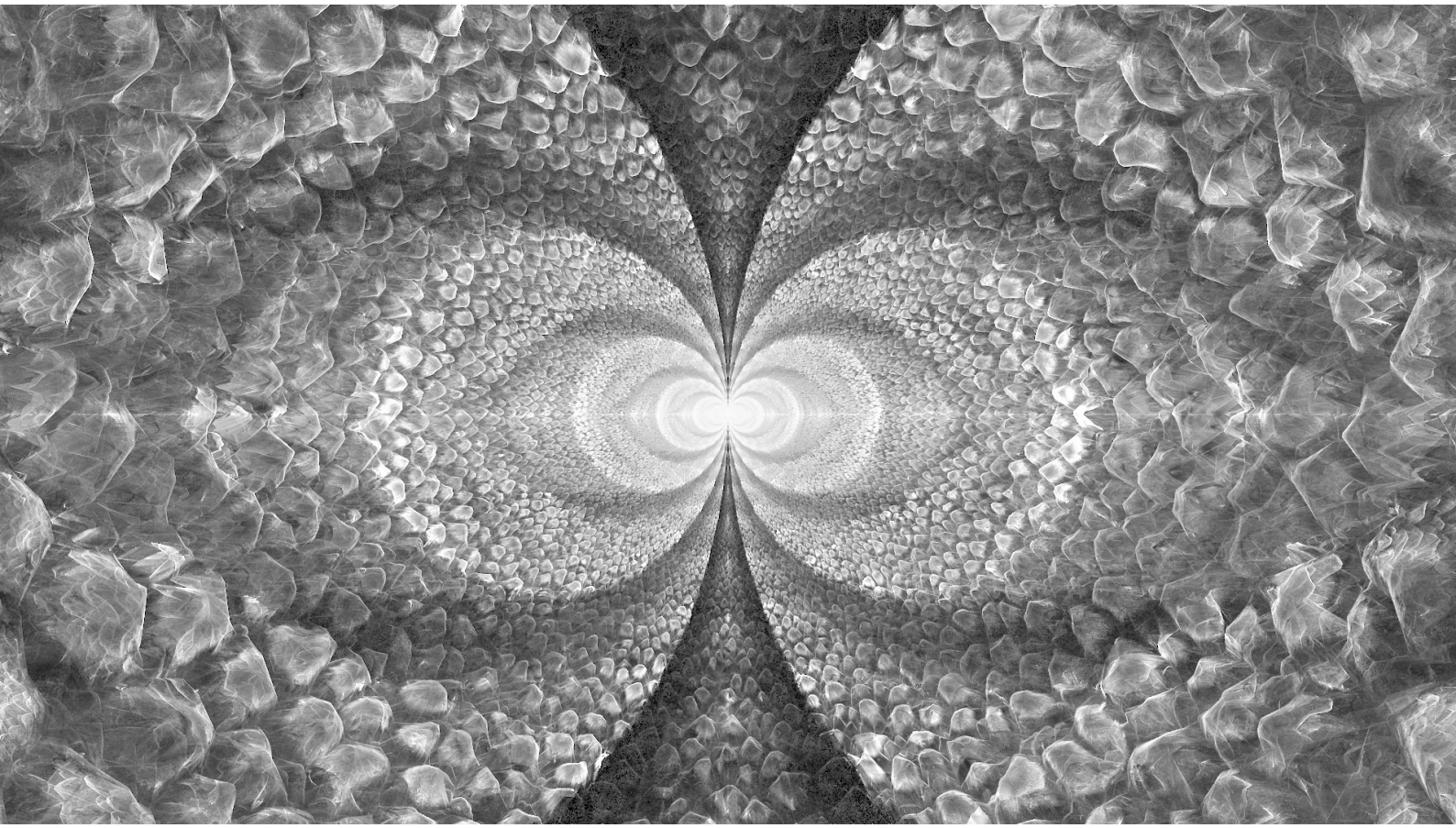
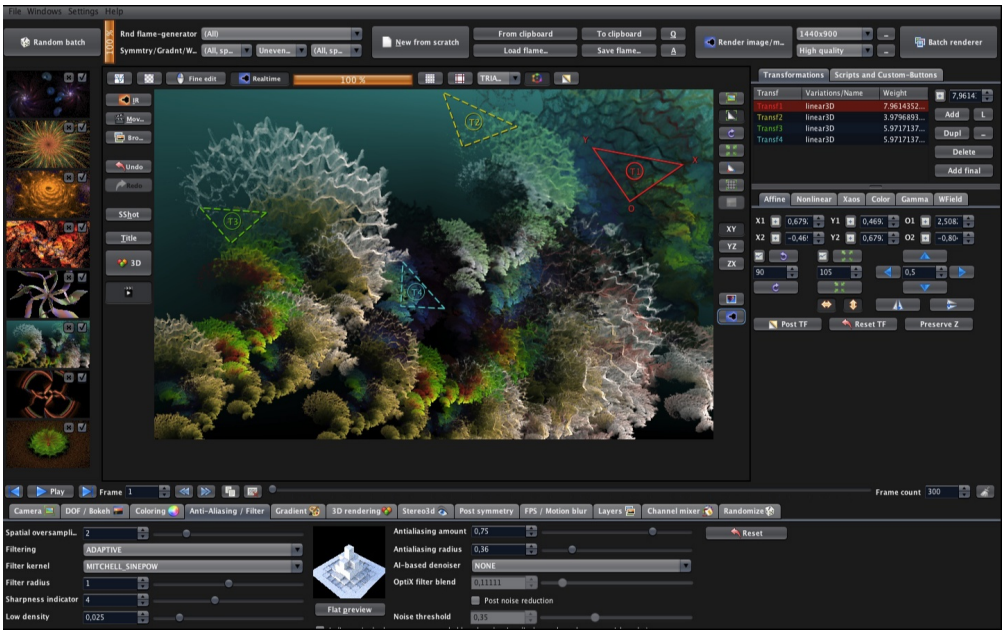


JWildfire users manual

Version 1.3.0

Table of Contents

Chapter 1. Introduction



Welcome to JWildfire and this user manual!

The goal of this book is to describe the numerous functions of the software and to explain how they work. After over 10 years of development there are many functions, so the book is more a reference book rather than one you might read from start to end.

This book does not attempt to teach you how to create fractals, and it does not include any practice materials or examples. That would be too much for this manual and will be part of a kind of "workbook" that I have plans to write.

