Mandelbulber

Tutorial & User Manual

Version: 0.9.1 (22nd, September 2016)

based on Mandelbulber Version 2.0.9

Download: https://sourceforge.net/projects/mandelbulber/

Project Website: https://github.com/buddhi1980/mandelbulber2

Forum: http://www.fractalforums.com/mandelbulber/

Editors:

Krzysztof Marczak (buddhi1980@gmail.com), Graeme McLarekin (mclarekin@gmail.com), Sebastian Jennen (sebastian.jennen@gmx.de)

Table of Contents

1. What are fractals?

Fractals are objects with self-similarity, where the smaller fragments are similar to those on a larger scale. A characteristic feature is to have subtle details even at very high magnification.

1.1. Mandelbrot set

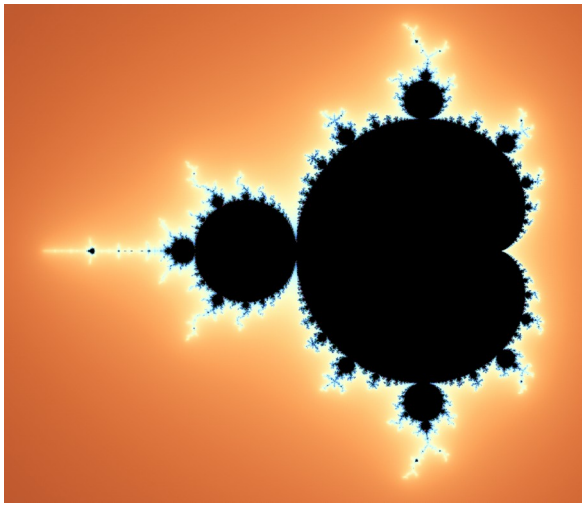


Illustration 1: Mandelbrot Set

This is a typical example of a two-dimensional fractal generated mathematically. This image is created with a very simple formula, which is calculated in many iterations:

$$z_{n+1} = z_n^2 + c$$

“z” is a complex number (*a* + i*b*), where i = (imaginary number). The number is made of two parts : “*a*” the real part and “i*b*” the imaginary part.

“c” is the coordinates of the image point to be iterated.

In 2D, “z” is a vector containing two complex number coordinates , x and y, (these points represent the pixel location where x represents the real part of the number [a] and y represents the imaginary part of the number [b]). Because they are complex numbers, they can be positive or negative, but also there will still be a mathematical solution if a function requires the square root of a negative number.

Each original point (pixel position) is tested in the formula iteration loop, to determine if it belongs to the formula specific mathematical fractal set.

The initial value of point “z” is assigned to equal “c”, (z0 = c), this parameter is then used repeatedly in the iteration loop.

$z_{n+1} = z_n^2 + c$ -iteration-> $z_{n+2} = z_{n+1}^2 + c$ -iteration-> $z_{n+3} = z_{n+4}^2 + c$ etc.

Termination conditions are applied to ensure the formula does not iterate to infinity. The most common conditions used are called Bailout and Maxiter. Maxiter is simply a condition to stop iterating when a maximum numbers of iterations is reached.

The Bailout condition stops the iteration loop if the formula transforms (moves) the point further than a set distance away from an “origin”.

In the Mandelbrot formula, after each iteration, the modulus of a complex number is calculated; in other words, the length of the vector from the origin (x = 0, y = 0) to the current “z” point. This vector length is often called “r” for it is the radius from the center (origin) to the current “z” point.

In this example, when the length r > 2 (i.e Bailout = 2), the termination condition has been met, then the iteration process is stopped and the resulting image point is marked with a light color. When, after many repeated iterations, “r” is still less than 2, then it can be considered for simplicity that such a result will continue indefinitely. Iterations are therefore interrupted after a certain number of iterations (Maxiter). This point is marked on the image with black. This results in a “set” of points that do not reach bailout termination (black) and the rest of the points given lighter colors (dependent on a chosen coloring method).

1.2. 3D fractals

The three dimensional fractal type, the "Mandelbulb," is calculated from a fairly similar pattern to the Mandelbrot set. The difference is that the vector “z” contains three complex numbers (x, y, z) or four dimensions (x, y, z, w). As they are part of the “z” vector, they are denoted as (z.x, z.y, z.z). Examples being Hypercomplex numbers and quaternions.

They can also be created by modification of quaternions or by a specific representation of trigonometric vectors. Generally, common maths operators are used, e.g., addition, multiplication, squaring, and powers), and also conditional functions (e.g., if z.x > z.y, then z.x = something).

Some other types of 3D fractal objects are based on iterative algorithms (IFS - Iterated Function Systems). An example would be the famous Menger Sponge.

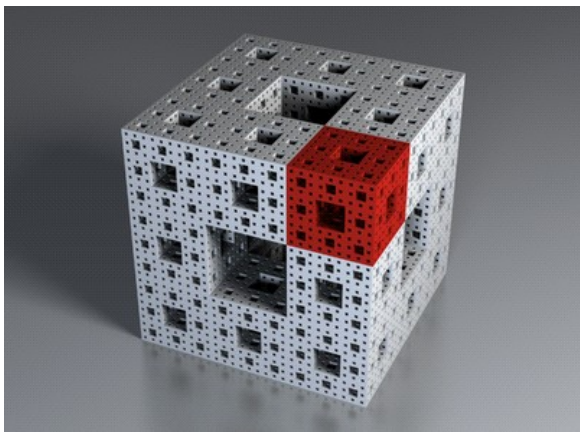


Illustration 2: Menger Sponge with highlighting of self similarity

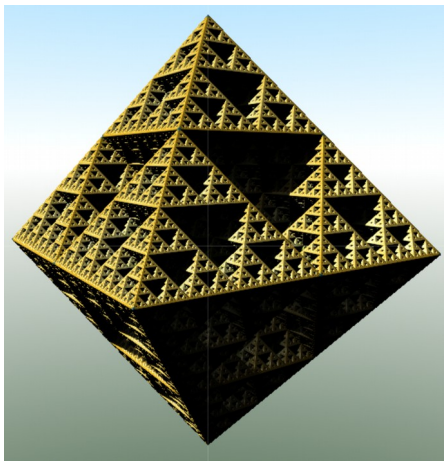


Illustration 3: Sierpinski pyramid

1.3. Mandelbulber Program

Mandelbulber is an easy to use, handy application designed to help you render 3D Mandelbrot fractals called Mandelbulb and some other kind of 3D fractals like Mandelbox, Bulbbox, Juliabulb, Menger Sponge, …. The following sections cover the program interface and give useful information about how to use it.

2. Distance Estimation

Distance Estimation (DE) is the calculation of an estimated distance from the given point to the nearest surface of the fractal. As suggested by the word 'estimate', it is an approximate value. It is calculated using simplified algorithms based on analytical (Analytical DE) or numerical (Delta DE) calculations of gradients.

DE is the most important algorithm required to render three-dimensional fractals within a reasonable time. It achieves a great reduction in the number of steps needed to find the exact area of the fractal while tracking a “photon” traveling toward the object along a ray (a simulated beam of light from the camera eye). A ray is generated for each pixel (1000 x 1000 resolution = 1,000,000 rays). They match FOV from the camera eye (i.e. they are not parallel.)

Without the DE calculation, the proximity of the photon to the fractal surface would need to be repeatedly calculated after each of many very small steps. For example, without an estimate of where the fractal surface is, you may need up to 10,000 steps to trace a ray of light, for every pixel of the image.

