs of the Layer Properties tool window, you can specify a solid color. The solid color can be used by coloring algorithms for special purposes.

To change the solid color, click on the **Solid Color** swatch in the Inside or Outside tab. By default, it is set to black, but you can choose any color. You can also change the opacity. By setting the opacity to 0, the solid color and thus the areas colored with the solid color will become transparent, so the lower layers will become visible.

By setting the **Transfer function** to **None**, the entire inside or outside area is filled with the solid color. This is useful if you do not want to use a coloring algorithm for that area. If you use a transparent solid color, the area will become transparent.

For example, you can use this if you want to color the inside area with a different gradient. Duplicate the layer and make the inside solid color in the top layer transparent. Now, you can change the gradient for the lower layer. Only the inside area of the lower layer, and the outside area of the top layer will be visible.

Notes

- If you make the solid color transparent, layer transparency will be enabled automatically. See Transparent layers.
- Setting the Transfer function to None will not disable the coloring algorithm. For maximum efficiency, make sure the None coloring algorithm is selected as well.

Next: Direct coloring algorithms

See Also
Coloring algorithms
Solid color (transformations)

Direct coloring algorithms

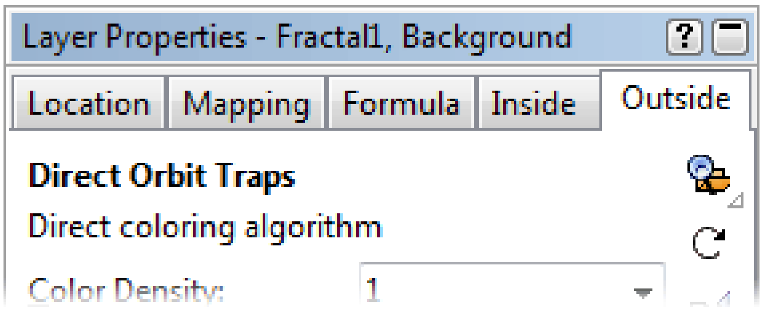
Normal coloring algorithms return an index value that is looked up in the gradient to produce a color for each pixel. This enables you to easily change the colors by editing the gradient. On the other hand, it limits the colors that can appear in the layer to the colors available in the gradient.

Unlike normal coloring algorithms, direct coloring algorithms directly return a color for each pixel. They are more powerful because they can return any desired color, and perform sophisticated merging operations internally.

Direct coloring algorithms can access the gradient and use its colors, but they are not limited to those colors. Because of this flexibility, editing the gradient will cause the layer to be recalculated. You can still use the coloring settings (such as Color Density) to change the appearance of the gradient.

Note that because the coloring algorithm decides how the gradient is used, the resulting colors in the layer may or may not be directly related to the colors in the gradient.

You can tell when a direct coloring algorithm is selected in the Inside or Outside tab because the text "Direct coloring algorithm" is visible below the title.



Direct Orbit Traps
Direct coloring algorithm
Color Density: 1

An example of a direct coloring algorithm is Direct Orbit Traps.

Next: Standard coloring algorithms

See Also
Coloring algorithms
Working with coloring algorithms

Standard coloring algorithms

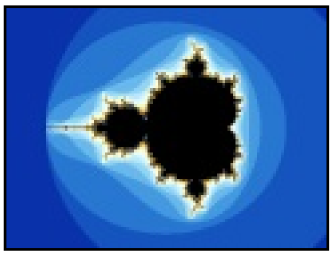
Ultra Fractal comes with a set of standard coloring algorithms. They are located in the file Standard.ucl in the Formulas folder. It contains the following coloring algorithms:

- Basic
- Binary Decomposition
- Decomposition
- Direct Orbit Traps
- Distance Estimator
- Emboss
- Exponential Smoothing
- Gaussian Integer
- Generic Coloring (Gradient, Direct)
- Gradient
- Image
- Lighting
- None
- Orbit Traps
- Smooth (Mandelbrot)
- Triangle Inequality Average

All standard coloring algorithms are also implemented as coloring classes in Standard.ulb. This makes it possible to use them with Generic Coloring or in other formulas that accept coloring classes. See About classes and Example 2 - Orbit trap classes for more information.

See Also
Standard fractal formulas
Standard transformations
Coloring algorithms
Public formulas

Basic



The Basic coloring algorithm implements four simple and classic ways of coloring the outside areas of a fractal. It is useful for reproducing fractals created with older fractal software.

Basic is available as a coloring algorithm in Standard.ucl and as a coloring class in Standard.ulb.

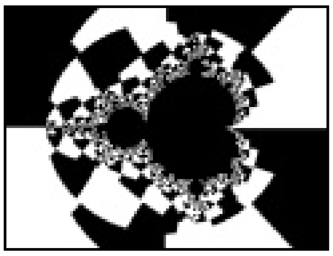
The **Coloring Type** parameter selects how the fractal should be colored:

The **Iterations** option colors a pixel according to the number of iterations that were necessary for the fractal formula to bail out (to decide that it's an outside pixel). This is the oldest method used to color fractals. It creates images with bands of solid colors. To smoothen the bands, use the Smooth (Mandelbrot) coloring algorithm.

The **Real**, **Imaginary**, and **Sum** options use the last value of z in combination with the number of iterations to color pixels. They create smooth, true-color images. Good results are usually obtained if the bail-out parameter of the fractal formula is not set too high. Try 4, for example.

See Also
Binary Decomposition
Distance Estimator
Standard coloring algorithms

Binary Decomposition



The Binary Decomposition coloring algorithm uses just two colors from the gradient. It colors fractals according to the "angle" of the last value of z from the fractal formula. This results in quite abstract and elegant images.

The two colors used are at the beginning and at the middle of the gradient.

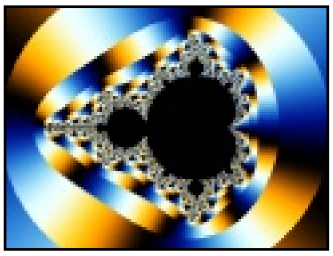
There is one parameter that selects between two different flavors of the same algorithm. Each option creates different patterns. The second option reproduces the coloring used for many fractals in the classic Beauty of Fractals book.

It often works well to combine this coloring algorithm with other layers that contain more different colors. Furthermore, low values (for example 4) for the bail-out parameter of the fractal formula usually give the best results.

Binary Decomposition is available as a coloring algorithm in Standard.ucl and as a coloring class in Standard.ulb.

See Also
Basic
Decomposition
Standard coloring algorithms

Decomposition



The Decomposition coloring algorithm colors fractals according to the "angle" of the last value of z from the fractal formula. The angle is decomposed and distributed over the full range of the gradient.

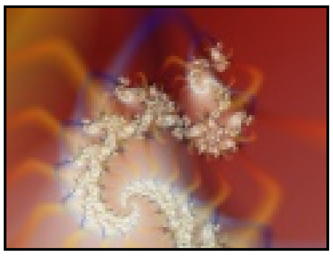
This coloring algorithm tends to create circular bands all over the image that contain all colors from the gradient. Low values (for example 4) for the bail-out parameter of the fractal formula usually give the best results.

With convergent fractal types, such as Newton or Nova, Decomposition usually does not create smoothly colored images, but it can still be interesting.

Decomposition is available as a coloring algorithm in Standard.ucl and as a coloring class in Standard.ulb.

See Also
Basic
Binary Decomposition
Standard coloring algorithms

Direct Orbit Traps



The Direct Orbit Traps coloring algorithm is a direct coloring algorithm. This means that it the resulting images are not limited to the colors in the gradient. It usually creates softly shaded, pastel-like images.

Direct Orbit Traps is available as a coloring algorithm in Standard.ucl and as a coloring class in Standard.ulb. As a coloring class, it uses several class parameters instead of normal parameters to make it easier to add new trap shapes and coloring options. See Example 2 - Orbit trap classes and Combining fractals with images for more information. You need to combine the Direct Orbit Traps class with Generic Coloring (Direct).

Direct Orbit Traps works by calculating a color for every iteration. These colors are merged together to obtain the final color for each pixel. This is like using multiple layers within a single coloring algorithm.

The colors are taken from the gradient and merged onto the background color. If you change the gradient, the layer is recalculated with the new colors. It can sometimes be difficult to predict the effect of changes to the gradient, because the gradient colors are merged with each other and with the background color.

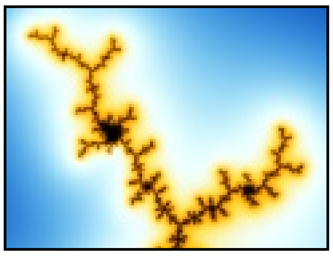
Most of the parameters are shared with Orbit Traps. There are some additional parameters that specify how the colors from each iteration are merged:

Base Color	Specifies the background color. All other colors are merged on top of the background, so the background color interacts with the colors from the gradient. The background color can also be transparent.
Trap Color Merge	Specifies the merge mode used to merge colors on top of the background. All layer merge modes are supported here. Remember to adjust the background color so it will work well with the selected merge mode. For example, use a dark background color with **Screen**, and a light background color with **Multiply**.
Additional Alpha	If set to **distance**, the opacity of the color calculated for each iteration is reduced according to the distance from the trap shape. This can create very soft and smooth images. (In the class version of Direct Orbit Traps, this parameter is grouped with Color Trap.)
Trap Merge Opacity	Sets the opacity (between 0 and 1) of the color calculated for each iteration. The opacity of the gradient is also taken into account, just like when merging layers.
Trap Merge Order	Sets the order in which traps are merged. The **bottom-up** option merges later iterations on top of the existing iterations, while the **top-down** option merges later iteration underneath the existing iterations.

Color Trap	(Class version only.) This selects the color trap class which is responsible for converting z values to colors. Normally, **Trap Shape Wrapper** is selected, which wraps a trap shape class, a trap coloring class and a gradient class to do this. However, you could also select another class, such as **Image Trap** from common.ulb which returns colors from an imported image.

See Also
Orbit Traps
Standard coloring algorithms
Example 2 - Orbit trap classes
Combining fractals with images

Distance Estimator



The Distance Estimator coloring algorithm estimates the distance between a pixel and the boundary of the fractal (for example the boundary of the Mandelbrot set). The pixel is colored accordingly.

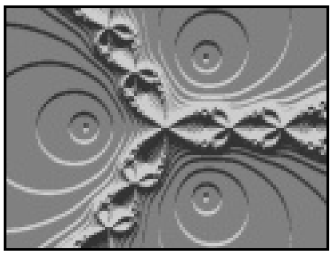
This coloring algorithm is especially good at showing the thin connecting lines and miniatures that exist everywhere in the Mandelbrot set. It works correctly for divergent fractal formulas like Mandelbrot, Julia, and Phoenix.

The **Exponent** parameter should be set to match the exponent or power of the fractal formula (this is usually also a parameter). Higher values (like 128) for the bail-out parameter of the fractal formula give the best results.

Distance Estimator is available as a coloring algorithm in Standard.ucl and as a coloring class in Standard.ulb.

See Also
Basic
Decomposition
Standard coloring algorithms

Emboss



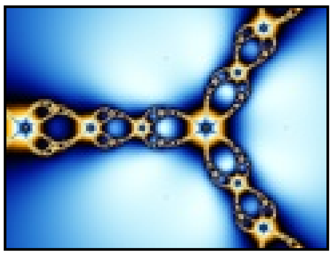
The Emboss coloring algorithm interprets results from one of the Embossed fractal formulas to create fractals with 3D contour lines. It is unlikely to give good results with other fractal formulas.

See Embossed (Julia, Mandelbrot, Newton) for more information.

Emboss is available as a coloring algorithm in Standard.ucl and as a coloring class in Standard.ulb.

See Also
Lighting
Standard coloring algorithms

Exponential Smoothing



The Exponential Smoothing coloring algorithm creates smoothly colored outside areas. It works well for both convergent and divergent fractal types, which means that it can be combined with almost any fractal formula.

Fractal formulas like Mandelbrot, Julia, and Phoenix have only divergent orbits, while types like Newton and Nova have only convergent orbits. The Magnet fractal formulas have both divergent and convergent orbits.

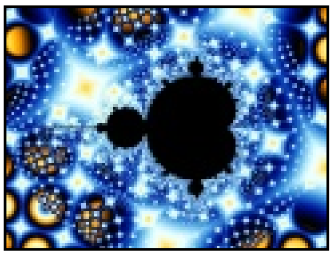
With the **Color Divergent** and **Color Convergent** parameters, you can enable coloring for divergent and convergent orbits. You should always enable at least one option. The formula runs slightly faster if you just enable the necessary options (only Magnet-like fractals require both parameters to be enabled).

The **Divergent Density** parameter can be used to tweak the color density for divergent parts of a fractal. It is only useful when both divergent and convergent orbits exist in the fractal.

Exponential Smoothing is available as a coloring algorithm in Standard.ucl and as a coloring class in Standard.ulb.

See Also
Smooth (Mandelbrot)
Standard coloring algorithms

Gaussian Integer



The Gaussian Integer coloring algorithm colors fractals according to how the calculated orbits are related to Gaussian integers.

Gaussian Integer is available as a coloring algorithm in Standard.ucl, or as a trap shape class in Standard.ulb that can be used together with the Orbit Traps or Direct Orbit Traps coloring classes. See Example 2 - Orbit trap classes.

Gaussian integers are complex numbers normalized to integer values. This coloring algorithm examines the values of z calculated by the fractal formula, and tests them against nearby Gaussian integers.

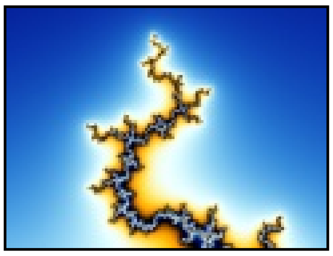
The resulting images are richly textured, containing many circles, dots, and stars. By tweaking the provided parameters, many variations are possible.

The following parameters are available:

Integer Type	Specifies the rounding method to use to find the nearest Gaussian integer. The **round(z)** option usually gives smoother images than the other options.
Color By	Selects how the color of each pixel is determined. For example, it can be colored by the minimum distance from a value of z to the nearest Gaussian integer.
Normalization	Chooses between several ways of normalizing the distance to the nearest Gaussian integer. If you select **factor** or **f(z)**, an additional parameter will appear that specifies the normalization factor or function to use.
Randomize	If checked, a small randomization factor is added to each value of z before examining its behavior. Additional parameters will appear to specify the amount of randomization and a seed value. Each seed value will give different randomization patterns. In the trap shape class version, this option is not available. However, you can achieve the same effect by selecting the Randomize transformation class from Standard.ulb for the Trap Position parameter in the Orbit Traps class.

See Also
Example 2 - Orbit trap classes
Orbit Traps
Standard coloring algorithms

Generic Coloring (Gradient, Direct)



Generic Coloring is a skeleton coloring algorithm that lets a coloring class do the actual work. All coloring algorithms in Standard.ucl are also implemented as coloring classes. You can select any of these as the coloring class to be used together with Generic Coloring.

There are two versions of Generic Coloring: a gradient version and a direct version. Generic Coloring (Gradient) is a normal coloring algorithm and works only with gradient coloring classes which return an index into the gradient.

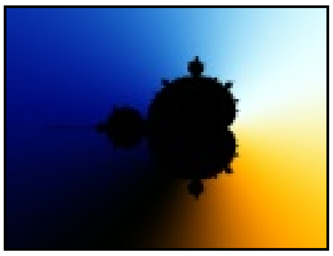
Generic Coloring (Direct) is a direct coloring algorithm and works with coloring classes that directly return a color. It also works with gradient coloring classes, but in that case changes to the gradient trigger an unnecessary recalculation.

The **Coloring Algorithm** class parameter sets the coloring class to be used. By default, the Smooth (Mandelbrot) class from Standard.ulb is selected for Generic Coloring (Gradient), while Generic Coloring (Direct) defaults to the Direct Orbit Traps class.

An example of using Generic Coloring is given in Example 2 - Orbit trap classes.

See Also
About classes
Standard coloring algorithms
Standard classes

Gradient



The Gradient coloring algorithm ignores the information from the fractal formula and fills the fractal with all colors from the gradient.

To obtain a completely filled image, select the Gradient coloring algorithm in both the Inside and the Outside tabs of the Layer Properties tool window. This ensures that all pixels are colored in the same way.

The Gradient Type parameter selects how the gradient should be displayed: linear (from left to right), radially, or as a cone. Use zooming and panning to reposition the gradient as desired.

This coloring algorithm is handy for creating special effects with multi-layer images. You can, for example, use it in a mask together with a suitable transparent gradient to show only selected portions in a regular pattern of the layer that is masked.

Gradient is available as a coloring algorithm in Standard.ucl and as a coloring class in Standard.ulb.

See Also
Standard coloring algorithms

Image

The Image coloring algorithm enables you to load an external image into a fractal. It simply displays the image in the inside or outside area of the fractal. If you want the entire layer to show the image, the Image coloring algorithm is best combined with the standard Pixel fractal formula.

The **Image** parameter contains the image that is displayed. See Using images for more information on how to use this parameter.

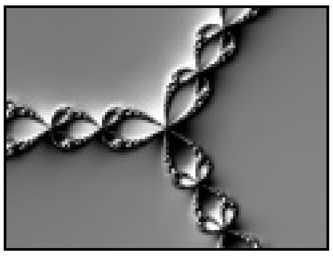
In the layer, the image is scaled such that it occupies the entire layer if the default location is set. To achieve this, load the Pixel fractal formula and click the Reset Location button on the Location tab. Of course, you can pan, zoom, and rotate at will to frame or distort the image.

When zooming in, the individual pixels in the image are interpolated using sophisticated bicubic interpolation methods. For the best results when zooming out, it is recommended to render the fractal with anti-aliasing enabled.

Image is available as a coloring algorithm in Standard.ucl and as a coloring class in Standard.ulb.

See Also
Standard coloring algorithms
Using images

Lighting



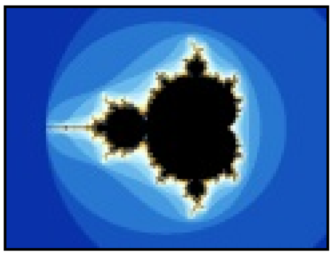
The Lighting coloring algorithm interprets results from one of the Slope fractal formulas to create fractals with 3D lighting effects. It will probably not give very good results with other fractal formulas.

See Slope (Julia, Mandelbrot, Newton) for more information.

Lighting is available as a coloring algorithm in Standard.ucl and as a coloring class in Standard.ulb.

See Also
Emboss
Standard coloring algorithms

None



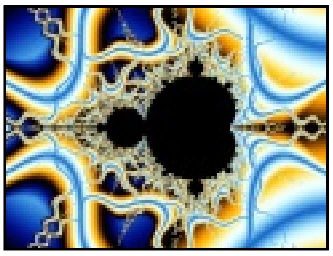
The None coloring algorithm is the simplest coloring algorithm available. It is loaded by default in Ultra Fractal when a new fractal is created.

None reproduces the standard iterations coloring algorithm found in most fractal software.

None is available as a coloring algorithm in Standard.ucl and as a coloring class (called Default) in Standard.ulb.

See Also
Basic
Standard coloring algorithms

Orbit Traps



The Orbit Traps coloring algorithm is an extremely versatile general-purpose coloring algorithm. It can be applied to almost any fractal formula with good results, on both the Inside and Outside tabs.

Orbit Traps is available as a coloring algorithm in Standard.ucl and as a coloring class in Standard.ulb. As a coloring class, it uses several class parameters instead of normal parameters to make it easier to add new trap shapes and coloring options. You need to combine the Orbit Traps class with Generic Coloring (Gradient). See Example 2 - Orbit trap classes for an example of how this works in practice.

Orbit Traps works by examining the value of z (as calculated by the fractal formula) for each iteration. It tests how close z is to a fixed shape (the orbit trap), and colors the pixel according to the closest distance, for example.

The possibilities are almost unlimited because there are so many combinations of parameters available. It is a good idea to take some time to explore the different options in here.

The following parameters are available:

Trap Position	(Class version only.) Provides options to transform the value of z before it is passed to the trap shape class. This essentially transforms the trap shape. The default Trap Position class provides many common options, but it is also possible to select any other transformation class, or even to merge multiple transformations with Transform Merge in common.ulb.
Trap Shape	Specifies the shape of the orbit trap. Some options may not look too exciting with the default settings of the other parameters, but try changing Trap Coloring and Trap Mode in that case. Not all options work equally well with all fractal types. In the class version, Trap Shape is a class parameter that accepts any trap shape class. All trap shapes from the Orbit Traps coloring algorithms are available as trap shape classes in Standard.ulb. Some trap shapes have additional parameters:
Diameter	Specifies the diameter or size of the trap. Larger values usually create decorations further away from the center of the fractal.
Order	Specifies the order for the trap, such as the number of leaves for the pinch trap shape. It depends on the trap shape how this is interpreted. Larger values usually give more complex traps.
Frequency	Specifies the frequency of ripples or waves (where applicable). Larger values create "busier" trap shapes with more frills.
Trap Transfer	(Class version only.) Provides options to transform the distance that is returned by the trap shape class. The default Trap Transfer class is quite powerful, but it is also possible to select another transfer class.

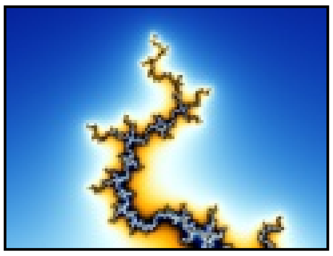
Trap Mode	Selects how the values gathered at each iteration are interpreted to produce a color. For example, the magnitude at the closest distance to the trap shape is used if Trap Coloring is set to **magnitude**, and Trap Mode to **closest**. Experiment to see which combinations work well together. Some trap modes only work well with specific trap coloring settings, for example. The **trap only** option will show the trap shape only. This is useful for learning how the other options work. In the class version, Trap Mode is a class parameter that accepts any trap mode class, and the original trap mode options are now also available as trap mode classes in Standard.ulb.
Trap Threshold	Specifies the width of the trap area, used for most trap modes. (In the non-class version, this parameter is called **Threshold**.)
Trap Coloring	The Trap Coloring parameter selects what information is gathered at each iteration. This is later filtered and combined to produce the final color. In the class version, Trap Coloring is a class parameter that accepts any trap coloring class, and the original trap coloring options are now also available as trap coloring classes in Standard.ulb.
Use Solid Color	If checked, areas outside the trap shape will be colored with the solid color (allowing them to be transparent).

The non-class version of Orbit Traps does not have the Trap Position class parameter that allows any transformation to transform the trap shape, but it does offer some transformation options:

Trap Center	Specifies the center of the trap shape. Values other than (0, 0) will distort the trap shape into the direction of the trap center. Use the eyedropper to select good values for this parameter.
Aspect Ratio	Changes the aspect ratio of the trap shape. Values larger than 1 will stretch the trap horizontally; values smaller than 1 will stretch it vertically.
Rotation	Rotates the trap shape in clockwise direction (specified in degrees).

See Also
Tutorial: Masking
Example 2 - Orbit traps classes
Direct Orbit Traps
Standard coloring algorithms

Smooth (Mandelbrot)



The Smooth (Mandelbrot) coloring algorithm creates smoothly colored outside regions with fractal formulas such as Mandelbrot and Julia.

It works with most divergent fractal formulas. For Newton and Nova fractals, use Exponential Smoothing instead.

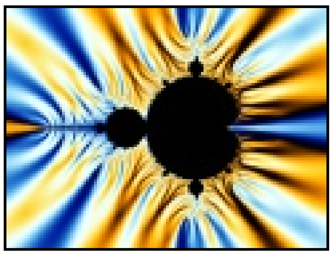
There are two parameters available: **Exponent** and **Bail-out value**. These should be set to match the corresponding parameters of the fractal formula. Otherwise, the coloring will not be perfectly smooth.

Usually, the best results are obtained when the Transfer Function in the Outside tab is set to **Log**.

Smooth (Mandelbrot) is available as a coloring algorithm in Standard.ucl and as a coloring class in Standard.ulb.

See Also
Basic
Standard coloring algorithms

Triangle Inequality Average



The Triangle Inequality Average coloring algorithm creates smoothly colored fractals with large flame-like patterns that extend from the fractal outwards.

Because it uses the same smoothing as Smooth (Mandelbrot), it only works with most divergent fractal formulas, such as Mandelbrot and Julia.

There are two parameters available: **Exponent** and **Bailout**. These should be set to match the corresponding parameters of the fractal formula. Otherwise, the coloring will not be smooth.

Use very large bail-out values for good results. The default value 1e20 (a 1 with 20 zeroes) is a good starting point. The Mandelbrot (Built-in) formula cannot handle such large values, so use the non-built-in Mandelbrot instead.

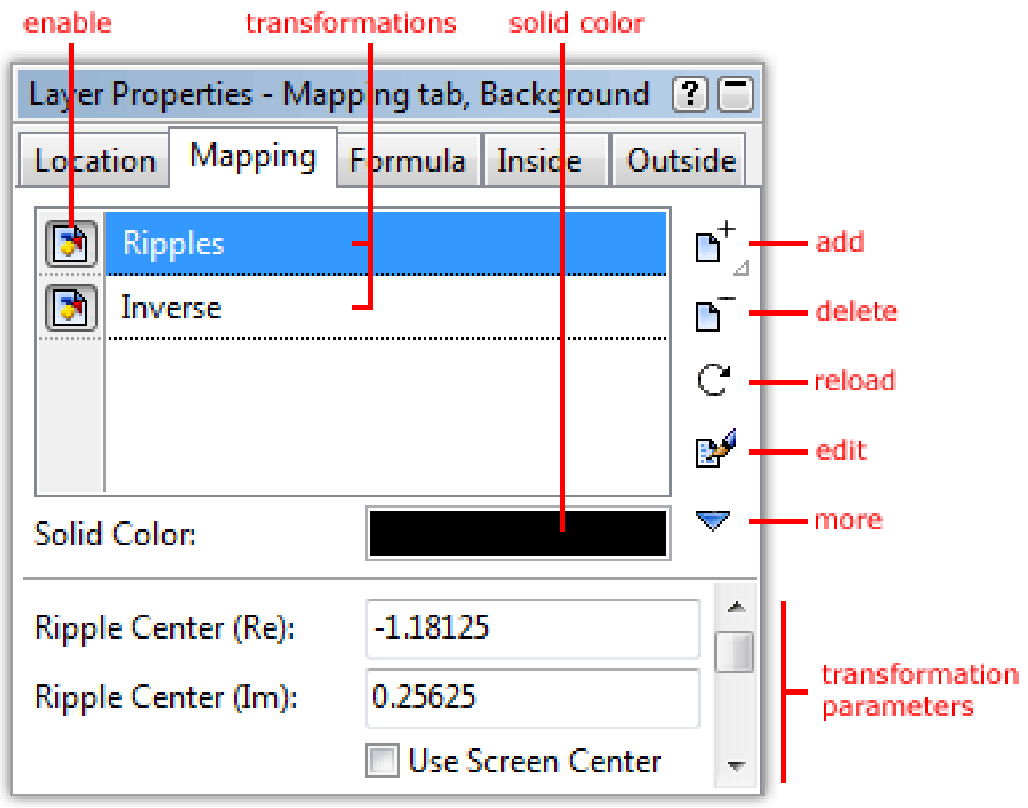
Triangle Inequality Average is available as a coloring algorithm in Standard.ucl and as a coloring class in Standard.ulb.

See Also
Orbit Traps
Standard coloring algorithms

Transformations

Transformations globally transform and warp the shape of a fractal. You can combine various transformations to create complex effects. Of course, you can also write your own transformations.

Transformations are managed in the Mapping tab of the Layer Properties tool window:



- The **Add** button opens a modal browser to select a new transformation. The transformation is then added to the list.
- The **Delete** button removes the selected transformations from the list.
- The **Reload** button reloads the selected transformation from disk and recalculates the layer.
- The **Edit** button opens the selected transformation in the formula editor.
- The **More** button shows a menu with additional commands.
- The **Enable** icon before a transformation quickly enables and disables the transformation.
- The **Solid Color** swatch specifies the solid color for the selected transformations. The solid color can be used by a transformation for special purposes. See Solid color.
- The **transformation parameters** are additional parameters specific to the selected transformations. See Formula parameters.

You can resize the transformations list by dragging the area just above the line that divides the Solid Color setting from the transformation parameters. The Reload and Edit buttons hide themselves automatically when there is not enough space. In this case, these commands can be found on the More menu.

Next: Working with transformations

See Also
Tutorial: Learning about transformations
Standard transformations

Fractal formulas
Coloring algorithms

Working with transformations

You work with transformations in the Mapping tab of the Layer Properties tool window. The Mapping tab shows a list with the transformations used by the active layer.

Transformations are stored in transformation files (*.uxf). Each file can contain multiple transformations.

To add a transformation, click the **Add** button. This opens a modal browser that shows the transformation files on your computer and the transformations that they contain. Double-click on a transformation to add it.

Hold down the Add button to open a menu with transformation presets. See Presets.

To remove the selected transformations, click the **Delete** button.

To rename the active transformation, click it again or press F2 (like in Windows Explorer).

To change the order in which transformations appear in the list, drag them up or down. See Multiple transformations.

The **Enable** icon before each transformation enables and disables it. Use it to temporarily disable a transformation so you can judge its effect, or adjust other transformations.

Some transformations contain additional help. To access it, click the **More** button, and then click Help from the menu that opens. This menu also provided commands to **copy** and **paste** the settings and parameters for the selected transformation, and to **reset** all parameters to the default values.

The bottom pane of the Mapping tab contains parameters specific to the selected transformations. These parameters work the same as the parameters for fractal formulas. See Formula parameters.

Notes

- You can select multiple transformations by holding down the Ctrl key while clicking a transformation. Hold down the Shift key while clicking to select a range of transformations. This enables you to easily edit multiple transformations together, like you can do with layers.
- If multiple layers are selected, the list in the Mapping tab only shows those transformations that all selected layers have in common.
- Right-click inside the list of transformations to open a menu with frequently used commands.

Next: Multiple transformations

See Also
Tutorial: Learning about transformations
Transformations
Standard transformations
Public formulas

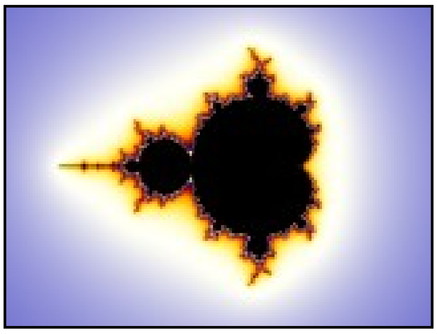
Multiple transformations

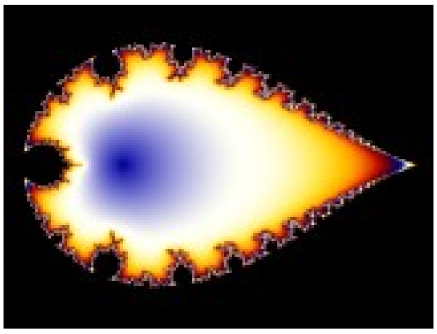
Ultra Fractal lets you combine multiple transformations to achieve more complex effects. With more than one transformation, the order in which the transformations appear in the list in the Mapping tab of the Layer Properties tool window is important.

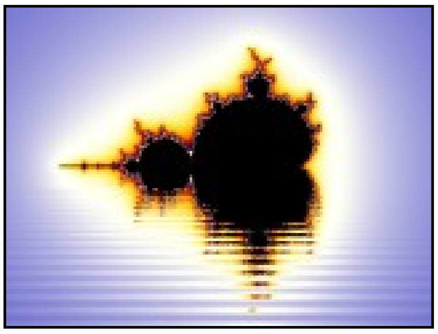
You can view a transformation as if it transforms the image of the layer as produced by the fractal formula and the coloring algorithms. (In fact, it works differently, but you can ignore that unless you are writing your own transformations.)

The transformations are then processed from the bottom of the list to the top. So, if you have two transformations, the top one works on the "image" produced by the second transformation.

Here is an example to illustrate this:

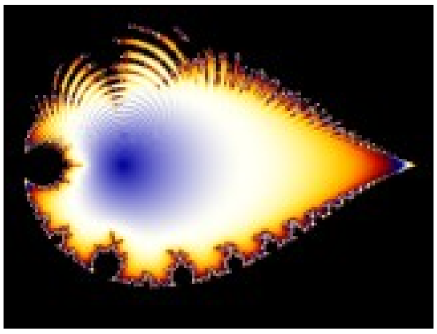

Original

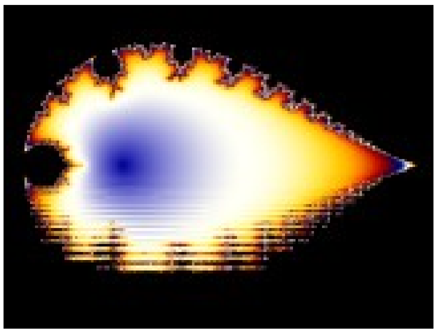

Inverse


Lake

Two variations on the original image are shown. The first uses the Inverse transformation that turns an image "inside out". The second variation uses the Lake transformation that mirrors the image horizontally and creates the illusion of water ripples.

What happens if we combine the two transformations?


First **Lake**, then **Inverse**


First **Inverse**, then **Lake**

If we put Inverse above Lake, we get the first image. If we put Lake on top instead, the second image is produced. This shows that a transformation works on the intermediate result produced by the transformations below it.

Notes

- You can freely experiment with the order of the transformations by dragging them up or down in the list.

- When you add a new transformation, it is always inserted above the selected transformation, so it works on the intermediate image produced by that transformation.