Anyway, I'm hoping this book will help you to have more fun with the software and achieve the results you are looking for.

Have fun!

Andreas Maschke, Grambek (Germany) 2021

1.1. Acknowledgement

Thanks to Brad Stefanov, D. Aaron Sawyer, Nancy Pierce, my wife Patricia Maschke, and Rick Sidwell for their valuable comments and suggestions for improving this document.

Thanks to the whole fractal art community for their never-ending enthusiasm which always pushes me forward.

Chapter 2. *Main* menu

After launching the program the Fractal-Flame Editor is opened automatically, because it is the main module of JWildfire. There are additional modules and useful functions which can be reached from the *Main* menu.

The *Main* menu contains the following items:

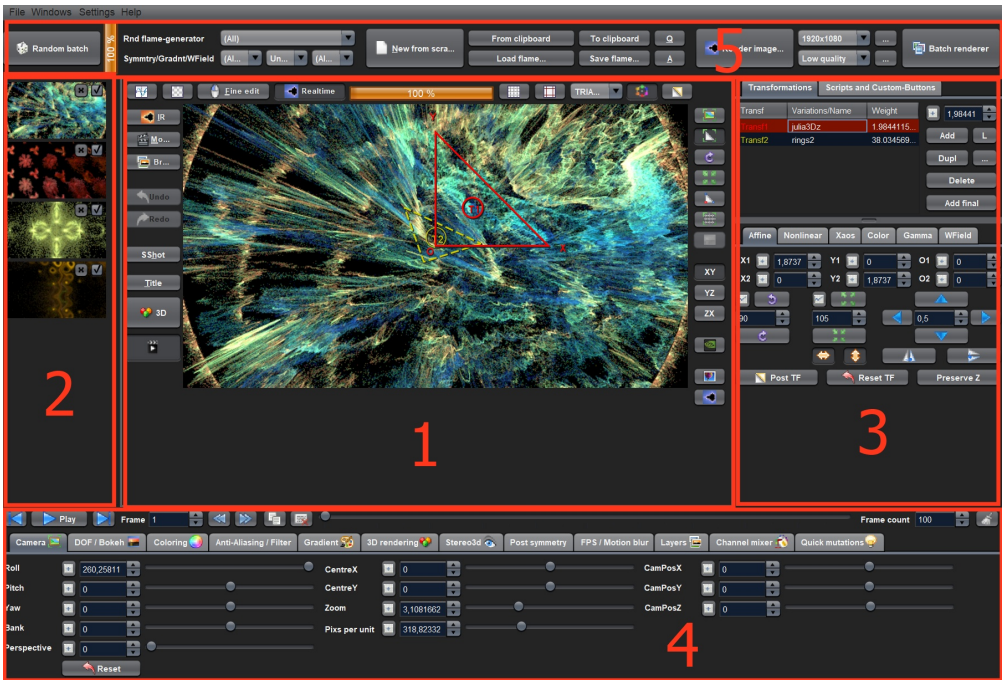
- **File**: some file-related functions
  - **Open flame**: open a flame-file to be edited in the Fractal-Flame Editor
  - **Open image**: open an image-file and display it in a separate window
  - **Quit JWildfire**: quits the program
- **Window**: to open/close the various modules of JWildfire (An opened module is indicated by a small checkmark left of the menu item. If it is not visible it might be in background.)
  - **Fractal flames: Editor**: open the Fractal-Flame Editor
  - **Fractal flames: MutaGen**: open the MutaGen module
  - **Fractal flames: Interactive renderer**: open the Interactive renderer
  - **Fractal flames: Flame browser**: open the Flame browser
  - **Fractal flames: Easy movie maker**: open the Easy Movie Maker
  - **Fractal flames: Dancing flames**: open the Dancing flames module
  - **Fractal flames: Batch renderer**: open the Batch renderer
  - **Fractal flames: Quilt renderer**: open the Quilt renderer
  - **Fractal flames: Mesh generator**: open the Mesh generator
  - **Fractal flames: GPU render**: open the GPU renderer
  - **Fractal flames: Help**: open a popup which displays some helpful information regarding fractal flame
  - **IFlames**: open the IFlames module
  - **Image processing**: open a popup which offers some image-processing possibilities. Goes back to the old Amiga days.
  - **Formula explorer**: open a popup which contains a simple function plotter
- **Settings**: options to customize the program
  - **UI Theme (Look and Feel)**: open a popup to change the UI theme of the whole program
  - **Preferences**: open the *Preferences*-window
  - **Startup settings (Steam)**: only available in the *Steam version*: opens a window to edit the Startup options for the *Steam Edition* of JWildfire
- **Help**: some useful information and the official documentation
  - **User manual**: the official user manual (the file you are currently reading)
  - **System information**: opens a popup to display some useful information about your system

relating to JWildfire, for example, available memory

- **Welcome to JWildfire**: an ancient Welcome-dialog with some useful links, not available in the *Steam Edition* of JWildfire
- **List of changes**: displays the ever growing list of changes. It is a recommended read after each release of the software.
- **Message log**: displays a log of all (error)messages which occured during the current session. You may also find a copy of these messages in the file JWildfire.log in your home folder.
- **GPU rendering**: some compact documentation about GPU-rendering
- **Supported AI-Post-Denoisers**: some compact documentation about *AI-based denoisers*
- **Tip of the day**: a small popup which shows small tips at the startup of JWildfire

For executing functions of the software without the user-interface, there is also a CLI (which currently is limited to rendering and to generating random-flames).

Chapter 3. Fractal-Flame Editor



The Fractal-Flame Editor is the heart of JWildfire. Here you create random flames or fine-tune your artwork. It consists of the following elements:

1. **Preview/Editor Area**
2. **Thumbnail ribbon**
3. **Transformations and Scripts**
4. **Flame settings**
5. **Global functions**

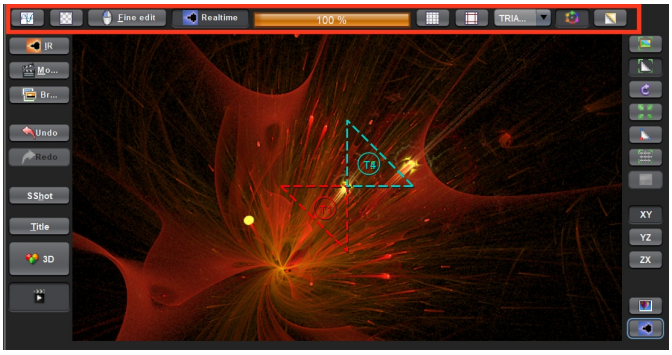
You will find a description of each of these elements as a separate sub-chapter in this manual.