Using DE, the size of the steps along the ray of light can be increased, based on the calculated estimate of where the fractal surface should approximately be located. The process of moving along the ray and testing for the surface location is called ray-marching.

Ray marching looks like the illustration below. In each step, an estimation on the distance to the nearest fractal surface is calculated. The photon is moved along the ray by this distance. The next step is re-calculated based on the estimated distance. This distance is less so this time the Photon is moved a smaller distance. The ray-marching becomes more accurate closer to the surface of the fractal. The ray marching ceases when the photon becomes within a set “distance threshold” from the surface or after a maximum number of iteration if the option “stop at maximum iteration” is enabled.

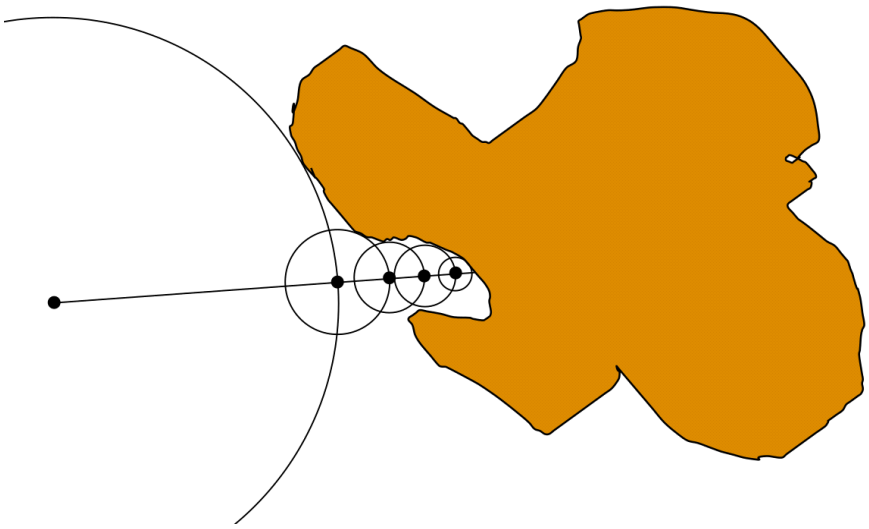


Illustration 4: Raymarching from the camera point along the ray close to the surface

Since the estimation contains some error (sometimes quite large), there is a risk that the step of moving the "photon" will be too large, and incorrectly it will flow into the surface of the fractal. This may result in visible noise in the rendered image.

To prevent this, the "photon" can be moved by the estimated distance multiplied by a number between 0 and 1 ("ray-marching step multiplier"). Steps are then smaller, so there is less risk of "overshooting" the surface, but the rendering time increases due to more steps being required.

Below is an example for a step multiplier of 0.5:

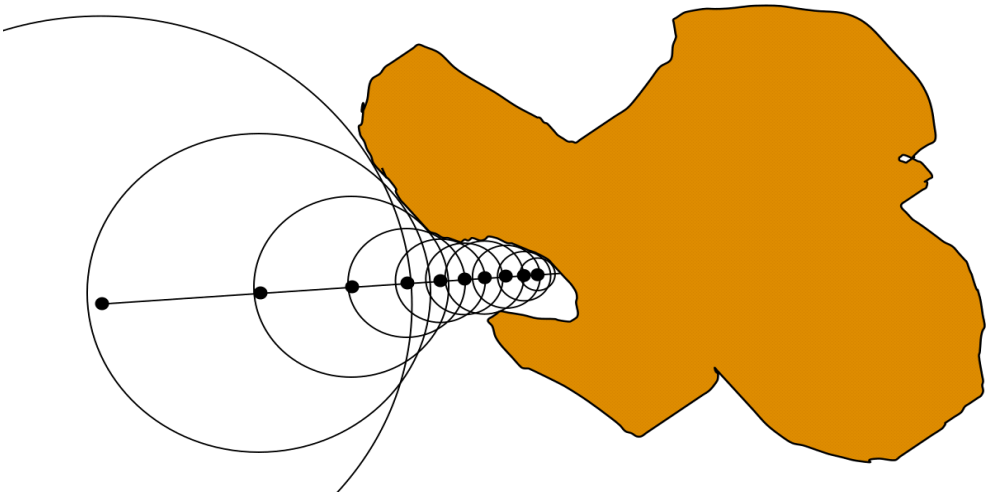


Illustration 5: Raymarching with step multiplier 0.5 produces high iteration count and good DE

Each formula is assigned a DE mode and function (“preferred”). In most cases the preferred mode is Analytical DE (fastest).

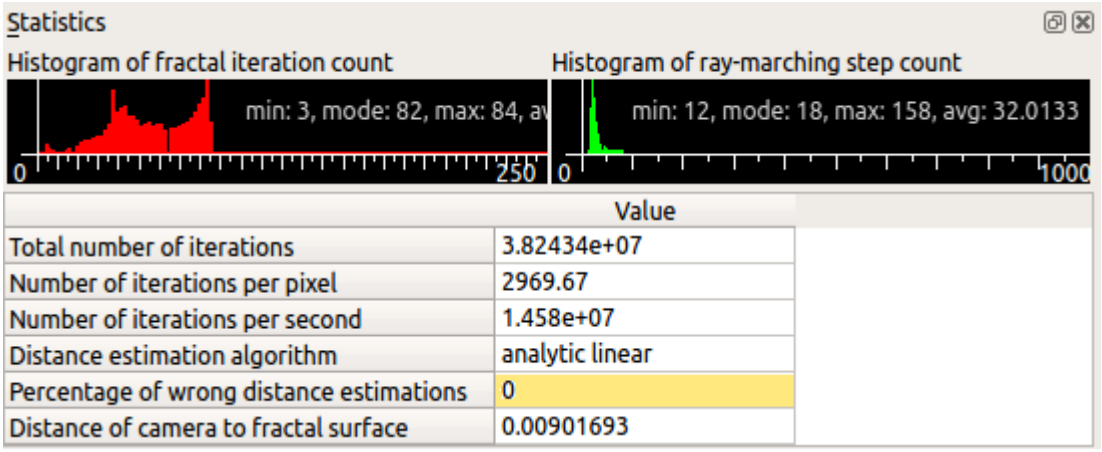
The preferred function is assigned based on whether the formula is transforming in a linear or logarithmic manner. These setting can be varied on the Render Engine tab.

Analytical DE mode is faster than Delta DE mode to calculate. However with some formulas only Delta DE mode will produce a good quality image. The DE modes can be used with either linear or logarithmic DE functions.

Example linear `out->distance = r / fabs(DE);`

Example logarithmic: `out->distance = 0.5 * r * log(r) / DE;`

The quality produced by the DE mode and function combinations is formula specific. The setting of formula parameters can also greatly affect the quality produced by the DE. In some cases the choice of fractal image is determined by what location and parameters can produce good DE quality.



	Value
Total number of iterations	3.82434e+07
Number of iterations per pixel	2969.67
Number of iterations per second	1.458e+07
Distance estimation algorithm	analytic linear
Percentage of wrong distance estimations	0
Distance of camera to fractal surface	0.00901693

Illustration 6: Statistics Tab with histogram data

In the Statistics (enable in View menu) you can see Percentage of Wrong Distance Estimations ("Bad DE"). As a general rule less than 0.01 is good, but it is case specific and 3.0 sometimes is OK and 0.0001 sometimes is not.