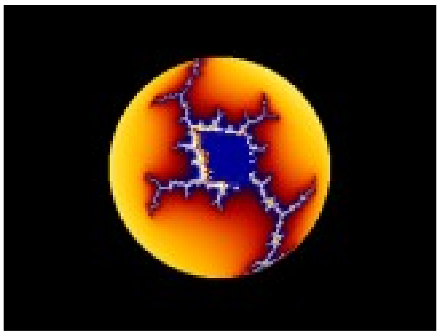
Next: Solid color

See Also
Transformations
Working with transformations

Solid color

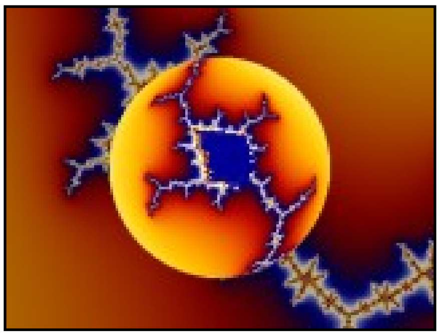
In the Mapping tab of the Layer Properties tool window, you can specify a solid color for each transformation. A transformation can use this color for special purposes.

For example, a transformation that maps a fractal onto a plane in 3D space also needs to color the area that is above or below the plane. This area is usually colored with the solid color.



This example shows a simple fractal mapped onto a sphere with the 3D Mapping transformation. The area outside the sphere is given the solid color (in this case black).

To change the solid color, click on the **Solid Color** swatch in the Mapping tab. By default, it is set to black, but you can choose any color. You can also change the opacity. By setting the opacity to 0, the solid color and thus the solid areas will become transparent, so the lower layers will become visible.



In effect, the transformation not only transforms the shape of the fractal, but also generates a mask for the layer.

Some transformations are only intended to create masks, and do not transform the pixels at all. You should use them with a transparent solid color. An example is the Clipping transformation.

Notes

- If you make a solid color transparent, layer transparency will be enabled automatically. See Transparent layers.
- The masks created with transformations always have sharp edges. For soft masking with more control over the masking shapes, use layer masks instead. See Masks.

Next: Standard transformations

See Also

Transformations

Working with transformations

Standard transformations

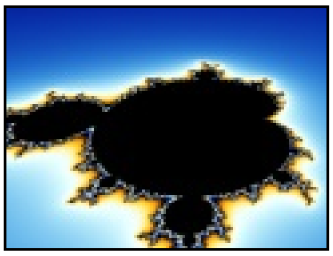
Ultra Fractal comes with a set of standard transformations. They are located in the file Standard.uxf in the Formulas folder. It contains the following transformations:

- 3D Mapping
- Aspect Ratio
- Clipping
- Generic Clipping
- Generic Transformation
- Glass Hemisphere
- Inverse
- Kaleidoscope
- Lake
- Mirror
- Ripples
- Twist

All standard transformations are also implemented as transformation or clip shape classes in Standard.ulb. This makes it possible to use them with Generic Transformation or Generic Clipping, or in other formulas that accept transformation classes. See About classes for more information.

See Also
Standard fractal formulas
Standard coloring algorithms
Standard classes
Transformations
Public formulas

3D Mapping



The 3D Mapping transformation maps a fractal onto a three-dimensional shape, such as a plane or a sphere. It is available as a transformation in Standard.uxf and as a transformation class in Standard.ulb.

Once this transformation is in effect, normal zooming and panning will just move the 3D shape around with the fractal on it. If you want to zoom into the fractal as it is mapped onto the shape, use the **Fractal Center**, **Fractal Magnification**, and **Fractal Rotation** parameters.

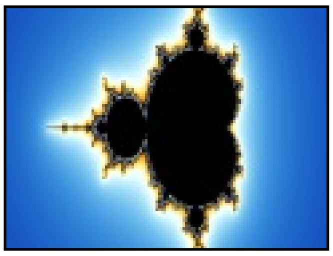
To use the rotation and translation parameters effectively, you need to understand the left-handed 3D coordinate system used by the transformation. Here, the X-axis points to the left, the Y-axis points upwards, and the Z-axis points into the screen. So, if you use a positive Z-translation, the 3D shape will appear to move away, into the screen.

The following parameters are available:

Shape	Selects the type of 3D shape that the fractal is mapped onto. In the class version of 3D Mapping, this is a class parameter that lets you select any class that implements a mapping shape.
X Rotation **Y Rotation** **Z Rotation**	Rotates the 3D shape around the X, Y, or Z axis. To predict the direction of rotation, hold up your left hand with your thumb pointing into the positive axis direction (for example, to the left for X rotation). Your (curled) fingers now show the direction of positive rotation around that axis.
X Translation **Y Translation** **Z Translation**	Moves the shape around in 3D space. You always need some positive Z translation to move the shape "into" the screen, otherwise you will be "inside" the shape and it won't be visible. With the plane shape, use a negative Y translation to look down upon it.
Fractal Center **Fractal Magnification** **Fractal Rotation**	Specifies the location of the fractal as it is mapped onto the shape. When adding the transformation, the current coordinates will be used (remember to reset the location to get a proper view of the 3D shape). The easiest way to specify a location is to copy the coordinates from the Location tab of a different fractal window that contains the same fractal formula without the 3D Mapping transformation.

See Also
Tutorial: Learning about transformations
Standard transformations

Aspect Ratio



The Aspect Ratio transformation can be used to create fractals for media with non-square pixels. You should add it to all layers of the fractal. It is available as a transformation in Standard.uxf and as a transformation class in Standard.ulb.

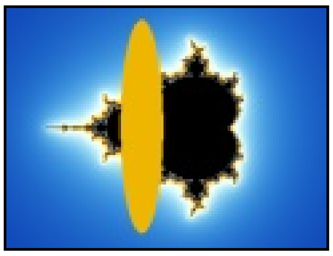
The **Aspect Ratio** parameter specifies the aspect ratio (height / width) of the final media. The final media is the computer screen or printed poster that will eventually display the fractal. If the value of the parameter is equal to the height divided by the width of the fractal window, no stretching occurs.

For example, suppose you would like to display a fractal on a 15" x 10" screen that has a resolution of 400 x 300 pixels. Here, the pixels are wider than tall. You would set the width and height of the fractal window to 400 x 300 to obtain the desired size. Then, set the **Aspect Ratio** parameter of this transformation to 0.6667 (10" / 15").

Most computer monitors and printers have square pixels. In this case, you do not need to use this transformation. If you just want to stretch the fractal, use the **Stretch** parameter in the Location tab of the Layer Properties tool window instead.

See Also
Standard transformations

Clipping



The Clipping transformation cuts a geometric shape out of a fractal. The shape is filled with a solid color. It can also be transparent, to make parts of the underlying layers visible.

Clipping is available as a transformation in Standard.uxf and as a clip shape class in Standard.ulb. In class form, it can be used together with Generic Clipping. In this case, it also shows handles in the fractal window to show where the edges of the clipping shape are.

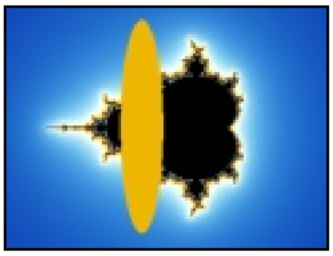
Both rectangular and circular shapes are available. You can choose to cut either the region inside or outside the shape. This makes the Clipping transformation also useful for creating frames.

The following parameters are available:

Center	Specifies the coordinates of the center of the clipping shape. Use the eyedropper (right-click and click Eyedropper) to select the center by clicking inside the fractal window.
Right Edge	Specifies the coordinates of the right edge of the clipping shape. Use the eyedropper to select this.
Top Edge	Specifies the coordinates of the top edge of the clipping shape. Use the eyedropper to select this.
Shape	Selects the type of shape to use. If you select **circle** or **square**, the Top Edge parameter is ignored.
Allow Rotation	If checked, the shape is allowed to rotate. In this case, the Right Edge parameter also defines the rotation to use.
Region	Selects whether to cut the region outside the clipping shape, or inside the clipping shape. (This parameter is not available in the Clipping class.)
Screen-Relative	If checked, all coordinates are interpreted as relative to the screen. This makes it harder to enter coordinates (because you can no longer use the eyedropper), but it preserves the location of the clipping shape relative to the screen when zooming. (This parameter is not available in the Clipping class.)

See Also
Tutorial: Learning about transformations
Standard transformations

Generic Clipping



The Generic Clipping transformation cuts a shape out of a fractal. The shape is determined by a clip shape class, and Generic Clipping allows any compatible clip shape class to be selected.

By default, the Clipping class from Standard.ulb is selected.

The following parameters are available:

Clipping Region	Selects whether to cut the region outside the clipping shape, or inside the clipping shape.
Screen-Relative Coordinates	If checked, all coordinates are interpreted as relative to the screen. This makes it harder to enter coordinates (because you can no longer use the eyedropper), but it preserves the location of the clipping shape relative to the screen when zooming. The coordinates are interpreted with (0, 0) at the lower left corner of the fractal window and (1, 1) at the upper right corner.
Clipping Shape	This is the clip shape class parameter. You can select any class that represents a clipping shape.

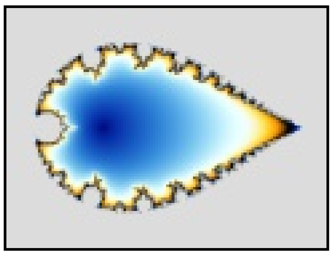
Generic Clipping can also show handles around the edges of a clipping shape. The number and position of these handles are determined by the selected clipping shape. By default, handles are shown in the fractal window, but not when rendering or on previews. Under **Handle options**, this can be configured:

Show Handles	Sets whether handles are displayed at all.
Not on previews	If checked, handles are hidden on previews, for example in the browser or for explore or eyedropper previews.
Not on renders	If checked, handles are hidden when rendering to disk.
Show Numbers	If checked, a number or label is displayed next to each handle if applicable.
Show Lines	If checked, lines are displayed if applicable.
Handle Size	Adjusts the apparent size of the handles.
Handle Thickness	Adjusts the apparent thickness of the handles.

Note: Handles are shown by toggling the solid color of the transformation on and off, so you might not see them properly if the solid color does not contrast with the fractal. Also, they do not work well in combination with the Guessing drawing method.

See Also
About classes
Standard transformations
Standard classes

Generic Transformation

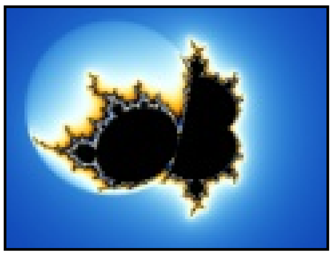


Generic Transformation is a skeleton transformation that lets a transformation class do the actual work. All transformations in Standard.uxf are also implemented as transformation classes. You can select any of these as the transformation class to be used together with Generic Transformation.

The **Transformation** class parameter sets the transformation class to be used. By default, the Inverse class from Standard.ulb is selected.

See Also
About classes
Standard transformations
Standard classes

Glass Hemisphere



The Glass Hemisphere transformation displays the fractal as though it is viewed through a spherical lens. It is available as a transformation in Standard.uxf and as a transformation class in Standard.ulb.

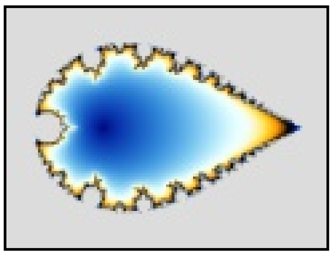
The location, size, and apparent refractive index of the lens can be changed. The refractive index adjusts the strength of the lens.

The following parameters are available:

Refractive index	Specifies the refractive index of the lens. You can use this to simulate various glass-like materials. Larger values will increase the strength of the lens.
Width	Specifies the width of the lens in fractal coordinates.
Center	Specifies the coordinates of the center of the lens. Use the eyedropper (right-click and click Eyedropper) to select the center by clicking inside the fractal window.
Use Screen Center	If checked, the center of the screen is used instead of the Center parameter, so the lens is always centered on the screen, even when zooming in.

See Also
Standard transformations

Inverse



The Inverse transformation turns a fractal inside out. The original center of the fractal is put infinitely far away, and points that were far away end up near the center. Inverse is available as a transformation in Standard.uxf and as a transformation class in Standard.ulb.

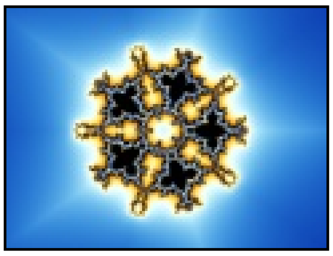
This can change the shape of the fractal in unexpected ways. The example image shows the Inverse transformation applied to a standard Mandelbrot set. (The inside areas of the Mandelbrot set are colored gray.)

The following parameters are available:

Radius	Specifies the radius of the inversion circle. The transformation inverts all points around this circle. Larger values will simply magnify the inverted fractal.
Center	Specifies the coordinates of the center of the inversion circle. This will drastically change the shape and form of the fractal. Use the eyedropper (right-click and click Eyedropper) to select the center by clicking inside the fractal window.
Use Screen Center	If checked, the center of the screen is used instead of the Center parameter, so the inversion circle is always centered on the screen. This can give unexpected effects when zooming in.

See Also
Standard transformations

Kaleidoscope



The Kaleidoscope transformation fills the screen with rotated copies of a small radial slice of the fractal, creating a kaleidoscope effect. It is available as a transformation in Standard.uxf and as a transformation class in Standard.ulb.

By tweaking the parameters, you can simulate many different kinds of symmetry. By default, the slices are aligned and mirrored to make the edges match, but there are also other options that produce sharp transitions.

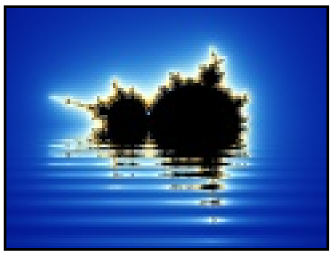
Try experimenting with the **Center** and **Rotation angle** parameters to obtain good results. Some sections of the fractal lend themselves much better to the kaleidoscope effect than others.

The following parameters are available:

Symmetry Order	Sets the symmetry order. This is the number of times the slice of the fractal is copied and rotated to obtain the final image.
Symmetry Mode	Selects the symmetry mode to use. Only the **reflective** option is guaranteed to produce seamless images. Use the **slice only** option to view the slice of the fractal that is used as a base for the symmetry effect.
Center	Specifies the coordinates of the symmetry center. Together with the Rotation angle parameter, this selects the slice of the fractal that is used. Try changing this to see the various effects that are possible. Use the eyedropper (right-click and click Eyedropper) to select the center by clicking inside the fractal window.
Use Screen Center	If checked, the center of the screen is used instead of the Center parameter, so the symmetry center is always centered on the screen. This can give unexpected effects when zooming in.
Rotation angle	Rotates the fractal before determining the slice that will be used as a base for the symmetry effect. This can drastically change the resulting image.

See Also
Mirror
Standard transformations

Lake



The Lake transformation mirrors the fractal in a rippled lake. The top part of the fractal is not altered, but below the water level, everything is mirrored. It is available as a transformation in Standard.uxf and as a transformation class in Standard.ulb.

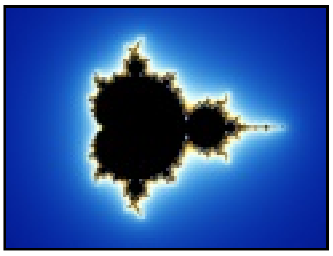
By changing the parameters, you can adjust the height and rotation of the water level, and change the size and frequency of the waves.

The following parameters are available:

Water level	Selects the water level. Only the imaginary part of this parameter is used. Use the eyedropper (right-click and click Eyedropper) to select the water level by clicking on a point inside the fractal window.
Use screen center	If checked, the water level is always centered on the screen. In this case, the Water level parameter is ignored.
Rotation angle	Rotates the water level. To rotate the fractal instead of the water, also enter the same value in the Rotation angle parameter on the Location tab.
Use Location tab angle	If checked, the rotation angle from the Location tab is used instead of the Rotation angle parameter. This ensures that the water level is always horizontal.
Amplitude	Specifies the amplitude of the waves.
Frequency	Specifies the frequency of the waves.

See Also
Ripples
Standard transformations

Mirror



The Mirror transformation mirrors the fractal along an arbitrary axis. It can be useful for mirroring effects with multiple layers. It is available as a transformation in Standard.uxf and as a transformation class in Standard.ulb.

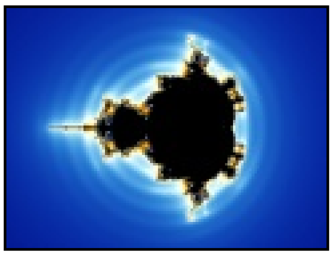
There are presets for a horizontal and a vertical reflection axis, but you can also specify any angle. The center of the axis is configurable, too.

The following parameters are available:

Reflection Axis	Selects the reflection axis. The axis points into the mirroring direction, so it is perpendicular to the imaginary mirror. Select **arbitrary** to specify any angle for the axis.
Reflection Angle	Specifies the rotation angle of the reflection axis, in degrees.
Center	Selects the center of the reflection axis. Use the eyedropper (right-click and click Eyedropper) to select a point by inside the fractal window.
Use screen center	If checked, the reflection center is always centered on the screen. In this case, the Center parameter is ignored.

See Also
Kaleidoscope
Standard transformations

Ripples



The Ripples transformation adds a water ripple effect to the fractal. The center, strength, and frequency of the ripples are adjustable. It is available as a transformation in Standard.uxf and as a transformation class in Standard.ulb.

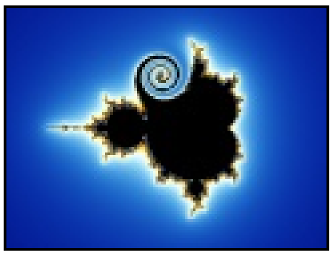
Interesting interference effects are obtained by adding multiple Ripple transformations to a fractal, with different center and strength values.

The following parameters are available:

Ripple Center	Specifies the center of ripples. Use the eyedropper (right-click and click Eyedropper) to select this by clicking on a point inside the fractal window.
Use Screen Center	If checked, the ripple center is always centered on the screen. In this case, the Ripple Center parameter is ignored.
Ripple Strength	Specifies the strength of the ripples. Larger values give a larger distortion.
Ripple Frequency	Specifies the frequency of the ripples. Larger values will create more and smaller ripples.
Ripple Fade	Specifies how soon the ripples fade out. Larger values cause the ripples to fade out over a larger distance (so more ripples are visible).
Ripple Type	Selects how the ripples distort the fractal. The default **Forward and Back** option gives the most natural water-like effect, but the other options are also interesting.

See Also
Lake
Twist
Standard transformations

Twist



The Twist transformation adds a twisted spiral to the fractal. It distorts a small part of the fractal in the form of a spiral, like a vortex. It is available as a transformation in Standard.uxf and as a transformation class in Standard.ulb.

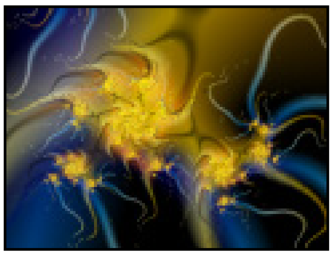
The center, strength, and size of the vortex are adjustable. This transformation is often combined with the Ripples transformation to obtain interference effects.

The following parameters are available:

Twist Center	Specifies the center of the twisted spiral. Use the eyedropper (right-click and click Eyedropper) to select this by clicking on a point inside the fractal window.
Strength	Specifies the strength of the twist. Larger values create more strongly twisted spirals.
Decay Factor	Specifies how soon the spiral loses its strength. Larger values decrease the size of the spiral.

See Also
Lake
Standard transformations

About classes



Since the first release of Ultra Fractal in 1998, the formulas written for it have become larger and more complex. Even though many formulas are quite similar, they could not share any common code, but had to be written as stand-alone formulas. For example, if you wanted to add just one new feature to an existing formula written by someone else, you had to create a new formula, duplicate all the code from the original formula, and only then add some new things.

To improve this situation, formulas in Ultra Fractal 5 can re-use common code via **classes**. You can view a class as a "black box" that adds functionality to a formula. Via class parameters, a formula enables you to select the actual class that performs an action for the formula.

The formula declares what kind of functionality it needs and you can select any compatible class to actually implement it. This makes it possible for formula authors to write a new class to implement just the new feature that they want to add to an existing formula. The formula doesn't have to be changed at all. You just select the new class when you use the formula.

For formula authors, classes make it easier to write and manage complex formulas. For everyday users, classes make it possible to combine existing elements in new ways without needing to do any programming.

Next: Example 1 - Formula classes

See Also
Fractal formulas
Writing formulas

Example 1 - Formula classes

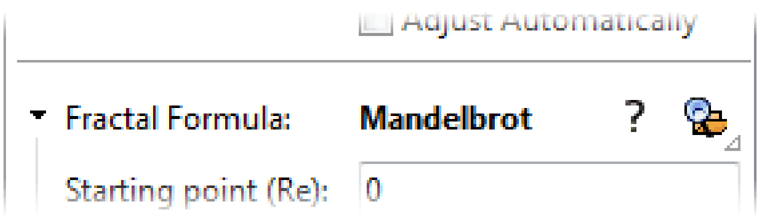
Tip: This example assumes that you are familiar with Ultra Fractal basics as explained in the tutorials.

As a first example of how classes work in practice, let's take a look at some of the standard classes that come with Ultra Fractal.

First, use the browser to open the Default parameter set in Examples.upr.

Click the **Browse** button in the Formula tab of the Layer Properties tool window to select Generic Formula in Standard.ufm.

Generic Formula is a "skeleton" formula that lets a class implement the actual fractal formula behavior. It has just one class parameter called Fractal Formula which is set to the Mandelbrot class by default:



Fractal Formula: Mandelbrot
Starting point (Re): 0

The Mandelbrot class has its own parameters like Starting point, which appear grouped with the class parameter. Let's select a different formula class.

Click the Browse button for the **Fractal Formula** class parameter to select a new class. This opens the browser in class mode, so it shows only those classes that are compatible with this class parameter.

Locate the Newton class in Standard.ulb and click OK.

Now, a Newton fractal appears instead of a Mandelbrot fractal. Anyone could write a formula class that works with Generic Formula to create a different fractal, without having to modify Generic Formula itself. Of course, this doesn't really do anything new, since you could also just write a new fractal formula. But it does make the formula behavior available in a generic form.

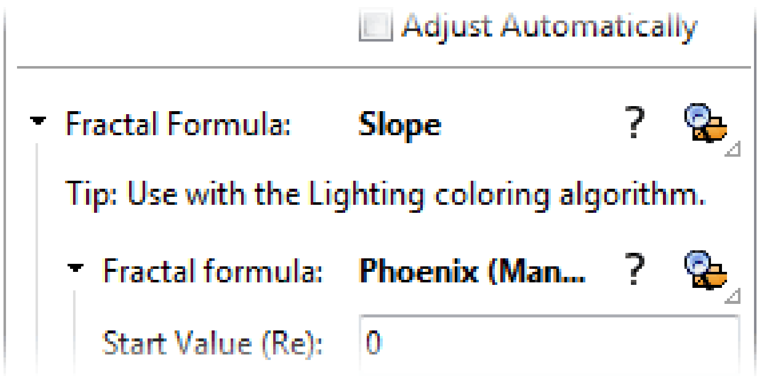
Click the Browse button for the **Fractal Formula** class parameter again, and this time, select the Slope class.

Briefly go to the **Outside** tab of the Layer Properties and click the **Browse** button there to select the Lighting coloring algorithm. Then go back to the **Formula** tab again.

The combination of the Slope formula class and the Lighting coloring algorithm creates a 3D lighting effect. Previously, there were three different Slope formulas for Mandelbrot, Julia, and Newton fractals (these are still in Standard.ufm). Each of these formulas duplicates the code for the lighting effect; the only different code is the where the actual fractal formula calculation is done. Adding a new fractal type could only be done by creating yet another variation of this formula.

However, now the Slope formula class contains another class parameter that enables you to select the internal fractal formula. You can select any formula class, just like with Generic Formula.

Click the Browse button for the **Fractal Formula** class parameter inside the **Slope** class (which currently is set to Mandelbrot) and select Phoenix (Mandelbrot) instead.


Adjust Automatically

Fractal Formula: **Slope**

Tip: Use with the Lighting coloring algorithm.

Fractal formula: **Phoenix (Man...**

Start Value (Re): 0

Suddenly, we have a Slope formula that works with any fractal type. New fractal types can be added simply by writing them as a formula class, which makes them available for Generic Formula, Slope, or any other place where a fractal formula can be inserted.

Next: Example 2 - Orbit trap classes

See Also
About classes
Fractal formulas
Writing formulas

Example 2 - Orbit trap classes

Tip: This example assumes that you are familiar with Ultra Fractal basics as explained in the tutorials.

In the previous example, we saw how a class can implement the behavior of an entire fractal formula. However, it's also possible to use classes in a more fine-grained way, exposing all aspects of a formula for maximum flexibility.

The Orbit Traps coloring class is an example of this. The old Orbit Traps coloring algorithm (still available in Standard.ucl) contained parameters such as trap shape, trap mode and trap coloring, each with a number of options. In the new version, implemented as a coloring class in Standard.ulb, these are now class parameters. Every option is now a separate class. Let's have a look:

First, use the browser to open the Default parameter set in Examples.upr.

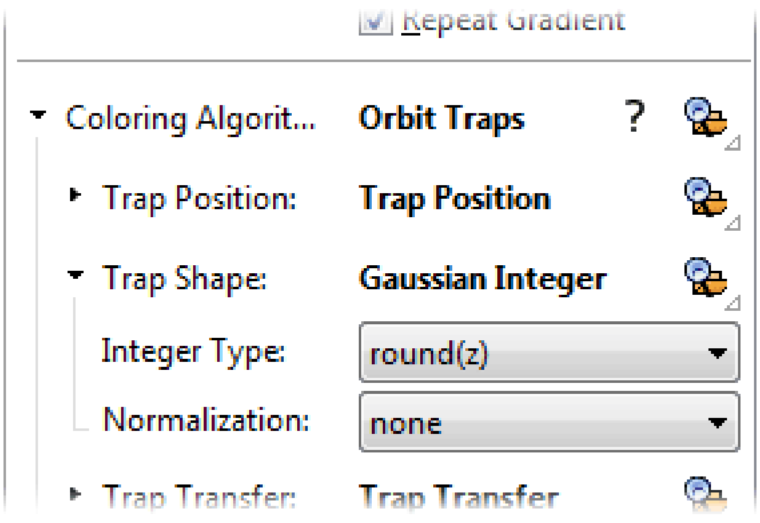
Click the **Browse** button in the Outside tab of the Layer Properties tool window to select Generic Coloring (Gradient) in Standard.ucl.

Like Generic Formula, Generic Coloring (Gradient) is a skeleton coloring algorithm that lets a coloring class do the work. By default, it loads with the Smooth algorithm.

Click the Browse button for the **Coloring Algorithm** parameter to select the Orbit Traps class in Standard.ulb.

As you can see, the Orbit Traps class uses class parameters for all major parameters. All trap shapes, for example, are now implemented as separate classes. Let's select a different trap shape.

Click the Browse button for the **Trap Shape** parameter to select the **Gaussian Integer** class.


Repeat Gradient
Coloring Algorit... Orbit Traps
Trap Position: Trap Position
Trap Shape: Gaussian Integer
Integer Type: round(z)
Normalization: none
Trap Transfer: Trap Transfer

The old Gaussian Integer coloring algorithm in Standard.ucl actually is a variation on an orbit trap, just with a different trap shape. It had limited options to color the trap shape, but now that it is available as a general trap shape class, we can combine it with any trap mode or trap coloring.

For example, you can select **Angle to Origin 2** as the trap coloring to get an effect that you couldn't create with the original Gaussian Integer coloring algorithm.

When formula authors write new trap shapes, trap modes or trap colorings and make them available in class library files via the public formula library, you can just drop them into this standard Orbit Traps class to use them.

Next: Working with classes

See Also
About classes
Coloring algorithms
Writing formulas

Working with classes

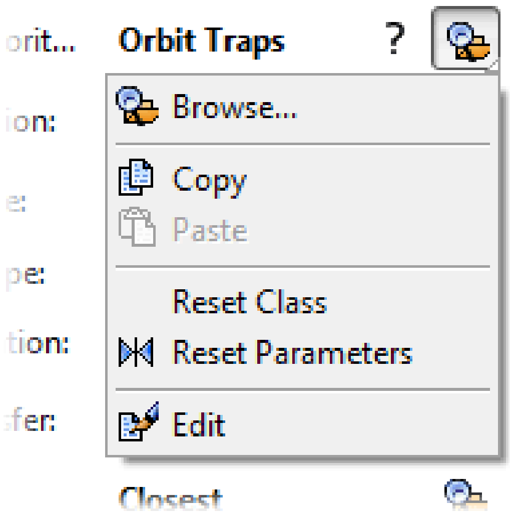
Classes are commonly stored in class library files with an .ulb extension. Like all formula files, you can examine and organize class library files with the browser. Make sure all files are visible, or set the file type to Classes.

Usually though, you'll work with classes via class parameters.

Click the **Browse** button for a class parameter to select a different class. A modal browser window opens, which shows only those classes that are compatible with this class parameter.

Tip: In the browser window, click the Find Entries button and leave all fields blank to get a list of all compatible classes in the formula library.

Hold down the Browse button for a menu with other options:



Click **Copy** to copy the currently selected class with all its sub-parameters to the Clipboard.

Click **Paste** to copy the class information on the Clipboard to the class parameter. The class parameter must be compatible with the class parameter that you copied the class information from, otherwise an error message is shown.

Click **Reset Class** to reset the class to the default class for the class parameter.

Click **Reset Parameters** to reset the sub-parameters for the currently selected class to their default values. The class itself stays the same.

Click **Edit** to open the class declaration in the formula editor.

Next: Standard classes

See Also
About classes
Working with formulas
Writing formulas

Standard classes

All standard transformations, fractal formulas and coloring algorithms that come with Ultra Fractal are also available as classes in the **Standard.ulb** file. This file is installed with the other standard formula files in the main formula folder. For more information, see the documentation for the standard formulas:

- Standard transformations
- Standard fractal formulas
- Standard coloring algorithms

For tips on how to use the standard classes, see the class usage examples.

Generally, to get as much formulas and classes working together as possible, it is necessary for them to share the definition for common types of classes. These shared definitions are in the file **common.ulb**. By adhering to these definitions, different formulas and classes from different formula authors can easily work with each other. The common.ulb file is installed in the folder for public formulas so it can be updated automatically via the online formula database when necessary.

The common.ulb file is intended mainly for formula authors. For everyday users, it does contain some useful classes:

- **Clip Shape Merge** allows a single clip shape parameter to accept multiple clip shapes. See Generic Clipping.
- **Image Trap** allows an image to be used as a color trap for the Direct Orbit Traps coloring class.
- **Transform Merge** allows a single transformation parameter to accept multiple transformations. See Generic Transformation.
- **Trap Shape Block** and **Trap Shape Merge** allows a single trap shape parameter to accept multiple trap shapes, and to transform trap shapes. See also Example 2 - Orbit trap classes.

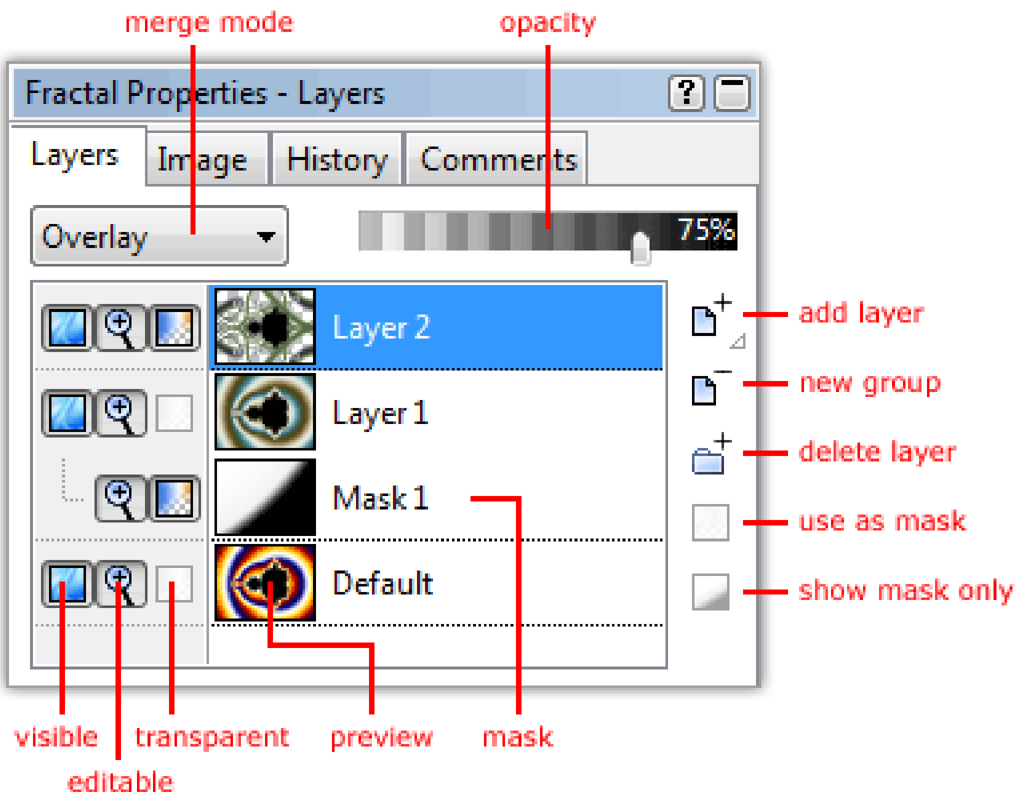
For formula authors, the common.ulb file is invaluable and almost any class should be based on one of the base classes found here. If you think modifications or new base classes are needed, please contact info@ultrafractal.com with your suggestion.