Most controls have tool tips; short descriptions that appear when you hover over them for a second.

3.1. Preview/Editor area

This area is both for previewing fractals and editing them in realtime. The most simple and intuitive way to edit a fractal is to drag or rotate one of the visible controls on top of the fractal (usually in the shape of triangles). However, there are also many other precise ways to modify a fractal flame.

3.1.1. Quick buttons on the top of the preview area



These buttons primarily control the previewing/editing options.

3.1.1.1. *Display/hide variation effect*

Displays the contribution of the currently selected transformation (in the Transformations-area) in the form of a transformed grid. For example, if the current transformation is a rotation, it will show a rotated grid. If there are nonlinear transformations (also known as variations or "plugins"), then the displayed grid may change/deform much more.

3.1.1.2. *Display/hide transparency*

Switch preview-rendering with transparent background on or off. Please note that this only affects the preview display and not the transparency setting of the flame itself.

3.1.1.3. *Fine edit*

Turn on or off "fine editing". Fine editing means that changes are less sensitive to the mouse or touchpad which allows for editing with a higher level of detail.

3.1.1.4. *Realtime*

Turns progressive rendering of the preview on or off. Progressive rendering means that the image starts to render in very low quality and improves with time. Usually this works well, and allows for fluid editing. On slower machines (for example notebooks), this may cause a heavy load, and on some fractals it may cause flickering. Turning it off will produce a render preview of a lower (non-progressive) quality.

3.1.1.5. *Progress bar*

The progress bar indicates the current render progress in the following cases:

- refreshing the *Preview* by using the *Render image preview* button
- rendering an image or movie by using *Render image/movie* button

3.1.1.6. *Turn grid on/off*

Turns on or off a grid which may help to align controls. Note that this grid can be moved (see *Edit view of the controls*), so is intended for control alignment, not image alignment.

3.1.1.7. *Turn guides on/off*

Turns on or off "artistic guides" for

- **center point**
- **rule of thirds**
- **golden ratio**

which may help to find the right camera position/view.

3.1.1.8. *Style of editor controls*

You may select one of the following styles of editor-controls:

- **axis**
- **crosshair**
- **rectangle**
- **triangle** (default)
- **hidden**

3.1.1.9. *Toggle monochrome/colored controls*

Per default, each control is drawn in its own color to make it more distinguishable. Sometimes this may distract you from the fractal. Using this option, you may switch to a less intrusive monochrome display of controls.

3.1.1.10. *Toggle post transform*

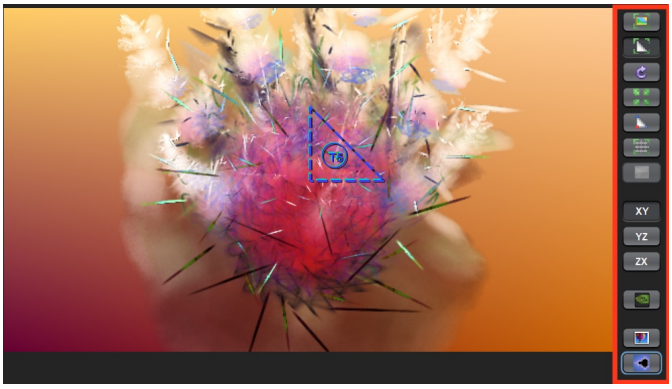
Quick-switch to toggle between editing of the affine-post-transform and the regular affine transform.

Longer explanation: Transformations are executed in the following order:

1. **affine transform**
2. **nonlinear transforms**
3. **affine post-transform**

In the preview area you may either edit the affine transform or the post-affine-transform by moving the visual controls with the mouse or touchpad. This button allows you to switch easily between these two options.

3.1.2. Quick buttons on the right of the preview area



These buttons primarily control the type of editing and refreshing the preview.

3.1.2.1. *Enable view editing mode*

In this mode you change the camera settings (view) when dragging the mouse.

- using the mouse-wheel, you may zoom in/out.
- by dragging the mouse with the left mouse-button pressed, you move the camera.
- by dragging the mouse right or left with the right mouse-button pressed, you rotate the camera.

3.1.2.2. *Move triangles mode*

In this mode you edit the affine or post-affine transformation when dragging the mouse.

- using the mouse-wheel, you may scale up/down.
- by dragging the mouse with the left mouse-button pressed, you move the control, causing a translation
- by dragging the mouse right or left with the right mouse-button pressed, you rotate the control, causing a rotation.

3.1.2.3. *Rotate triangles mode*

This mode is similar to *Move triangles mode*, but you are restricted to rotations. The rotation is done by dragging the mouse right or left while pressing the left mouse-button.

3.1.2.4. *Scale triangles mode*

This mode is similar to *Move triangles mode*, but you are restricted to scale up/down. This operation is done by dragging the mouse right or left while pressing the left mouse-button.

3.1.2.5. *Enable free point editing*

In this mode you may freely edit the affine transform by dragging the three corner points of the triangle-controls. This way you can easily create a free combination of translation, rotation and shear.

3.1.2.6. *Edit view of the controls*

In this mode you edit the appearance of the controls. This doesn't change the flame; only how the

controls look. It can be used to see controls that are outside the window by moving them or scaling them down. It also controls how sensitive the controls are; scale them up for finer placement.

- using the mouse-wheel, or by dragging the mouse left or right with the right mouse-button pressed, you may scale them up/down.
- by dragging the mouse with the left mouse-button pressed, you move the controls

See "Turn grid on/off" above.

3.1.2.7. *Edit focus point*

Edit the focus-point (parameters FocusX, FocusY and FocusZ at the "DOF/Bokeh → DOF" - tab) using the mouse.

- by dragging the mouse with the left mouse-button pressed, you change FocusX- and FocusY-parameters
- using the mouse-wheel you change the FocusZ-parameter

3.1.2.8. *Affine XY edit plane*

Sets the current editing-plane for editing affine-transform to the x-y-plane.

The editing of affine-transforms takes place in one plane, the default setting is the x-y-plane. When editing "classic" 2d-fractals, the x-y-plane is the only plane. But, when editing 3d-fractals, you may choose different planes, in order to modify all three coordinates.

3.1.2.9. *Affine YZ edit plane*

Sets the current editing-plane for editing affine-transform to the y-z-plane.