3. Ray-marching - Maximum number of iterations vs. distance threshold condition

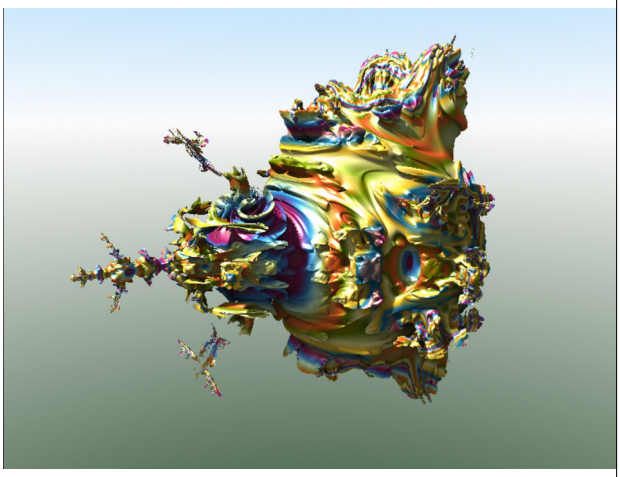
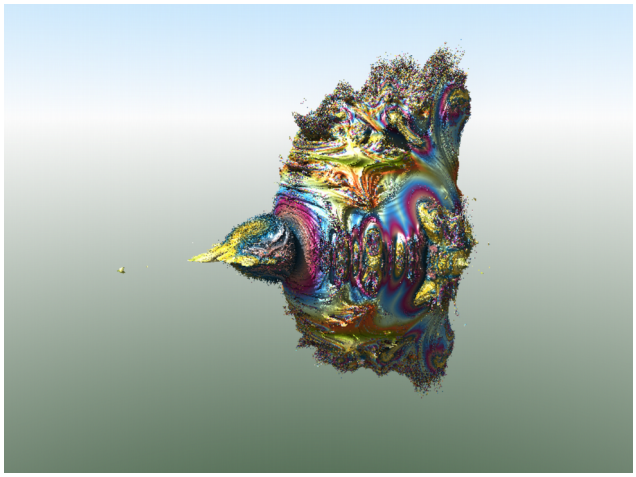
The ray marching distance threshold is the condition where the photon marching along the ray comes within a specified distance from the fractal surface and the ray-marching stops. This controls the size of the detail in the image, and is normally set to vary such that greater detail is obtained for the surface closest to the camera, (in the further regions of the fractal the distance threshold will be larger such that only bigger details are visible). Enabling “Constant Detail Size” on the Rendering Engine tab will make the distance threshold uniform.

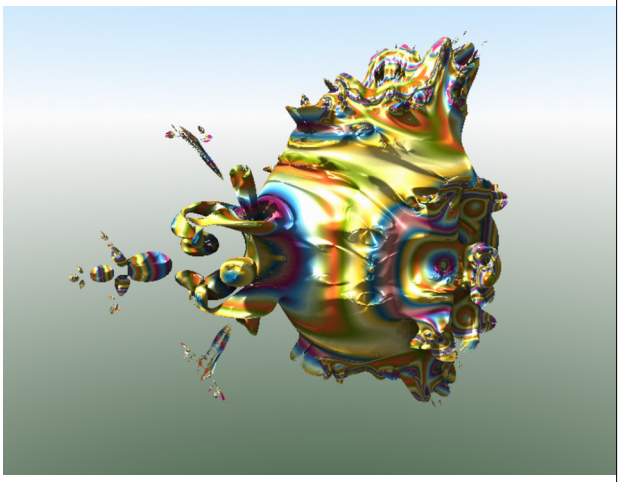
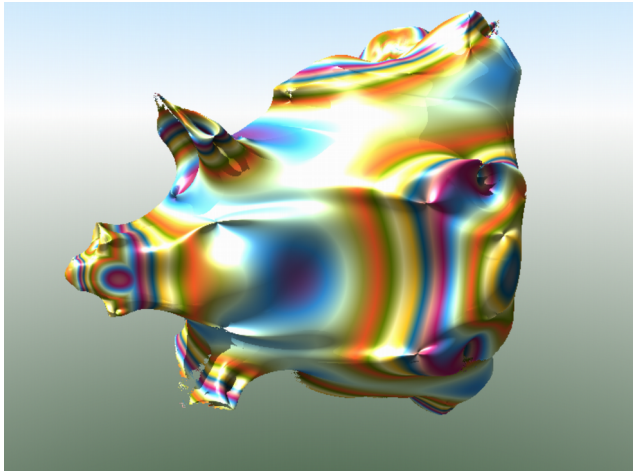
There are two modes of stopping the ray-marching of each image pixel.

1st case: Stop ray-marching at distance threshold (“Stop at maximum iteration” is disabled).

2nd case: Stop ray-marching at point when a maximum number of iterations is reached (“Stop at maximum iteration” is enabled).

First important note: “Stop at maximum iteration” doesn't control the fractal iteration loop. The iteration loop always runs to achieve Bailout, (then if bailout is not reached the iteration stops at Maxiter.).

Example for 1st case:	**Example for 2nd case:**
Ray-marching stops at distance threshold. In most cases the fractal iteration loop stops on bailout condition, (because away from surface it is not possible to reach Maxiter).	Ray-marching stops at the photon step when the maximum number of iterations is reached (ray-marching distance threshold is ignored). In many cases iteration loop stops on bailout condition (away from fractal surface), but on the fractal surface the maximum number of iterations is calculated (when bailout is not reached).
 Illustration 7: Stop ray-marching at bailout	 Illustration 8: Stop ray-marching at maxiter

Example for 1st case:	**Example for 2nd case:**
When maximum number of iterations is set to 4. Even if Maxiter is reached the ray-marching is continued until the ray marching distance threshold is reached.	When maximum number of iterations is reached, then ray-marching is stopped even if distance threshold is not reached.
 Illustration 9: Stop ray-marching at bailout with low bailout	 Illustration 10: Stop ray-marching at maxiter with low maxiter

4. Navigation

To set the current view there are two elements:

- **Camera** - represents a point where the camera is located
- **Target** - represents the point to which the camera is orientated with (i.e., the camera is always looking at the target.)

4.1. Camera and Target movement step

The relationship between the camera point and the target point can be altered manually by changing the numbers in the spin boxes, or by navigating with distance and rotation “steps” defined by the user.

For rotations, the camera is moved by the parameter **rotation step** (default 15 degrees). For movements of the camera and/or the target in a linear direction, the parameter **step** (default 0.5) is used. There are two modes for its use:

a) Relative step mode

The **step** for moving the camera and/or target in a linear direction is calculated relative to the estimated distance from the surface of the fractal. The closer to the surface that the camera is located, the smaller the step. This prevents movement of the camera beneath the surface of fractal, because very close to the surface, a step needs to be very small.

The actual step is equal to the distance from the fractal multiplied by the parameter **step**.

Example: If in this mode, the step is set at 0.5 and the nearest point of the fractal is 3.0, this camera will be moved 1.5 (no matter in which direction).