See Also
About classes
Writing formulas

Layers

One of Ultra Fractal's key features is the ability to use multiple layers. Each layer contains a separate fractal image. By using multiple layers, you can achieve many special and wonderful effects that are not possible with single-layer images.

Layers are managed in the Layers tab of the Fractal Properties tool window:



The layers list shows all layers of the active fractal window, complete with previews. It also selects the active layer. The active layer is edited by the Layer Properties tool window and the gradient editor.

- The **Add** button duplicates the active layer or group. Hold down the Add button to open a menu with more options and layer presets. See Presets.
- The **New Group** button adds a new layer group to the layers list. See Layer groups.
- The **Delete** button deletes the currently selected layers.
- The **Merge Mode** input box selects the merge mode of the selected layers.
- The **Opacity** slider changes the opacity of the selected layers.
- The **Visible**, **Editable** and **Transparent** icons before each layer toggle the visibility, editability, and transparency of the layer. See Working with layers.
- The **Use as Mask** button turns a layer into a mask and back. The **Show Mask Only** button makes it easier to edit a mask. See Masks.

Next: How layers are merged

See Also
Tutorial: Working with layers
Keyboard shortcuts for the Fractal Properties tool window
Layer groups

Animating layers

How layers are merged

Within a multi-layered fractal, Ultra Fractal merges the different layers to create the resulting image. This image appears in the fractal window. The layers are merged by superimposing them.

Ultra Fractal starts with the bottom layer and places the second layer on top of it. The third layer (if any) is in turn placed on top of the result, and so on. If a layer is completely opaque, the layers below it will be hidden. If a layer is completely transparent, it will not be visible.

Most layers will be more or less transparent, so they are visible while still allowing the lower layers to shine through. There are four ways to make layers transparent:

- Reduce the opacity of the layer. By default, the opacity is set to 100%, making the layer fully opaque. Move the opacity slider to the left to make the active layer more transparent.
- Change the merge mode of the layer. By default, the merge mode is set to Normal. The other merge modes create special effects that allow lower layers to be partially visible even if the opacity of the layer is set to 100%. See Merge modes.
- Make only parts of the layer transparent. The previous two options affect the entire layer. You can, however, also change the opacity of only certain areas in the layer. See Transparent layers.
- Add a mask to the layer. The mask allows even more control over which areas of the layer will be transparent. See Masks.

Of course, you can freely mix these options. It is common, for example, to use a merge mode like Hard Light and set the opacity to less than 100%.

Next: Working with layers

See Also
Layers
Tutorial: Working with layers

Working with layers

Layers are managed in the Layers tab of the Fractal Properties tool window.

Click the **Add** button to add a new layer, duplicating the active layer.
Hold down the Add button to open a menu with layer presets. See Presets.

Click the **Delete** button to delete the currently selected layers.

Click the **New Group** button to create a new layer group. See Layer groups.

Most commands and tool windows (such as the Layer Properties tool window) work on the currently selected layers. So, by selecting layers in the layers list, you change what is being edited by these other tool windows.

- To **select** a layer, simply click it in the layers list.
- To **select multiple** layers, hold down the Ctrl key and click the layers you want to add to the selection. Ctrl-click a selected layer to deselect it again.
- To **select a range** of layers, click the first layer. Then, hold down the Shift key and click the last layer.
- The **Merge mode** drop-down box selects the merge mode of the selected layers. The merge mode determines how the layer is combined with the layers below it.
- The **Opacity** slider adjusts the opacity of the selected layers. Move it to the left to make the layer less visible (more transparent). Move it to the right to make it more opaque.
- To **rename** the active layer, click it again or press F2 (like in Windows Explorer).
- To **move** layers around in the list, drag the selected layers up or down. Of course, this affects the way the layers are composited.

To the left of each layer is a row of icons. These icons toggle various properties.

The **Visible** icon toggles the visibility of the layer. Use it to temporarily hide a layer, so you can more clearly see the other layers.

The **Editable** icon selects whether a layer is editable. Only editable layers are affected by zooming operations. By default, all layers are editable, so if you want to zoom in on only one layer, you should clear this icon on the other layers first.

The **Transparent** icon selects whether transparent areas in a layer are visible. See Transparent layers.

Notes

- By holding down the Shift key while clicking on the **Visible**, **Editable**, or **Transparent** icons, you toggle all other layers instead. If you want to see just one layer, for example, Shift-click its **Visible** icon and all other layers will be turned off. Shift-click it again to show all layers.
- If a layer is not editable, its properties can still be changed with the Layer Properties tool window. The Editable icon only affects zooming operations.

- To copy the selected layers to another fractal window, drag them from the list of layers to the other fractal window.
- Right-click in the list of layers to open a menu with frequently used commands. This menu also contains **Copy** and **Paste** commands that are another way of copying layers to other fractal windows.

Next: Merge modes

See Also
Tutorial: Working with layers
Working with masks
Keyboard shortcuts for the Fractal Properties tool window
Animating layers

Merge modes

The **Merge mode** drop-down box at the top of the Layers tab of the Fractal Properties tool window selects the merge mode of the selected layers. The merge mode defines how a layer is combined with the underlying layers to create the final image. The table below lists all merge modes and what they do.

The best way to learn how to use the different merge modes is to experiment. Also try to use various settings of the Opacity slider and see how it controls the intensity of the merging effects.

Next: Transparent layers

Merge mode	**Description**
Pass Through	This mode is only available for layer groups. The layers in the group will be merged as if they were outside the group.
Normal	Directly returns colors from the layer. Use this if you don't want any special effects.
Multiply	Multiplies the layer with the underlying layers. The result is always a darker color, thus darkening the underlying layers.
Screen	Multiplies the inverse of the layer with the inverse of the underlying layers, and inverts that again. The result is always a lighter color, thus brightening the underlying layers. Screen is the inverse of Multiply.
Overlay	Multiplies or screens the colors, depending on the color in the underlying layers. Creates color blending effects between the layer and the underlying layers.
Hard Light	Multiplies or screens the colors, depending on the color in the layer. Emphasizes the dark and light regions in the layer, while the areas with medium brightness become transparent. Useful if the layer contains shadows or embossing effects.
Soft Light	Darkens or lightens the colors, depending on the color in the layer. Creates an effect similar to Hard Light, but with less emphasis on the dark and light areas in the layer.
Darken	Returns the darkest of the color in the layer and the color in the underlying layers.
Lighten	Returns the lightest of the color in the layer and the color in the underlying layers.
Difference	Returns the difference between the layer and the underlying layers. Often creates unusual and unexpected color transitions.

Hue	Returns the hue of the layer, and the saturation and luminance of the underlying layers. Colors the underlying layers with the hue of the layer.
Saturation	Returns the saturation of the layer, and the hue and luminance of the underlying layers. Changes the saturation of the underlying layers depending on the layer.
Color	Returns the hue and saturation of the layer, and the luminance of the underlying layers. Colors the underlying layers with the layer. The underlying layers control the brightness of the resulting image.
Luminance	Returns the luminance of the layer, and the hue and saturation of the underlying layers. The layer controls the brightness of the underlying layers. Luminance is the inverse of Color.
Addition	Directly adds the layer to the underlying layers, limiting the resulting colors at white (255, 255, 255).
Subtraction	Directly subtracts the layer from the underlying layers, limiting the resulting colors at black (0, 0, 0). Difference is similar, but returns the absolute value after subtracting.
HSL Addition	Adds the layer to the underlying layers using the HSL color model. Creates unusual effects.
Red	Returns the red part of the layer, and the green and blue parts of the underlying layers.
Green	Returns the green part of the layer, and the red and blue parts of the underlying layers.
Blue	Returns the blue part of the layer, and the red and green parts of the underlying layers.

See Also

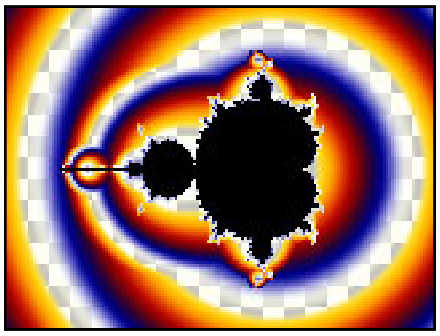
How layers are merged

Layers

Transparent layers

To make a layer transparent, you can use the **Opacity** and **Merge mode** settings, but these work on the entire layer. You can also make only certain parts of a layer transparent, which gives you more artistic control.

The easiest way to create transparent areas in a layer is to use a transparent gradient. The transparent parts of the gradient will create transparent areas in the layer. If you make all the other layers invisible, a pattern of blocks will show the transparent areas.



Another way of creating transparent areas is to use the Solid Color setting of transformations and coloring algorithms. Solid colors with an opacity value of less than 255 create transparent areas. Many transformations, such as Clipping in Standard.uxf, use this to create masking effects.

The **Transparent** icon before the layer toggles transparent areas on and off. Use it to quickly verify the transparent areas and to see what difference they make to the final image. By default, transparency is off, but it is automatically turned on when you make changes to transparent areas in the layer.

Next: Masks

See Also
Tutorial: Working with layers
How layers are merged
Layers

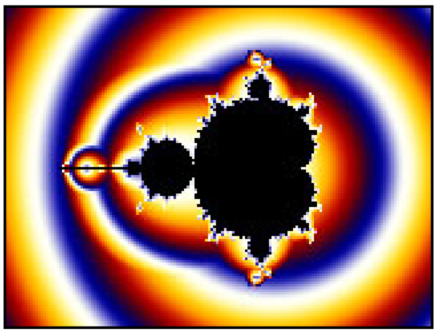
Masks

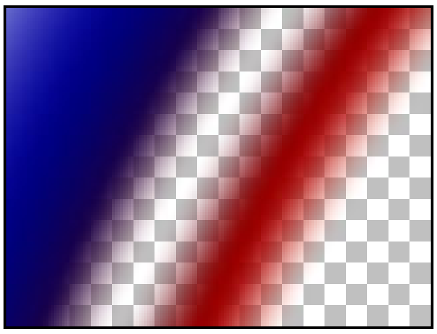
If you create transparent areas in a layer using a transparent gradient, the shape of the transparent areas is controlled by the selected coloring algorithms. Certain colors in the gradient are transparent, so those colors will become transparent in the layer, too. This means that you cannot use this method to create arbitrarily shaped transparent areas.

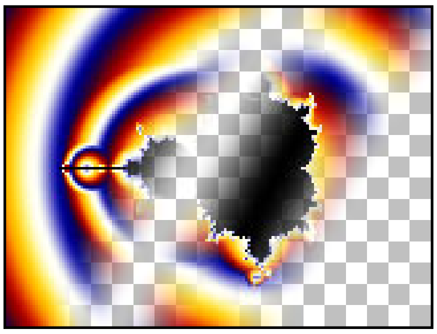
If you use transformations instead and set the opacity of the solid color to less than 255, you can create arbitrarily shaped transparent areas in the layer. You do need a transformation that will output the area you need, but you could write it yourself. Still, a limitation of this technique is that pixels are either set to the solid color (transparent), or they are colored according to the gradient and the selected coloring algorithms. You can only create sharp edges, not smooth transitions.

Masks overcome these problems. A mask is an invisible layer that is attached to the layer that needs transparent areas. The mask contains transparent areas, that are created with an ordinary transparent gradient. Since the mask is invisible, these transparent areas are invisible as well.

Instead, the layer owning the mask "borrows" the transparent areas of the mask. The shape of these areas is defined by the selected fractal formula, the selected coloring algorithms, and the gradient of the mask. The shape is independent from the layer owning the mask.


Layer that needs transparent areas


Mask with transparent areas


Layer with the mask applied

As you can see, the layer uses the transparent areas of the mask. The colors in the mask layer are ignored.

Layers can have multiple masks to add differently shaped transparent areas. If the layer contains transparent areas itself (for example created with a transparent gradient), these are also taken into account.

Next: Working with masks

See Also
Tutorial: Masking
Layers

Working with masks

Masks are managed in the Layers tab of the Fractal Properties tool window.

To duplicate the active layer as a mask, hold down the **Add** button and click **Duplicate as Mask** in the menu that appears.

To delete a mask, select it and click the **Delete** button.

To turn an existing layer into a mask, click the **Use as Mask** button. The layer will become a mask, attached to the layer directly above it.

Click the **Show Mask Only** button to disable the layer owning the mask. Instead, the transparent areas in the mask will be shown. White areas are opaque, black areas are transparent. This makes it easier to edit the mask.

Usually, the mask will be entirely white because its gradient is not yet transparent. The first thing to do is to open the gradient editor. Note that only the opacity view of the editor is enabled, since the colors in the gradient do not matter for a mask. Add a few extra control points to the gradient so it will start to show some transparency (darker regions).

By toggling the **Show Mask Only** button, you can alternatively work on the mask and judge the effects that it has on the layer that owns it. The mask can be edited like any layer. For example, you can zoom in on it, select another coloring algorithm, and so on.

Notes

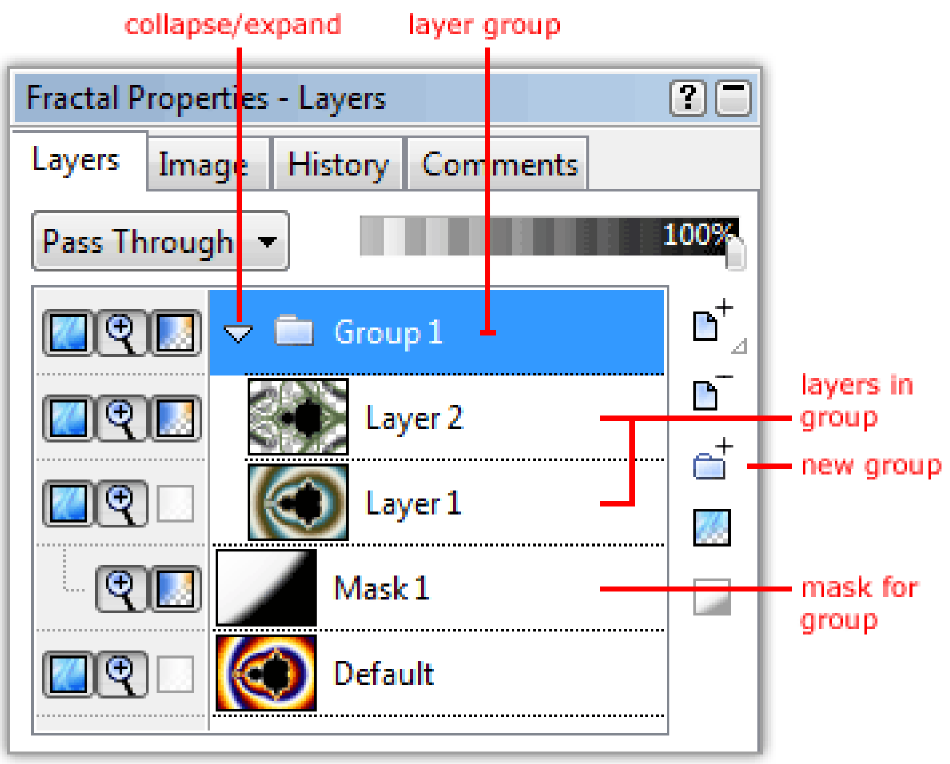
- You can duplicate a mask layer like any other layer by clicking the **Add** button.
- Layer groups can also have a mask. Just drag an existing mask layer onto the layer group, or position the layer to use as a mask just below the group (but not in it) and click the **Use as Mask** button.
- Masks can be moved to other layers or layer groups simply by dragging them onto the desired layer or layer group.
- A layer or layer group can have more than one mask. In this case, the transparent areas in the masks are combined, so each additional mask makes the layer more transparent.

Next: Layer groups

See Also
Tutorial: Masking
Masks
Working with layers
Layers

Layer groups

You can organize the layers in a fractal with **layer groups**. A layer group can contain any number of layers and nested layer groups. By logically arranging your layers in groups, the layers list becomes easier to manage. In addition, you can apply masks and merge modes to the entire group, which makes new creative effects possible.



Layer groups appear in the layers list with a folder-like group icon and a triangular **collapse/expand** button. Click this button to hide or show the items in the group. Layers in a group appear below it, indented to show their relationship.

Click the **New Group** button to create a new layer group. The layer group starts empty. To fill the group, simply drag one or more layer onto the group to move them inside. (Remember that you can Ctrl-click or Shift-click to select multiple layers.)

To delete a layer group, select it and click the **Delete** button. This also deletes all items that the layer group contains. Tip: you can always undo unwanted deletions.

You can easily move layers in and out of groups by dragging them around. While you are dragging, a thick black line shows where the dragged layers will end up. You can also nest groups simply by dragging one group inside another. Another way to move layers in and out of groups is with the **Move Up** and **Move Down** commands in the right-click menu for the layers list.

Layer groups have a merge mode just like layers, but with an additional **Pass Through** mode which is the default. If Pass Through is selected, the layers in the group are merged with the underlying layers one by one, as if they were not in a group at all. With the Pass Through merge mode and Opacity at 100%, the layer group organizes its layers without introducing any visual changes.

If a merge mode other than Pass Through is selected, the layers in the group are first merged together. Then, that intermediate result is in turn merged onto the underlying layers with the merge mode for the layer group. The resulting effect is often not possible to create without layer groups.

A group can also have a mask. In this case, the mask is applied to all layers in the group. Such a mask appears below all other items in the group, aligned with the group itself, as shown in the screen shot above. To turn a layer into a group mask, position it just under the group (but not inside it), and click the **Use as Mask** button. To apply an existing mask to a group, simply drag it onto the group. See also Working with masks.

See Also

Layers

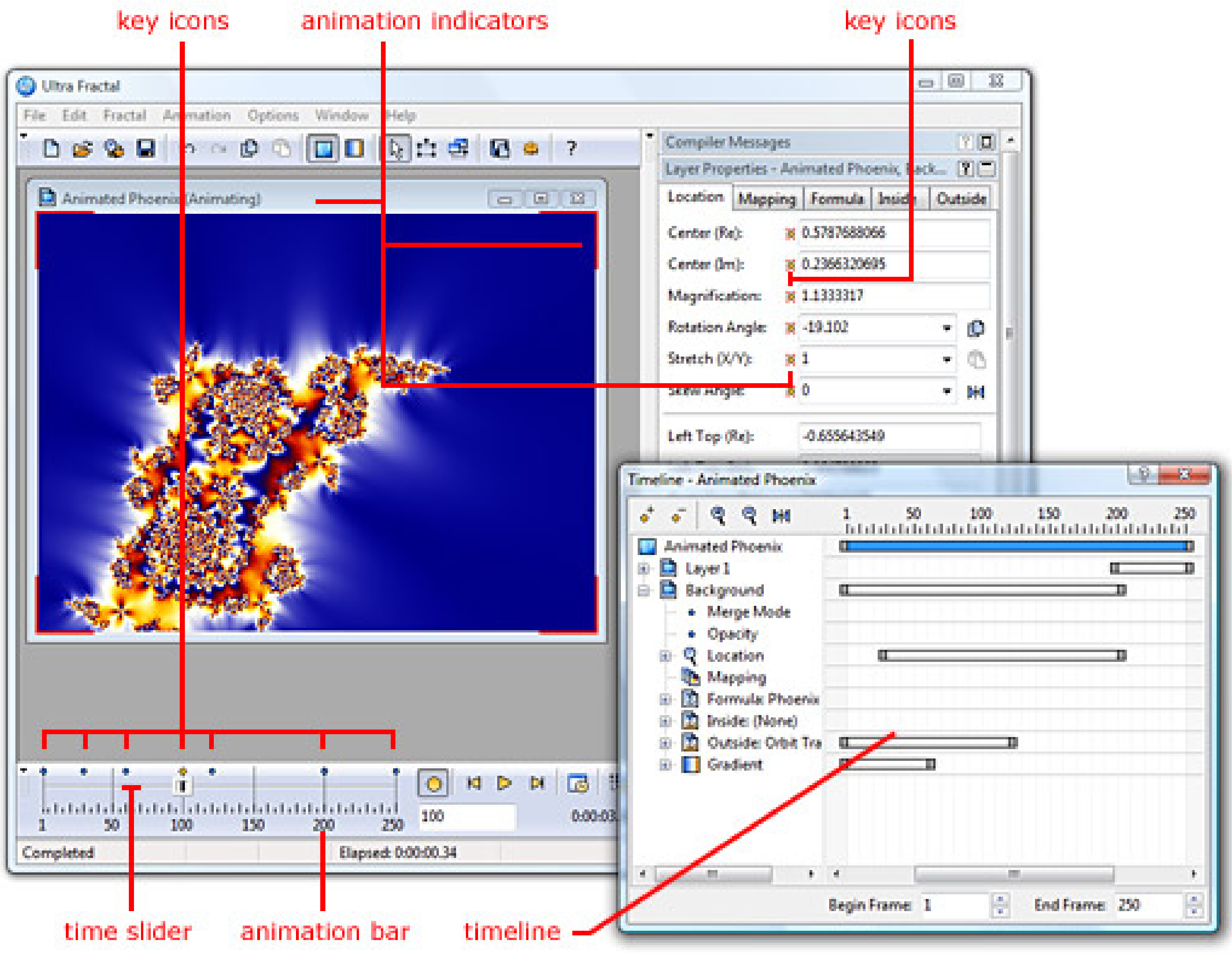
How layers are merged

Masks

Animation

Note: You need Ultra Fractal Animation Edition to work with animations.

Any fractal in Ultra Fractal can easily be turned into an animation. You can animate all parameters of the fractal at will and see the result immediately in the fractal window. Finally, render the animation to watch it as a movie clip.



Creating and editing animations is done with various tools:

- The **animation bar** contains the time slider and vital animation controls that enable you to create, edit, and play back animations. See Animation bar.
- With the **time slider**, you set the current frame. The fractal window always displays the image for the current frame.
- The red **animation indicators** are shown if the fractal is currently in Animate mode. In Animate mode, changes that you make to the fractal are only applied to the current frame. If Animate mode is off (the default), your changes are applied to the entire range of frames.
- The **key icons** show at which frames and for which parameters keys have been recorded. See Animation keys.
- The **Timeline tool window** provides an in-depth view of all animated settings and parameters and can be used to edit and tweak your animations. See Timeline.

The following topics will explain how these tools work, and how to use them effectively.

Next: Creating animations

See Also

Tutorial: Working with animations

Creating animations

Creating an animation from a normal fractal is easy, but perhaps different than in other fractal programs. These steps show how to create a simple zoom movie.

1. Click **New** on the File menu, and then click **Fractal** to open the formula browser. Select **Mandelbrot** in Standard.ufm, and then click **Open**. This creates a new default Mandelbrot fractal.

   (**Note:** Skip this step if you already have a fractal window open that you would like to turn into an animation.)
2. Move the time slider at the bottom to the far right. This sets the current frame to 100. (If you cannot see the time slider, click **Animation Bar** on the Options menu to reveal it.)
3. Click the **Animate** button on the animation bar to turn Animate mode on. In Animate mode, changes you make to the fractal are applied to the current frame only. This is necessary because we want to keep frame 1 as it is, and change frame 100 to something else.

   Note that the fractal window now shows red animation indicator marks in its corners, and "(Animating)" in the title bar. This shows that Animate mode is on.
4. Shift-click inside the fractal window, hold the mouse button down, and drag upwards to zoom in. Release the mouse button when you are satisfied with the result. (See Normal mode for more information about zooming.)
5. Congratulations! You have just made your first zoom movie. Drag the time slider to the left and right to see a real-time preview.

   Note that above the time slider, two key icons have appeared, one at frame 1, and one at frame 100. This shows that keys have been recorded at those frames.
6. To make the movie more interesting, let us add a rotate effect. Move the time slider to frame **50**, and ensure that Animate mode is still on.
7. Enter **90** in the **Rotation Angle** input box on the Location tab of the Layer Properties tool window. This will rotate the fractal 90° clockwise at frame 50.
8. Click the **Animate** button again to turn off Animate mode, because we are done recording this animation for now. It is a good habit to leave Animate mode off normally to avoid unintended changes to your animations.
9. Click the **Play** button on the Animation bar to start playing a preview of the animation, or drag the time slider back and forth.

   Observe that the animation starts unrotated, rotates to 90° at frame 50, and then rotates back to normal at frame 100, while zooming in all the time. The frames where we did not explicitly set new values are interpolated to create a smooth animation.

Notes

- In Ultra Fractal, there is no fundamental difference between animations and still (normal) fractals. A still fractal is simply a fractal without any animation keys. If you only want to create still fractals, just hide the animation bar and ignore the Animation menu.
- As you can see, the fractal does not interpolate from one set of parameters to another, like in some other fractal programs. Instead, every parameter and setting has its own set of keys and interpolates between them independently. This makes creating and editing animations much easier and enables you to create more complex animations.
- To create a movie clip of your animation, render it to disk.

Next: Animation keys

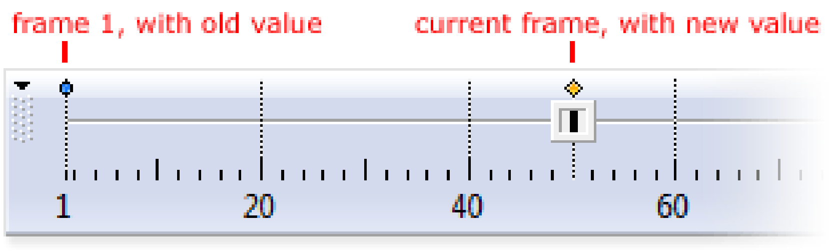
See Also
Tutorial: Working with animations
Animation

Animation keys

While you are creating animations, you are actually creating animation keys for the parameters that are animated. The animation keys define how the values of those parameters change over time.

Each parameter that can be animated has its own list of animation keys, which is initially empty. In this case, the parameter just has a static value that is the same for the entire frame range, and it is not animated.

When you first change a parameter while Animate mode is on, Ultra Fractal inserts a new key for that parameter at frame 1 with its old value. Then, it inserts a second key at the current frame with the new value.



The time slider shows the keys as blue and yellow dots. Keys are displayed as yellow dots if they are located at the current frame. In this example, if you now move the time slider from frame 1 to frame 50, the parameter animates from its old value to the new value.

Notes

- The key at frame 1 with the old value is only inserted when the parameter does not yet have any keys. If the parameter already has one or more keys, Ultra Fractal only inserts a key at the current frame.
- If Animate mode is on, and you change a parameter that already has a key at the current frame, Ultra Fractal adjusts the value of that key instead of inserting a new key.
- Use the Timeline tool window to edit and delete keys. You can also delete the key at the current frame from the right-click menu of an animated parameter. See Editing animations.
- Internally, a key stores its position as a time value, not as a frame number. This makes it possible to scale animations without introducing round-off errors. See Time settings.

Next: Animate mode

See Also
Tutorial: Working with animations
Editing animations
Animation

Animate mode

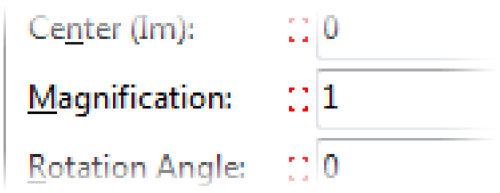
The Animate mode toggle controls what happens when you make changes to a fractal.

Set Animate mode on or off using the **Animate** button in the animation bar, the **Animate** command on the Animation menu, or the F3 key.

By default, Animate mode is **off**. In this case, changes that you make to the fractal, such as zooming in or adjusting a parameter, are applied to the entire range of frames. Changes never result in new animation keys; they only adjust the existing keys or the static value of non-animated parameters.

If Animate mode is **on**, changes that you make are applied to the current frame only. Ultra Fractal creates or adjusts keys at the current frame to accomodate your changes. This is the primary way to animate parameters and therefore create animations.

While Animate mode is on, the corners of the fractal window are marked with red animation indicators, and "(Animating)" is displayed in the title bar. Also, a small red animation indicator is shown next to each parameter that can be animated. The indicator reminds you that keys will be created or updated when you change that parameter.



Center (Im):	0
Magnification:	1
Rotation Angle:	0

If Animate mode is off, and you change a parameter that already has one or more keys, the values for all keys are adjusted. For example, if you zoom in, the entire animation will be zoomed in; if you apply a rotation, the entire animation will be rotated. This is very useful if you want to do global adjustments. In practice, you will usually switch Animate mode on and off while working with an animation to achieve the effects that you are after in an efficient way.

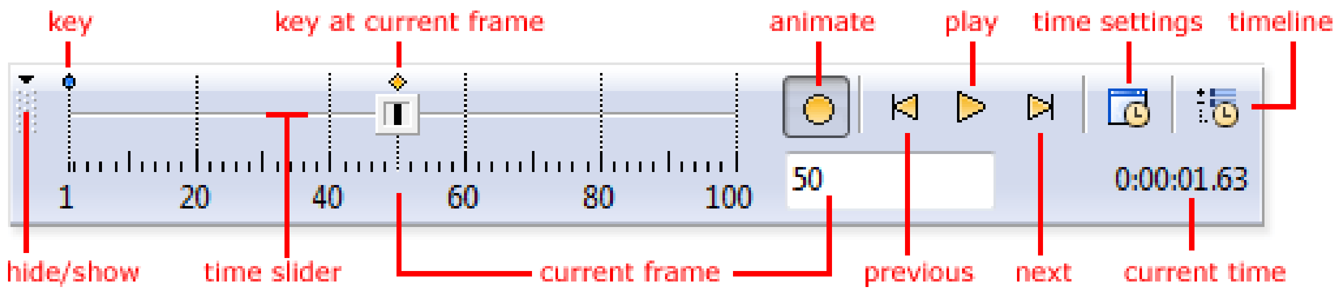
To adjust the values for all keys when Animate mode is off, Ultra Fractal calculates the difference between the new value and the old value, and adds that to the values for all keys. For floating-point parameters with exponential interpolation, however, Ultra Fractal divides the new value by the old value and multiplies the values for all keys by the result.

Next: Animation bar

See Also
Tutorial: Working with animations
Animation

Animation bar

The animation bar at the bottom of the screen contains the time slider and provides quick access to controls that you often use while working with animations.



- The button on the far left side **hides** or **shows** the animation bar. You can also click **Animation Bar** on the Options menu.
- The **time slider** sets the **current frame**. You can also enter the current frame directly in the input box to the right of the time slider. To the right of that input box, the **time** between the first frame of the animation and the current frame is shown, in hours, minutes, seconds, and 1/100 seconds.
- Above the time slider, blue and yellow **key icons** indicate where keys have been inserted. A key icon turns yellow when it is at the current frame. Click on a key icon to jump to that key's frame.
- The **Animate** button turns Animate mode on or off.
- The **Previous Key** and **Next Key** buttons jump to the first key before or after the current frame.
- The **Play** button starts or stops real-time playback of the animation.
- The **Time Settings** button opens the Time Settings dialog where you can scale the animation and adjust its length.
- The **Timeline** button activates the Timeline tool window where you can edit and delete animation keys, and adjust interpolation curves.

Notes

- Although the position of keys are shown above the time slider, you cannot directly edit keys from there. Use the Timeline tool window instead.
- Most commands are also available on the Animation menu with keyboard shortcuts.

Next: Playing animations

See Also
Tutorial: Working with animations
Animation

Playing animations

While you are working with an animation, you will often want to see a quick live preview. You can drag the time slider back and forth to slowly preview a part of the animation in the fractal window. If the fractal is not too slow to calculate, you can also play a real-time preview in the fractal window.

Click the **Play** button in the animation bar to start or stop playing a preview of the animation, or click **Play** on the Animation menu.

While playing the preview for an animation with multiple layers, Ultra Fractal gives some layers more priority than others. The active layer is calculated with the highest priority, and then the other visible layers are calculated from top to bottom, with the editable layers first. To speed up the preview, you can temporarily hide one or more layers.

The preview is always played at a constant frame rate, independent of the frame rate of the animation. Click **Options** on the Options menu and then click the Fractal tab to change the **Animation preview speed**. Higher values give smoother animation, but allow less time per frame, so individual frames will show less detail. The best value depends on the speed of your computer and the complexity of the animations.

Notes

- Playback will not work well if the drawing method of a layer is set to **One-pass linear**. Use **Guessing** or **Multi-pass Linear** instead.
- To create a final movie clip of your animation, or to create a quick preview movie, render it to disk.

Next: Animating locations

See Also
Tutorial: Working with animations
Rendering animations
Animation

Animating locations

Although you can do much more with animation in Ultra Fractal, the most obvious thing to animate is the location of the fractal to create **zoom movies**. The location is controlled by five parameters on the Location tab of the Layer Properties tool window: Center, Magnification, Rotation Angle, Stretch, and Skew Angle.

To animate the location, first set Animate mode to **on** and move the time slider to the frame where you want to create animation keys, as described in Creating animations.

Use the same zooming, panning, and rotation features that you would normally use (see Normal mode and Select mode). They work the same as when Animate mode is off, except that only the current frame is modified, instead of the entire range of frames.

If you use Select mode, Ultra Fractal always inserts keys at the current frame for all five location parameters, even when the value of the parameter has not been changed. For example, if you just zoom in without stretching or skewing, keys for Stretch and Skew will also be inserted although their values have stayed the same. If you do not want this, use Normal mode instead.

Of course, you can also animate the location by changing one of the location parameters directly while Animate mode is on. You can also edit the corner coordinate parameters in the lower half of the Location tab, but this just indirectly changes the normal parameters in the upper half.

Always make sure whether Animate mode is on or off. If you zoom in while Animate mode is off, this will transform the entire animation, which usually is not what you want.

The behavior of the **Copy**, **Paste**, and **Reset** buttons in the Location tab behave depends on whether or not Animate mode is on, providing flexible ways to copy and clear animation keys. See also Editing animations.