3.1.2.10. *Affine ZX edit plane*

Sets the current editing-plane for editing affine-transform to the z-x-plane.

3.1.2.11. *AI Denoiser preview*

Apply the currently selected AI denoiser (OptiX or OIDN) and display it in the preview. This function uses a split preview, in order to allow you to compare the denoised and the original image. On the left side, you see the original image, on the right side of the split indicator you see the denoised image. This button only appears if an AI denoiser is selected in the Anti-Aliasing/Filter tab.

This function uses the current preview. So it is recommended to re-render the preview in higher quality before invoking it.

3.1.2.12. *External render preview*

Open an additional detached progressive preview; works especially well when you are using two monitors.

3.1.2.13. *Render image preview*

Re-renders the current preview with a much higher quality level.

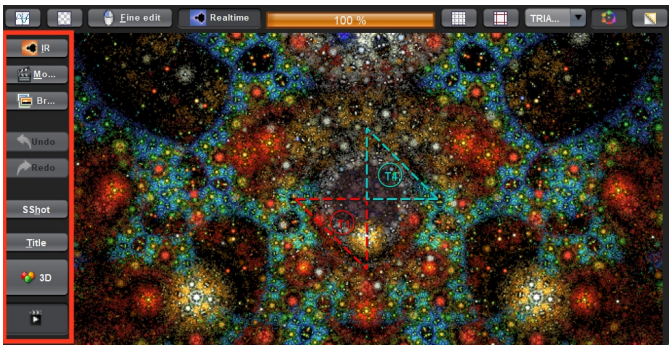
This can take some time. The progress bar on the top of the preview area shows the preview progress. If solid rendering is enabled, there will be a short delay after the progress bar reaches 100% before the result is displayed.

This is a very important function in the fractal-editing workflow, because certain parameter changes do not cause a complete re-render of the image, as this would be too slow. Instead, after such parameter changes, a raw in-memory version of the last preview to which the parameter change is applied is used.

This is especially helpful when progressive rendering is off (see the *Realtime* button), where the preview has very low quality.

Therefore, when fine-tuning colors, it is recommended to re-render the preview in higher quality using this button before changing the settings. Because this function is of such importance, is has been placed in a prominent place.

3.1.3. Quick buttons on the left of the preview area



In this area you find some quick-buttons to reach other modules of JWildfire and the Undo/Redo-buttons.

3.1.3.1. *Interactive renderer*

This is a shortcut for the Interactive Renderer. It opens the Interactive Renderer or brings it into the foreground. It serves no other function, and won't for example, load the current flame into the Interactive Renderer. This was intentional.

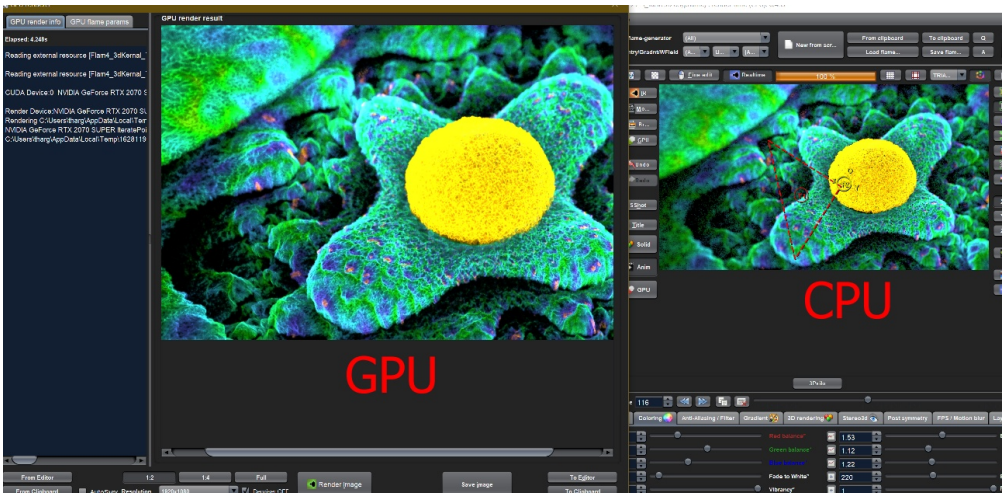
3.1.3.2. *Easy Movie Maker*

Opens the Easy Movie Maker. Automatically adds the currently loaded flame as the first keyframe inside the Easy Movie Maker.

3.1.3.3. *Flame Browser*

Opens the Flame Browser.

3.1.3.4. *GPU render*



Opens the GPU render module. Automatically imports the currently selected flame into the GPU render module and starts to render the flame.

3.1.3.5. *Undo*

Reverts the latest change to the currently selected flame. Each flame has its own undo-history. So undoing a change will not switch between flames.

Please note, that the undo history is not saved together with flames, so it is lost when you quit JWildfire.

3.1.3.6. *Redo*

Repeats the latest undone change to the currently selected flame.

3.1.3.7. *SShot (Snapshot)*

Create a snapshot of the current flame. This is an identical copy with new undo-history.

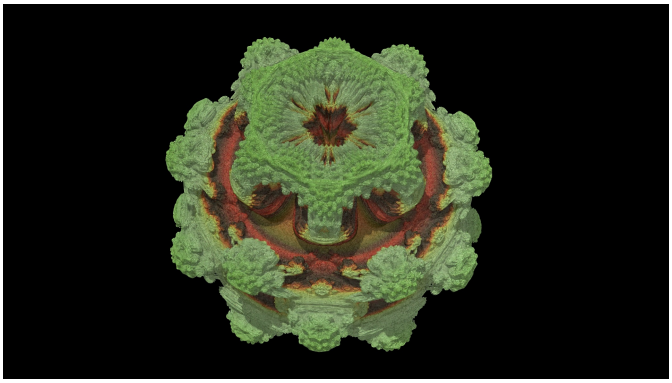
3.1.3.8. *Title*

Sets the title of the current flame. This is a good place to also put copyright information or to add your name (as the author of the artwork). You can set a default title for flames which are created using the *New from Scratch*-button, by setting the property **tinaDefaultNewFlameTitle** in the *Preferences*-window.

3.1.3.9. *Solid* **(formerly named "3D")**

Switch between "classic" rendering and solid rendering.