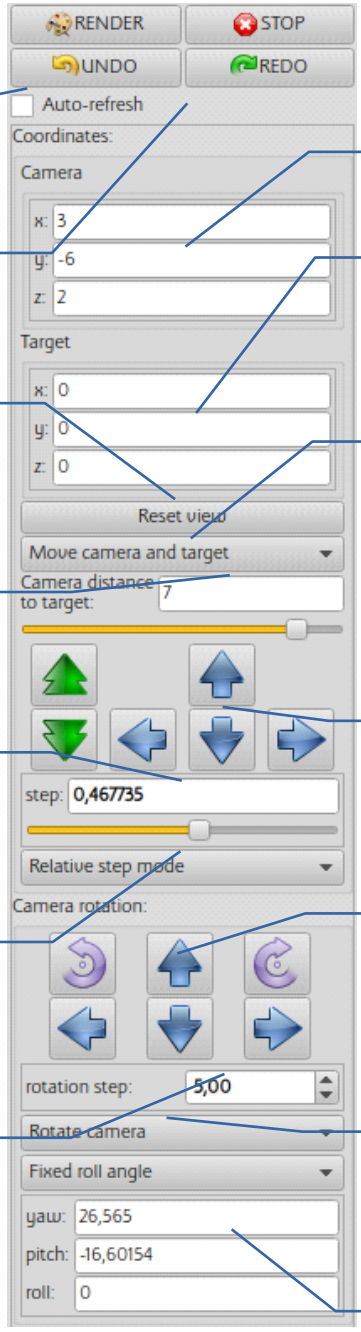
This mode is recommended in animations approaching the surface of the fractal.

b) Absolute step mode

Step movement of the camera and/or target is fixed. Therefore if the step is set at 0.5, the movement will be 0.5 in the direction of the arrow key or mouse pointer.

This mode is recommended for animation camera flying with a fixed (or strictly controlled) speed.

4.2. The camera and target functions



Enables automatic rendering of image after change of any fractal or effect parameter

Mode selection of calculation of rotation angles

Resets the position of the camera so you can see the fractal, while maintaining the last angle of the camera.

Setting the camera's distance to target. This will automatically update when navigating

A step distance for moving the camera and/or the target.

Calculation mode for movement step

Rotation step in degrees

Camera coordinates

Target coordinates

Selecting the camera movement mode.
- simultaneous movement of the camera and target
- only camera movement
- only target movement

Moves the camera and/or target up, down, left, right, backwards and forwards, in linear directions by a given step

Rotates the camera by the rotation step

Selection of camera rotation mode
- camera rotation (target moves around the camera)
- strafing the camera around the target

Camera rotation angles

Illustration 11: Navigation dock

4.3. Linear camera and target movement modes using the arrow buttons

A user can navigate by operating the arrow keys on the Navigation dock, with the user defined steps.

There are three modes for changing the relationship between camera and target.

a) Move camera and target mode

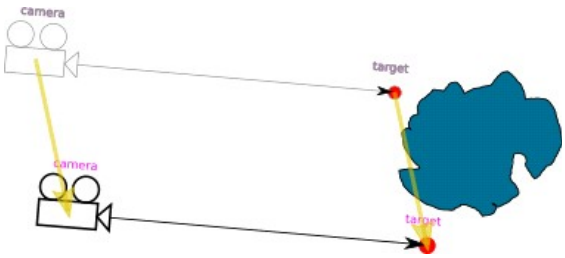


Illustration 12: Movement mode - camera and target

Moves both the camera and the target by the same distance in the same direction. The angle of camera rotation does not change.

b) Move camera mode

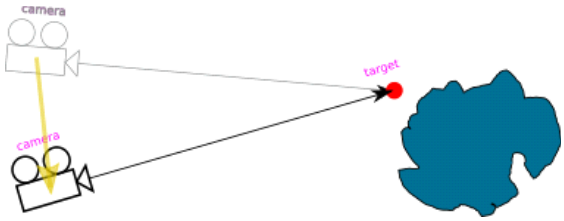


Illustration 13: Movement mode - camera

Moves only the camera and rotates it in respect to the motionless target.

c) Move target mode

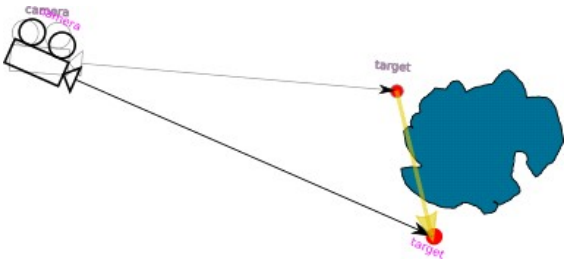


Illustration 14: Movement mode - target

Moves only the target while maintaining a fixed camera position. The camera rotates following the target. **Note:** In Relative Step Mode, the target is moved by distance related to distance of target to fractal surface. If target is inside the fractal (distance = 0), then this option will not work with Relative Step Mode.

4.4. Linear camera and target movement modes using the mouse pointer

A user can navigate in the render window with the user-defined **step** (absolute or relative) and the Mouse Click Function set to “Move the camera.”

a) Move camera and target mode

The target is moved to the point selected by the mouse. The camera is moved by a user defined step, and rotated.

In relative step mode, the camera is moved by distance equals to *distance_to_indicated_point* multiplied by *step* parameter.

In absolute step mode, the camera is moved by distance equals to *step* parameter.

b) Move camera mode

The camera is moved by the step (absolute or relative) in the direction of the mouse pointer, rotating the camera to look at the target. The target remains stationary.

c) Move target mode

The target is moved to the point selected by the mouse. The camera remains at the same point but rotates following the target.

4.5. Camera rotation modes using the arrow buttons

a) Rotate camera

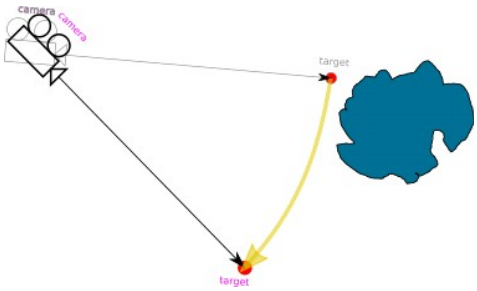


Illustration 15: Rotation mode – around camera

The camera is rotated by the rotation step around its axis and the target is moved accordingly. This is the standard mode for rotation of the camera.

b) Rotate around target

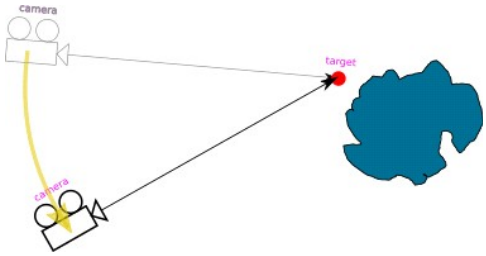


Illustration 16: Rotation mode – around target

The camera is moved around the stationary target by the rotation step, maintaining a constant distance to the target. The camera is rotated to look at the target.

4.6. Reset View

Camera Position is reset, by being zoomed out from the fractal but still maintaing the camera angles.

If the rotations are changed to zero before using Reset View, the camera will then be zoomed out from the target, and rotated to look down the y axis.

4.7. Calculation of rotation angles modes

a) Fixed-roll angle

In this mode, the angle gamma (roll) is constant. Pan the camera left or right always takes place around the global vertical Z-axis (not the render window vertical axis).

This mode can be likened to an aircrafts controls, where all turns are relative to the aircraft's axis, not the ground below. Rotate up / down raises / lowers the nose of the aircraft. Rotate left / right turns in the directions of the wings. Tilting the camera buttons  tilts the aircraft.

When the camera is pointing straight up or down, or when it is upside down in this mode, it is quite difficult to predict the result of the turn.

b) Straight rotation

The camera is rotated around its own local axis (local vertical axis is the render window vertical axis.)

This mode is a more intuitive way to rotate the camera, e.g., turn the camera left always give the visual effect of the camera rotating in that direction. The rotation angles are automatically converted so they are appropriate for the selected direction. This mode changes the gamma angle (roll).