Animate mode on	**Animate mode off**
Copies the location at the current frame to the Clipboard, without any animation keys.	Copies the location for the entire frame range to the Clipboard, including all animation keys.
Sets the current location to the location on the Clipboard, inserting animation keys when necessary, just as if you entered those values manually. You must turn off Animate mode before pasting locations with animation keys (copied when Animate mode is off).	Sets the current location to the location on the Clipboard, overwriting any animation keys. If the location on the Clipboard has no animation keys (copied with Animate mode on, or from a non-animated fractal), any animation keys in the current location will be removed.
Resets the location at the current frame to the default location for the current fractal formula, inserting animation keys when necessary.	Clears all animation keys and resets the location to the default location for the current fractal formula.

Next: Animating parameters

See Also

Tutorial: Working with animations

Animate mode

Timeline

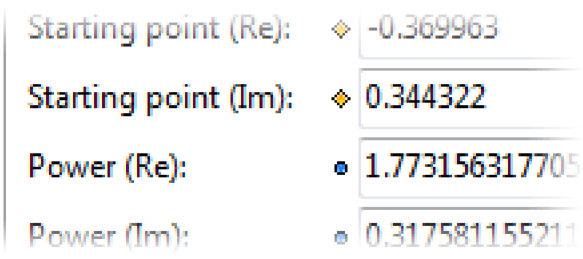
Animation

Animating parameters

You can animate almost anything in Ultra Fractal, including all formula parameters and settings such as Maximum Iterations and Color Density.

To animate a parameter, first set Animate mode to **on** and move the time slider to the frame where you want to create a new animation key, as described in Creating animations.

Parameters that can be animated will display a red animation indicator next to them while Animate mode is on. Simply type a new value or use the Explore feature to change the parameter.



Starting point (Re): ◆ -0.369963
Starting point (Im): ◆ 0.344322
Power (Re): ● 1.77315631770
Power (Im): ● 0.31758115521

A **blue dot** next to a parameter means that it is animated (i.e. has one or more animation keys). If the blue dot turns into a **yellow marker**, this means that the parameter contains an animation key at the current frame. In this case, editing the parameter while Animate mode is on will change the value of the key at the current frame instead of inserting a new key.

If Animate mode is off, editing an animated parameter will adjust the values of all its animation keys. For example, if a parameter animates from the value **1** at frame 1 to **4** at frame 100, and the current frame is 1 and you change the value to **2**, the value at frame 100 will change to **5**. Floating-point parameters with exponential interpolation, such as Magnification and Color Density, are scaled instead of translated.

Complex, floating-point, and integer parameters will be interpolated smoothly between animation keys. You can also animate enumerated and boolean parameters, but they will not be interpolated.

Right-click a parameter for a menu with options to **insert** a new key at the current frame or to **remove** the existing key, and to jump to the **previous** and **next** key for that parameter. See Editing animations.

Inserting a key manually is useful if you want to animate a parameter from frame 20 to frame 30, for example. If you just move the time slider to frame 30 and change the parameter, the parameter will be animated from frame 1 to frame 30, which is not what you want.

Instead, first move the time slider to frame 20 and insert a key there (which does not change the value of the parameter). Then move to frame 30, turn Animate mode on, and edit the parameter to animate it.

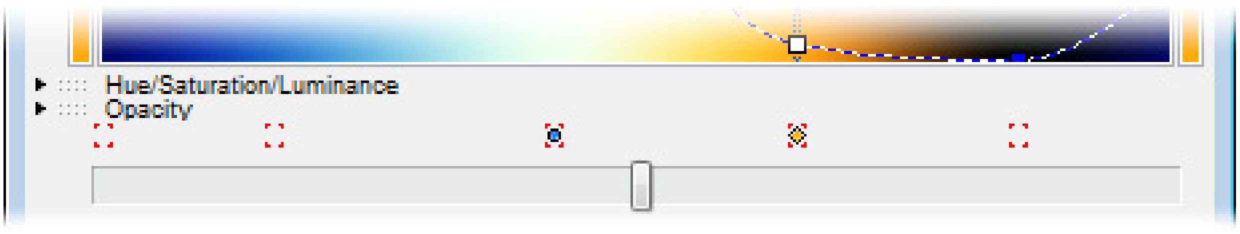
Next: Animating gradients

See Also

Tutorial: Working with animations
Animating locations
Timeline
Animation

Animating gradients

Of course, you can also animate gradients in Ultra Fractal to create color cycling movies or subtle color change effects during your animations.



The gradient editor contains a small bar just above the rotation slider that shows whether or not the control points above it are animated. When Animate mode is on, this bar shows red animation indicators below each control point to illustrate that it can be animated.

To animate a control point, first set Animate mode to **on** and move the time slider to the frame where you want to create a new animation key, as described in Creating animations.

Now simply drag the control point around. This will record keys for both the color and the position of the control point. You can also type new values in the input boxes in the gradient editor. See also Editing gradients.

As with parameters, a **blue dot** below a control point means that a control point is animated. It turns into a **yellow marker** if the control point has an animation key at the current frame. In this case, dragging the control point while Animate mode is on will change the value of the keys at the current frame instead of inserting new keys.

If Animate mode is off, dragging an animated control point will transform the values of all its animation keys. For example, if a control point animates from position **0** at frame 1 to **50** at frame 100, and the current frame is 1 and you drag it to position **20**, its position at frame 100 will change to **70**.

The following gradient adjustment options record keys when Animate mode is on and can be used to create animated gradients: **Randomize Custom** (if Randomize control points is not checked), **Adjust Colors**, **Reverse**, and **Invert**.

The **Randomize**, **Randomize Bright**, and **Randomize Misty** options clear all control points and then generate a new gradient, so they cannot animate the existing control points.

The **Copy** and **Paste** commands work differently depending on whether or not Animate mode is on:

	Animate mode on	Animate mode off
	Copies the control points at the current frame to the Clipboard, without any animation keys.	Copies the control points for the entire frame range to the Clipboard, including all animation keys.

📋	Pastes the control points from the Clipboard into the gradient, inserting animation keys as necessary. If the number of control points on the Clipboard is equal to the number of control points in the gradient, this will animate the control points in the gradient. You must turn off Animate mode before pasting gradients with animation keys (copied when Animate mode is off).	Pastes the control points from the Clipboard into the gradient, overwriting any animation keys. If the gradient on the Clipboard has no animation keys (copied with Animate mode on, or from a non-animated gradient), any animation keys in the gradient will be removed.

Notes

- You can create **color cycling** animations simply by changing the **Rotation** setting or dragging the rotation slider while Animate mode is on.
- The individual input boxes in the gradient editor do not show animation indicators. Refer to the animation indicators in the horizontal bar below the control points instead.
- You cannot directly insert or delete animation keys in the gradient editor. Use the Timeline tool window instead.
- Because each control point is animated separately, you cannot create an animation from one arbitrary gradient to another, and you cannot animate the insertion or deletion of a control point. To blend from one arbitrary gradient to another, duplicate the current layer, change the gradient on the new layer, and animate the opacity of the layer so it fades in to replace the original gradient.

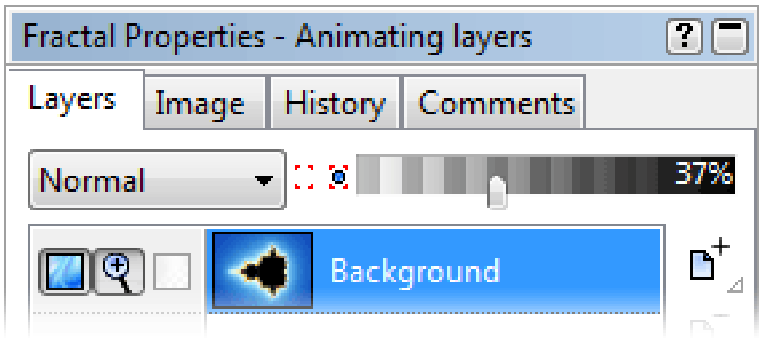
Next: Animating layers

See Also

Tutorial: Working with animations
Animating parameters
Gradients
Animation

Animating layers

Animating the opacity and blending of layers is the way in Ultra Fractal to animate blends between different fractals, gradients, color combinations, and so on.



To animate layer blending, first set Animate mode to **on** and move the time slider to the frame where you want to create new animation keys, as described in Creating animations.

Now simply drag the opacity slider to a new value, or set a new merge mode.

As with any parameter, a **blue dot** next to the Merge Mode input box or the opacity slider means that it is animated. It turns into a **yellow marker** if there is an animation key at the current frame. In this case, editing the parameter while Animate mode is on will change the value of the key at the current frame instead of inserting a new key.

Notes

- If you animate the merge mode, it will not be interpolated between animation keys, which gives sudden changes. For smooth transitions, create two identical layers, one with the first merge mode, and one with the second, and animate the opacity of both layers to create a smooth transition between the two.
- You cannot animate the visibility or the insertion or deletion of a layer. Instead, animate the opacity to 0% so the layer appears to be hidden.
- You also cannot animate a layer from or to a mask layer. Instead, animate the opacity part of the gradient of the mask layer to be 100% opaque (white) so the mask has no effect.

Next: Time settings

See Also

Tutorial: Working with animations
Animating parameters
Layers
Animation

Time settings

Each fractal starts with 100 frames and a frame rate of 30 frames per second, but you can of course easily change the length and the frame rate of the animation.

Click the **Time Settings** button on the animation bar, or click **Time Settings** in the Animation menu. This opens the Time Settings dialog where you can extend, reduce, or scale the animation.

The Time Settings dialog enables you to enter new values for the number of frames and the current frame rate (in frames per second). The total time for the animation is also shown and updated as you type, according to the following formula:

Time (seconds) = **Frames** / **Frame Rate** (fps)

Before you enter a new frame rate, review the **Lock frames** and **Lock time** radio buttons. If **Lock frames** is checked, the number of frames will not be changed, so the total time of the animation will change according to the new frame rate. Otherwise, the number of frames will be adjusted as you type in order to preserve the duration of the animation.

With the radio buttons in the **Existing keys** group, you can globally scale the existing animation keys according to the new length in frames.

- If **Scale to new length** is checked, the animation keys will be scaled such that their relative positions in the animation do not change. For example, if you have keys at frame **1** and **50** and the number of frames changes from 100 to 200, the keys will end up at frame **1** and **99**, so they stilll occupy the same part of the animation.
- If **Keep at first frame** is checked, the frame number of the animation keys will stay the same. If the number of frames is increased, this will leave the new part of the animation empty so you can add new animation keys at the end. If the number of frames is decreased, the last part of the animation will be chopped off (but not deleted; you can still work with it in the Timeline tool window).
- If **Keep at last frame** is checked, the frame number of the animation keys will be adjusted to keep the distance to the end of the animation the same. If the number of frames is increased, this will move the existing keys to the end, so you can add new keys at the beginning. If the number of frames is decreased, the first part of the animation will be chopped off (but not deleted; you can still work with it in the Timeline tool window).

Notes

- Because the position of an animation key is internally stored as a precise time value, you can scale an animation more than once without introducing round-off errors. For example, if you reduce an animation with 100 frames to 10 frames with **Scale to new length** checked, a key at frame 25 will end up at frame 3. If you now scale the animation back to 100 frames, the key will be back at frame 25.
- For more information on individual controls in the Time Settings dialog, click the ? button in the title bar, and then click a control.
- You can change the default fractal to have a different length and frame rate. See Default fractal.

Next: Editing animations

See Also

Tutorial: Working with animations
Creating animations
Animation keys
Animation

Editing animations

Ultra Fractal provides various ways to edit and change your animations after you have initially created them. Often you will use the Timeline tool window, which also reveals the structure of your animation. But you can also directly work with animation keys by right-clicking on animated parameters, which opens a menu with animation-related commands. This can be useful for small changes.

To insert a new key at the current frame, right-click a parameter and click **Insert Key**.

To delete a key at the current frame, right-click a parameter and click **Delete Key**. Use the Timeline tool window to delete multiple keys at once.

To jump to the first key before the current frame, right-click a parameter and click **Previous Key**.

To jump to the first key after the current frame, right-click a parameter and click **Next Key**.

Note that the **Previous Key** and **Next Key** commands in the right-click menu for a parameter jump to the previous and next keys for that parameter only. To jump to the previous and next keys in the entire animation, use the global commands in the animation bar or on the Animation menu instead.

A powerful way to make global changes to an animation is to copy and paste parts of it, using the **Copy** and **Paste** commands found everywhere in Ultra Fractal. You can copy locations, transformations, formulas, coloring settings, gradients, layers, and even entire fractals. The behavior of the Copy and Paste commands depends on whether or not Animate mode is on.

Animate mode on	**Animate mode off**
Copies the settings at the current frame to the Clipboard, without any animation keys.	Copies the settings for the entire frame range to the Clipboard, including all animation keys.
Copies settings from the Clipboard, inserting animation keys when necessary, just as if you entered all values manually. You must turn off Animate mode before pasting settings with animation keys (copied when Animate mode is off).	Copies settings from the Clipboard, overwriting any animation keys. If the settings on the Clipboard have no animation keys (copied with Animate mode on, or from a non-animated fractal), any existing animation keys will be removed.

Remember: if Animate mode is on, your actions only affect the current frame; if Animate mode is off, your actions are applied to the entire fractal.

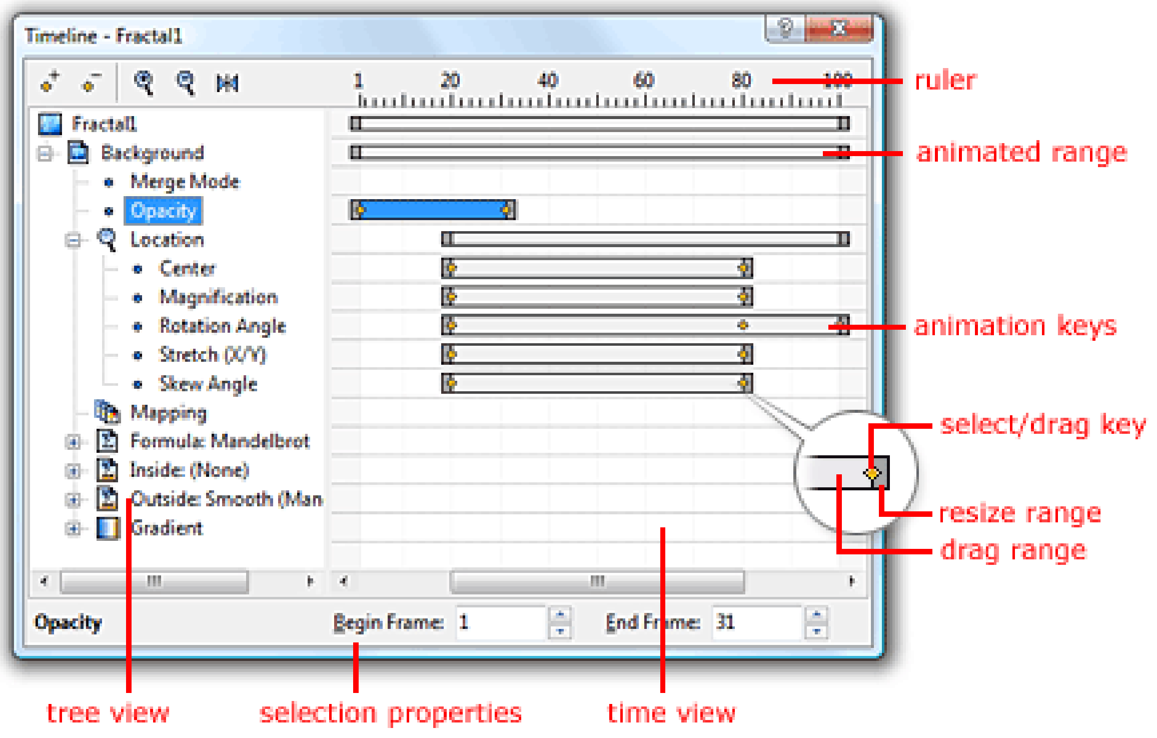
Next: Timeline

See Also
Tutorial: Working with animations
Animation keys
Time settings
Animation

Timeline

The Timeline tool window provides the most versatile way of editing your animations. It shows a tree view of all parameters in the fractal, grouped by layer and category, with an overview of their animation keys.

To open the Timeline tool window, click the **Timeline** button on the animation bar, or click Timeline on the Animation menu.



- The **tree view** on the left displays all layers of the active fractal window. Within each layer, the parameters are grouped by categories that correspond to the tabs of the Layer Properties tool window, combined with layer settings and the gradient.
- The **time view** on the right displays the **animated range** and **animation keys** for all categories and parameters. The animated range of a category contains the animation keys of all parameters within the category.
- The **selection properties** at the bottom enable you to edit the current selection. For animated ranges, you can change the begin and end frames, which scales the range. For animation keys, you can change the value of the key and its frame. You can also select how Ultra Fractal interpolates around the key with various interpolation options. See Interpolation.

It is easy to discover which parts of a fractal are animated by looking at the animated ranges for different categories. To **move** a complete animated range, grab it in the middle and drag it to the left or to the right. To **resize** an animated range, drag the resize gribs at the left or right ends. This powerful way to edit animations adjusts all animated keys within the category that corresponds with the animated range.

You can also click on a single animation key and **move** it, or **adjust** its value with the selection properties panel at the bottom. To select multiple keys or ranges, hold down Ctrl while you click. Hold down Shift to select a consecutive area of keys or ranges. This enables you to move or resize multiple items at once.

To insert a new key, click the **Insert** button and then click in the time view where you want to create a new key.

To delete the currently selected key or animated range, click the **Delete** button.

Click the **Zoom In** button to enlarge the area with frames at the center of the time view, so you can work more accurately.

Click the **Zoom Out** button to be able to see more frames at the same time.

To scale the time view such that all frames of the animation will just fit in the currently visible area, click the **Reset View** button.

Notes

- To quickly insert or delete keys, hold down Ctrl and click in the time view where you want to create a new key, or on the key that you want to delete.
- Right-click in the time view for a menu with frequently used commands.

Next: Interpolation

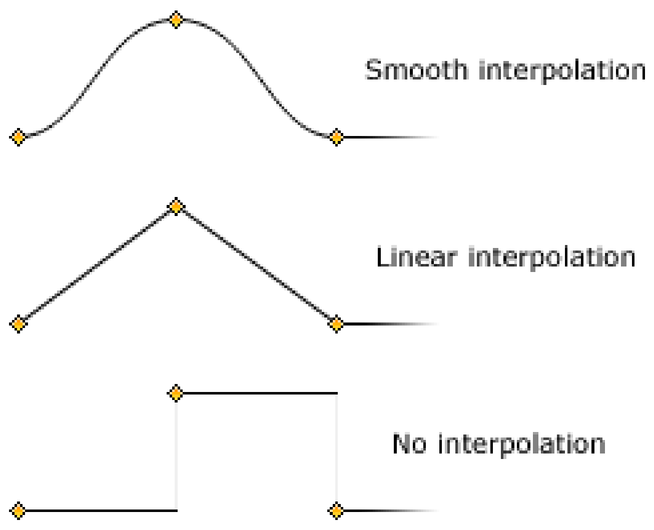
See Also

Tutorial: Working with animations
Time settings
Editing animations
Animation

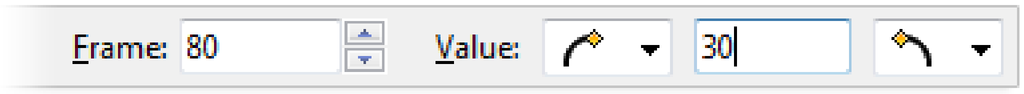
Interpolation

To create smooth animations, Ultra Fractal automatically interpolates between the animation keys that you have recorded. By default, a smooth interpolation method is used that damps sudden changes and creates smooth ease-in and ease-out effects around each key. However, you can select different interpolation methods for each animation key using the Timeline tool window.

There are three interpolation methods available:



To adjust interpolation methods for an animation key, select the key in the Timeline tool window. The properties panel at the bottom now shows the frame, the value, and the current interpolation settings for the key.


Frame: 80
Value: 30

You can set the interpolation method for both the curve to the left and the curve to the right of the key. In some cases, this will also change the interpolation settings for adjacent keys. For example, if you set the interpolation method for the curve to the right of the current key to No, the interpolation method for the curve to the left of the next key will also be set to No.

Sets **No interpolation**. The value of the parameter will remain the same until it suddenly jumps to the value of the next key.

Sets **Linear interpolation**. The value of the parameter will linearly change from one key to the next. This can result in noticeable changes in the speed at which the parameter changes around animation keys. If you want to create seamlessly looping animations with rotation, for example, you need to use linear interpolation to keep the rotation speed constant.

Sets **Smooth interpolation**. The value of the parameter will gradually change from one key to the next, slowing down and speeding up where needed to avoid sudden changes in the speed at which the parameter is changed. This is the default interpolation method.

Next: Exponential interpolation

See Also
Tutorial: Working with animations

Timeline
Animation

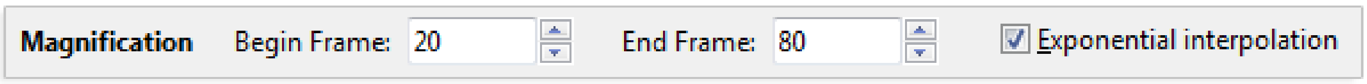
Exponential interpolation

For floating-point parameters, Ultra Fractal offers an additional interpolation setting called **exponential** interpolation. This can be set independently of the normal interpolation methods in the Timeline tool window.

Exponential interpolation should be used for parameters that are exponential in nature. This means that in order to experience the same apparent increase of the parameter, you need to double it each time, instead of adding something.

A perfect example is the Magnification parameter on the Location tab. It starts at 1. If you add 1 to make it 2, the fractal is magnified by a factor of two. If you add 1 again, the fractal is magnified by only a factor of 1.5. If you keep adding 1, the apparent difference gets less and less. At a magnification of 1000, you will probably not notice it if you change the magnification to 1001. In contrast, if you keep multiplying the magnification with two, you will each time experience the same zoom effect. Clearly, the Magnification parameter is exponential in nature. In contrast, the Rotation Angle parameter is linear in nature.

This is important when interpolating animations, because exponential parameters must also be interpolated exponentially. For example, if you create an animation of 99 frames where Magnification ranges from **1** at frame 1 to **16** at frame 99, it must be **4** at frame 50 to give the effect of a gradually increasing zooming level. With normal interpolation, it will be 8.5. Fortunately, the Magnification parameter uses exponential interpolation by default.


Magnification Begin Frame: 20 End Frame: 80 ✓ Exponential interpolation

To turn exponential interpolation on and off for a parameter, open the Timeline tool window, select the parameter in the tree view, and click the **Exponential interpolation** check box.

Notes

- Certain built-in parameters in Ultra Fractal, such as Magnification, Stretch, and Color Density in the Inside and Outside tabs, are exponential by default. When you are writing a formula, you can also specify whether or not floating-point parameters in your formula should be treated as exponential by default. See the exponential setting.
- Due to the nature of exponential values, exponential interpolation only works if both the begin and end points are nonzero and both positive or negative.

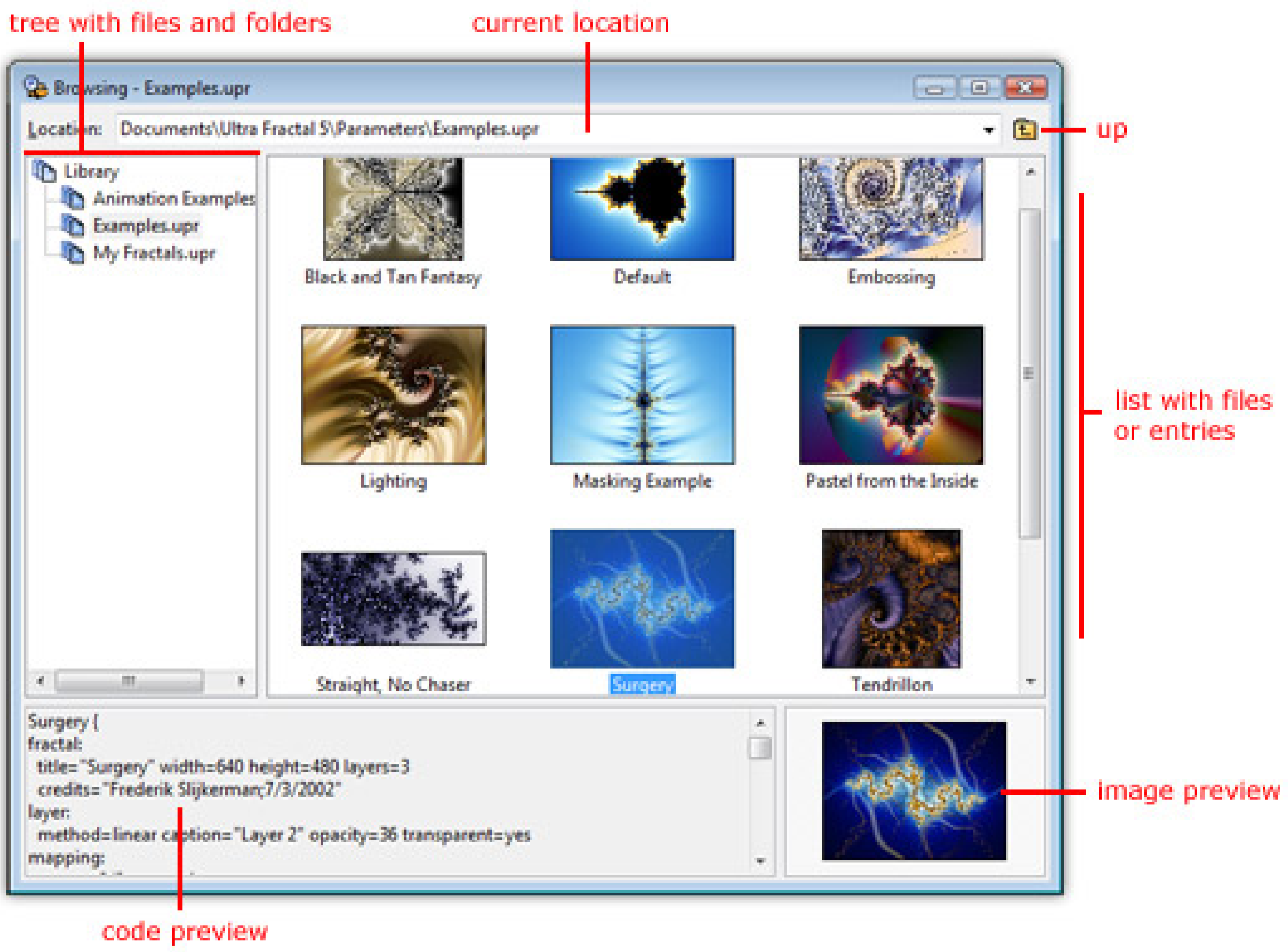
See Also
Tutorial: Working with animations
Timeline
Interpolation
Animation

Browsers

To explore and organize the various types of fractal-related files on your computer, Ultra Fractal includes a flexible file browser. It works much like Windows Explorer, but it also works with files containing multiple entries, such as parameter files or formula files.

To open a browser, click **Browse** on the File menu.

The browser is divided into four panes:



Location: Documents\Ultra Fractal 5\Parameters\Examples.upr

```
Surgery {
fractal:
  title="Surgery" width=640 height=480 layers=3
  credits="Frederik Slijkerman;7/3/2002"
layer:
  method=linear caption="Layer 2" opacity=36 transparent=yes
mapping:
```

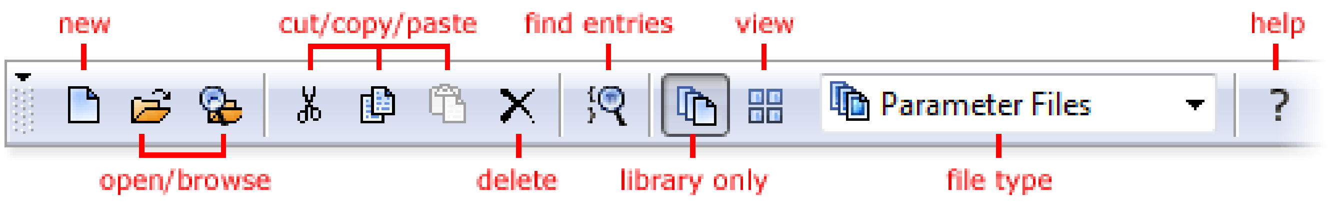
- The **location** input box at the top shows the file or folder that is currently selected. You can also type a new location here. Click the **Up** button next to the location input box to go up to the parent folder.
- The **tree** on the left shows an overview of all files and folders on your computer. If Library mode is active, only the files and folders in the library of the current file type are shown. See Library mode.
- The **list** on the right shows the contents of the file or folder selected in the tree. The name of this file or folder is displayed by the location input box. The list can show either item details, icons or thumbnails. See View style.
- The **code preview** shows the text corresponding to the entry selected in the list.
- The **image preview** shows a preview image for the entry selected in the list.

Next: Browser toolbar

See Also
Quick Start Tutorial
Modal browsers
Formula ratings
Workspace
Fractal windows

Browser toolbar

The toolbar for the browser contains commands to view and work with folders, files and entries:



- The **New** button creates a new fractal from scratch.
- The **Open** button opens a file from disk.
- The **Browse** button opens a new modeless browser. To duplicate the browser, click Duplicate on the File menu.
- The **Cut**, **Copy**, and **Paste** buttons are used to move and copy selected files, entries, and folders. See Organizing your work.
- The **Delete** button deletes selected files, entries, and folders.
- The **Find Entries** button opens a dialog where you can search for entries (such as parameter sets and fractal formulas) on your computer. See Finding files and entries.
- The **Library Only** button toggles library mode on and off. See Library mode.
- The **View** button selects the current view style: details, icons or thumbnails. See View style.
- The **File Type** drop-down box selects which files are currently visible. See File types.

The commands on the toolbar are duplicated on the File, Edit, and View pull-down menus. Frequently used commands are also on the menu that pops up when you right-click in the browser.

Next: Modal browsers

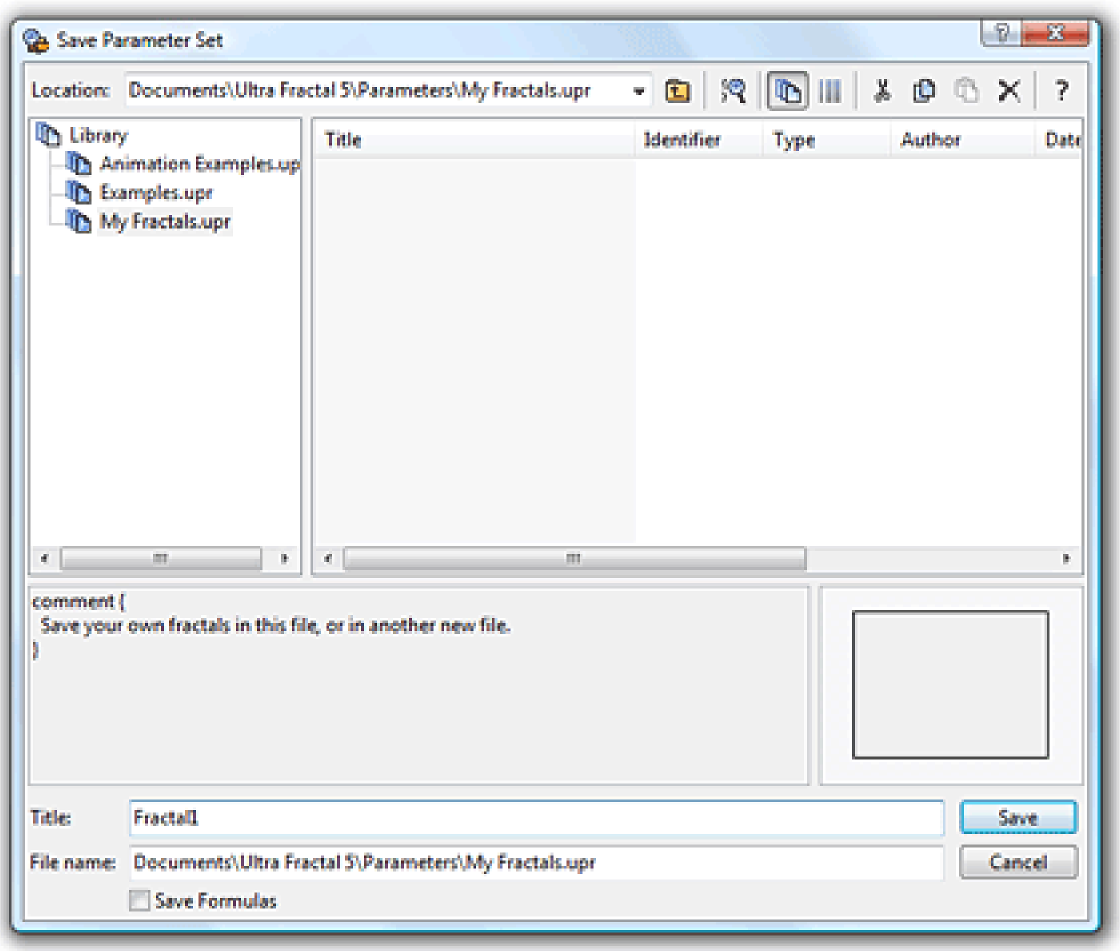
See Also
Keyboard shortcuts for browsers
Browsers
Workspace

Modal browsers

There are two types of browsers: modeless and modal browsers. A modeless browser is created when you click Browse on the File menu. Modeless browsers are typically used to organize files. They can stay open in the background while you work with other windows, such as fractal windows.