Solid rendering basically works by treating each sample of the fractal flame as a solid particle. The sum of all of these particles makes up a larger solid shape. This works only for certain types of flame fractals. It may fail if the structure is too thin or is in general too "chaotic" or too blurry. Solid rendering can be used to create really unusual and interesting things. Here is a render of the "classic" Mandelbulb (thanks to Whittaker Courtney for the idea):



3.1.3.10. *Animations controls on/off*

Using this button you may turn on or off animation controls. You may set up your preference for this setting in the *Preferences*-window.

3.1.3.11. *GPU-mode on/off*

This button is only available, when GPU-rendering is supported on your system.

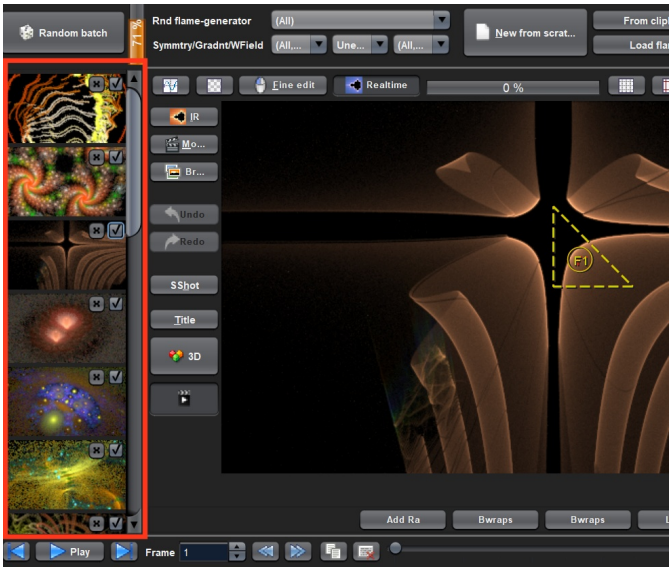
It is a switch to turn GPU-mode on or off.

GPU-mode helps you to integrate GPU-rendering into your artistic process. When activated, it changes the behaviour of JWildfire in the following ways: - random-flame-generators try to avoid features which are not supported by GPU-rendering (which uses heuristics and is currently limited) - the Render-function of the main-editor automatically uses GPU for rendering (without the need to transfer the flame to the separate GPU-renderer-window) - the Preview-Render-function (button at the lower right area of the prewiew area) also uses the GPU. - there is an implicit GPU-filter for the list of variations, so when creating new fractals you can select only variations which are supported on GPU

Please note, that the GPU-mode does not affect the real-time-preview, which is used for editing.

You may change the initial state of the GPU-mode by adjusting the property **tinaGpuModeDefaultEnabled** in the *Preferences*-window.

3.2. *Thumbnail ribbon*



The thumbnail ribbon holds thumbnails of all the flames you currently are editing. After starting the program, some random flames are generated and displayed in the thumbnail ribbon.

You can edit/view any flame of the thumbnail by just double-clicking (or right-clicking) on it.

If you want to return to a previous flame, just double-click the thumbnail of the other flame. Each flame in the thumbnail-ribbon has its own undo-history.

3.2.1. Selecting flames in the *thumbnail ribbon*

Each flame in the *thumbnail ribbon* has a little checkbox assigned in the upper right corner. By changing the value of this checkbox you change the selected state of the thumbnail.

Selection of individual flames is only important for saving and deleting.

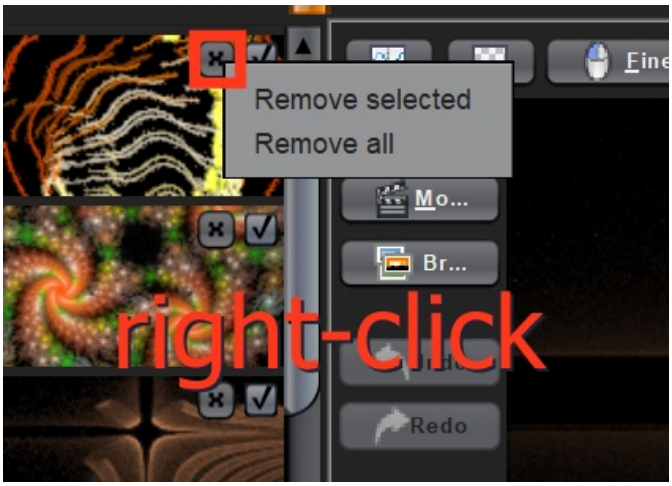
By pressing the right mouse-button on top of this checkbox you can open a pop-up menu to change the selection for all flames inside the *thumbnail ribbon*:

- **Toggle all**: invert the selection of all flames
- **Deselect all**: set all flames to deselected state

So, one way to select all flames is to deselect them all and then to invert the selection.

3.2.2. Deleting flames in the *thumbnail ribbon*

Each flame in the *thumbnail ribbon* has also a little delete-button assigned in the upper right corner. By pressing this button you may delete the corresponding flame from the *thumbnail ribbon*. Since this can not be undone, a popup will appear to confirm the action.



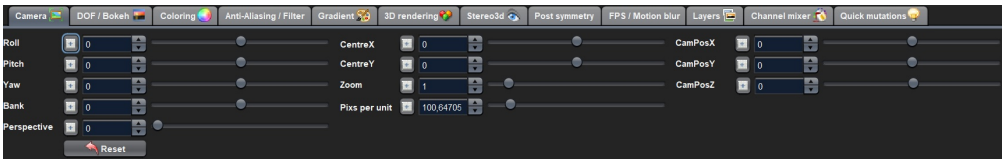
By pressing the right mouse-button on top of this button you can open a pop-up menu to delete a number of flames from the *thumbnail ribbon*:

- **Remove selected**: delete all flames which are selected (the corresponding checkbox is checked) from the *thumbnail ribbon*
- **Remove all**: empty the *thumbnail ribbon*

3.3. Flame settings

At this main tab, you may edit settings at the fractal-flame-level. Most of these settings will affect the final outcome in a global way, but do not affect the structure of the fractal itself. For example, you can change the view-settings or the final coloring, but not the shape itself or which details are computed.

3.3.1. *Camera* tab



With these controls you can control the camera-view. You can rotate and move the camera freely and zoom in/out.

"Classic" flame fractals are flat (2D); they have no depth. Basically the parameters Roll, CentreX, CentreY and Zoom/Pixels per unit apply to them. For 3D fractals there are a lot more controls.

But, you can "mix" them, of course. So you can apply all parameters to any fractal and also rotate a 2D flame in space or apply a perspective distortion to it.