4.8. Camera rotation in animations

With animation, the camera point and the target point can move independently following their own trajectories, with the camera always looking towards the target point. It is important to be aware that the rotation angle of the camera is the result of the camera coordinates and the target coordinates.

There are various ways of animating, depending on the objective.

The following example is a flight animation, with the camera trajectory approaching a location, with the camera rotating simultaneously so that the location is always observed, (as shown in the figure below). The camera positions represent three keyframes.

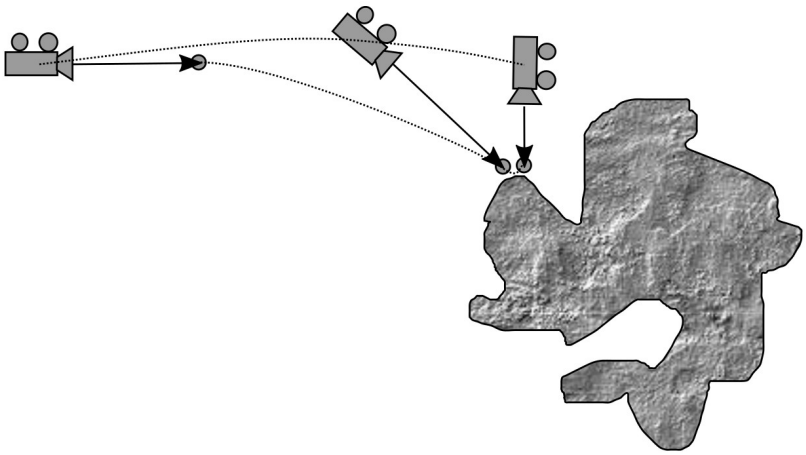
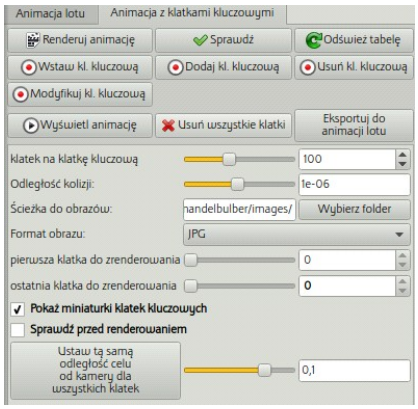


Illustration 17: Keyframe Animation with differing distances camera to target

Between the first and second keyframes, the camera and target both move large distances. But between the second and third keyframes, the camera moves a much greater distance than the target. This can sometimes lead to unexpected camera rotations between keyframes.

To compensate for this, on the Keyframe navigation tab use the button "Set the same distance from the camera for all the frames." This adjusts all keyframes by setting a constant distance between camera and target. It is important to note that the use of this function does not change the visual effect for the keyframes, and will help correct interpolation.



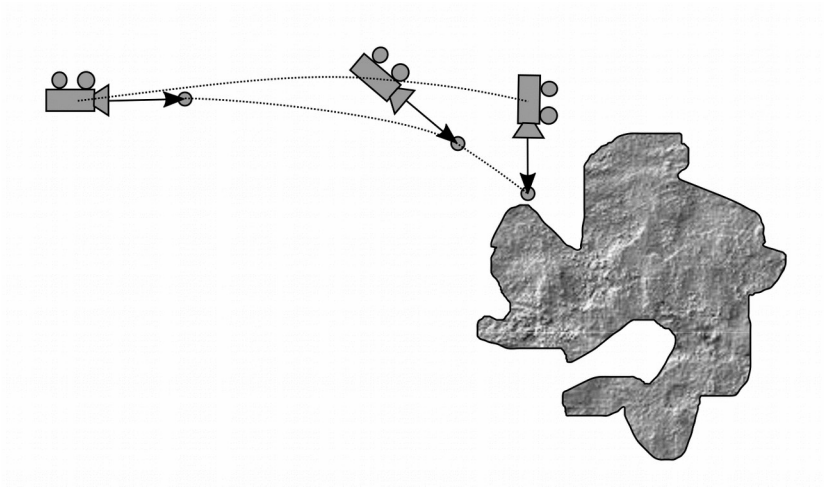
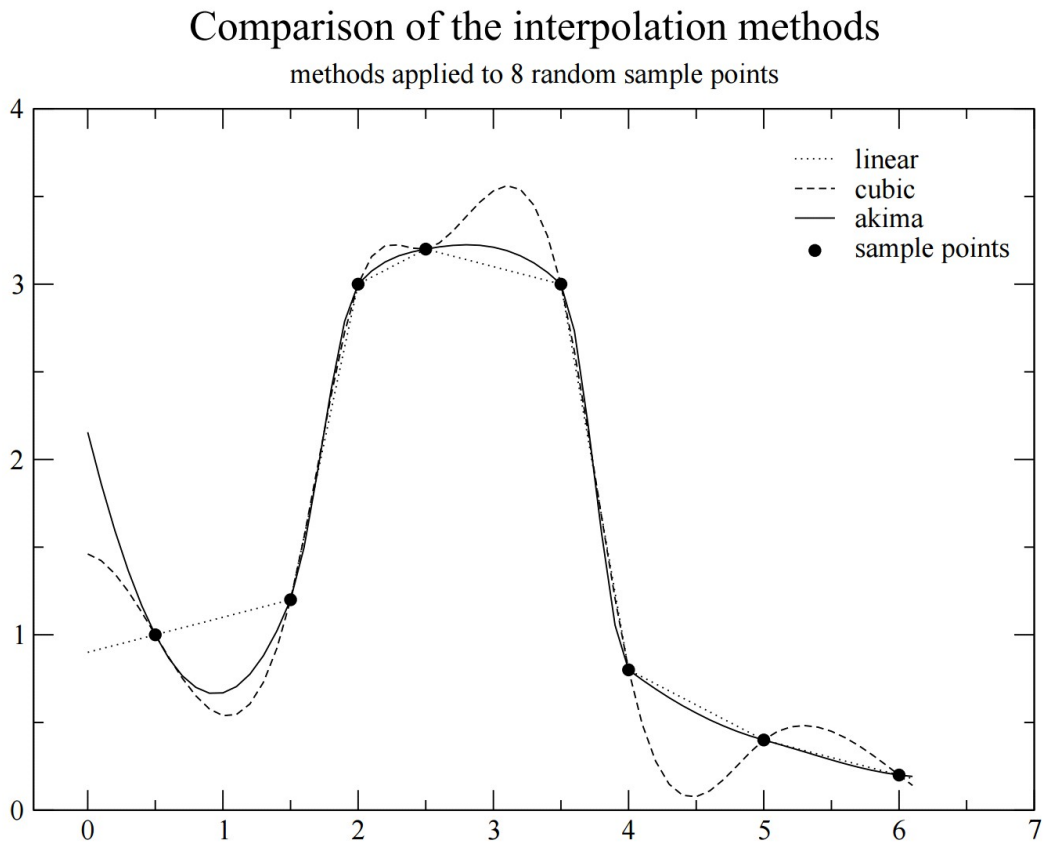


Illustration 18: Keyframe Animation with equal distances camera to target

5. Interpolation

5.1. Akima interpolation



Catmull-Rom and Akima interpolation can both be used to get a “smooth” transition of a value over the keyframes. These methods can mostly be used interchangeably, where Akima stays closer to the points where as Catmull-Rom oscillates a little bit stronger. 5.2 applies to both interpolation forms.

5.2. Catmul-Rom / Akima interpolation - Advices

a) Collision

Fast approaching the obstacle may cause inadvertent drag to the camera towards the center of the object. It is recommended to maintain the principle that one keyframe does not reduce the distance to the object more than 5-times.