A modal browser is shown when you must select a parameter set or formula. The modal browser looks like a modeless browser, but it contains a small built-in toolbar, and Open and Cancel buttons. You must close the modal browser by clicking Open or Cancel before continuing.

Modal browsers are also used to save parameter sets and gradients. In this case, they contain input boxes to enter a file name and a title, and a Save button.



Location: Documents\Ultra Fractal 5\Parameters\My Fractals.upr

comment {
Save your own fractals in this file, or in another new file.
}

Title: Fractal1

File name: Documents\Ultra Fractal 5\Parameters\My Fractals.upr

Save Formulas

You can compare modal browsers to standard Windows Open and Save dialog boxes, except that they work with entries (such as parameter sets and formulas) instead of files. In the same way, a modeless browser can be compared to Windows Explorer.

Next: File types

See Also
Browsers
Browser toolbar

File types

The browser works with all Ultra Fractal file types. You can set it to display all fractal-related files, or only files of a selected type.

To select a file type, click File Types on the View menu, and then click the file type. You can also use the drop-down box on the toolbar. The following file types are available:

Fractal Files	Displays fractal files (*.ufr). Fractal files contain complete fractals. See Opening and saving fractals.
Parameter Files	Displays parameter files (*.upr). Parameter files contain multiple parameter sets that describe a fractal without storing the calculated pixels. Older Fractint parameter files (*.par) are also shown. See Parameter files.
Gradient Files	Displays gradient files (*.ugr). Gradient files contain multiple gradients that store coloring information for a fractal. Older gradient files (*.ual) and Fractint palette files (*.map) are also shown. See Opening and saving gradients.
Transformations	Displays transformation files (*.uxf). Transformation files contain multiple transformation formulas. See Transformations.
Fractal Formulas	Displays fractal formula files (*.ufm). Fractal formula files contain multiple fractal formulas. Older Fractint formula files (*.frm) are also shown. See Fractal formulas.
Coloring Algorithms	Displays coloring algorithm files (*.ucl). Coloring algorithm files contain multiple coloring algorithms (formulas). See Coloring algorithms.
Classes	Displays class library files (*.ulb). Class library files contain multiple classes, which can be used in formulas to extend their behavior. See About classes.

Select **All Files** to display all these file types at the same time.

Notes

- If **All Files** is selected, Library mode is not available.
- Modal browsers cannot switch between file types, since their purpose is to open or save a file of a particular type.

Next: Library mode

See Also
Browsers

Browser toolbar

Library mode

The browser can be switched to library mode. In library mode, only the files and folders within the library of the visible file type are shown.

This prevents cluttering the tree on the left side of the browser with all folders on your computer (most of which will not contain any fractal-related files), and makes it easier to work with parameter files, gradient files, and formulas.

To activate library mode, click **Library Only** on the View menu. Click it again to turn library mode off.

The library for each file type is a folder (for example Documents\Ultra Fractal 5\Parameters) that contains files for that file type by default.

Usually, you will work in library mode, but if you want to open a file or entry that is not in the library, you have to turn library mode off in order to find it. In this case, you can view all files and folders on the local drives (including network drives) on your computer.

Notes

- Library mode does not work when all file types are visible, because files of different types do not share the same library folder.
- The location of the library of a file type can be set in the Folders tab of the Options dialog.
- To open a file located on a different (networked) computer, you have to map a shared folder on that computer to a network drive with Windows Explorer first.

Next: View style

See Also
Browsers
Browser toolbar

View style

The list of items in the browser can display the items with different view styles, just like in Windows Explorer.

To change the view style, click a new view style on the View menu. You can also click the **View** button on the tool bar and then click a view style on the pop-up menu that appears.

Click **Thumbnails** to display the items as large thumbnails that contain a preview of the item.

Click **Large Icons** to display the items as a collection of large icons, where each icon shows the type of the item rather than a preview.

Click **Small Icons** to display the items as a collection of small icons.

Click **List** to display the items as a list of small icons.

Click **Details** to display the items in a list with columns that contain more information on each item, such as the date, author and comments. You can click on a column header to sort the items by the information in that column. Next to Thumbnails, this is probably the most useful view style.

In Thumbnails view, the browser caches the thumbnails that it generates for parameter files, fractal files and formulas. These are stored in the preview cache that is also used for the preview window in the bottom-right corner of the browser.

You can adjust the size of the preview cache in the Browser tab of the Options dialog. If you have many parameter sets that you view regularly, you might want to increase the cache size, although this uses more space on your hard disk. You can also turn off thumbnail caching for formulas to leave more space in the cache for parameter and fractal thumbnails which are more time-consuming to regenerate.

Next: Opening files and entries

See Also
Browsers
Browser toolbar

Opening files and entries

Browsers are used to organize your work, but they can also open all types of files and entries. (This does not apply to modal browsers.)

To open a file or entry, double-click in the list on the right of the browser. If you double-click a folder, its contents will be shown.

Fractal files and parameter sets will be opened in a new fractal window. You can also drag them from the browser to any open fractal window.

Right-click on a fractal file, parameter file, or parameter set and click **Render to Disk** to render it to disk. You can also drag them to the Render to Disk tool window.

Right-click a parameter set and click **Open as Text** to open the parameter set in the formula editor to edit it manually.

Gradients will be opened in a new gradient editor. You can also drag them from the browser to any open gradient editor.

Transformations, fractal formulas, coloring algorithms and classes will be opened in the formula editor.

You can also drag transformations to the list of transformations in the Mapping tab of the Layer Properties tool window. Fractal formulas can be dropped on the top of the Formula tab, and coloring algorithms can be dropped on the top of the Inside and Outside tabs to select them.

Next: Organizing your work

See Also
Browsers
Browser toolbar
Modal browsers

Organizing your work

To organize your fractal-related files, you can move, copy, delete, and rename files and entries in the browser. Again, the browser works similar to Windows Explorer, except that it also manipulates the entries in parameter files, gradient files, and formula files.

To move a file, folder, or entry, select it in the list view, and click **Cut** on the Edit menu. Select the new location, and then click **Paste** on the Edit menu.

To copy a file, folder, or entry, select it in the list view, and click **Copy** on the Edit menu. Select the new location, and then click **Paste** on the Edit menu.

Click **Paste** on the Edit menu to move or copy an item that was previously cut or copied to the Clipboard.

To delete a file, folder, or entry, select it in the list view, and click **Delete** on the Edit menu.

To rename a file, folder, or entry, select it in the list view and click it again, or click **Rename** on the Edit menu.

These commands are also on the menu that pops up when you right-click an item in the list view or in the tree view. In this way, you can also move, copy, delete, and rename files and folders using the tree view.

Alternatively, you can drag items from one location to another to move them. Hold down Ctrl while dropping to copy the items instead.

Notes

- Items that are cut or copied to the Clipboard are not actually moved or copied until you use the Paste command.
- Deleting a file, entry, or folder will delete it immediately. It will not be moved to the Recycle Bin.

Next: Finding files and entries

See Also
Browsers
Browser toolbar

Finding files and entries

The browser can search for files and entries that match various criteria that you specify. This is useful when you are looking for certain formulas, parameter sets, or gradients, but you do not know their exact name or location.

To search for files and entries, click **Find Entries** on the Edit menu. This will open the Find Entries dialog box.

The Find Entries dialog box allows you to search for parameter sets, gradients, and formulas by title, comments, and identifier. You can also specify selected files and folders to search.

For parameter sets, you can also search for authors and formulas (identifiers and files) that are used.

Click **Find Now** to start the search. This will populate the list in the dialog box with results. Click a result to open it in the browser. The Find Entries dialog box will stay on top of the browser until you close it.

Tip: When you open the Find Entries dialog box while selecting a class for a class parameter, it will only display classes that are valid for that class parameter. If you don't enter any criteria, the list of results will show all matching classes in the formula library, which can be very useful.

Next: Formula ratings

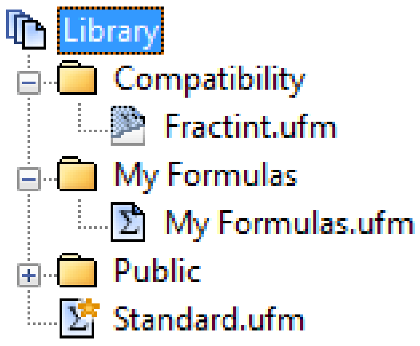
See Also
Browsers
Browser toolbar

Formula ratings

To help you to quickly see which formulas you should try first, Ultra Fractal contains a simple formula rating system. This divides all formulas into three categories:

- **Recommended**
  These formulas are the most versatile, or easiest to use. Try these first.
- **Average**
  All other formulas. They may be very worthwhile, but are perhaps not the first choice when you are new to Ultra Fractal or to a certain formula file.
- **Not recommended**
  These formulas may be obsolete or replaced by a newer formula.

Both individual formulas and entire formula files can be rated. In the browser, the formula icon shows the rating, as shown below:



Recommended formulas, such as Standard.ufm, have a little star in their icon, while formulas that are not recommended (Fractint.ufm) are greyed out slightly. All other formulas have their normal icon, such as My Formulas.ufm in the screen shot above.

By default, the browser groups formulas by rating. Click **Group By Rating** on the View menu to turn this on or off.

Ratings for formulas are updated automatically when you update the public formulas. However, you can also add your own ratings which will not be overridden by the public ratings. Select the items to want to rate in the browser, **right-click** them to open a pop-up menu, click **Rating**, and then click the desired rating.

How do formula ratings work?

Formula ratings are stored in two text files: the **private ratings file** in the main Formulas folder, and the **public ratings file** in the Formulas\Public folder. Both are called Ratings.txt. The public ratings file is automatically updated when new public formulas are downloaded. The private ratings file contains your own ratings that you have added via the Rating menu item of the pop-up menu in the browser.

In addition, formulas can also store their own rating. This is a way for formula authors to indicate which are recommended formulas in their formula files. (See the rating setting.)

If you have rated a formula yourself (so it is in the private ratings file), Ultra Fractal uses that. Otherwise, it uses the rating that is stored in the formula file. If the formula file doesn't contain a rating, it uses the rating from the public ratings file. This means that formula authors can override the public ratings in their own formula files, and you can in turn override those ratings.

For formula files, this works the same, except that there are no ratings stored in the formula file: there are only the private and public rating files. In this case, Ultra Fractal uses the public ratings unless you have defined your own rating.

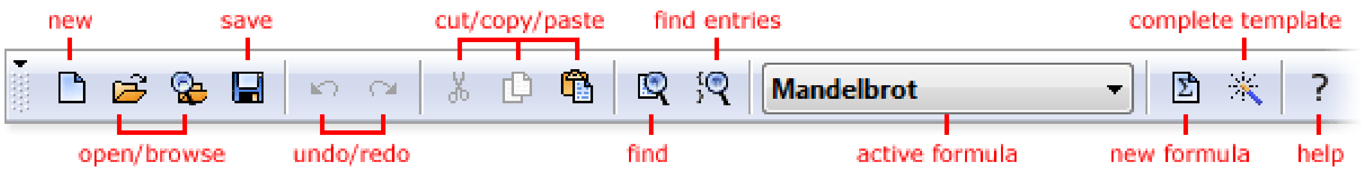
Currently, the public ratings are created by the formula database administrator, so this is not a public voting system. If you feel your formulas are rated incorrectly, please contact the rating administrator via ratings@formulas.ultrafractal.com.

See Also
Browsers
Formulas
Public formulas
rating setting

Formula editors

Ultra Fractal contains a built-in formula editor. It is a powerful text editor with syntax highlighting and extra productivity features for writing formulas efficiently.

The toolbar contains commands to edit and save the file you are working on:



- The **New** button creates a new fractal from scratch.
- The **Open** and **Browse** buttons open files from disk.
- The **Save** button saves your changes to disk.
- The **Undo** and **Redo** buttons undo changes you have made to the file.
- The **Cut**, **Copy**, and **Paste** buttons are used to move and copy blocks of text. See Editing formulas.
- The **Find** button opens a standard Find dialog where you can search for text.
- The **Find Entries** button opens a dialog where you can search for formulas in the file. See Finding text and formulas.
- The **Active Formula** drop-down box shows the formulas in the file and allows you to quickly jump to any formula.
- The **New Formula** button adds a new empty formula to the file.
- The **Complete Template** button completes the editor template at the cursor position. See Templates.

The commands on the toolbar are duplicated on the File, Edit, and Insert pull-down menus. Frequently used commands are also on the menu that pops up when you right-click inside the editor.

Next: Editing formulas

See Also
Keyboard shortcuts for formula editors
Fractal formulas
Coloring algorithms
Transformations

Editing formulas

To edit a formula, open it in the built-in formula editor. Either:

Click the **Edit** button on the Mapping, Formula, Inside, or Outside tabs on the Layer Properties tool window to edit the selected transformation, fractal formula, or coloring algorithm. To edit a class, hold down the Browse button for the class parameter and then click Edit.

Click **Browse** on the File menu to open a new browser, and double-click on a formula to edit it.

Click **Open** on the File menu and select a formula file to open it in the formula editor.

The editor works similar to other Windows text editors and supports common text editor features:

Click **Cut** on the Edit menu to move the selected text block to the Clipboard.

Click **Copy** on the Edit menu to copy the selected text block to the Clipboard.

Click **Paste** on the Edit menu to insert the text on the Clipboard into the file at the position of the cursor.

Click **Undo** on the Edit menu to undo your last change.

Click **Redo** on the Edit menu to cancel the last Undo command.

If you are editing a formula that is being used by an open fractal window, you can easily view your changes:

Click the **Reload** button on the Mapping, Formula, Inside, or Outside tabs on the Layer Properties tool window to save the formula file, recompile the formula, and recalculate the layer in one step.

The formula editor makes it easy to navigate through large formula files:

- The drop-down input box in the toolbar lists all formulas in the file, sorted by identifier. Simply click on a formula to jump to its first line.
- Within a formula, click **Next Section** or **Previous Section** on the Edit menu to go to different sections in the formula quickly.
- Click **Go to Line** on the Edit menu to jump to a specific line number in the file.

Notes

- Press **F1** or click **Help** on the Help menu to get help on the word at the cursor position. This works for reserved words, built-in functions, predefined symbols, settings, compiler directives, and labels.
- Click **Check Class Syntax** on the File menu to check all classes in the formula file for errors. Errors and warnings are displayed in the Compiler Messages tool window. (To check formulas in a file for errors, just load the file in the browser in thumbnail mode to see which formulas will not load.)
- In the status bar, the status of the file and the current row and column are displayed while you type.
- You can change the colors for various syntax elements in the formula editor on the Syntax tab of the Options dialog.

Next: Finding text and formulas

See Also

Formula editors
Keyboard shortcuts for formula editors
Writing formulas

Finding text and formulas

The formula editor provides various ways of finding text and formulas within formula files:

Click **Find** on the Edit menu to search for text using a standard Find dialog. Click on the Find Next button to start the search from the cursor position. The first occurrence of the text will be highlighted. Since the Find dialog will stay on top of the editor, you can keep it open while you edit the file.

Click **Replace** on the Edit menu to search for and replace text.

Click **Find Formulas** on the Edit menu to open the Find Formulas dialog. This dialog allows you to search for formulas within the file, based on various criteria. Click the Find Now button to start the search. Click a formula in the list of results to jump to it in the editor, where you can review and edit it. The Find Formulas dialog will stay on top of the editor until you close it.

Next: Indenting and commenting

See Also
Formula editors
Keyboard shortcuts for formula editors
Writing formulas

Indenting and commenting

The formula editor can indent and outdent blocks of code for you. Indentation is used to indicate the logical structure of code, to make formulas easier to read.

- Click **Indent Block** on the Edit menu to move the selected lines to the right. This will add a space at the start of each line.
- Click **Outdent Block** on the Edit menu to move the selected lines to the left. This will remove a space from each line, if there is one.

You can also easily comment and uncomment blocks of code. The compiler will ignore commented code, so this is an easy way to (temporarily) disable parts of a formula.

- Click **Comment Block** on the Edit menu to comment the selected lines. This will put a semicolon and a space at the start of each line.
- Click **Uncomment Block** on the Edit menu to uncomment the selected lines. This will remove a semicolon and a space from each line, if any.

Next: Templates

See Also
Formula editors
Keyboard shortcuts for formula editors
Writing formulas

Templates

To write formulas efficiently, you can use editor templates. An editor template is a commonly used piece of code that can be inserted easily.

Click **Complete Template** on the Insert menu (or press Ctrl+J) to expand the pattern at the cursor position to a template.

For example, to insert a parameter definition, enter "p" and press Ctrl+J. The pattern "p" will be expanded to a default parameter definition:

```
param |
  caption = ""
endparam
```

where | indicates the position of the cursor, so you're ready to type the name of the parameter.

If the pattern is not unique (for example when there are multiple patterns starting with "p"), a dialog box is shown where you can select the desired template.

Go to the Editor tab in the Options dialog to see the default patterns and templates and customize them.

See Also
Formula editors
Keyboard shortcuts for formula editors
Writing formulas

Exporting and rendering

To use the artwork that you create in Ultra Fractal for printing or on the web, you have to export it or render it to disk. This will create a bitmap image of the fractal, ready for further processing.

- You can directly export the image from a fractal window to a bitmap image. Click **Export Image** on the File menu, enter a file name, and click Save. See File formats for a description of the image file formats that are supported.

Usually though, you will want to render your artwork to disk. Rendering to disk provides the following benefits:

- Anti-aliasing for improved image quality. This is especially important if you are preparing images for the web. See Anti-aliasing.
- More accurate color blending and merging. When a fractal is rendered to disk, the colors are calculated more accurately than the fractal window does, producing smoother, higher quality images.
- Support for large images. You can render images up to 100,000 by 100,000 pixels. Because the images are rendered to disk, the size is not limited by available memory (RAM).
- Entire parameter files can be rendered to disk with one command, producing multiple images.

For animations, rendering is essential, since these cannot be exported normally.

Notes

- Exporting and rendering a fractal only saves the fractal as a bitmap image. You cannot open this image in Ultra Fractal. Always save your fractal as a parameter set or a fractal file as well to be able to open it in Ultra Fractal later.
- To export a fractal as an image, you can also click **Copy Image** on the Edit menu. See Copying and pasting fractals.
- If you are using an evaluation copy of Ultra Fractal, exported and rendered images will be marked with *Evaluation Copy* text. To fully enable exporting and rendering images, you must purchase your copy of Ultra Fractal. See Purchasing Ultra Fractal.

Next: Rendering images

See Also
Tutorial: Learning about transformations
Tutorial: Masking
Render jobs
Fractal windows

Rendering images

Rendering fractals to disk is the preferred way of exporting your artwork to bitmap images, ready for printing or publishing on the web.

To render the fractal in an open fractal window to disk, click **Render to Disk** on the Fractal menu.

Click the **Add** button on the Render to Disk tool window to render a parameter set to disk. Hold down the button and click **Add Fractal** to specify a fractal file to render.

This will open the Render to Disk dialog box. Here, you can specify a file name and file format for the image, the desired size and resolution, and the anti-aliasing settings. To get help on a control in the dialog box, click the **?** button in the title bar, and then click the control.

Click OK to start rendering the fractal. This will create a new render job that starts calculating the image in the background. Render jobs can be monitored and managed in the Render to Disk tool window. See Render jobs.

Notes

- You can also render fractals and parameter sets directly from the browser by right-clicking them and clicking **Render to Disk**, or by dragging them from the browser to the Render to Disk tool window. See Opening files and entries.
- The resolution specified when starting a render job is used to calculate the desired size in pixels if you enter the width and height of the image in cm or inches. See Resolution.
- When rendering very large images, you can use the **Split into tiles** option to automatically split the rendered image into multiple rectangular tiles, labeled A1, A2, B1, B2, and so on. This helps to keep the size of the individual image files at a reasonable level for any post-processing. You can later stitch the tiles together with software such as Adobe Photoshop.
- Check the **Force Linear drawing method** option for the highest image quality. This forces Ultra Fractal to use the One-pass linear option as the drawing method in all layers of the fractal to be rendered. In this way, you can work on the fractal with the Guessing drawing method for convenience without having to change the drawing method for a final render.
- Always save your fractals as a parameter set or a fractal file as well. Rendered images do not contain any fractal information and Ultra Fractal will not be able to open them later.

Next: Rendering animations

See Also
Exporting and rendering

Rendering animations

Note: You need Ultra Fractal Animation Edition to render animations.

To create a final version of a fractal animation, you have to render it to disk. This works the same as rendering images, except that the Render to Disk dialog shows additional options that are specific to animations.

To render the animation in an open fractal window to disk, click **Render to Disk** on the Fractal menu.

The Render to Disk dialog opens. Here, you can specify a file and file format for the animation. You can either render the animation to an AVI movie, or to a sequence of bitmap images in any format. It is recommended to render to a sequence of images for final renders. When you later want to compress these to a single movie (for example in MPEG format) with programs such as VirtualDub, it is much less time-consuming to experiment with different compression settings.

Apart from the regular settings such as the desired size and resolution and the anti-aliasing settings, the dialog also provides specific settings for animations. You can select which frames you want to render: the entire animation, the current frame, or a consecutive range of frames. When you are rendering to a sequence of images, you can optionally offset the frame number of the generated images. This is useful if you are dividing a long animation into several parts that are rendered individually. To get help on a control, click the **?** button in the title bar, and then click the control.

You can also apply motion blur to the rendered animation. The motion blur feature examines the amount of movement in each frame and blurs it accordingly, like a film camera would. This makes the movement in animations more natural and convincing.

Click OK to start rendering the animation. This will create a new render job that starts calculating the image in the background. Render jobs can be monitored and managed in the Render to Disk tool window. See Render jobs.

Notes

- It is recommended to use both anti-aliasing and motion blur when rendering animations to reduce the amount of jumping and flickering pixels.
- Motion blur is calculated directly from the coordinate movement that occurs when the fractal is zoomed or rotated, and therefore it is efficient to calculate and fairly accurate. The downside is that it only works with zooming, panning, rotating, and so on, and not with movements that are the result of changes to other parameters.
- Motion blur needs extra memory and temporary disk space, and because of this there is a limit to the maximum size of the image that does not normally apply to disk rendering. For example, a 1024x768 image takes 24 MB and a 4096x4096 image takes 512 MB of RAM.

Next: Rendering parameter files

See Also

Animation
Working with animations
Exporting and rendering

Rendering parameter files

You may sometimes wish to render all images and animations in a parameter file to disk. Instead of rendering each image separately, you can render the entire parameter file to disk with one command.

Hold down the **Add** button on the Render to Disk tool window, and click **Add Parameter File**. Select a parameter file to render in the dialog box that appears, and click Open.

This will open the Render Parameter File to Disk dialog box. Here, you can specify a folder that will contain the rendered bitmap images, the file format, the desired size and resolution, and the anti-aliasing settings.

You can choose to render all images in the parameter file, or only selected images, or only the images that do not yet exist in the destination folder (useful to update a folder that holds the images for a parameter file if you have added new parameter sets).

In addition, you can select if you only want to render the still images or the animations in the parameter file, or both. For example, you can render the still images in PNG format and the animations in AVI format by rendering the parameter file twice, selecting the Only Stills option the first time, and Only Animations the second time, with appropriate file format settings.

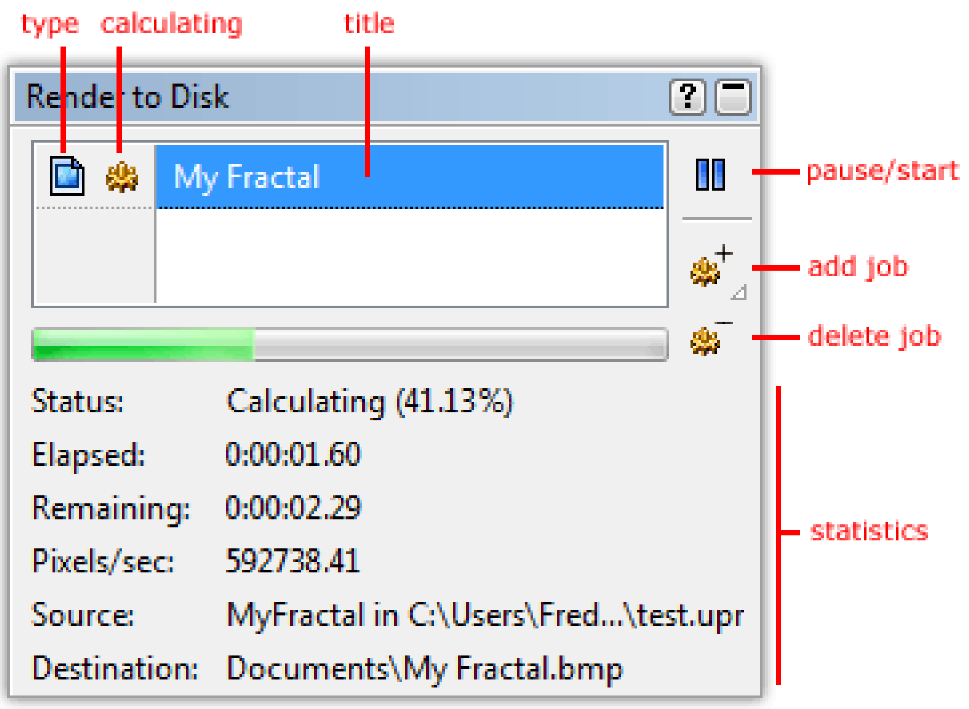
Click OK to start rendering the parameter file. This will create a new render job that starts calculating the images in the background. Render jobs can be monitored and managed in the Render to Disk tool window.

Next: Render jobs

See Also
Exporting and rendering
Rendering images
Rendering animations

Render jobs

When you start rendering a fractal or a parameter file, a render job is created that performs the requested calculations in the background. Render jobs are monitored and managed in the Render to Disk tool window.



Status: Calculating (41.13%)
Elapsed: 0:00:01.60
Remaining: 0:00:02.29
Pixels/sec: 592738.41
Source: MyFractal in C:\Users\Fred...\test.upr
Destination: Documents\My Fractal.bmp

The tool window contains a list of render jobs. Below the list, the status of the selected job is shown, including various statistics.

- Click the **Pause/Start** button to pause the selected job. To resume it, click the button again.
- Click or hold down the **Add** button to add a new job. See Rendering images and Rendering parameter files.
- Click the **Delete** button to cancel and delete the selected job.

The icons before each job in the list show its type (single image or parameter file), and whether it is currently calculating.

By default, only the job at the top is calculating. When it is finished, the next job in the queue is started. New jobs are added at the bottom, and therefore will be started automatically when all the other jobs have been completed.

To start another job, select a job in the list that is not calculating and click the **Pause/Start** button to start it. When this job is finished, it will start the next job in the queue, too.

You can reorder the jobs in the queue by dragging them up or down to adjust the order in which they will be calculated. For example, if you want a job to stay paused while the other jobs are calculated, drag it to the top of the list, so it will not be started automatically.

Notes

- To get help on a specific statistic, hover the mouse cursor over it while the Fractal Mode tool

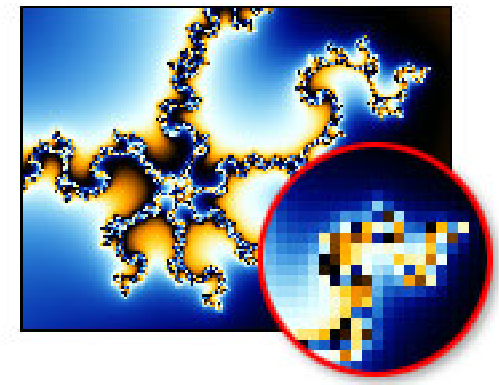
window is open.
- When you hide or close the tool window, the jobs will continue to calculate in the background. When you close Ultra Fractal, the jobs will be paused. They will be resumed automatically when you start Ultra Fractal again.
- Although Ultra Fractal is carefully designed to resume jobs correctly in case of a power failure or computer crash, you may want to back up time-consuming render jobs to be absolutely safe. To back up a render job, right-click it in the list and click **Backup Job**. This will save all calculated data to a single file (*.urj). To restore a job, hold down the **Add** button and click **Restore Job**.

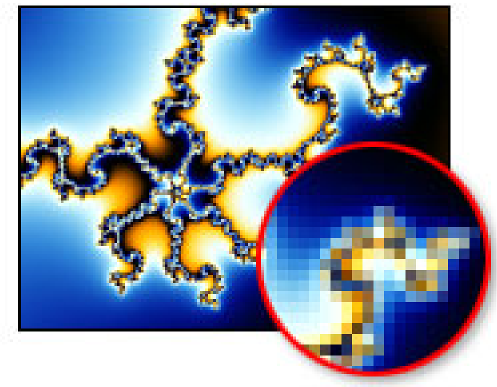
Next: Anti-aliasing

See Also
Exporting and rendering

Anti-aliasing

One of the reasons to render fractals to disk is to be able to use anti-aliasing. Anti-aliasing improves the quality of rendered images by sharpening and smoothening them, removing jagged edges.


No anti-aliasing


Normal anti-aliasing

Anti-aliasing increases the time it takes to render the image. The effect of anti-aliasing and the extra time required depends on the anti-aliasing settings you use when starting a render job (see Rendering images and Rendering parameter files).

The following settings are available:

Anti-aliasing	Selects common anti-aliasing settings. • **Off** turns anti-aliasing off completely. • **Quick** selects minimal anti-aliasing settings that calculate relatively quickly. • **Normal** selects normal anti-aliasing settings that provide good quality and reasonable calculation times. • **Non-adaptive** turns off adaptive anti-aliasing. This gives better results for some images, but it requires (much) longer to calculate. • **Custom** allows you to specify your own anti-aliasing settings.
Threshold	Specifies the threshold to use for adaptive anti-aliasing. Ultra Fractal anti-aliases a pixel only when the difference between the pixel and its neighbors (red + green + blue) is larger than or equal to the threshold. Use 0 to turn adaptive anti-aliasing off. This slows the calculation down, but is required if adaptive anti-aliasing does not work well (for example when the fractal contains many thin lines, such as with the Embossed fractal formulas).
Depth	Specifies the anti-aliasing depth (1 or greater). Greater depths give better quality at the expense of calculation time. Increasing the depth by one can easily double or triple the calculation time. The default value of 1 is usually sufficient.
Subdivisions	Selects how pixels are subdivided for anti-aliasing. The default value 9 (3x3) gives better quality and is recommended. Use 4 (2x2) only for quick renders, or when the depth is greater than 1.

The **Normal** setting is recommended, and will usually give the best results. If you are preparing images for printing, it may not be necessary to use anti-aliasing. This is because on prints, individual pixels are rarely visible due to the much higher resolution.

Next: File formats

See Also
Exporting and rendering

File formats

Ultra Fractal supports various image file formats for exporting and rendering images. You can select the file format when selecting the file name of the exported image and when starting a new render job (see Rendering images, Rendering animations, and Rendering parameter files).

The following file formats are supported:

Bitmap image	Saves the image as a Windows bitmap image (*.bmp). This format is supported by almost all Windows graphics programs.
Photoshop image	Saves the image as an Adobe Photoshop image (*.psd). This allows you to save layers individually, so they can be post-processed. This file format also supports transparent images. Note: Currently, layer groups are not preserved in the Photoshop file. Instead, the fractal is exported as a linear list of layers, and any masks attached to layer groups are applied to the last layer of the group. For this reason, it is a good idea to turn group masks into normal layers first, so you could re-apply them as mask later in Photoshop.
PNG image	Saves the image as a Portable Network Graphics image (*.png). This file format is readable by many graphics programs and supports lossless compression and transparent images.
JPEG image	Saves the image as a JPEG image (*.jpg). This format offers very good compression. You can set the quality of the saved image to adjust the file size. A value of 95% will usually give good results. Do not use this format if you want the best possible image quality.
Targa image	Saves the image as a Targa image (*.tga). This is a common format for high-end graphics programs, such as raytracers and 3D packages. It supports transparent images.
TIFF image	Saves the image as a TIFF image (*.tif). This format is often used by print shops and graphics designers working with Apple computers. It also supports transparent images.
AVI movie	Only for animations. Saves the animation as a Windows AVI movie (*.avi) with a selectable codec. Only codecs with a Video for Windows interface are supported. DirectX-only codecs are not supported. If you want the best quality, it is recommended to render animations to bitmap sequences instead, and compress them later with third-party software such as VirtualDub.

When you render an animation to a format other than AVI, it will be rendered as a sequence of bitmap images.

Next: Resolution

See Also

Exporting and rendering

File types

Resolution

In Ultra Fractal, you can set a resolution value for your fractals in the Fractal Properties tool window, and in the Render to Disk dialog. The resolution specifies the relation between the logical size of the fractal in pixels, and its physical size in centimeters or inches. Normally, the resolution is set in DPI (dots per inch), so the logical size and the physical size are related by this formula:

LogicalSize [Pixels] = **PhysicalSize** [Inches] * **Resolution** [DPI]

For example, a 1600x1200 fractal at 300 DPI is 5.33x4 inch or 13.5x10.2 cm. If you are going to print your fractal, you probably know the resolution that your printer uses. If you type that in the Render to Disk dialog, for example, you can directly set the size in centimeters or inches.

Ultra Fractal always uses pixels internally. It will just use the resolution value to convert physical sizes to pixel values, and it will store the resolution value with the exported or rendered images. Resolution values are not supported by the JPEG and Targa file formats.

It is important to realize that the only thing that really matters is the logical size in pixels. You can always open a rendered image in Adobe Photoshop, for example, and change its resolution, which will "magically" change the physical size as well. The ability to specify the resolution in Ultra Fractal saves you from having to convert the desired physical size to a logical size in pixels yourself, but remember that it is just a placeholder value.

See Also
Exporting and rendering

Network calculations

Note: You need Ultra Fractal Animation Edition for network calculations.