3.3.1.1. *Roll*

Camera roll angle in degrees. Rotates around the virtual z-axis and also applies to 2D fractals.

3.3.1.2. *Pitch*

Flame pitch angle in degrees. Rotates the flame around its x-axis.

3.3.1.3. *Yaw*

Flame yaw angle in degrees. Rotates the flame around its z-axis.

3.3.1.4. *Bank*

Flame bank angle in degrees. Rotates the flame around its y-axis.

3.3.1.5. *Perspective*

Camera perspective. A value of 0 indicates no perspective, while higher values produce more dramatic perspective effects. Values can be negative for more interesting effects.

3.3.1.6. *CentreX*

Offset (translation) of the fractal x-direction.

3.3.1.7. *CentreY*

Offset (translation) of the fractal y-direction.

3.3.1.8. *Zoom*

Camera zoom.

3.3.1.9. *Pixels per unit*

Another (classic) measure of zoom. "Pixels per unit" means how many pixels are needed to fill a numerical distance of 1.0 (in fractal coordinates). The higher the value, the more details (like zooming in).

3.3.1.10. *CamPosX*

Position of the camera in x-direction. Has an effect similar to CentreX for 2D fractals, but a different effect for 3D-fractals.

3.3.1.11. *CamPosY*

Position of the camera in y-direction. Has an effect similar to CentreY for 2D fractals, but a different effect for 3D-fractals.

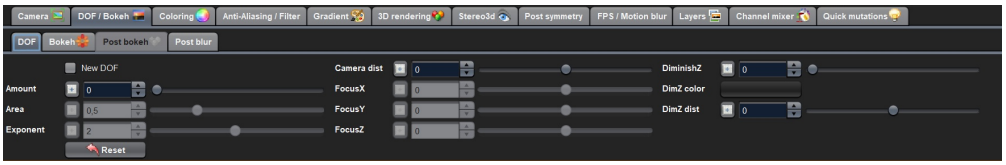
3.3.1.12. *CamPosZ*

Position of the camera in z-direction.

3.3.2. *DOF/Bokeh* tab

DOF (Depth Of Field) is the blurring of parts of the flame based on the distance from the camera. Whether to use it is a personal decision. Some people like it a lot (including myself), some people do not like it at all. So, this sub-chapter describes a rather "special" topic. Bokeh refers to the shape of out-of-focus points of light in a real camera. JWildfire can simulate some bokeh effects in conjunction with DOF.

3.3.2.1. *DOF* subtab



This tab contains some of the more basic settings regarding to DOF.

3.3.2.1.1. *New DOF*

Switches between classic and more sophisticated (new) DOF modes. Usually, the "new mode" is recommended. The classic mode is still there because of backwards compatibility.

3.3.2.1.2. *Amount*

Amount of DOF. A value of zero means no DOF at all.

3.3.2.1.3. *Area*

Measure of the sharp area around the focus point.

3.3.2.1.4. *Exponent*

Measure of the falloff of the sharp area around the focus point.

3.3.2.1.5. *Camera dist*

Only applies to the classic mode and is a measure of the distance from the camera to the focus point.

3.3.2.1.6. *FocusX*

X-position of the focus point (a point which is sharp). May also be edited interactively in the preview area (use the *Edit focus point* button).

3.3.2.1.7. *FocusY*

Y-position of the focus point (a point which is sharp). May also be edited interactively in the preview area (use the *Edit focus point* button).

3.3.2.1.8. *FocusZ*

Z-position of the focus point (a point which is sharp). May also be edited interactively in the

preview area (use the *Edit focus point* button).

3.3.2.2. *Bokeh* subtab

At this tab you can select options to create bokeh-like styles of DOF-effects. These options are not available when using solid rendering. For solid rendering there is an option to apply post-bokeh effects, but these are limited in comparison to the options you find at this tab.

3.3.2.2.1. *Shape*

Selects the shape of the bokeh-effect

- **Bubble**: the typical and default shape
- **Cannabiscurve**: some nice-looking organic shapes
- **Cloverleaf**
- **Flower**
- **Heart**: little hearts, makes nice Valentine-cards
- **NBlur**: has some interesting options to create very different shapes
- **Perlin Noise**: an unusual but interesting style
- **Rect**: very simple but effective style. You can also create "rain"-like scenes with this
- **SineBlur**: a popular circular bokeh
- **StarBlur**: tiny stars
- **Taurus**: another unusual but interesting shape
- **Snowflake**: a complex snowflake, created with the algorithm described in the paper "A local cellular model for snow crystal growth" by Clifford A. Reiter.
- **SubFlame**: This is the most versatile option: you may use any other fractal as bokeh-style.

3.3.2.2.2. *Scale*

Global scale of the bokeh-shapes.

3.3.2.2.3. *Rotate*

Global rotation-angle of the bokeh-shapes.

3.3.2.2.4. *Fade*

Global falloff of the bokeh-shapes.

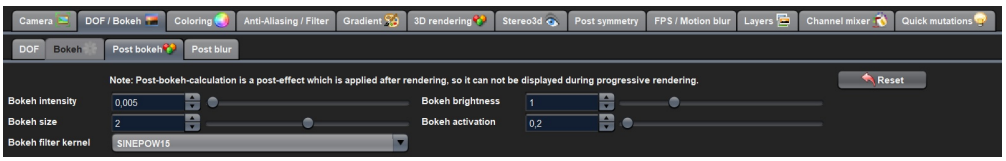
3.3.2.2.5. *Shape specific parameters*

Depending on the bokeh-shape, there are additional parameters. Most of them are the same as in the variation of the same name. For example, the "Taurus"-bokeh-style has a parameter N (which is the number of corners). The "taurus"-variation has the same parameter. Not all parameters of a variation are exposed to the bokeh-style of the same name.

3.3.2.2.6. Flame (for the Sub-Flame-bokeh-style)

This parameter selects the flame you want to use as bokeh-style. It is a number, starting at 1. Before you can use a flame as bokeh-style, you must save it to your default flame-folder by using a filename in the form **"_dof_XXXX.flame"**, where XXXX corresponds to the number you select here. For example, when you enter "23" as parameter flame, JWildfire will look for a flame-file with the filename **_dof_0023.flame**.

3.3.2.3. *Post bokeh* subtab



At this tab you can select options to create bokeh-like effects when using solid rendering. These options are only available in solid rendering and are limited to the bokeh-related options which are available for regular flames.

3.3.2.3.1. *Bokeh intensity*

Overall amount/intensity of bokeh-effects.