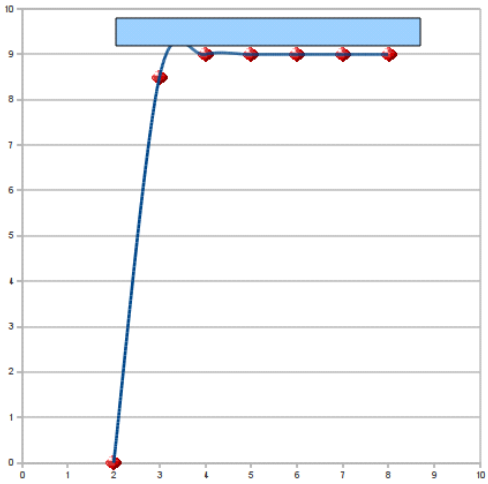


Illustration 19: Catmull-Rom - Collision with bad keyframes

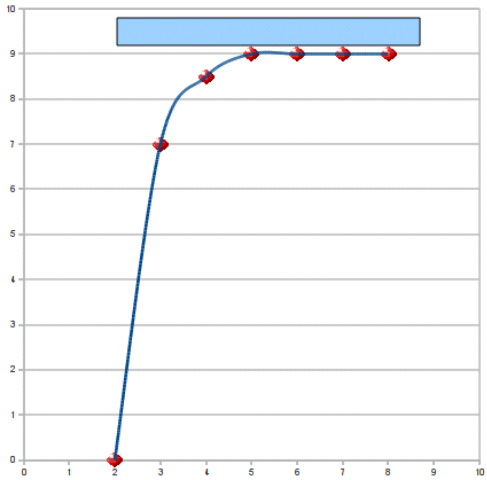


Illustration 20: Catmull-Rom - Collision averted with good keyframes

b) Fly through the gap

It is recommended to place a keyframe at the point where the camera flies through the gap

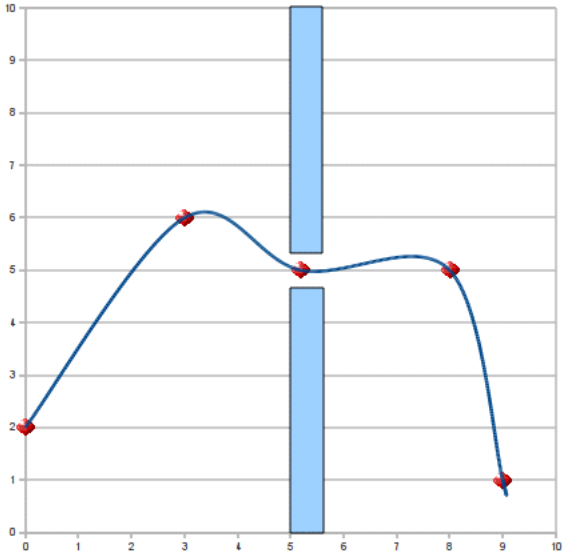
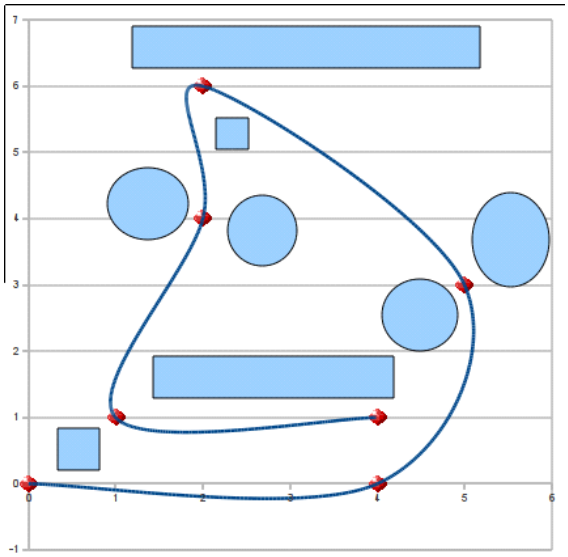


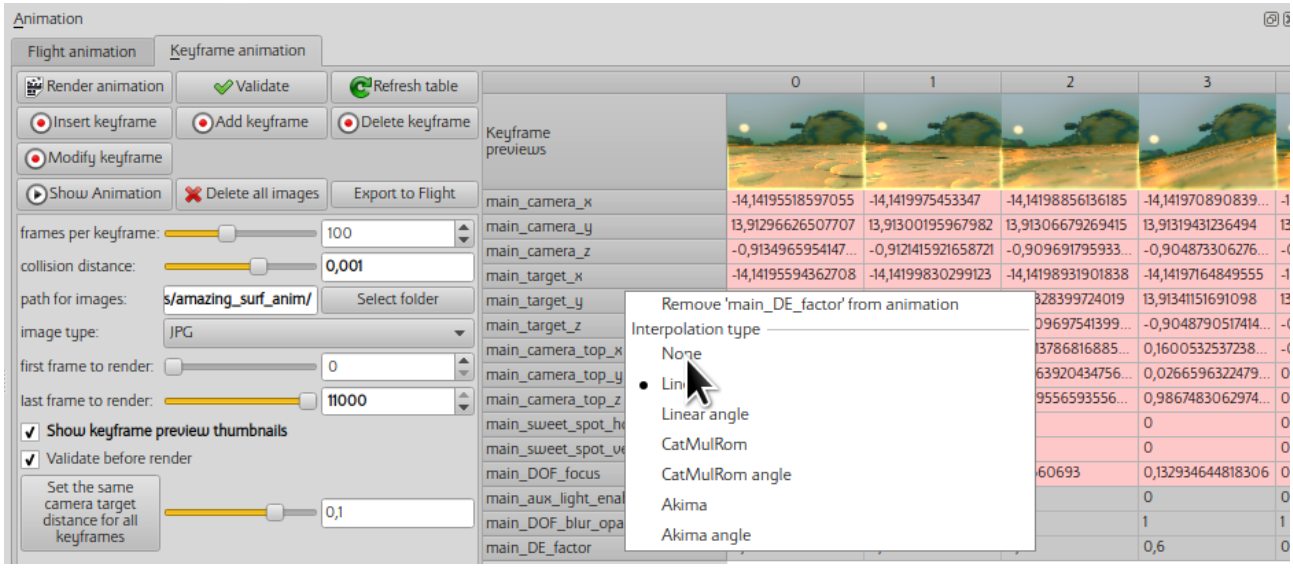
Illustration 21: Catmull-Rom - Fly through gap with keyframe in center of gap

c) Proper conduct cameras between objects



5.3. Changing interpolation types

To change the interpolation algorithm, right click on the parameter list and the options appear. In this example the main_formula_weight parameters have been changed from Akima to Linear. Interpolation type is color coded e.g. Linear parameters are highlighted in grey.



6. NetRender

NetRender is a tool that allows you to render the same image or animation on multiple computers simultaneously. If you have multiple computers connected to an Ethernet network, you can greatly increase overall computing power.

One of the computers (server) manages the process of rendering. It sends a requests to the connected computers (clients) and collects the results of rendering. Other computers (clients) render different portions of the image and send it to the server. There can be only one server (master) but clients (slaves) can be any number. The more clients, the faster the rendering will be.