Ultra Fractal enables you to distribute calculations over multiple computers connected with a network. This can greatly increase the speed at which complex fractals are calculated, especially in combination with deep zooming.

Network calculations work by running a separate program called **Ultra Fractal Server** on remote computers, and creating connections to those computers with the Network tool window in Ultra Fractal. One server can accept multiple connections from different computers running Ultra Fractal, and one computer running Ultra Fractal can create connections to multiple servers.

When you have successfully created one or more connections, calculations will immediately be divided among all available computers. You can add or remove connections at any time. Both calculations performed by fractal windows and calculations performed by render jobs can be distributed.

Ultra Fractal uses the TCP/IP protocol for network calculations, so you can connect to computers both on a local network and on the Internet.

Next: Network servers

See Also
Connections
Fractal windows
Exporting and rendering

Network servers

To be able to connect to a remote computer, the Ultra Fractal Server program must be running on that computer. The server accepts connections from other computers running Ultra Fractal and performs requested calculations.

To start the server, click Programs on the Start menu, and then click **Ultra Fractal 5 Server**. (The name can differ slightly depending on the version number.)

To run the server on another computer, you can install Ultra Fractal on that computer. Alternatively, share the drive that Ultra Fractal is installed on (with Windows Explorer) so you can access it from other computers, and then start the Server.exe program located on your own computer in the Ultra Fractal folder from a remote computer.

The server shows a list with the current connections and a log that displays all activity. In the status bar, the number of connections and the current IP address are shown.

By default, the server listens on port 8691 for connections, but you can change this if it causes conflicts with other programs. If there is a firewall on the server computer, you need to configure it to allow incoming connections on this port.

Click **Options** on the File menu of the server to set connection and security options for the server. See Security.

Notes

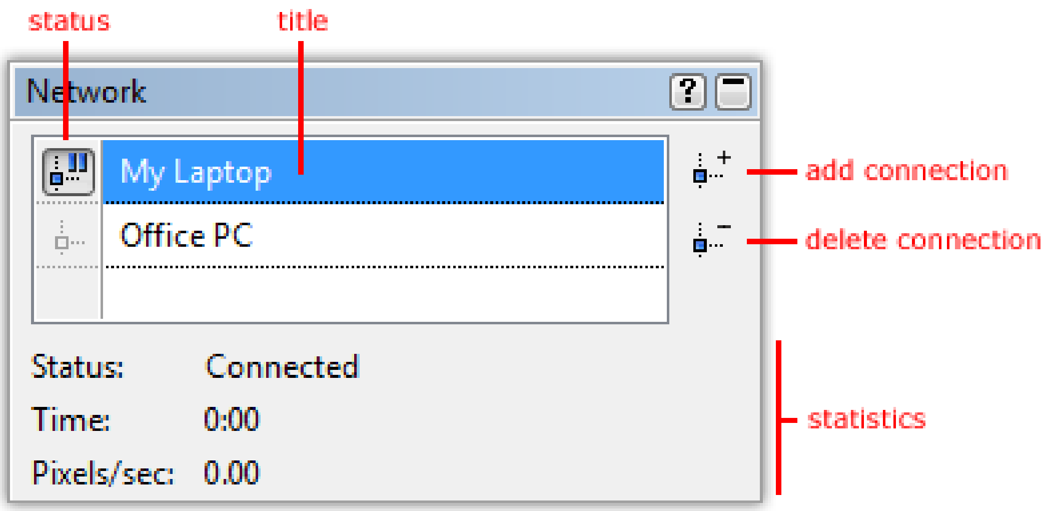
- If you want to keep the server running continously on a remote computer, create a shortcut to it in the Startup folder in the Start menu. By minimizing the server window, it will run unobtrusively in the taskbar tray area.
- You do not need an extra license to run Ultra Fractal Server on other computers.
- The computers that run Ultra Fractal Server do not require any fractal formulas, so you do not need to have an updated collection of formulas on those computers.

Next: Connections

See Also
Network calculations

Connections

In the Network tool window, you can create and manage connections to other computers running Ultra Fractal Server.



Status: Connected
Time: 0:00
Pixels/sec: 0.00

- The **Add Connection** button creates a new connection. This will open a dialog box where you can enter the address of the computer to connect to. To get help on a control in this dialog box, click the **?** button in the title bar, and then click the control.
- The **Delete Connection** button deletes the selected connection.
- The **status** icon before each connection shows its current status. Click on the icon to disable and enable the connection.
- In the list, the **titles** of the connections are displayed. Click on the title of the selected connection to rename it. Double-click a connection to edit its properties. You can drag connections up or down to organize them.
- Below the list, various **statistics** on the selected connection are displayed.

Right-click inside the list to open a pop-up menu with frequently used commands.

Next: Tips

See Also
Network calculations

Tips

The following tips will help you to use network calculations effectively.

- Network calculations work best with images that are slow to calculate. Especially deep-zoomed fractals will benefit. If the fractal calculates relatively quickly, the network communication overhead can sometimes outweigh the extra calculation speed.
- Large images do not necessarily benefit more from network calculations than small images.
- Connect to local computers rather than to computers on the Internet if you can help it, because the communication overhead is likely to be much smaller on a local network (LAN). On a LAN, use 100 Mbps or Gpbs Ethernet instead of 10 Mbps.
- If you can, avoid using a personal firewall because it might make the network calculations less efficient.
- Try to connect to remote computers that are roughly as fast as your own computer. It does not help much to connect to slower computers.

Next: Security

See Also
Network calculations
Network servers

Security

By default, a computer that runs Ultra Fractal Server accepts connections from any user on any computer. However, you have several options to restrict access to the server.

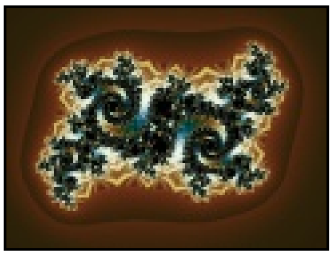
Click **Options** on the File menu of the server, and then click the Security tab to set security options.

- You can restrict access to selected IP addresses or ranges of IP addresses. Using this feature, you can for example only allow users on your local network to connect to the server.
- You can require users to supply a password when creating a connection to the server. Users without a valid password will not be able to create a connection. Passwords are transmitted securely.

In the log at the bottom of the server window, you can see all connection activity. The log also shows incoming connections that were not accepted because of a blocked IP address or an invalid password. The log can also be written to a file so you can analyze it later.

See Also
Network calculations
Network servers

Writing formulas



In Ultra Fractal, every part of the fractal calculation process is controlled by formulas. There are three types of formulas: fractal formulas, coloring algorithms, and transformations. A formula can be seen as a small, specialized computer program that is compiled and executed by Ultra Fractal.

By writing your own formulas, you can completely customize how a fractal is calculated. It is recommended to first learn how to use Ultra Fractal and get comfortable with it before you start writing formulas.

This chapter explains how to write your own formulas. It is divided into five sections:

- The **Language basics** section introduces the syntax and elements of the formula language. You should study this first.
- The **Functions and classes** section describes how to declare and use your own functions and classes, including inheritance and class parameters.
- The **Formulas** section shows how to use the formula language in practice to write fractal formulas, coloring algorithms, and transformations.
- The **Reference** section describes all operators, built-in functions, predefined symbols, etc. that are available.
- The **Tips** section contains additional information on debugging and publishing your formulas.

Use the Contents tab to navigate to topics in these sections.

Next: Creating a new formula

See Also
Fractal formulas
Coloring algorithms
Transformations
Copyright and tweaking

Creating a new formula

This topic shows step by step how to create your first fractal formula, a basic Mandelbrot set.

1. Create a new fractal. It does not matter which formula is selected, since it will be replaced by your own.
2. Click **New** on the **File** menu, and then click **Fractal Formula File**. The formula editor appears with an empty file.
3. Click **New Formula** on the **Insert** menu. Enter "My Mandelbrot" as the title and click OK. The new entry should now appear in the formula editor.
4. After the init: label, insert the following line:
   ```
   z = 0
   ```
   This initializes the complex variable z to (0,0).
5. After the loop: label, insert the following line:
   ```
   z = z * z + #pixel
   ```
   This is the main equation for the Mandelbrot set. #pixel refers to the coordinates of the pixel being computed and will be different for every pixel. The statements in the loop section will be executed repeatedly.
6. After the bailout: label, insert the following line:
   ```
   |z| < 4
   ```
   This defines when Ultra Fractal should stop repeating (or iterating) the statements in the loop section. With this condition, the loop section will be iterated as long as |z| (equal to sqr(real(z)) + sqr(imag(z))) remains smaller than (4,0).
7. The formula should now look like this:

   ```
   MyMandelbrot {
   init:
     z = 0
   loop:
     z = z * z + #pixel
   bailout:
     |z| < 4
   default:
     title = "My Mandelbrot"
   }
   ```

8. Save the new formula by clicking **Save As** on the **File** menu. Enter "My Formulas.ufm" as the filename and click Save.
9. Now, click the **Browse** button in the Formula tab of the Layer Properties tool window. Select "My Mandelbrot" from the file "My Formulas" and click OK.
10. Congratulations! You have just created your first fractal formula.

Notes

- If compiling the formula results in compiler errors, make sure you have entered the formula correctly. Highlight the error in the Compiler Messages tool window and click the **Trace** button to see where the first error occurred. Correct the error and click the **Reload** button in the Formula tab to try again (changes in the formula will be saved automatically).
- Try to experiment with the loop section. For example, change z * z into z * z * z and see what happens. Click the **Reload** button in the Formula tab to reload the formula after you have changed it.

Next: Formula files and entries

See Also

Fractal formulas

Formula editors

Formula files and entries

Formula files are plain text files with the extensions .ufm (fractal formula files), .ucl (coloring algorithm files), or .uxf (transformation files). A formula file can contain any number of formulas (called entries).

Each entry starts with an entry identifier, followed by the contents of the entry between curly brackets { and }. After the entry identifier, an optional value between parentheses can be added. Examples:

```
My-Formula {
; entry contents
}

Mandelbrot(XAXIS) {
; this formula uses an optional setting
}

comment {
  The comment entry always consists of comments.
}
```

The entry identifier may consist of almost all characters, except spaces and tabs, of course. Since you can specify an additional descriptive title for the formula, the identifier can be cryptic as the user never notices it (but it is shown by the browser). The identifier is used to distinguish between formulas in the same file, so no two formulas in the same file can have the same identifier.

The semicolon ; is used to add comments to a formula. After a semicolon, the rest of the line is ignored by the compiler. You can also add global comments like copyright information to a file by placing them inside a special *comment { }* entry, which is also ignored (but the browser shows it when the file is selected).

To facilitate working with entries, the built-in formula editor can automatically create new entries and search for existing entries. See Formula editors.

Next: Sections

See Also
Creating a new formula
Fractal formulas
Inheritance

Sections

Each entry in a formula file is divided into one or more sections. It depends on the formula type which sections are supported and what they can contain. This topic describes sections in general. For specific information about a formula type, see Transformations, Fractal formulas, and Coloring algorithms.

Here is an example of a fractal formula:

```
MyMandelbrot {
init:
  z = 0
loop:
  z = sqr(z) + #pixel
bailout:
  |z| < 4
default:
  title = "My Mandelbrot"
}
```

This formula contains four sections. There are three types of sections:

- Sections containing statements. These statements can be executed by Ultra Fractal; how and when this happens depends on the particular section. Each statement must be on a separate line, or they must be separated by commas. Examples of these sections are **init** and **loop**.
- Sections containing a boolean expression. There's only one section of this type, the **bailout** section.
- Sections containing settings related to the formula. Settings always have the form "setting = value". An example is the **default** section.

As you can see, sections are divided by labels. A label is just the name of the section followed by a colon.

For compatibility reasons, it is sometimes allowed to omit the label in some sections. For more information, see the formula-type specific documentation.

You can break long statements and settings across several lines using the backslash \ character. The backslash character must be the last character on the line for this to work. Any spaces or tab characters on the next line are removed, so if you want to add one or more spaces, you must put them on the previous line (the line containing the backslash). Example:

```
default:
  title = "A very long title \
           for my \
           favorite formula"
```

This is equal to:

```
default:
  title = "A very long title for my favorite formula"
```

Next: Expressions

See Also

Formula files and entries
Global sections
Member visibility

Expressions

To recap briefly, a formula is divided into multiple sections, separated by labels. Some sections (for example **init** and **loop**) can contain statements. Most statements are just expressions, and this topic explains what expressions really are.

Almost everything is an expression. Variables, parameters and constants are all expressions. By using operators or functions, expressions can be composed of one or two subexpressions. Examples:

```
a
3
3 + 2
sin(a)
(3 + sin(a)) / 2
```

These are all valid expressions (note that "a" is a variable). The important property of an expression is that you can always calculate its value, no matter how complicated the expression is. Moreover, you can do something with this value, for example assigning it to a variable by using the = (assignment) operator:

```
b = 3 + 2
b = (3 + sin(a)) / 2
```

These expressions are called assignments. It is important to realize that assignments are expressions themselves, so this is also a valid expression:

```
c = b = 3 + 2
```

This gives both c and b the value 5.

Now, remember that statements can be expressions, so that means any expression is a valid statement. Of course, it does not make sense to use expressions like "3" or "(3 + sin(a)) / 2" as statements, since nothing is done with the value of the expression, so the statement is ignored by the compiler (and results in a warning message). Only assignments actually use the value of the expression, therefore statements usually are assignments.

Note: do not confuse the = (assignment) operator with the == (equality) operator. The == operator is used to test if two expressions are equal, and returns a boolean value (**true** or **false**).

Next: Types

See Also
Sections
Operators
Built-in functions

Types

To write formulas, you need to be aware of the concept of types. Expressions (constants, variables, parameters, etc.) in formulas can have different types. There are five different built-in types. The following table shows each built-in type and explains its use.

bool	Boolean expressions can only have two values: **true** or **false**. This type is returned when expressions are compared: 3 < 4 ; **true** **false** == **true** ; **false**
int	Integers are useful for counting. They can have values from -2147483648 to 2147483647. Most arithmetic operations can be performed on integers. Example: 3 -2
float	Floating-point numbers are the most familiar type of numbers. They can be very small or large, and have virtually unlimited precision (see Arbitrary precision). Their greatest benefit is that they can represent fractional values. Examples: 3.1 -2.9 1.3e7
complex	Complex expressions represent complex numbers. They consist of two floating-point numbers and are used to perform complex arithmetic, which is very useful for fractal calculations. Examples: (2.8, 4) 2.8 + 4i
color	Color expressions represent a color. You can perform operations on colors and store them in color variables. Internally, a color is stored as four floating-point values, corresponding to red, green, blue, and alpha (opacity) components. Each value ranges from 0 to 1. Colors are intended only for direct coloring algorithms.

Note: You can add new types yourself by declaring classes. This is not necessary when you are beginning to write formulas, though.

Next: Constants

See Also
Expressions
Type compatibility
Operators

Functions

Constants

Constants are used in formulas to specify fixed values. Example:

```
x = 3 * x + 4
```

Here, 3 and 4 are constants. This topic explains how constants are used in Ultra Fractal formulas, and how Ultra Fractal determines the type of a constant (you should be aware of this).

Boolean constants are of type bool. There are only two boolean constants: **true** and **false**.

Integer constants are of type int. Integer constants are signed numbers within the range -2147483648 .. 2147483647. Examples:

```
5
-23
```

Floating-point constants are of type float. They consist of a signed mantissa, optionally followed by the character E and a signed integer exponent to denote a power of ten. The exponent may range from -4931 to 4931. Examples:

```
3.0
-1.23482
98.283E-3  ; 0.098283
-1e5       ; -100000
```

Complex constants are of type complex. They consist of two floating-point numbers, separated by a comma and surrounded by parentheses. Alternatively, you can specify an imaginary number by typing the letter **i** right behind a normal floating-point number. Examples:

```
(2, 3)
(3e2, 3.239)  ; (300, 3.239)
2.3i          ; (0, 2.3)
2 + 1.63i     ; (2, 1.63)
```

Color constants are of type color. They are created by supplying the rgb, rgba, hsl, and hsla functions with constant arguments. Examples:

```
rgb(0.5, 1, 0)        ; orange
hsla(0, 0, 0.5, 0.5)  ; transparent red
```

Next: Variables

See Also

Expressions

Types

Variables

Formulas are often composed of several separate calculations. Therefore, it is necessary to store intermediate results in memory. To do this, you must use variables.

Identifiers are used to refer to variables. An identifier must start with a letter, followed by one or more letters, digits or the underscore character "_". Here are some examples of valid identifiers:

```
x
MyOwnVariable
var_32
```

It does not matter if the letters are lower- or uppercase, so "myvar" and "MyVar" represent the same variable. Some identifiers (called keywords) are reserved by the compiler for other purposes: you cannot use them as variables.

Before you can read from a variable, you must first write to it. To do this, use the = (assignment) operator:

```
x = 3
y = 3 + 28 / 7
```

The variable is created when it is first written to. By default, the type of the variable is complex, but you can change that by placing a type keyword (bool, int, float, complex or color) in front of the assignment. Alternatively, you can declare the variable before using it.

```
int i
i = 2
int x = 3
bool MyBooleanVariable = x > 3
```

Assignments are expressions themselves, so their result can be assigned again. This example gives both x and y the value 1:

```
x = y = 1
```

To read from a variable, use the identifier in an expression. This example reads the value in x and stores it in y:

```
y = x
```

Of course, you can also perform calculations when doing this:

```
y = x * 3 + 7
```

Next: Parameters

See Also
Arrays
Type compatibility

Parameters

Parameters are used to allow users to customize formulas without needing to rewrite them. Parameters are used just like variables, with two exceptions: you cannot write to them, and you must prefix them with the @ character (so the compiler can determine they are parameters instead of variables). Here is an example that uses a parameter:

```
Mandelbrot {
init:
  z = 0
loop:
  z = z^@power + #pixel
bailout:
  |z| < 4
}
```

The parameter used here is @power. In the Formula tab, the user can now enter a value for this parameter and view the Mandelbrot set of any power using just one formula.

By default, the type of a parameter is complex. You can change that by adding a param block to the **default** section of your formula and providing the necessary settings. Using the param block, you can also specify enumerated parameters. With enumerated parameters, the user does not have to enter a numerical value, but rather chooses an item from a drop-down list.

It is also possible to use user functions. These can be used just like normal functions, with the exception that the user can choose the actual function from a list of all available built-in functions. Here is an example with a user function:

```
Mandelbrot {
init:
  z = 0
loop:
  z = @myfunc(z) + #pixel
bailout:
  |z| < 4
}
```

Now, the user can use **sqr**, but also **sin** and **cos** and many other functions with this formula. You can customize the behavior of the user function using the func block in the **default** section.

Notes

- It is possible to write to parameters, although this is not recommended and can make your formulas run slower. It is provided for compatibility with old Fractint formulas.
- There are six predefined parameters: p1..p6, and four predefined functions: fn1..fn4. You do not need to use the @ prefix with these parameters and user functions. However, it is recommended to use your own names instead of these (with the @ prefix) to make your formulas easier to understand.

- The param and func blocks also provide settings for changing the caption, default, minimum and maximum values of parameters and user functions, as well as adding a small help text.
- Parameters are sorted alphabetically in the user interface, unless you provide a param block for each parameter. In this case, the parameters appear in the order of the param blocks in the formula file.
- In the user interface, you can also group parameters together by adding headings.
- For more information on using parameters with classes, see Class parameters.

Next: Arrays

See Also
Providing help and hints
Writing direct coloring algorithms

Arrays

Arrays are used to store multiple variables in an organized way. Before you can use an array, you must declare it. Example:

```
int myArray[8]
```

This declares a static array called myArray with 8 elements. (It is called a static array because the number of elements is fixed. To declare a dynamic array that allows the number of elements to be changed later, see Dynamic arrays.)

You can use the elements in the array like normal variables:

```
myArray[0] = 1
myArray[myArray[0]] = 2 * myArray[0] + 1
```

You access an element in an array by specifying the index of the element between square brackets [ ]. The index must be an integer expression, ranging from 0 to the number of elements - 1 (the value 0 corresponds to the first element in the array).

You can also declare and use multi-dimensional arrays by specifying multiple values between the brackets. When declaring an array, the size of each dimension must be a constant integer expression. Parameters and most predefined symbols also qualify as constants. Example:

```
float a[10, 2 * 6 - 2]
bool flags[#width, #height]
a[3, 5 - 3] = 1.5
flags[0, 0] = a[1 + 2, 2] > 0
```

You can directly assign arrays with the same size to one another:

```
color x[10]
color y[10]
x = y
```

Notes

- Unlike other types of variables, arrays are not initialized upon declaration. Be sure to explicitly initialize all elements in arrays before reading from them. They will contain random values otherwise. Reading from an array element that has not been initialized does not generate a warning either, so you have to check your formulas carefully.
- You will often use loops to iterate through the elements of an array.
- Arrays are often filled with pre-calculated values in the global section to speed up calculations.
- If you use constant array indices outside the supported bounds, the compiler will produce an

error message. If you use expressions that evaluate to an invalid index value while the formula is executed, a run-time message is written to the Compiler Messages tool window if the **debug** compiler directive is defined. See Compiler directives.

- If you assign arrays to one another that are incompatible and the size of the arrays is unknown at compile time, a run-time message occurs, too.

Next: Dynamic arrays

See Also
Types
Variables

Dynamic arrays

In addition to static arrays, you can also declare dynamic arrays. Unlike static arrays, which have a predetermined number of elements, the number of elements in a dynamic array can be changed at any time. Example:

```
int anotherArray[]
```

This declares a dynamic array called anotherArray. Before you can store anything in this array, you need to call the setLength function to set the number of elements that it can contain:

```
setLength(anotherArray, 12)
```

You can now use the elements in the dynamic array:

```
anotherArray[0] = 8
anotherArray[1] = anotherArray[0] * 2
```

Like static arrays, the index of the element can range from 0 to the number of elements - 1. The index value 0 corresponds to the first element in the array.

At any time, you can call setLength again to resize the array. The length function always returns the current size of the array:

```
setLength(anotherArray, 20)
int currentSize = length(anotherArray)  ; returns 20
```

Unlike static arrays, dynamic arrays cannot be multi-dimensional. If you need a multi-dimensional array, you can simulate it like this:

```
; Simulate a 10x10 two-dimensional array
int twoDimArray[]
setLength(twoDimArray, 10 * 10)
; Initialize the element at position [0, 0]
twoDimArray[0 * 10 + 0] = 13
; Initialize the element at position [8, 2]
twoDimArray[8 * 10 + 2] = 14
```

Also unlike static arrays, it is not possible to copy dynamic arrays with a single assignment. This will compile, but result in the Arrays are not compatible run-time error message:

```
int a[]
int b[]
setLength(a, 10)
setLength(b, 10)
a = b   ; Generates run-time error message
```

Use a loop to copy the individual elements instead.

Notes

- Unlike static arrays, the elements of a dynamic array are zero-initialized by the setLength function.
- As with static arrays, if you try to access elements of the array that are out of range, a run-time error message will be written to the Compiler Messages tool window if the **debug** compiler directive is defined.
- Dynamic arrays are slightly less efficient than static arrays, so use a static array if possible.

Next: Type compatibility

See Also
Arrays
Types

Type compatibility

As explained in Types, every expression has a type. If an expression is a variable, a constant or a parameter, its type is equal to the type of the variable, constant or parameter, of course. This topic explains what happens if an expression is composed of two subexpressions with different types.

Consider this example:

```
3 + 2.1
```

The type of "3" is int, and the type of "2.1" is float. The type of the resulting expression is always the "highest" type of the subexpressions. High means "most descriptive" here. For example, **complex** is higher than **float**, which is in turn higher than **int** (boolean expressions are treated differently). So, this means the type of the expression above must be float. Since its result should be 5.1, this behaves as you would expect.

This means, however, that "3", a constant of type int, must be converted to "3.0", a constant of type float. This conversion is performed automatically by the compiler. You should be aware of the fact that the compiler can only convert types to higher types. For example, it cannot convert a float to an int. So this statement is illegal and results in an error message:

```
int i = 3.1
```

Functions like round and real are available to perform these conversions manually, if this is necessary.

Be aware of the fact that some operators and most functions always return float or complex values. For example, this statement is illegal, since the division operator / cannot return an integer result (it returns 1.5):

```
int i = 3 / 2
```

You can look up the behavior of all operators and functions in the Reference section.

The compiler can convert booleans to other types and vice versa, but it does generate a warning for each conversion. The value **true** is converted to 1, and the **value** false to 0. An expression of another type (int, float or complex) is converted to **false** if it is equal to 0, otherwise it is converted to **true**. Please note that this is only supported for compatibility with old Fractint formulas. It is not recommended to use this in new formulas.

Colors cannot be converted automatically. Use the rgb, rgba, hsl, and hsla functions to convert floating-point values to colors. Use red, green, blue, hue, sat, lum, and alpha to convert colors to floating-point values.

For the type compatibility rules for classes, see Inheritance and Casting.

Next: Conditionals

See Also

Expressions

Variables

Conditionals

This topic describes the **if** statement. The **if** statement is used to write pieces of code that should only be executed under certain circumstances (conditions). Here is an example:

```
if a > 3
  b = 2
else
  b = 1
  a = 2
endif
```

In this code snippet, b is given the value 2 when a is greater than 3, otherwise b is given the value 1, and a is given the value 2.

Here is the complete syntax of the **if** statement:

```
if <boolean-expression>
  <statements>
[elseif <boolean-expression>
  <statements>]
[else
  <statements>]
endif
```

The parts between brackets [ ] are optional. There can be as many **elseif** blocks as you find useful. Here is an example which finds the largest of three variables a, b and c and places it in x:

```
if a > b && a > c
  x = a
elseif b > c
  x = b
else
  x = c
endif
```

The usage of **elseif** is not really necessary. The following example does exactly the same thing:

```
if a > b && a > c
  x = a
else
  if b > c
    x = b
  else
    x = c
  endif
endif
```

This example uses a nested **if** statement (an **if** statement within another **if** statement). **If** statements may be nested as much as you want.

When evaluating the boolean operators && and ||, Ultra Fractal uses *short circuit evaluation*. The expression is evaluated from left to right and Ultra Fractal stops as soon as the answer is known. This is more efficient and it allows you to write expressions like these:

```
int i = -1
float a[10]
if i >= 0 && a[i] > 2
  ...
endif
```

In this case, Ultra Fractal detects that i is negative, so it will never even try to read a[i], which is exactly what we intend because -1 would not be a legal index value for the array.

Notes

- In Fractint, it is necessary to surround the boolean expressions after if and elseif with parentheses. Ultra Fractal does not require this, but it is not harmful.
- For more information on boolean operators like &&, see Operators in the Reference section.

Next: Loops

See Also
Expressions
Types

Loops

Sometimes you may want to repeat a sequence of statements. Ultra Fractal provides two constructs to do this: the **while** loop and the **repeat** loop. Here is the syntax:

```
while <boolean-expression>
  <statements>
endwhile
```

```
repeat
  <statements>
until <boolean-expression>
```

The **while** loop repeats the statements as long as the boolean expression is **true**. If the boolean expression is or becomes **false**, the statements are never or no longer executed. The **repeat** loop, however, repeats the statements until the boolean expression becomes **true**. This means that the statements are always executed at least once.

If you want the statements to be executed at least once, use the **repeat** loop. Otherwise, use the **while** loop.

Here is an example that calculates x = n! (defined as x = 1*2*3*...*(n-2)*(n-1)*n), first using a **while** loop, and then using a **repeat** loop:

```
int n = 23   ; or another value
float x = 1  ; float because 23! is very large
while (n > 1)
  x = x * n
  n = n - 1
endwhile
```

```
int n = 23
float x = 1
if n > 1
  repeat
    x = x * n
    n = n - 1
  until n == 1
endif
```

Obviously, the **while** loop is better suited to calculate the factorial of a number, but in other cases the **repeat** loop may be better.

As with **if** statements, loops may be nested as much as you want.

Next: Functions

See Also
Conditionals
Arrays

Functions

As you write larger formulas, it becomes desirable to avoid code duplication and to have a way to better structure the formula code. To do this, you can group code in functions that you can call just like the built-in functions.

Here is a complete Mandelbrot formula that shows how to declare and use a function that calculates one iteration of the Mandelbrot set:

```
MandelbrotWithFunction {
init:
  complex func calculateMandelbrot(const complex x)
    return sqr(x) + #pixel
  endfunc

  z = 0
loop:
  z = calculateMandelbrot(z)
bailout:
  |z| < 4
}
```

- Functions start with the **func** keyword and end with the **endfunc** keyword. (These keywords are also used for function blocks.)
- The **return type** of the function is specified before the func keyword. For a function with no return value, leave out the type before the func keyword.
- Arguments to the function are declared after the name of the function between parentheses. See Function arguments for more information.
- A function needs to return its value with the **return** keyword. The return keyword causes an immediate exit of the function with the specified return value. It is required in functions with a return value. In functions without a return value (also called void functions or procedures in other languages), you can use return to exit the function prematurely, but it is not required. See Function arguments for an example.
- Functions can declare local variables, but they can also access the variables from the formula in which they are declared.
- Functions declared in a formula can only be accessed by the formula itself. If you want to access a function from another formula, declare it as a static class method instead.
- Function declarations can occur anywhere in a formula where you can write "normal" formula code. The order in which functions are declared does not matter: it is legal to call a function that is declared later in the formula. A function can also call itself recursively.

Functions declared in a global section can write to global variables, but you cannot call these from outside the global section, unless the entire function is declared as **const** (in which case it can only read from any variables):

```
Test {
global:
  float x

  float func getXSquared() const
```

```
    return x * x
  endfunc

init:
  z = getXSquared()
  ...
}
```

Next: Function arguments

See Also
Built-in functions
Methods

Function arguments

Most function declarations include one or more function arguments. Function arguments are declared after the name of the function between parentheses.

- Each argument declaration contains the type and the name of the argument. Within the formula, the argument can be used just like a normal variable. Function arguments can be of type bool, int, float, complex, color and object. You cannot pass an entire array as a function argument, but you can of course pass individual array elements. To pass an entire array, wrap it in an object (see the array wrapper classes in common.ulb).
- Optionally, the **const** keyword can be put before the type of the argument. This tells the compiler that the function will not change the value of the argument, which can make calling the function more efficient (currently this matters only for float, complex, and color arguments).
- Sometimes, a function needs more than one return value. In this case, you can declare a **by-reference argument** with a **&** symbol before the name. Such an argument is passed by reference, instead of by value, which enables the function to write a value back to it that the caller will receive.
- Objects are always passed by reference, so it is almost never needed to use the & symbol with object arguments. You should use it only in case you want a function to be able to actually change the object variable owned by the caller.

Examples:

```
complex func square(const complex x)
; Squares x and returns the result.
  return x * x
endfunc

func square2(complex &x)
; Squares x and returns the result in x.
  x = x * x
endfunc

bool func isPowerOfTwo(int value, int &powerOfTwo)
; Checks if value is a power of 2, such as 32 or 64. If so, returns true
; and returns the power of two in powerOfTwo. Otherwise returns false.
  powerOfTwo = 0
  if value > 0
    int mask = 1
    int i = 0
    while i < 31
      if value == mask
        powerOfTwo = i
        return true
      endif
      mask = mask * 2
      i = i + 1
    endwhile
  endif
  return false
endfunc

; Examples of how to call these functions:
complex c = (2, 0)
c = square(c)  ; c = (4, 0)
```

```
square2(c)     ; c = (16, 0)
int pow
if isPowerOfTwo(64, pow)
  ; pow = 6 here, because 2^6 = 64.
  ...
endif
```

Next: Classes

See Also
Built-in functions
Methods

Classes

While function declarations in formulas are useful, you cannot access them from another formula. To be able to re-use code in multiple formulas, you need to use classes instead.

- A **class** is a declaration of a set of variables and functions that operate on these variables.
- An object is an instance of its class. This class is the type of the object, just like **int** is the type of an integer variable.
- The variables of a class are called fields.
- The functions of a class are called methods. Both fields and methods are also known as the **members** of a class.

Classes are declared in formula files, or in class library files with the .ulb file extension. A class looks a bit like a regular formula entry, but the entry identifier is preceded by the **class** keyword. Here is an example of a simple class that represents a point with integer coordinates:

```
class Point {
public:
  func Point(int aX, int aY)
    x = aX
    y = aY
  endfunc

  float func distance(Point p)
  ; Returns the distance to the second point p.
    return sqrt(sqr(p.x - x) + sqr(p.y - y))
  endfunc

  int x
  int y
}
```

This class, named Point, contains a constructor, which is a special function that initializes an instance of the class. The constructor always has the same name as the class. Also, there is an additional method called distance, and two fields called x and y that store the coordinates of the point.

Next: Objects

See Also
Inheritance
Class parameters

Objects

To use a class, you have to create instances of the class, which are called **objects**. In Ultra Fractal, you always use a reference to the object. Such an object reference can be stored in a variable with the class name as type.

Let's create some objects using the example Point class given in Classes:

```
; Declare a new object variable
Point p
; Create a new Point object and store it in p.
p = new Point(3, 4)
float d = p.distance(new Point(0, 0))  ; Returns 5
```

The new operator takes the name of the class as a function, together with the arguments for the constructor. It creates a new instance of the Point class and returns a reference to it.