3.3.2.3.2. *Bokeh size*

Size of the bokeh-effects. The larger the effect, the more computation time is required.

3.3.2.3.3. *Bokeh filter kernel*

Type of bokeh-shape. The default setting produces "classic" flat circular shapes.

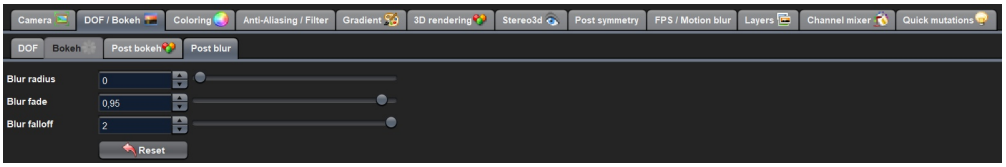
3.3.2.3.4. *Bokeh brightness*

Intensity of the bokeh in comparison the the original image.

3.3.2.3.5. *Bokeh activation*

Intensity level of the original image for placing bokeh "seeds". The larger the value, the more seeds and the more bokeh.

3.3.2.4. *Post blur* subtab



Post-blur is not technically a DOF-effect, but it creates a type of blur, therefore it is located at this tab. It is not a popular option and primarily resides here for backwards compatibility.

Post-blur works by smudging each rendered sample of the fractal flame. The effect diminishes with distance to center. It usually makes a fractal appear more bold and "glassy".

3.3.2.4.1. *Blur radius*

Radius of the blur.

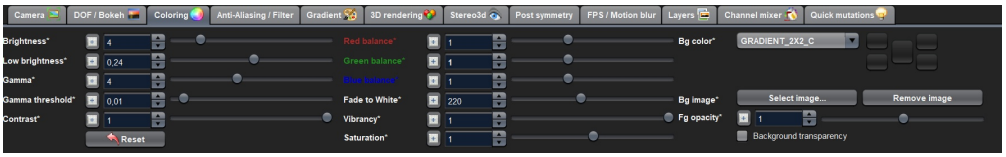
3.3.2.4.2. *Blur fade*

(Inverse) intensity of the blur. The maximum value means maximum fade out, or no blur effect at all.

3.3.2.4.3. *Blur falloff*

Strength of diminishing of the effect dependent of the distance to the center of the individual sample.

3.3.3. *Coloring* tab



At this tab you may edit numerous coloring/tonemapping options to turn your fractals into real artworks.

3.3.3.1. *Brightness*

Overall brightness level of the rendered image. For some fractals even very high values (above 20) may work nicely.

3.3.3.2. *Low brightness*

A tool to increase the brightness of areas with low sampling density. May help to reveal background details of the fractals which would otherwise be hidden.

3.3.3.3. *Gamma*

Gamma correction level. Small values (below 2.5) increase the brightness of already bright areas and darken already dark areas, causing a kind of glowing effect.

3.3.3.4. *Gamma threshold*

A density-threshold defining when to apply the gamma correction. In areas with a density below the threshold a linear transformation is applied, while at areas with higher density the actual gamma correction is applied. Playing with this value may also help to increase details of areas with low density.

3.3.3.5. *Contrast*

Overall contrast level of the fractal per default maximum contrast. In the most cases it should not be necessary to alter this value, because lowering the contrast usually leads to less appealing images.

3.3.3.6. *Red balance*

Overall red balance of the image. A convenient option for fine-tuning the final result without the need to change the gradient.

3.3.3.7. *Green balance*

Overall green balance of the image. A convenient option for fine-tuning the final result without the need to change the gradient.

3.3.3.8. *Blue balance*

Overall blue balance of the image. A convenient option for fine-tuning the final result without the need to change the gradient.

3.3.3.9. *Fade to White*

A measure for the saturation-level of final colors. It can be used to achieve glowing-effects (decrease the value) or to reduced glowing and make more details visible (increase the value). Works similar to the gamma-value.

3.3.3.10. *Vibrancy*

Overall vibrancy-level (measurement of the number of different color values). Per default set to the maximum value, which is very typical for fractal rendering.

3.3.3.11. *Saturation*

Saturation-level of the colors. A convenient option for fine-tuning the final result without the need to change the gradient.

3.3.3.12. *Bg color*

Background-color which is mixed with the fractal in the foreground to create a smooth looking final result. The options are:

- **Single color**: one single color for the whole background
- **Gradient 2x2**: create a background by blending 4 colors at the corners of a rectangle
- **Gradient 2x2_c**: create a background by blending 4 colors at the corners of a rectangle and one color at the center

3.3.3.13. *Bg image*

Background image which is mixed with the fractal in the foreground to create a smooth looking final result. Please note that a background image has higher priority than a background color. Therefore, if the background image and the background color are chosen, only the background image is rendered.

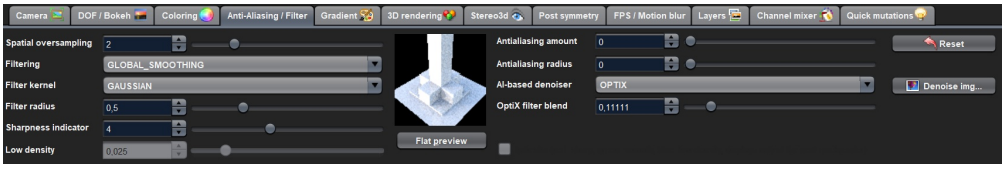
3.3.3.14. *Fg opacity*

Mixing-level of the foreground (fractal) when applying a background color or background image. Lower values make the foreground less transparent, but may increase antialiasing effects.

3.3.3.15. *Background transparency*

Turns rendering with transparent background (alpha channel) on or off. This option is usually used when you want to use your fractal images inside other software packages like Adobe Photoshop to compose a final artwork out of different images or parts. Please note that you may also activate a preview of the background-transparency in JWildfire (in the top area of the preview-window).

3.3.4. *Anti-Aliasing/Filter* tab



At this tab you find some very important settings which help to improve the visual quality or "cleanliness" of your fractal artworks. For details about filtering see the sub-section *Filter kernel preview/How filtering works*.

3.3.4.1. *Spatial oversampling*

Factor of spatial oversampling. Spatial oversampling works by internally rendering an image of a larger size and then creating a final of (usually) better quality by taking all this additional information into account.

- a value of 1 means no spatial oversampling
- a value of 2 means rendering internally an image of double size (double the width and double the height, so it requires about 4 times the memory)
- a value of 3 means rendering internally an image of triple size (triple the width and triple the height, so it requires about 9 times the memory)

The recommended value is 2.