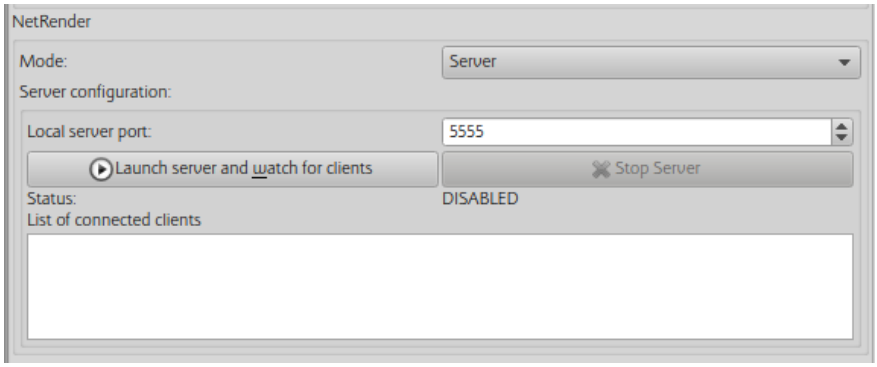
The Server is also the computer which renders the combined image .

The total number of CPUs (cores) used is the sum of server's CPUs cores + all client's CPUs cores.

6.1. Starting NetRender

a) Server configuration

On the computer which will be used as the Server, Mode is set to *Server.*



Local server port should be set to one which is not used by other applications, and is passed through routers (if any are used) and firewall. The default is 5555.

If settings are correct, press *Launch server and watch for clients* button to connect server to existing clients.

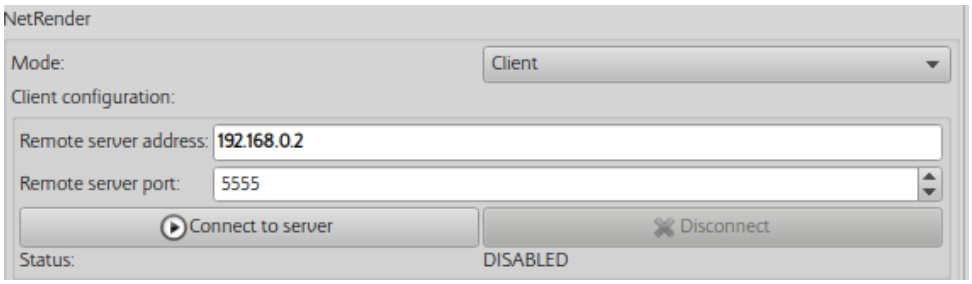
At this point, the server is ready to work

Alternative way to launch the server is to use command line. Example:

$ mandelbulber2 --server --port 5555 pathToFileToRender.fract

b) Configuring the clients

On the computers which will be used as Clents, Mode is set to *Client*

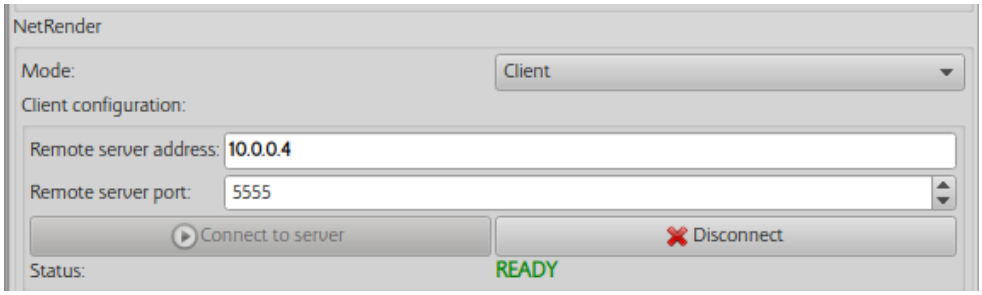


The remote server address must be set to the same as the Server computer which is running Mandelbulber in Server mode. The address can be given as an IP address or a computer name.

The remote server port number must be exactly the same as the setting on the Server.

Press *Connect to server* button to connect to the server

Once the connection is established correctly, the client application should show the status READY



Alternative way to establish NetRender client is to use command line:

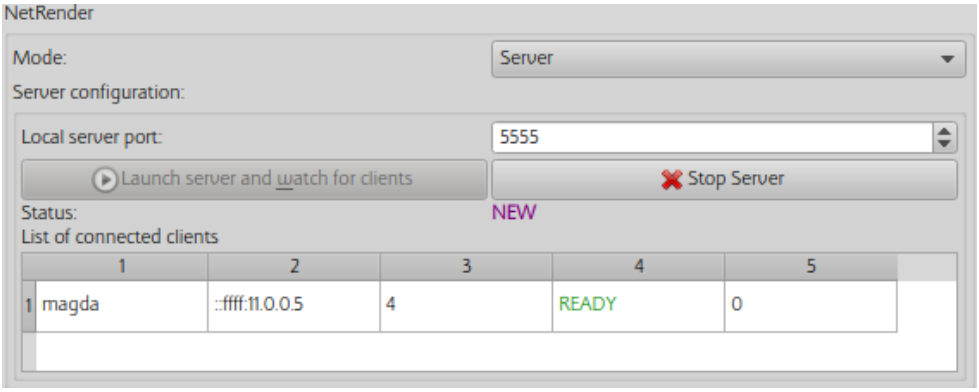
$ mandelbulber2 --nogui --host 10.0.0.4 --port 5555

when connection is successfully established the program should return following message:

NetRender - Client Setup, link to server: 10.0.0.4, port: 5555
NetRender - version matches (2090), connection established

On the Server computer, in the table "List of connected clients" should be shown the name and address of the connected clients and the number of available processors (cores).

i.e magda computer has 4 cores and is READY.



c) Rendering

Only the Server can initiate rendering on the computers connected using NetRender. When the RENDER button is pressed on the server, all the connected computers commence rendering.

In the table "List of connected clients'" in the column "Done lines" will be shown the number of lines the image rendered by each of the computers.

when rendering is finished, the Server computer, will display the complete image. On the Client computers, only that portion of the image which was rendered by that client will be displayed.