When you copy an object variable, you only copy the reference to the object, not the object itself. Copying an object variable therefore does not create a new object. Example:

```
Point pt1 = new Point(3, 4)
Point pt2 = pt1 ; This copies a reference!
pt2.x = 10      ; Now pt1.x is also 10
print(pt1.x)    ; This will print 10
pt2 = new Point(pt1.x, pt1.y) ; Creates a true copy
pt2.x = 6       ; Only modifies the copy
print(pt1.x)    ; This still prints 10
print(pt2.x)    ; This prints 6
```

You can also set an object variable to the special value 0, the **null reference**. This clears the reference, so it points to no object at all. This is often useful when programming, for example when building a linked list. You can test if an object value is empty by comparing with 0, or by treating it as a boolean expression:

```
Point p = 0
if p != 0
  ; This will not be executed.
endif
if p
  ; Will not be executed, same as above.
endif
int x = p.x  ; Compiles, but illegal!
if p
```

```
   x = p.x     ; This is safe
endif
```

When an object reference can be null, as in the example above, you need to test it first. Only use the object reference if it is non-null. If you access a field or method using a null object reference, Ultra Fractal will generate a run-time error and halt execution of the current calculation.

Next: Member visibility

See Also
Classes
Memory management

Member visibility

Like formulas, class declarations are divided in three sections, called **public**, **protected**, and **private** (in this order). Each section can contain member declarations: fields and methods. The type of section determines the visibility of the members that it contains.

- Members in the **public** section are accessible to formulas and other classes. This is the public interface of the class: the part that is intended to be visible to the outside world.
- Members in the **protected** section are accessible only to derived classes. This is often useful to change the inherited behavior of a class in a descendant. See Inheritance.
- Members in the **private** section can only be accessed by methods from the same class. Formulas or other classes cannot access them. This is useful for storing the internal data for the class, because you are sure that no other class can interfere with it.

Generally, the public section contains the constructor for the class, and the methods that make up its public interface. All fields and internal methods should go into the private section. If it is necessary to access a field from outside the class, the best design is to create a method to access it (commonly called a getter or setter). This allows you to change the internal representation of the class later, while keeping the public interface the same.

If you do not specify a section, it is assumed to be public. Many examples in this help file use this for brevity, but for real classes, it is recommended to define the visibility explicitly.

Here is an example of a simple Texture class that can distort a complex value:

```
class Texture {
public:
  func Texture()
    fAmount = 0.1
    fSeed = 1234
  endfunc

  complex func distort(const complex x)
  ; Adds a small random value to x and returns the result.
    return real(x) + getRandomValue() + flip(imag(x) + getRandomValue())
  endfunc

  float func getAmount()
  ; Returns the amount of distortion.
    return fAmount
  endfunc

  func setAmount(const float amount)
  ; Sets the amount of distortion
    fAmount = amount
  endfunc

private:
  float fAmount
  int fSeed

  float func getRandomValue()
  ; Return a new random value using the current seed.
    fSeed = random(fSeed)
    return fAmount * (fSeed / #randomrange)
  endfunc
}
```

All fields in this class are private, and the amount of distortion can be accessed with getAmount and setAmount. The getRandomValue method is private because it is only used within the class.

For completeness, here is how you would use this class:

```
Texture tx = new Texture
print(tx.distort((2, 3)))  ; Prints (1.968, 2.945)
print(tx.distort((2, 3)))  ; Prints (2.001, 3.062)
```

Note: classes can also have a default section for specifying the title and parameters of the class. See Class parameters.

Next: Inheritance

See Also
Classes
Fields
Methods

Inheritance

Class inheritance enables you to create a new class that derives from an existing class. The derived class is called a **descendant**; the class it derives from is called the **ancestor** or **base class**. A descendant inherits all fields and methods from the ancestor and can add new fields and methods. Also, the descendant can change the behavior of the ancestor by overriding one or more methods.

To derive from a class, specify its name within parentheses after the name of the class. Example:

```
class Base {
  int x
  int func getXPlusOne()
    return x + 1
  endfunc
}

class Derived(Base) {
  int y
  int func getXPlusY()
    return x + y
  endfunc
}
```

The Derived class contains both the members from itself, and the members from Base. Let's test this:

```
Derived d = new Derived
d.x = 2
d.y = 3
print(d.getXPlusOne())  ; Prints 3
print(d.getXPlusY())    ; Prints 5
```

You can assign a derived object to a variable that has the type of the ancestor class, because a derived object is always compatible with its ancestor:

```
Derived d = new Derived
d.x = 4
d.y = 5
Base b = d
print(b.x)  ; Prints 4
print(b.y)  ; Compilation error!
```

- Use inheritance sparingly. Inheritance is only suitable if the descendant class has an "is-a" relationship with the ancestor. For example, you should never derive a Rect class (representing a rectangle) from a Point class, because a rectangle is not a point. You could however derive a Car class from a Vehicle ancestor, because a car is a type of vehicle. If there is not an "is-a" relationship, use the other class as a field. For example, a Rect class could have two fields of type Point: leftTop and bottomRight.

- In Ultra Fractal, inheritance is primarily useful in combination with class parameters.
- You can include a file name with the ancestor if it is declared in another file. See Importing classes.
- The chain of classes that is built with inheritance is called a **class hierarchy**.
- If no ancestor is specified, the class descends from the internal Object base class. This means that all classes ultimately have Object as a common ancestor. See also Casting.
- See Overriding for more information about overriding methods and changing the behavior of a descendant class.

Next: Fields

See Also
Classes
Casting

Fields

Member variables in classes are called fields. Fields are used to store the internal data for a class.

- It is recommended to always declare all fields in the private section of a class, unless the class is very simple and mainly used to group a set of variables. As an illustration, compare the Point example with the Texture example.
- By convention, field names start with a lowercase "f", such as fMyData. This allows you to easily see the difference between fields and local variables or arguments.
- Fields can be of any type, just like variables. They can be arrays, dynamic arrays, or object references. The type of a field can even be the type of the class that it is declared in, so you can create recursive data structures like a tree or a linked list.
- Unlike formula variables or local function variables, all fields of a class are automatically initialized to 0. Use a constructor to initialize them explicitly.

Here is a very simple example of a linked list node that refers to itself:

```
class Node {
  Node next
  int data
}
```

Usage example:

```
Node head = new Node
head.data = 2
head.next = new Node
head.next.data = 3
Node n = head
while n
  print(n.data)  ; Prints 2 and 3
  n = n.next
endwhile
```

Note that it is not necessary to initialize the next reference when creating a node, because it is automatically initialized to 0, the null reference.

Next: Methods

See Also
Classes
Variables

Methods

Member functions in classes are called methods. Methods typically perform operations on the fields of a class. Methods are declared just like functions in a formula.

Methods always operate on an instance of the class (except static methods). The instance is returned by the reserved **this** reference. Here is an example:

```
class Data {
  int x
  func printMe(Printer p)
    p.printData(this)
  endfunc
}

class Printer {
$define debug
  func printData(Data d)
    print(d.x)
  endfunc
}
```

If you call a method on an object reference, the **this** reference inside the method will return the object reference that it was called on.

```
Data d = new Data
d.x = 12
d.printMe(new Printer)  ; Prints 12
```

In this code fragment, the **this** reference in Data.printMe() will return the d reference, which is then passed to the Printer object.

Note that objects are always passed to functions and methods by reference (see also Function arguments). For example, the Printer.printData() method could change the x value of the Data object that it receives. If you include the & symbol with the function argument (as you would do for normal by-reference argument passing), this lets the function modify the actual object reference which is often not needed.

Next: Overriding

See Also
Classes
Fields
Static methods

Overriding

In a derived class, you can **override** inherited methods to change their behavior. When overriding a method, the declaration must match exactly, including the argument names and types and the return type. (Ultra Fractal supports covariant return types, so if the return type is a class, it can also be a descendant of the return type of the inherited method.)

In the overridden method, you can call the inherited method using the ancestor class name and the member operator. Here is an example where the Derived class overrides a method inherited from Base, and calls the inherited method:

```
class Base {
  int func test()
    return 3
  endfunc
}

class Derived(Base) {
  int func test()
    return Base.test() * 2
  endfunc
}
```

Ultra Fractal always uses **virtual** method dispatching for all methods. This means that when you call a method on an object, the actual type of the object is determined and the method for that object type is called. This matters when using inheritance and method overriding:

```
Base b = new Base
print(b.test())  ; Prints 3
Derived d = new Derived
print(d.test())  ; Prints 6
; Test virtual method dispatching
Base x = b
print(x.test())  ; Prints 3
x = d
print(x.test())  ; Prints 6
```

When x.test() is executed, the actual type of x is determined and the corresponding method is called. In the first case, the actual type of x is Base, so Base.test() is called, which returns 3. In the second case, Derived.test() is called because the actual type of x is Derived, even though x is declared as a reference to Base. This behavior is extremely useful with class parameters.

Note: Unlike methods, member variables are never overridden. If you declare a variable in a derived class with the same name as a variable in the ancestor class, the new variable will exist in addition to the (now hidden) inherited variable. Because the new variable doesn't have any relationship with the inherited variable, it is legal for it to be of a different type. Generally, declaring variables with the same name as inherited variables is not recommended since it can easily lead to confusion.

Next: Constructors

See Also

Classes
Inheritance
Methods

Constructors

As briefly explained in Classes, a class can contain a constructor. A constructor is a special method that is called automatically by the new operator when an instance of the class is created. The task of the constructor is to initialize the fields of the just created object.

The constructor has the form of a method without a return type, and its name must be equal to the name of the class. Optionally, the constructor can have one or more arguments, which must be passed when creating an object of the class. This makes it possible to immediately initialize the object to the desired state.

```
class Test {
  func Test(int arg)
    x = arg
  endfunc
  int x
  int y
}
```

Before the constructor is called, all fields of the class are initialized to 0. You only need to initialize those fields that must have a different initial value.

```
Test t = new Test(3)
print(t.x)  ; Prints 3
print(t.y)  ; Prints 0
```

When using **inheritance**, the constructor of the derived class must have the same arguments as the ancestor class. (This is required because the constructor is also a virtual method.)

The constructor of the derived class can call the inherited constructor by using the name of the ancestor as a function. Calling the inherited constructor is almost always necessary because otherwise the ancestor will not be initialized correctly. Example:

```
class Derived(Test) {
  func Derived(int arg)
    Test(4)  ; Call the inherited constructor
    w = arg
  endfunc
  int w
}
```

```
Derived d = new Derived(7)
print(d.x)  ; Prints 4
print(d.w)  ; Prints 7
```

You can also call a constructor on an already constructed object, in which case it will execute as a normal function. This might be useful to reset the object.

Next: Static methods

See Also
Classes
Inheritance
Methods

Static methods

Normal methods in a class always operate on an instance of that class, so they need to be called on an object reference. In the method, the object reference that it was called on is returned by the **this** keyword.

Static methods, in contrast, do not operate on an instance of the class. They can only perform operations on the function arguments and on local variables. Therefore, you can use static methods as global functions, except that they are grouped by the class that contains them.

You declare a static function with the **static** keyword, which must appear just before the function declaration. Example:

```
class Utils {
  static int func min(int a, int b)
    if a < b
      return a
    else
      return b
    endif
  endfunc
}
```

You can call a static method with the class name, or with an object reference (an instance of the class) and the member operator:

```
; Call with class name (preferred)
print(Utils.min(3, 5))  ; Prints 3
; Call with object instance
Utils u = new Utils
print(u.min(3, 5))      ; Prints 3
```

Calling a static method is slightly more efficient than calling a normal method.

Next: Casting

See Also
Classes
Variables

Casting

When you assign an object reference to a variable, or pass it to a function, the type of the reference and the expected type should be compatible. As explained in Inheritance, a derived object is always compatible with its ancestor. Let's try this:

```
class Base {
  int x
}
class Derived(Base) {
  int y
}
```

```
Derived d = new Derived
Base b = d        ; Legal because Base is ancestor of Derived
b.x = 4
print(d.x)        ; Prints 4
Derived d2 = b    ; Compilation error: incompatible types!
```

The last assignment is illegal because b has type Base, which is not compatible with Derived. However, b really points to an object of type Derived, so it should be possible to get the 'original' object back. You can do this with a **cast**. A cast looks like a function call, using the name of the class to cast to as the name of the function:

```
Derived d = new Derived
d.y = 5
Base b = d
Derived d2 = Derived(b)   ; Cast b to Derived
print(d2.y)               ; Prints 5
Base b2 = new Base
d2 = Derived(b2)          ; This cast fails and returns 0
print(d2)                 ; Prints 0
```

A cast checks if the object is really an instance of the class to cast to. If so, it returns a reference with the desired type. Otherwise, it returns a null reference. You can use this to query the type of object that a reference actually points to.

Because all classes ultimately descend from the internal Object class, you can use Object as a common type to store different types of objects in an array, for example. With a cast, you can later test what type of object an element in the array really contains.

For debugging purposes, you can also print the value of an object reference, which will print the class name and the reference count of the object:

```
Object obj = new Derived
print(obj)  ; Prints Derived (1)
```

Note: With a class parameter, you can also determine the currently selected class by comparing it to a class with the == operator or the != operator. See Extending class parameters.

Next: Class parameters

See Also
Classes
Inheritance

Class parameters

Like formulas, classes can also have parameters and a title, which are declared in the default section. This matters only if the class is in turn declared as a **class parameter** for a formula. In this case, the class appears in the list of parameters in the Layer Properties tool window, with the parameters for the class grouped with the title of the class. (See Example 1 - Formula classes for a screen shot.)

Here is a simple example of a class that implements a simple bailout condition for a Mandelbrot-like fractal formula. The bailout value is exposed as a parameter.

```
class Bailout {
public:
  func Bailout()
    ; Empty constructor, see Extending class parameters
  endfunc
  bool func hasBailedOut(const complex z)
    return |z| > @bailout
  endfunc
default:
  title = "Simple Bailout"
  float param bailout
    caption = "Bailout value"
    default = 4
    min = 1
  endparam
}
```

To use this class and include its parameters in the parameter list for a formula, declare a class parameter in the formula. A class parameter is declared by a parameter block with the name of the class as its type.

```
MandelbrotTest {
init:
  z = (0, 0)
  Bailout bo = new @bailoutParam
loop:
  z = sqr(z) + #pixel
bailout:
  !bo.hasBailedOut(z)
default:
  Bailout param bailoutParam
    caption = "Bailout Test"
  endparam
}
```

Notes

- Normally, you create an instance of a class with parameters using new and the class name. In this case, all parameters always remain at the default value. The class parameter bailoutParam in the MandelbrotTest formula represents a variant of the Bailout class that has different parameter values, set by the user. To create an object that uses these user-set parameter values, use the new operator with the class parameter as shown above.
- You can create more than one object from a single class parameter, and you can have multiple class parameters of the same class type, which have their own independent set of parameters.
- Inside a class, all parameters must be declared by a parameter block. (In contrast, formulas can have undeclared parameters for Fractint compatibility.)
- Optionally, a class can also contain a rating setting in its default section.

Next: Extending class parameters

See Also
Classes
Parameter blocks
Function blocks
Headings

Extending class parameters

Class parameters make it easy to create different formulas with common features, but you can even take this a step further. If you load the MandelbrotTest example in Ultra Fractal, a Browse button is visible next to the name of the class in the list of parameters. This button enables the user to select a different class than the default Bailout class, as long as it descends from Bailout. (See Example 1 - Formula classes for an example.)

Let's write a descendant class that implements more bailout options:

```
class ExtendedBailout(Test.ufm:Bailout) {
public:
  bool func hasBailedOut(const complex z)
    return (@test == "mod" && |z| > @bailout) ||       \
      (@test == "real" && sqr(real(z)) > @bailout) || \
      (@test == "imag" && sqr(imag(z)) > @bailout)
  endfunc
default:
  title = "Extended Bailout"
  int param test
    caption = "Bailout Test"
    default = 0
    enum = "mod" "real" "imag"
  endparam
}
```

If you put this code in a class library file such as Test.ulb, you can select the new class by clicking on the Browse button next to the Bailout class parameter. The MandelbrotTest formula will now use the bailout code from the ExtendedBailout class. As you can see, this system enables you to add new features to a formula even after it has been published, without having to modify the formula itself!

- The descendant class must be declared in a class library file with the .ulb file extension so the browser can find it. Therefore, the file name of the ancestor needs to be specified as well. The example code above assumes that both MandelbrotTest and the Bailout class are declared in a file called Test.ufm. See also Importing classes.
- The ExtendedBailout class overrides the hasBailedOut method to change its behavior. Note that it introduces a new parameter called test. The bailout parameter is inherited from the Bailout base class.
- If you want the class to appear in the browser when selecting a value for the class parameter, make sure it has a title. Classes without a title are not visible in the browser. (This is useful for abstract base classes, as explained in the next paragraph.)
- In this example, the Bailout base class already has a default behavior and a parameter. In practice, it might be better to have an **abstract base class** that only declares the methods that form the public interface for the class. Do not include a title for the abstract base class (the browser will only show classes that have a title). In the class parameter declaration for the formula that uses an abstract class, you can then use the default setting to select a descendant of the base class that is declared in a corresponding class library file. In this way, the abstract base class is never used on its own, just as a base class.
- A descendant can also **override parameters** from its ancestor by including a parameter block for an inherited parameter. This parameter block completely replaces all previous

settings for the parameter. With this, you can for example change the default value of a parameter, or hide it by setting visible to false. The type of the parameter must remain the same.

- In case you do not want to provide the option to the user to select a descendant class for a class parameter, set the selectable setting of the class parameter to false.
- The base class must have a constructor (possibly empty), otherwise the constructor of the descendant class will never be called. The compiler will generate a warning if you use a base class without a constructor for a class parameter, and the selectable setting is not false.
- You can determine the currently selected class of a class parameter by comparing the class parameter to the class directly with the == operator or != operator. This allows you to show or enable other parameters depending on the currently selected class. (You can also determine the selected class with a cast, but that requires the creation of an object of the class, something you cannot do in visible or enabled settings.)

Next: Parameter forwards

See Also

Classes
Class parameters
Inheritance
Overriding

Parameter forwards

When you rewrite an existing formula to take advantage of class parameters, this changes the format of parameter sets that are saved while this formula is in use. This means that existing parameter sets cannot be opened anymore. To solve this problem, use parameter forwards.

A parameter forward declares a new parameter name in the formula, and lets it point to a sub-parameter of a class parameter. For example, suppose that an earlier version of the MandelbrotTest example handled its bailout condition itself, with a float parameter named bailout. Now, however, the equivalent parameter is bailout from the bailoutParam class parameter. The following parameter forward expresses this:

```
MandelbrotTest {
; ...
default:
  Bailout param bailoutParam
    caption = "Bailout Test"
  endparam
  param bailout = bailoutParam.bailout
}
```

A parameter forward starts with the **param** keyword, the name of the old parameter, the = symbol, and then the "path" to the new parameter, which contains parameter identifiers separated by dots.

In this case, when Ultra Fractal loads an old parameter set, the parameter set will contain the value for a parameter named bailout. Because the formula does not contain a parameter named bailout, Ultra Fractal will search its parameter forwards and will subsequently set the bailout parameter of the class that is the default class for the bailoutParam class parameter (in this case the Bailout class).

For this to work correctly, obviously the default class for the class parameter must be compatible with the previous behavior of the formula. (You can set a default class with the default setting.) This is your responsibility as formula author.

Next: Importing classes

See Also
Classes
Class parameters

Importing classes

When a formula or a class contains a class name, for example to declare a variable or a class parameter, Ultra Fractal searches for the class in the list of imported files, and then in the current file. To import a file so its classes become available, use the **import** keyword.

```
class ExtendedBailout(Test.ufm:Bailout) {
public:
  import "common.ulb"
  ; ...
}
```

This modified version of the ExtendedBailout example first imports the file Test.ufm because that is where its ancestor is located. Then it imports the class library file common.ulb.

- The **import** keyword expects a file name within double quotes, without any path information. The file is located using the current formula search paths as set in the Folder tab of the Options dialog.
- If a file name is specified in the **ancestor specification** of a class, this behaves as if this file was imported before any other files. The browser, however, needs the specification of the ancestor file name here to be able to check the inheritance of classes when you select another class for a class parameter.
- The **order** in which import statements occur is important. When searching for a class, Ultra Fractal searches the file that was last imported first, and so on. Other than that, import statements can occur anywhere in a formula or class. Even the statements before an import statement are affected by it, because import statements are read and applied before the rest of the code is checked for consistency and type compatibility. Therefore it is recommended to put your import statements at the top of a formula or class.
- **Class library files** with the .ulb extension have two purposes. When looking for classes for class parameters, the browser will only look in .ulb files. So any derived classes intended to extend class parameters must be in a class library file. Also, class library files provide a logical place to put classes that are intended to be shared by different fractal formulas, coloring algorithms, and transformations.
- The common.ulb file that is installed with Ultra Fractal provides many useful base classes. See The common.ulb file.
- You can also import classes from .ufm, .ucl, or .uxf files if you have to, but this is recommended only for files that you control yourself. If you are a formula author and you want to provide common functionality to other authors, put them in your own .ulb file. Do not import a class from another author's .ufm/.ucl/.uxf file because it might be changed in a future revision of that file. See also Publishing your formulas.

Next: The common.ulb file

See Also
Classes
Inheritance
Class parameters

The common.ulb file

Classes can be useful on their own, but they show their true power when you combine them with each other. When used well, classes can eliminate duplicate code in everyone's formulas. However, to realize this potential, there needs to be a common foundation so all classes can work with each other easily.

The common.ulb file installed in the Public formula folder provides such a foundation. It contains an extensive collection of base classes and utility classes. Here is a quick overview of the most important classes:

- The Array classes provide a way to pass arrays to functions, since Ultra Fractal doesn't allow this directly. See also Function arguments.
- The Generator and IntegerGenerator classes are base classes for objects that generate a sequence of values, such as random number generators.
- The Transfer and IntegerTransfer classes are base classes for objects that take a value and transform it.
- The Formula class and descendants such ConvergentFormula, DivergentFormula and ConvergentDivergentFormula wrap a complex fractal formula as an object. See Example 1 - Formula classes for an example of how this works in practice.
- The Coloring, GradientColoring and DirectColoring classes wrap coloring algorithms as an object.
- The Transform, UserTransform and ClipShape classes wrap transformations as an object.
- The TrapShape, TrapMode, TrapColoring, TrapPosition, TrapTransfer and ColorTrap classes provide a framework to implement orbit trap coloring algorithms where every aspect of the algorithm is extensible. See Example 2 - Orbit trap classes.

The goal is to provide flexible base classes that help formula authors to write classes and formulas that work with all other classes. It is highly recommended to read through the common.ulb file thoroughly before you start writing your own classes. Also have a look at how the standard classes in Standard.ulb are implemented.

The common.ulb file will be continually updated via the online formula database. The UF-programmers mailing list is the best place to discuss the library of classes and to suggest modifications.

Documentation for the common.ulb file and any other classes in the online formula database is accessible via formulas.ultrafractal.com/reference.

Next: Memory management

See Also
Classes
Importing classes

Memory management

Ultra Fractal manages the memory for all objects that are created automatically, and normally you do not have to worry about this. However, if you create a large number of objects, or if your objects are large because they use large arrays, you might want to know how you can influence memory usage during a calculation.

The lifetime of an object is managed with reference counting. As long as there is a reference to an object, it will not be freed. If the last reference to an object goes out of scope, if another object is stored to the reference, or if it is set to 0, the object is freed immediately.

For debugging purposes, you can print the name and reference count of an object. This example uses the Point example class:

```
Point p = new Point(0, 0)
print(p)         ; Prints Point (1)
Point pt2 = p    ; Create a second reference
print(p)         ; Prints Point (2)
p = 0            ; Clears the first reference
print(pt2)       ; Prints Point (1)
pt2 = 0          ; Frees the Point object
print(pt2)       ; Prints 0
```

As you can see, you can free an object by setting all reference that point to it to 0, the null reference. You can use this to reclaim memory during a calculation.

When using recursive data structures, you can easily get the situation where two objects contain references to each other. In this case, the reference count of both objects will never reach zero. To solve this, Ultra Fractal keeps a running list of all live objects and will free them once the entire calculation is finished. If you need to reclaim memory earlier, you need to explicitly set the references to 0 so the objects can be freed. You could say that Ultra Fractal uses deterministic garbage collection, in contrast to environments like Java or .NET which use nondeterministic garbage collection.

Again, normally you do not have to worry about this because the amount of memory consumed by objects is usually very modest. But in case memory usage becomes a problem, you can use this information to track down the reference counts of objects and make sure they are freed when necessary.

Next: Writing transformations

See Also
Classes
Objects

Writing transformations

Transformations are put in transformation files with the .uxf extension. They can have the following sections, in this order:

- **global**
- **transform**
- **default**

If a transformation does not start with a label, it is assumed to start with the **transform** section. The optional setting within parentheses after the entry identifier is ignored.

The **global** section is executed only once per image and can be used to fill look-up tables and initialize read-only variables. See Global sections.

The **transform** section contains one or more statements. The purpose of these statements is to take the predefined symbol #pixel, which contains the coordinates of the pixel currently being calculated, transform it and put the result back in #pixel. They can also set the predefined boolean symbol #solid to **true** to give the pixel a solid color instead of calculating it. In this case, no further calculations are done for the pixel. The solid color is adjustable on the Mapping tab of the Layer Properties tool window.

The **default** section can contain the following settings:

- helpfile
- helptopic
- precision
- rating
- render
- title

It can also contain one or more parameter blocks.

Next: Writing fractal formulas

See Also
Writing formulas
Transformations

Writing fractal formulas

Fractal formulas are put in fractal formula files with the .ufm extension. They can have the following sections, in this order:

- **global**
- **builtin**
- **init**
- **loop**
- **bailout**
- **default**
- **switch**

If a fractal formula does not start with a label, it is assumed to start with the **init** section. If a fractal formula contains an empty label (a single colon), it is assumed to start the **loop** section, and in this case, the last statement in the **loop** section is assumed to be the boolean expression from the **bailout** section (so the formula should not contain a separate **bailout** section). This set of rules ensures compatibility with old Fractint formula. It is not recommended to use this when writing new formulas, though.

The optional setting within parentheses after the entry identifier specifies the symmetry of the fractal formula. See Symmetry.

The **global** section is executed only once per image and can be used to fill look-up tables and initialize read-only variables. See Global sections.

The **builtin** section is used for accessing built-in fractal formulas. If this section is used, the **global**, **init**, **loop**, and **bailout** sections are not allowed. The **builtin** section can contain the following settings:

- type

The **init** section is executed only once per pixel, and is useful for initializing variables.

The **loop** section is executed once per iteration. It should update the value of the predefined complex variable z (you can also write to #z) using the old value of z.

The **bailout** section contains a single boolean expression. The **loop** section is repeated as long as this expression evaluates to **true** (but it is always executed at least once).

The **default** section can contain the following settings:

- angle
- center
- helpfile
- helptopic
- magn
- maxiter
- method
- periodicity
- precision
- rating

- render
- skew
- stretch
- title

It can also contain one or more parameter blocks.

The **switch** section is used to implement switching from one formula type to another (for example from Mandelbrot to Julia). See Switch feature.

Next: Writing coloring algorithms

See Also
Writing formulas
Fractal formulas

Writing coloring algorithms

Coloring algorithms are put in coloring algorithm files with the .ucl extension. They can have the following sections, in this order:

- **global**
- **init**
- **loop**
- **final**
- **default**

If a coloring algorithm does not start with a label, it is assumed to start with the **final** section (in that case, the **init** and **loop** sections are not allowed).

The optional setting within parentheses after the entry identifier specifies whether the coloring algorithm can be used for inside coloring, outside coloring, or both. The possible values are:

INSIDE	The coloring algorithm is only intended for coloring the inside of a fractal.
OUTSIDE	The coloring algorithm is only intended for coloring the outside of a fractal.

If the setting is omitted, or has another value, the coloring algorithm is supposed to be useful for both inside and outside coloring. See also Inside and outside.

The **global** section is executed only once per image and can be used to fill look-up tables and initialize read-only variables. See Global sections.

The **init** section is executed only once per pixel, and is useful for initializing variables.

The **loop** section is executed once per iteration, right after the **loop** section of the fractal formula has been executed. It can read the current value of #z and perform some calculations on it.

The **final** section is executed afterwards to determine the actual index into the gradient (this index is further transformed by the various settings on the Inside or Outside tabs). The index is a float value and should be written to the predefined symbol #index. If the settings on the Inside or Outside tab are set to their default values (Density = 1, Transfer = Normal and Offset = 0), the entire gradient range corresponds to the range 0..1 of the index value.

To create a direct coloring algorithm, use the predefined symbol #color instead of #index.

It is also possible to set the predefined symbol #solid to **true**: this gives the pixel the solid color set in the Inside or Outside tab of the Layer Properties tool window.

The **default** section can contain the following settings:

- helpfile
- helptopic
- precision
- rating
- render
- title

It can also contain one or more parameter blocks.

Next: Writing direct coloring algorithms

See Also
Writing formulas
Coloring algorithms

Writing direct coloring algorithms

Direct coloring algorithms directly output a color instead of an index value. This is done by assigning a value to the #color predefined symbol instead of to #index.

To compute this color, you can use color expressions, color variables, and color arithmetic. The following arithmetic operations are available:

c1 + c2	Returns a color where each component is the sum of the respective components from c1 and c2. So, red(c1 + c2) is equal to red(c1) + red(c2).
c1 - c2	Subtracts the color components in c2 from the respective color components in c1. So, red(c1 - c2) is equal to red(c1) - red(c2).
c * f	Multiplies each component of c with a float value. So, red(c * f) is equal to red(c) * f. **Note**: the float value must be at the righthand side of the * operator.
c / f	Divides each component of c by a float value. So, red(c / f) is equal to red(c) / f.

For example, to calculate the average of two colors, use (c1 + c2) / 2. Be aware of the fact that the alpha value is treated just like the other components. So, c / 2 will not only darken a color, it will also make it more transparent.

The following conversion functions are available:

- rgb
- rgba
- hsl
- hsla
- red
- green
- blue
- hue
- sat
- lum
- alpha

There are some functions for blending and retrieving gradient colors:

- gradient
- blend
- compose

There are also functions to reproduce all layer merge modes. They are called **mergeX**, where **X** stands for the name of the merge mode. See Merging functions.

You can use color parameters to let user specify a color. You can also use special user functions that allow a user to select a merge mode. See Parameter blocks.

Next: Global sections

See Also
Writing coloring algorithms

Direct coloring algorithms

Global sections

Sometimes, initializing variables of a formula can require a large number of calculations. Since you would normally initialize variables in the **init** section of a fractal formula or a coloring algorithm (or the **transform** section of a transformation), these calculations are performed again and again for every pixel.

Often, they only depend on parameter settings, and therefore the results are the same for every pixel. To avoid doing many repeated calculations, you can move them to the **global** section. This is a special section at the start of a formula that is executed once per image.

Use the global section to perform per-image calculations and store the results in variables that can be read in other sections. Variables declared here are treated as read-only in other sections, so you cannot use this to share variables across pixels (that would not work reliably).

In the following example, the global section is used to pre-calculate an array of random values that is the same for every pixel. These random values are subsequently used to disturb a standard Mandelbrot set.

```
MandelbrotModified {
global:
  float values[#maxiter]
  int i = 0
  int seed = 12345678
  while i < 100
    seed = random(seed)
    values[i] = seed / #randomrange
    i = i + 1
  endwhile
init:
  z = (0,0)
  int iter = 0 ; "i" is already taken
loop:
  z = sqr(z) + #pixel + values[iter]
  iter = iter + 1
bailout:
  |z| 4
}
```

Notes

- Global sections are often combined with arrays to compute look-up tables that can speed up the formula tremendously.
- You can even declare an array equal to the size of the image (with the #width and #height predefined symbols) and calculate the entire fractal in the global section of a coloring algorithm. The **final** section is then used only to return colors or index values from this array. This enables you to implement fractal types like IFS and flame fractals that are not natively supported by Ultra Fractal. This technique does have some limitations: it can require a fair amount of memory and the progress of the calculation is not reported. It is very slow

and memory-intensive with rendering to disk, especially with anti-aliasing, and it does not work well with network calculations either. See also the render setting.

Next: Image parameters

See Also

Arrays

Sections

Image parameters

Formulas in Ultra Fractal can import external images. This is typically used only in direct coloring algorithms, which can subsequently use the information from the image to return colors for each pixel that is calculated. The standard Image coloring algorithm is a good example.

To be able to import an image, the formula needs to declare an image parameter. This takes the form of a regular class parameter with the built-in Image class as the class type:

```
default:
  ...
  Image param imageParam
    caption = "Image"
  endparam
```

The parameter block can contain the following settings: caption, enabled, hint, and visible. You cannot specify a default image. By default, the image is always empty.

In the list of parameters in the Layer Properties tool window, the image parameter shows up as an image preview and an Open button to select an image file. See Using images for an example. The user can select any image from the computer and it will be loaded in the preview window.

To access the data from the image, you need to create an Image object from the image parameter, just like with class parameters:

```
Image img = new @imageParam
int numPixels = img.getWidth() * img.getHeight()
```

You can now access all pixels in the image using the methods of the built-in Image class. The Image object created is associated to the specified image parameter, so you can easily have multiple image parameters if necessary.

Note: You can also create a stand-alone Image object with `new Image`. For example, you could make a copy of an imported image and modify it.

Next: Random values

See Also
Classes
Image class

Random values

Some formulas need to calculate random values. Ultra Fractal offers two ways of obtaining pseudo-random values.

The predefined symbol #random returns a new complex random number for every pixel. This exists primarily for compatibility with old Fractint formulas.

The preferred way of obtaining a random value is the random function. This function accepts an integer seed and returns a new random seed. To generate a series of random numbers, you should call the function repeatedly, each time supplying the seed returned by the previous call. Example:

```
int seed = 123456789 ; initial value
seed = random(seed)
; seed is now the first random number
seed = random(seed)
; seed is now the second random number
```

To obtain a random floating-point number between 0 and 1, divide abs(seed) by the predefined symbol #randomrange. To obtain a random integer between 0 and n - 1, use abs(seed) % n.