7. Q&A , Examples and Hints

Example of MandelboxMenger UI

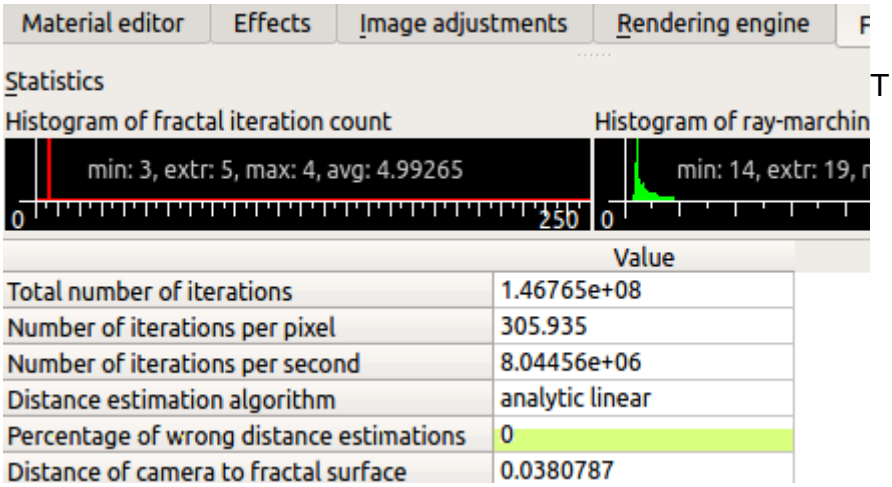
Example settings

(copy to clipboard, then load in Mandelbulber using : File – Load settings from clipboard):

```
# Mandelbulber settings file
# version 2.08
# only modified parameters
[main_parameters]
ambient_occlusion_enabled true;
camera 1.872135433718922 -2.023030528885091 1.871963531652841;
camera_distance_to_target 0.005814178381115117;
camera_rotation -28.76425655707408 26.3550335393397 3.450283685696816;
camera_top -0.1604796308669786 -0.4174088010201082 0.894436236356597;
DE_factor 0.6;
dont_add_c_constant_1 true;
flight_last_to_render 0;
formula_1 91;
formula_2 61;
formula_iterations_2 5;
formula_start_iteration_2 4;
formula_stop_iteration_2 5;
fractal_constant_factor 0.9 0.9 0.9;
fractal_enable_2 false;
fractal_rotation 0 -90 0;
keyframe_last_to_render 0;
main_light_beta 44.34;
main_light_intensity 2;
mat1_coloring_palette_offset 12.83;
mat1_coloring_palette_size 255;
mat1_surface_color_palette fd6029 698403 fff59c 000000 0b5e87 c68876 a51c64 3b9fee d4ffd4 aba53c;
SSAO_random_mode true;
target 1.874642452030676 -2.018463533070165 1.874544631933419;
view_distance_max 28.58330790625501;
volumetric_fog_colour_1_distance 3.55841069795292e-06;
volumetric_fog_colour_2_distance 7.116821395905841e-06;
volumetric_fog_distance_factor 7.116821395905841e-06;
[fractal_1]
fold_color_comp_fold 0.3;
mandelbox_color -0.27 0.05 0.07000000000000001;
mandelbox_rotation_main 9 1.74 3;
mandelbox_scale -1.5;
transf_addCpixel_enabled_false true;
transf_int_1 12;
transf_scaleB_1 0;
transf_scaleC_1 0;
transf_start_iterations_M 4;
transf_stop_iterations_M 5;
```

1) In the example the MengerSponge part is run only on iteration 4. A single iteration of another fractal to make a hybrid is often the best practice.

2) In the Statistics (enable in View menu) you can see Percentage of Wrong Distance Estimations ("Bad DE") is 0, which is good!!. As a general rule less than 0.01 is good, but it is case specific and 3.0 sometimes is OK and .0001 sometimes is not.



The Raymarching step multiplier or "fudge factor" (Rendering Engine tab) is set at 0.6, which is good for a hybrid. If I change it to 0.7 the Percentage of Bad DE leaps up to 0.25 and you can see the areas of quality loss on your image.

Now if we disable the addCpixel Axis swap Constant Multiplier, we find we can now increase the Raymarching Step Multiplier to 0.9, and get a faster render and visually the same quality. So monitoring Percentage of Wrong Distance Estimations is a guide to managing quality. (Note when doing animations you may want to drop the Raymarching step down a bit to allow for what might happen between keyframes.)

3) MandelboxMenger Hybrids can behave a bit differently to a lot of hybrids, in the fact that the Percentage Bad DE often improves when you zoom in.

maximum view distance

Located : Rendering Engine - Common Rendering settings

It is important to optimize this setting to minimize render time. You can reduce until the furthest part of the 3D object(s) starts to disappear. However with animation an allowance should be made for changes between keyframes.

hint/note.
When navigating in Relative step mode, mouse click on spherical_inversion, camera zooms out, tand maximum view distance becomes set on 280. If you don’t reset it your render times will be increased.

Magic Angle Benesi Mag Transforms

In mathematics the Magic Angle = 54.7356° .

When rendering basic mag transforms the image does not render parallel to the standard x,y,z global axis. On the fractal dock, in “Global parameters” set y-axis rotation to 35.2644° (= 90° - 54.7356°). The fractal will then render parallel to the x-y plane.

Example of using Transform_Menger Fold to make Hybrid

Mandelbulber settings file

```
# version 2.08
# only modified parameters
[main_parameters]
ambient_occlusion_enabled true;
camera -1.528388569045064 -1.23063017895654 -0.0251755516595821;
camera_distance_to_target 0.0004503351519815117;
camera_rotation -14.07789975269277 -44.28785609194563 3.773777260910995;
camera_top 0.2333184436621841 0.6598138513697914 0.7142885869084139;
DE_factor 0.7;
flight_last_to_render 0;
formula_1 1052;
formula_2 1010;
formula_3 1052;
formula_4 1009;
formula_iterations_1 5;
formula_start_iteration_4 45;
formula_stop_iteration_2 12;
formula_stop_iteration_4 5;
fractal_constant_factor 0.9 0.9 0.9;
fractal_enable_4 false;
hdr true;
hybrid_fractal_enable true;
keyframe_last_to_render 0;
main_light_alpha 2.6;
main_light_beta 1.59;
mat1_coloring_palette_offset 46.51;
mat1_coloring_palette_size 255;
mat1_coloring_random_seed 647723;
SSAO_random_mode true;
target -1.528310155903731 -1.230317492741513 -0.02549000429402527;
volumetric_fog_colour_1_distance 3.55841069795292e-06;
volumetric_fog_colour_2_distance 7.116821395905841e-06;
volumetric_fog_distance_factor 7.116821395905841e-06;
[fractal_1]
transf_addition_constantA_000 -0.071633 0 0;
transf_function_enabledy false;
transf_int_1 12;
transf_scale 0.5;
transf_scaleC_1 0;
transf_stop_iterations_1 2;
[fractal_2]
transf_scale3D_333 1.055556 1.027778 0.861111;
[fractal_3]
transf_function_enabledx false;
```

On this transform UI, the standard menger sponge formula is split into a start and end function. The simplest way to use this transform is in Hybrid Mode, having the menger fold transform in slots 1 & 3. In slot 2 place any linear type formula or transform. (ie more mengers, kifs, mboxes, amazing surf, folds, rotation , Benesi T1 etc).

In slot 1 disable the stop function and in slot 3 disable the start function, resulting in a standard menger sponge with something in the middle.

BTW in fact you can mix around with the start and stop functions have all enabled if you wish. Generally linear functions all work well together in making hybrids.

In Statistics, maximum is approx. 80 iterations. Generally hybrids take longer to render than standard formulas.

As well as adjusting formula parameters, you can use the iteration controls to tweak hybrids. In this example the first slot is set to repeat for 5 iterations before moving to slot 2. Slot 2 is set to stop at iteration 12, whereas slots 1 & 3 can continue to termination conditions are met (bailout or maximum number of iterations).

In the example above, slot2 of the hybrid sequenced ended at iteration 12. 12 was chosen because how it fitted into the iteration sequence, as follows:

Slot1 x 5
0,1,2,3,4 iteration numbers (note first iteration is iteration number 0.)

slot2
5

slot3
6

slot1 x 5
7,8,9,10.11

slot2
12 // last use of slot2

sequence continues slot1 x 5, slot3, ...to bailout.

So slot2 is used only twice in the iteration process. If I had entered 11 instead of 12 for slot2's stop_iterations, then the slot would have been used only once, if I entered 19 then it would run three times.

7.1. Q&A . How do you get different materials on different shapes?

This is how I have been doing it.

Rectangle at the bottom marked A.

This is where you start a new material or load an existing. The active material is highlighted in blue. Meaning it is active in the **material editor** where you create or modify the material.

Rectangle at top left marked B.

One way to use a material is to go to Global Parameters, click on the material preview image, and the **Material Manager** UI will appear with the materials you have loaded or created. Click on the one you want to use, then close that UI.
Similarly with primitives, click on the material preview image. And with Boolean Mode each fractal/transform has it's own material preview image when you scroll down.