You can generate multiple independent, reproducible series of random numbers just by declaring and using multiple seeds. The random function will always return the same result for the same seed value.

Next: Symmetry

See Also
Global sections

Symmetry

Some fractal formulas always create images that are symmetric. Ultra Fractal can take advantage of the symmetry to speed up the calculation. To enable this, you must specify the symmetry of the formula as the optional setting between parentheses right after the entry identifier. Example:

```
Mandelbrot(XAXIS) {
  ...
}
```

This formula is symmetric around the horizontal x-axis, therefore it uses the XAXIS setting. This table lists all possible values for the symmetry setting:

XAXIS	Forces symmetry around the horizontal x-axis, or the real axis.
YAXIS	Forces symmetry around the vertical y-axis, or the imaginary axis.
XYAXIS	Forces symmetry around both the horizontal and the vertical axes.
ORIGIN	Forces rotational symmetry around the origin. This is useful for Julia sets.
PI	Not implemented.

You can append **_NOPARM** to all values (thus obtaining **XAXIS_NOPARM**, etc) to make sure symmetry is only applied when all complex parameters are set to (0, 0). The XAXIS setting also allows the suffixes **_NOREAL** and **_NOIMAG** to disable symmetry for non-zero real and imaginary parts of all complex parameters.

Notes

- Symmetry is always disabled if the rotation angle set in the Location tab is not zero, or if a coloring algorithm that reads the value of #z is selected.
- It is not recommended and not reliable to use this to enforce symmetry that does not exist in the formula itself. It is only intended as a speed-up for formulas that naturally exhibit symmetry.

Next: Switch feature

See Also
Writing fractal formulas
Formula files and entries

Switch feature

The switch feature allows you to switch easily between related fractal types. One fractal type can be used as a map for another. This is very useful, since Mandelbrot sets, for example, are in fact maps of the corresponding Julia sets.

To use the switch feature with your own formulas, you must include the **switch** section as the last section in your fractal formula. Here is an example of a typical Mandelbrot formula using the **switch** section:

```
Mandelbrot {
init:
  z = 0
loop:
  z = sqr(z) + #pixel
bailout:
  |z| < @bailout
switch:
  type = "Julia"
  seed = #pixel
  bailout = bailout
}
```

The type setting specifies the identifier of the formula (in the same file) to switch to. The other settings can copy parameters and the pixel value from the source formula to the destination formula (the formula Ultra Fractal is switching to). The #pixel symbol returns the coordinates of the point in the fractal window where the user clicked to initiate the switch.

When switching, Ultra Fractal now loads the Julia formula, and tries to find the parameters **seed** and **bailout** in the Julia formula. If these parameters can be found, they are set to the pixel value and the bailout of the Mandelbrot formula. Otherwise, the settings are ignored.

So, you need to take the following steps to use the switch feature:

1. Append the **switch** section to the end of your formula.
2. Insert the type setting and use the entry identifier of the formula you want to switch to as the setting value, enclosed in double quotes (like all string values).
3. Insert the setting "destination-parameter = #pixel" to let the switch feature be dependent on the point where the user clicked inside the fractal window. If you want, you can use this setting more than once, or just leave it out. The parameter in the destination formula must be complex, otherwise this setting is ignored.
4. Optionally, insert additional settings "destination-parameter = source-parameter" to copy other parameters from the source formula to the destination formula. Be sure that the types of the source and destination parameters are the same; otherwise, the setting is ignored. Parameters not explicitly copied here are set to the default values.

Here is an example of a Julia formula that could be used with the Mandelbrot formula shown above. Note that the Julia formula allows you to switch back to the Mandelbrot formula (of course without using the pixel value).

```
Julia {
init:
  z = #pixel
loop:
  z = sqr(z) + @seed
bailout:
  |z| < @bailout
switch:
  type = "Mandelbrot"
  bailout = bailout
}
```

Next: Providing help and hints

See Also

Writing fractal formulas
Switch mode

Providing help and hints

To make your formulas easier to use, you might want to add help. There are two ways to provide help for formulas.

The easiest way to add help to your formula is to provide hints for all parameters. A hint is a small explanatory message that is shown in the Fractal Mode tool window when the user hovers over the paramter, or when the user clicks the **?** button in the title bar of the Layer Properties tool window, and then clicks the parameter.

- To add a hint to a parameter, use the hint setting in the parameter block.

Although parameter hints are certainly helpful, they cannot provide an overview of the purpose and intended use of the formula. To overcome this, you can create a separate help file and specify its name and location in the formula file so Ultra Fractal can open it.

Ultra Fractal supports help in HTML files, in Windows Help files (*.hlp), in Windows HTML Help files (*.chm), in Adobe Acrobat files (*.pdf), in Microsoft Word files (*.doc), and in plain text files (*.txt). These help files are usually installed in the **Help on Formulas** folder. By default, its location is "My Documents\Ultra Fractal 5\Help on Formulas", but you can change this in the Folders tab of the Options dialog.

- To link a formula to an external help file, use the helpfile and helptopic settings in the **default** section.

Ultra Fractal launches the help file when the user clicks the **Help** button in the Layer Properties tool window.

Next: Debugging

See Also
Publishing your formulas
Parameter blocks

Debugging

When you are writing complex formulas, it is likely that they will not immediately function as intended. Although the compiler tries to catch most of the common mistakes and reports them as errors or warnings, some mistakes will go unnoticed until you try the formula.

The process of trying a formula and correcting it until it works is called **debugging**, because you are essentially removing bugs (programming mistakes). To debug a formula, you use run-time messages.

Run-time messages can be generated by a formula while it is executed. They appear in the Compiler Messages tool window, where you can examine them.

To enable run-time messages, define the **DEBUG** symbol. Run-time messages are caused by an array index that is out of bounds, an assignment of incompatible arrays, or by the print function. Here is an example:

```
int a[4]
int i = 5
a[i] = 4               ; out of bounds, no run-time message
print("Hello?")        ; ignored
$define DEBUG
a[i] = 3               ; out of bounds, causes run-time message
print("Hello, world") ; causes run-time message
```

Use the print function to examine the values of variables while the formula is executed, so you can understand why it is not working properly.

By not defining the **DEBUG** symbol, run-time messages are not generated. When you are publishing a formula, you should make sure the **DEBUG** symbol is not defined, since the users of your formula will probably not appreciate the run-time messages.

Next: Optimizations

See Also
print function
Compiler directives
Memory management

Optimizations

The compiler performs optimizations to make sure that all formulas run as fast as possible. It is helpful to know something about these optimizations when writing formulas.

The optimizations can be divided into two categories:

- Evaluation of constant expressions. Constant expressions are expressions whose value can be evaluated by the compiler. Constant expressions can contain operators, built-in functions, constants and parameters. This is possible because the formula is re-compiled each time a parameter changes, so parameters can be treated as constants by the compiler.
- Replacement of slow operations by faster ones, such as replacing 2*x by x+x (adding is faster than multiplying).

When writing formulas, this has the following consequences:

- Writing to parameters is not recommended. This can make the formula run slower, because parameters cannot be treated as constants anymore.
- You can write constant expressions wherever you want. You do not have to precalculate them, because the compiler can do that for you. So, using 2 + 1/3 is just as fast as using 2.3333. Just use whatever you find most convenient.
- **If** statements with constant expressions (like **if** 3 < 2) are completely eliminated by the compiler. This is very helpful when you use lots of **if** statements depending on the values of enumerated parameters. Therefore making your formula versatile by providing many options does not cause it to run slower.
- You can use descriptive operators like z^4 instead of z*z*z*z or sqr(sqr(z)), since Ultra Fractal automatically chooses the most efficient operation independent of the operators or functions used.

Next: Compatibility

See Also
Conditionals

Compatibility

Ultra Fractal 5 accepts almost all formulas written for earlier versions of Ultra Fractal. There are many changes in the compiler since **Ultra Fractal 4**, but the language has been changed in such a way that existing formulas will still compile. The new language features are accessed with semi-reserved keywords, meaning that you can still use them as variable names. This ensures that there will not be a conflict with existing formulas.

There are some minor differences in the compiler since **Ultra Fractal 2**:

- There are a three new keywords: **color**, **heading**, and **endheading**. Variables with these names will have to be renamed. Parameters with these names should not be renamed to avoid breaking backwards compatibility. Instead, add a @ character to the parameter name in the parameter block that describes it, so the compiler will recognize it as a parameter, not as a keyword. For example, "param color" should be changed to "param @color".
- Some formulas might not run well with double precision (Ultra Fractal 2 always uses extended precision). In this case, either correct the formula, or adjust the **Additional Precision** value in the Formula tab of the Layer Properties tool window so extended precision is used. You can verify this in the Statistics tool window. Also see Arbitrary precision.
- Built-in functions that can return complex values with float arguments are treated differently. If the return value is assigned to a complex variable, the float argument is converted to complex and the complex version of the function is called instead of the float version. This fixes a number of bugs like:

  ```
  complex c = sqrt(-1)
       ; should be (0, 1)
  ```

  Unfortunately, sometimes it breaks backwards compatibility because Ultra Fractal 2 would assign an invalid value to c in this case. To work around this, use:

  ```
  float f = sqrt(-1)

      complex c = f
  ```

The formula compiler is largely compatible with **Fractint**'s parser and most Fractint formulas can be used without modification. There are a few exceptions, however:

- Writing to parameters is not recommended, since Ultra Fractal can perform special optimizations when parameters are read-only. Formulas writing to parameters will be accepted, though.
- Writing to predefined symbols that are read-only, such as #pixel, is not allowed. Formulas writing to these predefined symbols will not be accepted.
- Formulas using the predefined symbol LastSqr will not be accepted. The usage of LastSqr serves no purpose; it is intended as a speed-up but it only makes formulas run slower (even in Fractint). Furthermore it makes formulas very hard to write and to understand.
- Formulas using the function cosxx will not be accepted either. This function results from an early version of Fractint which contained a bug in the cos function. The cosxx function allows you to reproduce this bug in later versions of Fractint. If you still want to use the cosxx function, you can write conj(cos(a)) instead of cosxx(a).
- In Fractint, built-in functions with invalid arguments often return other values than in Ultra Fractal. For example. log(0) returns 0 in Fractint, but it returns -infinity in Ultra Fractal. This

can cause problems. In general, if a Fractint formula gives a blank screen instead of a fractal, you should check the formula for this kind of errors (log(0), division by zero, recip(0), etc).

- Fractint iterates fractal formulas not the number of times given by the maximum iterations value, but one time less. Ultra Fractal does iterate fractal formulas the number of times given by the maximum iterations value. The Fractint PAR import feature in Ultra Fractal takes this into account and subtracts one from the maximum iterations value used by Fractint.

Generally, formulas that depend on the forgiving behaviour of Fractint's parser will not be accepted. Often Fractint accepts formulas that contain syntax errors and it will still produce a picture. Ultra Fractal will refuse to load such formulas. This is useful, because it helps you to write clear and understandable formulas, and it might point you in the direction of possible errors.

Next: Execution sequence

See Also
Invalid operations

Execution sequence

To help you understand the way Ultra Fractal executes the various sections in all formula types, here is an overview of the execution sequence per pixel, written in pseudo-code similar to the formula language.

Before calculating the image, the global sections are executed:

```
for each transformation
  execute global section of transformation
execute global section of fractal formula
execute global section of inside coloring algorithm
execute global section of outside coloring algorithm
```

Then, for each pixel, the following calculations are performed:

```
for each transformation
  #solid = false
  execute transform section of transformation
  if #solid == true
    stop and give pixel the solid mapping color
endfor
execute init section of fractal formula
execute init section of inside coloring algorithm (if it exists)
execute init section of outside coloring algorithm (if it exists)
int iter = 0
repeat
  execute loop section of fractal formula
  bool b = the expression in the bailout section of the fractal formula
  if b == true
    ; not yet bailed out
    execute loop section of inside coloring algorithm (if it exists)
    execute loop section of outside coloring algorithm (if it exists)
  endif
  iter = iter + 1
until (b == false) || (iter == #maxiter)
#numiter = iter
if #numiter == #maxiter
  ; pixel is inside
  execute final section of the inside coloring algorithm
else
  ; pixel is outside
  execute final section of the outside coloring algorithm
endif
color the pixel
```

Next: Invalid operations

See Also
Writing transformations

Writing fractal formulas
Writing coloring algorithms
Global sections

Invalid operations

Formulas can easily perform invalid operations, such as dividing by zero. Rather than showing an error message when an invalid operation occurs, Ultra Fractal just ignores the error and calculates further. This means that the resulting image can be unpredictable if you do not pay proper attention to these special cases. See also the isInf and isNaN functions, and Compatibility.

Invalid operations include dividing by zero (also with the % operator), and using some built-in functions with invalid arguments. The range of valid arguments for a function is always discussed (if there are invalid values) in the description of the function. It does not hurt to use arguments outside the valid range, as long as you remember that the results may be unpredictable.

If you are using loops, you should avoid writing infinite loops at all times. An infinite loop is a loop which repeats forever, without stopping. Here is are two examples:

```
while true
  ...
endwhile
repeat
  ...
until false
```

You must make sure that the loop is exited at some time. This means that the condition in a while loop should eventually become false, and that the condition in a repeat loop should eventually become true.

If, for some reason, the formula still enters an infinite loop, the fractal window will remain black and no pixels will be calculated at all because the formula is still busy to calculate the first pixel, which will never be finished. Ultra Fractal is still able to terminate an infinite loop. Just close the fractal window, or select another formula.

Next: Publishing your formulas

See Also
Debugging

Publishing your formulas

When you have written a new formula, you might want to publish it, so other people can use it as well. You can publish it using the online formula database at formulas.ultrafractal.com. When you register with the formula database, you reserve a two- or three-letter abbreviation that you can use to name your formula files.

Before publishing your formula, take a step back and ask yourself the following questions:

- Does your new formula add something that is not already available in the formula database? In other words, is it worth trying for other people?
- Have you carefully considered all parameters and the effect that they have? Usually, formulas with fewer parameters are more effective and easier to use. Don't add extra parameters just because they look interesting.
- Have you added help and hints to make your formula easier to use?

In general, you should take responsibility for the formulas that you publish. For complex formulas, you might want to have them beta-tested by a small group of people first. Consider that it might not be possible to fix problems later without breaking compatibility with parameter sets produced by earlier versions.

When you are revising and improving your formula files, make sure that you do not break backwards compatibility either. Here are some guidelines to help you:

- Do not give the formula a different entry identifier, unless you know what you are doing. Parameter files reference the formula by its identifier, so they cannot be restored anymore if the identifier has been changed. You can always change the title of the formula, of course.
- Do not change the identifiers of existing parameters in your formula, since parameter sets reference parameters in a formula by their identifiers. Instead, change their captions.
- When using enumerated parameters, do not change the names of the existing items. It is possible though to change their order and add new items without breaking existing parameter sets.
- When using classes and class parameters, see also Parameter forwards and Importing classes.

See Also
Public formulas
Compatibility

General keyboard shortcuts

The following general keyboard shortcuts apply to any document window in Ultra Fractal, and some can also be used when no document window is open at all.

Shortcut	Menu command	Description
Ctrl+N	File|New|Fractal	Creates a new fractal window.
Ctrl+O	File|Open	Opens an existing document file.
Ctrl+B	File|Browse	Opens a new browser window to open and organize your documents.
Ctrl+D	File|Duplicate	Duplicates the active document window.
Ctrl+S	File|Save	Saves the active document.
F11	-	Toggles keyboard focus between the active document window and the tool windows.
Ctrl+F11 Shift+Ctrl+F11	-	Moves keyboard focus between the tool windows.
Shift+F2..F10	Window|Tool Windows	Shows and hides a tool window.
F12	Options|Tool Windows	Shows and hides all tool windows.
F1	Help|Help	Shows context-sensitive help.
Shift+F1	Help|Contents	Shows the help table of contents.
Ctrl+F1	Help|Index	Shows the help index.

You can use several keyboard shortcuts with most input boxes for layer parameters. These are shortcuts for the commands found in the right-click popup menu for the input box.

Shortcut	Popup menu	Description
Shift+Ctrl+K	Insert/Delete Key	Inserts a key for this parameter at the current frame, or deletes the existing key. See Editing animations.
Shift+Ctrl+X	Explore	Starts the Explore feature for this parameter.
Shift+Ctrl+E	Eyedropper	Starts the Eyedropper feature for this parameter.
Shift+Ctrl+C	Copy Complex Value	For complex parameters, copies both the real and imaginary value to the Clipboard so you can transfer it to another complex parameter.
Shift+Ctrl+V	Paste Complex Value	Copies the complex value on the Clipboard to this parameter.

Next: Keyboard shortcuts for fractal windows

See Also
Workspace

Keyboard shortcuts for fractal windows

The following keyboard shortcuts apply to fractal windows:

Shortcut	Menu command	Description
Ctrl+A	File|Save Parameters	Saves a parameter set describing the fractal. See Parameter files.
Ctrl+E	File|Export Image	Exports the fractal as an image. See Exporting and rendering.
Ctrl+Z	Edit|Undo	Undoes the last action. See Fractal history.
Ctrl+Y	Edit|Redo	Cancels the last Undo command.
Ctrl+C	Edit|Copy	Copies the parameter text of the fractal to the Clipboard. See Copying and pasting.
Ctrl+I	Edit|Copy Image	Copies the image of the fractal to the Clipboard.
Ctrl+V	Edit|Paste	Pastes the contents of the Clipboard into the fractal.
F4	Fractal|Gradient	Shows the gradient editor to edit the colors of the fractal.
F9	Fractal|Zoom In	Zooms in to the center.
F10	Fractal|Zoom Out	Zooms out of the center.
Ctrl+F	Fractal|Full Screen	Shows the fractal in full-screen mode.
F5	Fractal|Normal Mode	Selects Normal mode for zooming, panning, and rotating by dragging the mouse.
F6	Fractal|Select Mode	Selects Select mode for zooming using a zoom box.
F7	Fractal|Switch Mode	Selects Switch mode for switching to a related fractal type.
Ctrl+R	Fractal|Render to Disk	Adds the fractal to the queue of render jobs. See Rendering images.
Ctrl+1..4	Right-click, Gradient|Randomize	Randomizes the gradient of the active layer.
Ctrl+J	Right-click, Gradient|Adjust Colors	Adjusts the colors of the active layer.
Ctrl+] Ctrl+[	Right-click, Gradient|Cycle Colors	Cycles the colors in the active layer.
Ctrl+arrow keys	-	Pans the fractal in steps of one pixel.
Shift+Ctrl+arrow keys	-	Pans the fractal in steps of ten pixels.

Next: Keyboard shortcuts in Select mode

See Also
General keyboard shortcuts
Fractal windows

Keyboard shortcuts in Select mode

The following keyboard shortcuts can be used in a fractal window when Select mode is active:

Shortcut	**Description**
Cursor left/right/up/down	Moves the selection box.
Shift+cursor up/down Delete/Insert	Stretches the selection box.
Page Up	Shrinks the selection box.
Page Down	Enlarges the selection box.
Numeric keypad -/+	Rotates the selection box.
Home/End	Skews the selection box.
Enter	Zooms in.
Ctrl+Enter	Zooms out.
Esc	Cancels Select mode.

Hold down the Ctrl key for fine adjustments.

Next: Keyboard shortcuts for animations

See Also
Keyboard shortcuts for fractal windows
Fractal windows

Keyboard shortcuts for animations

Note: You need Ultra Fractal Animation Edition to work with animations.

The following keyboard shortcuts are used in fractal windows to create and edit animations:

Shortcut	**Menu command**	**Description**
F3	Animation|Animate	Enables or disables Animate mode.
Ctrl+Space	Animation|Play	Starts or stops animation playback.
Shift+Ctrl+<	Animation|Previous Key	Jumps to the first key to the left of the current frame.
Shift+Ctrl+>	Animation|Next Key	Jumps to the first key to the right of the current frame.
Ctrl+<	Animation|Previous Frame	Moves to the previous frame.
Ctrl+>	Animation|Next Frame	Moves to the next frame.
Ctrl+T	Animation|Timeline	Opens the Timeline tool window. This shortcut also works in gradient editors.

The following keyboard shortcuts work in the Timeline tool window:

Shortcut	**Description**
Cursor up/down	Selects the first animated range above or below the current selection.
Cursor left/right	Selects the first animation key to the left or to the right of the current selection.
Ctrl+Cursor left/right	Moves the current selection to the left or right.
Ins	Toggles Insert mode. In Insert mode, click inside the timeline view to insert a new animation key.
Del	Deletes the current selection.
Ctrl+T	Closes the Timeline tool window. This shortcut works as a toggle.

Next: Keyboard shortcuts for gradient editors

See Also
Keyboard shortcuts for fractal windows
Animation
Fractal windows

Keyboard shortcuts for gradient editors

The following keyboard shortcuts apply to gradient editors:

Shortcut	**Menu command**	**Description**
Ctrl+R	File|Replace	Opens an existing gradient, replacing the current gradient.
Ctrl+Z	Edit|Undo	Undoes the last action.
Ctrl+Y	Edit|Redo	Cancels the last Undo command.
Ctrl+C	Edit|Copy	Copies the gradient to the Clipboard.
Ctrl+V	Edit|Paste	Pastes the contents of the Clipboard into the gradient.
Ins	Edit|Insert	Inserts a new control point.
Ctrl+Del	Edit|Delete	Deletes the selected control points.
Ctrl+A	Edit|Select All	Selects all control points.
F4	Gradient|Fractal	Shows the fractal window that owns the gradient editor. This shortcut works as a toggle.
Ctrl+F2	Gradient|Color	Activates the color bars.
Ctrl+F3	Gradient|Opacity	Activates the opacity bar.
Ctrl+L	Gradient|Link Color and Opacity	Links or unlinks the color and opacity bars. See Transparent gradients.
F5..F8	Gradient|Randomize	Randomizes the gradient.
Ctrl+J	Gradient|Adjust Colors	Adjusts the color balance and brightness of the gradient. See Adjusting gradients.
Ctrl+Enter	Right-click, Select Color	Selects the color of the selected control points.
Ctrl+Left	-	Selects the previous control point.
Ctrl+Right	-	Selects the next control point.

Hold down Ctrl while clicking in the gradient editor to add a new control point or to delete the selected control point. When dragging control points, hold down Shift to constrain the movement to horizontal or vertical only.

Next: Keyboard shortcuts for the Layer Properties tool window

See Also
General keyboard shortcuts
Gradients

Keyboard shortcuts for the Layer Properties tool window

The following keyboard shortcuts apply to the Layer Properties tool window. Most shortcuts work on the active tab. Use Ctrl+Tab and Ctrl+Shift+Tab to activate other tabs.

	Shortcut	Description
	Ctrl+Alt+Enter	Opens the browser to select another fractal formula or coloring algorithm.
	Ctrl+Alt+R	Reloads the fractal formula, coloring algorithm, or selected transformation.
	Ctrl+Alt+E	Edits the fractal formula, coloring algorithm, or selected transformation.
	Ctrl+Alt+C	Copies the fractal formula, coloring algorithm, or selected transformation (including parameter settings) to the Clipboard. If the Location tab is active, the location is copied.
	Ctrl+Alt+V	Pastes the contents of the Clipboard into the active tab.
	Ctrl+Alt+Z	Resets the parameters of the fractal formula, coloring algorithm, or selected transformation. If the Location tab is active, the location of the active layer is reset.
	Ctrl+Alt+F1	Shows help (if available) on the fractal formula, coloring algorithm, or selected transformation.
	Ctrl+Alt+A	Adds a new transformation.
	Ctrl+Alt+D	Deletes the selected transformation.
	Ctrl+Alt+T	Enables or disables the selected transformation.
	F2	Renames the selected transformation.

Next: Keyboard shortcuts for the Fractal Properties tool window

See Also
General keyboard shortcuts
Formulas
Coloring algorithms
Transformations

Keyboard shortcuts for the Fractal Properties tool window

The following keyboard shortcuts apply to the Fractal Properties tool window:

	Shortcut	Description
	Shift+Alt+A	Duplicates the active layer or group and inserts the new copy in the layers list.
	Shift+Alt+G	Creates a new layer group and inserts it in the layers list.
	Shift+Alt+D	Deletes the active layer.
	Alt+Up	Activates the next layer.
	Alt+Down	Activates the previous layer.
	Shift+Alt+K	Turns the active layer into a mask for the layer above it, or turns it into a normal layer again. See Masks.
	Shift+Alt+O	Shows the transparency of the mask in grayscale, making it easier to edit.
	Shift+Alt+M	Selects the merge mode of the active layer.
	Shift+Alt+1..9, 0, Left, Right	Changes the opacity of the active layer.
	Shift+Alt+F	Shows or hides the active layer.
	Shift+Alt+E	Makes the active layer editable or not editable.
	Shift+Alt+T	Turns layer transparency on and off. See Transparent layers.
	F2	Renames the active layer.
	Shift+Alt+Up	Moves the active layer one place up.
	Shift+Alt+Down	Moves the active layer one place down.
	Shift+Alt+C	Copies the active layer to the clipboard.
	Shift+Alt+V	Pastes the contents of the Clipboard into the layers list.

When clicking on the **Visible**, **Editable**, or **Transparent** icons, hold down Shift to toggle all other layers on and off.

Next: Keyboard shortcuts for formula editors

See Also
General keyboard shortcuts
Layers

Keyboard shortcuts for formula editors

The following keyboard shortcuts apply to formula editors:

Shortcut	**Menu command**	**Description**
Ctrl+P	File|Print	Prints the active document.
Ctrl+Z	Edit|Undo	Undoes the last action.
Ctrl+Y	Edit|Redo	Cancels the last Undo command.
Ctrl+X	Edit|Cut	Moves the selected text to the Clipboard.
Ctrl+C	Edit|Copy	Copies the selected text to the Clipboard.
Ctrl+V	Edit|Paste	Pastes the contents of the Clipboard into the editor.
Ctrl+A	Edit|Select All	Selects all text.
Ctrl+F	Edit|Find	Finds text in the active document. See Finding text and formulas.
Ctrl+H	Edit|Replace	Finds and replaces text in the active document.
Ctrl+G	Edit|Go to Line	Jumps to the specified line number.
Ctrl+E	Edit|Find Formulas	Finds formulas in the active document.
Shift+Ctrl+Cursor up/down	Edit|Previous Section Edit|Next Section	Jumps to the previous or next section in the formula, for quick navigation.
Ctrl+I	Edit|Indent Block	Moves the selected text to the right. See Indenting and commenting.
Ctrl+U	Edit|Outdent Block	Moves the selected text to the left.
Ctrl+L	Edit|Comment Block	Turns the selected text into a comment.
Ctrl+K	Edit|Uncomment Block	Turns the selected comment into normal text.
Ctrl+M	Insert|New Formula	Inserts a new formula.
Ctrl+J	Insert|Complete Template	Replaces text just before the cursor with the corresponding code template. See Templates.
Ctrl+Q	-	Quickly jumps to another formula.
F1	Help|Help	Searches for help on the word at the cursor position. This works for reserved words, built-in functions, predefined symbols, settings, compiler directives, and labels.

Next: Keyboard shortcuts for browsers

See Also
General keyboard shortcuts
Formula editors

Keyboard shortcuts for browsers

The following keyboard shortcuts apply to modeless browsers. Most shortcuts also work in modal browsers.

Shortcut	Menu command	Description
Ctrl+X	Edit|Cut	Removes the selected items and copies them to the Clipboard. The items are not actually removed until you use the Paste command. See Organizing your work.
Ctrl+C	Edit|Copy	Copies the selected items to the Clipboard.
Ctrl+V	Edit|Paste	Inserts the items you have copied or cut into the selected location.
Ctrl+Del	Edit|Delete	Deletes the selected items.
F2	Edit|Rename	Renames the selected item.
Ctrl+A	Edit|Select All	Selects all items in the window.
Ctrl+I	Edit|Invert Selection	Reverses which items are selected and which are not.
Ctrl+F	Edit|Find Entries	Finds entries in files. See Finding files and entries.
Ctrl+L	View|Library Only	Shows the library folders only, or shows all folders. See Library mode.
F5	View|Refresh	Refreshes the contents of the browser.
Ctrl+R	Right-click, Render to Disk	Renders the selected fractal file, parameter file, or parameter set to disk. See Exporting and rendering.

See Also
General keyboard shortcuts
Browsers

Purchasing Ultra Fractal

If you want to continue to use Ultra Fractal after the 30-day trial period, you must purchase it. You will then receive a personal license key that turns your evaluation copy into a full registered version.

The full version does not mark exported and rendered images, and it does not show any evaluation reminder dialog boxes. In addition, you are entitled to free support via e-mail and upgrades for a reduced price.

Ultra Fractal 5 comes in two editions: the **Standard Edition** and the **Animation Edition**. The difference is that the Animation Edition supports animations and network calculations, and the Standard Edition does not. To verify the edition that you are currently using, click About on the Help menu. If you own the Standard Edition, you can purchase an additional upgrade to the Animation Edition.

You can place your order online via a secure server. All major credit cards and PayPal are supported. To order, go to the Ultra Fractal web site:

- Purchase Ultra Fractal

When your order is complete, your personal license key will be sent to you by e-mail in a few minutes. Please contact info@ultrafractal.com if you have any questions!

Next: Entering your license key

See Also
License information
Support

Entering your license key

After you have purchased Ultra Fractal, your personal license key will be sent to you via email in a few minutes. This license key turns your evaluation copy of Ultra Fractal into a full version.

1. Start Ultra Fractal, and click **Enter License** in the Evaluation Reminder dialog that appears. (Alternatively, click **Enter License** on the Purchase menu if Ultra Fractal is already opened.)
2. Copy and paste your license key from the email that you have received in the Enter License dialog. Click Next to proceed.
3. If you are upgrading from an earlier version of Ultra Fractal, or if you have purchased an upgrade from the Standard Edition to the Animation Edition, you will be prompted to enter a previous license key as well. Ultra Fractal will search for previous license keys and fill them in if possible. (Contact info@ultrafractal.com if you have lost your previous license key.)
4. Click Restart to complete the registration process.

Be sure to make a backup of your license key, so you can enter it again in case you have to reinstall Ultra Fractal, for example on a new computer.

Please contact info@ultrafractal.com if you have any questions. Thank you for purchasing Ultra Fractal 5!

Next: License information

See Also
Purchasing Ultra Fractal
Support

License information

If you purchase Ultra Fractal 5, you will receive a personal license. This personal license entitles you to install and use the full version of Ultra Fractal 5 on any computer, as long as you are the only person using the installed copies.

The license key that you receive upon purchasing Ultra Fractal 5 is personal and confidential. You may not publish it, give it away, or sell it in any way. If you would like to introduce Ultra Fractal to someone else, please point him or her to the Ultra Fractal web site at www.ultrafractal.com so he or she can download a fresh evaluation copy.

In case you would like to purchase multiple copies of Ultra Fractal, for example for classroom use, you can purchase a site license for a reduced price. Contact info@ultrafractal.com for details and pricing.

See Also
Purchasing Ultra Fractal
Support

Support

If you need help using Ultra Fractal, refer to the other topics in this help file first. You will usually find the information you need here. Try to do some of the included tutorials if you are new to Ultra Fractal. The help file is also available as a printable PDF manual on www.ultrafractal.com.

In case the online help does not solve your problem, ask a question on the public Ultra Fractal mailing list. This is also a great place for sharing tips and parameter sets. See Mailing list.

To find links to tips, articles, and more tutorials, visit the **Resources** section on the Ultra Fractal web site at www.ultrafractal.com. Also take a look at the **FAQ** page for answers to frequently asked questions.

For technical support, you can also contact the author of Ultra Fractal via email at info@ultrafractal.com.

Next: Mailing list

See Also
Getting help
Purchasing Ultra Fractal
Tutorials

Mailing list

The Ultra Fractal mailing list is a great place to ask questions and to share tips or fractal images with other Ultra Fractal users. See www.ultrafractal.com/mailinglist.html for more information and instructions for signing up.

For programming-related questions and discussion, there is a separate UF Programmers mailing list, also accessible via the page above.

Warning: the mailing list generates about 20 to 50 messages per day. To make it easier to handle this amount of messages, you can set up your e-mail software to automatically place messages from the Ultra Fractal mailing list in a separate folder.

In Outlook Express, you can do this with Tools|Message Rules|Mail. In Mozilla Thunderbird, use Tools|Message Filters. Messages from the Ultra Fractal list always contain [ultrafractal] in the subject line, so that is a good way to identify them.

Next: Acknowledgements

See Also
Copying and pasting fractals
Copyright and tweaking
Support

Acknowledgements

First of all, I would like to thank the members of the beta test team for spending so much of their time testing and evaluating Ultra Fractal 5 (in no particular order): Toby Marshall, Ken Childress, Janet Parke, Jim Blue, David Makin, Ron Barnett, Kerry Mitchell, Jos Leys, Mark Townsend, Damien Jones, Paul DeCelle and Linda Allison.

Special thanks go to Damien Jones for writing most of the common.ulb file and for hosting the Ultra Fractal web site, the formula database, and the mailing list, and to Janet Parke for writing almost all tutorials included in this help file.

Finally, I would like to thank all Ultra Fractal users for their continued support that enables me to keep improving Ultra Fractal. I want to thank the formula authors in particular for publishing such a wealth of formulas to the online formula database, which has made Ultra Fractal much more valuable for everyone.

Ultra Fractal 5 was developed with Borland Delphi 7. Ultra Fractal uses the following third-party libraries:

- The zlib compression library, written by Jean-Ioup Gailly and Mark Adler
- The JPEG library, written by Jacques Nomssi Nzali
- Indy internet components
- FastMM memory manager
- SynEdit and VirtualTree components

See Also
Support