Please note that, while higher levels of spatial oversampling require much more memory, they do not need much higher rendering times.

3.3.4.2. *Filtering*

Global filtering strategy for building the final image by taking into account the fractal iteration information:

- **Global sharpening**: creates an image with consistent high sharpness
- **Global smoothing**: creates an image with consistent high smoothness
- **Adaptive**: tries to sharpen details, while smoothing "unsharp" areas as well as areas with low density

3.3.4.3. *Filter kernel*

Select the filter kernel you wish to apply. There are different choices, depending on the selected *Filtering* option.

The following values are recommended (but feel free to play with other settings as well):

- **Global sharpening**: **Mitchell-Filter**
- **Global smoothing**: **Sinepow10**
- **Adaptive**: **Mitchel_Sinepow** (**Mitchel**-Filter for sharpening details, **Sinepow10**-Filter for smoothing areas of low density)

3.3.4.4. *Filter kernel preview/How filtering works*

Creating a fractal image is an elaborate process consisting of several steps. Filtering is one of these steps and very important for the final outcome. It takes place after the iteration-process which creates a lot of individual "measure points" of the fractal, called samples.

One pixel of the final image is usually calculated considering several raw pixels that are in close proximity to the source pixel. Here, a kind of averaging of all pixel values around the middle pixel takes place, which ultimately leads to an improved quality of the final image (than if only one raw pixel per final pixel was considered).

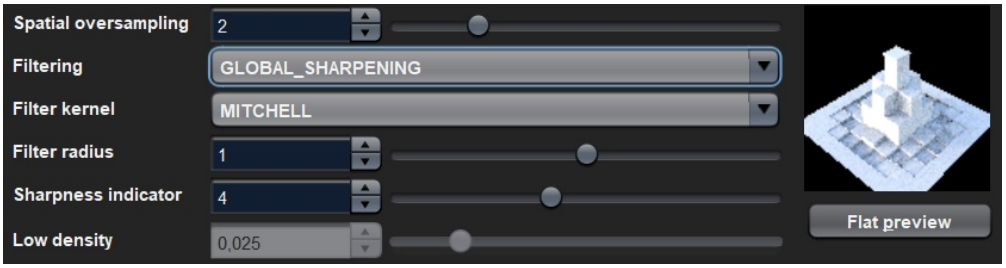
The filter-kernel-preview in the middle of the *Anti-Aliasing/Filter* - tab shows a visual representation of this behavior. Higher bars correspond to a higher influence of a sample, while lower bars correspond to a lower influence. Usually, the highest influence is in the center, which makes sense because the sample at the position of the final pixel usually should have an important influence.

3.3.4.4.1. *Example: Gaussian-filter*

The Gaussian-filter is a very popular filter for smoothing. You can view this behavior in the preview:

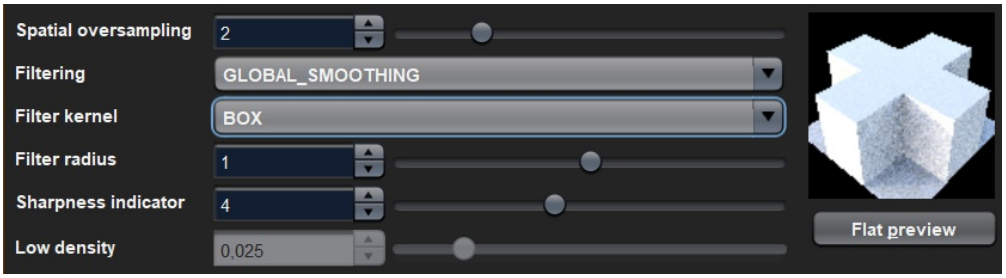
- the middle sample has the largest influence.
- samples which have a certain distance to the center, have a smaller influence.
- this influence is smaller the more distant the sample is.

3.3.4.4.2. *Example: Mitchell-filter*



The Mitchell-filter is a very popular filter for sharpening. In comparison to a smoothing filter, it has small areas with negative contributions, which cause the sharpening effect.

3.3.4.4.3. *Example: Box-filter*



The box-filter is a special filter because it creates averages from a number of samples. You can see this clearly in the preview, having only bars of the same size, i.e. each sample has the same contribution.

3.3.4.5. *Filter radius*

A numerical value indicating the size of the effective filter kernel around each sample. A value of 0 means no filter kernel at all, higher values lead to a growing of the filter kernel, you can see this very well in the filter-kernel-preview.

Useful values are in the range of 0.5 to 1.5, depending on the filter kernel and amount of spatial oversampling.

3.3.4.6. *Sharpness indicator*

This value helps the Adaptive filter to decide which portions of the image should be sharpened and which should be smoothed.

The higher the value, the more areas are treated as smooth areas. The lower the value, the more areas are sharpened.

This is a very abstract value, but there is a preview-mode which helps to visualize the different areas. This feature is described in the sub-section *Adaptive filter preview* .

3.3.4.7. *Low density*

This value helps the Adaptive filter to decide which portions of the image should be treated as areas with low density, and which areas should be smoothed regardless of density.

Usually, smoothing in areas of low density is more intense than in regular smoothing areas, therefore the distinction.

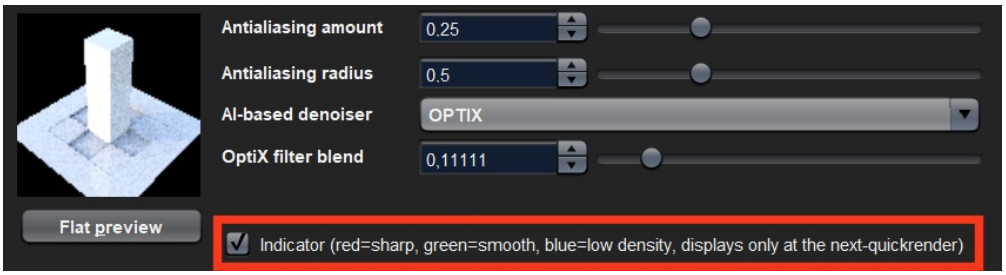
The higher the value, the more areas are treated as low density areas. The lower the value, the more areas are treated a general smoothing areas.

This is a very abstract value, but there is a preview-mode which helps to visualize the different areas. This feature is described in the sub-section *Adaptive filter preview*.

3.3.4.8. *Flat preview*

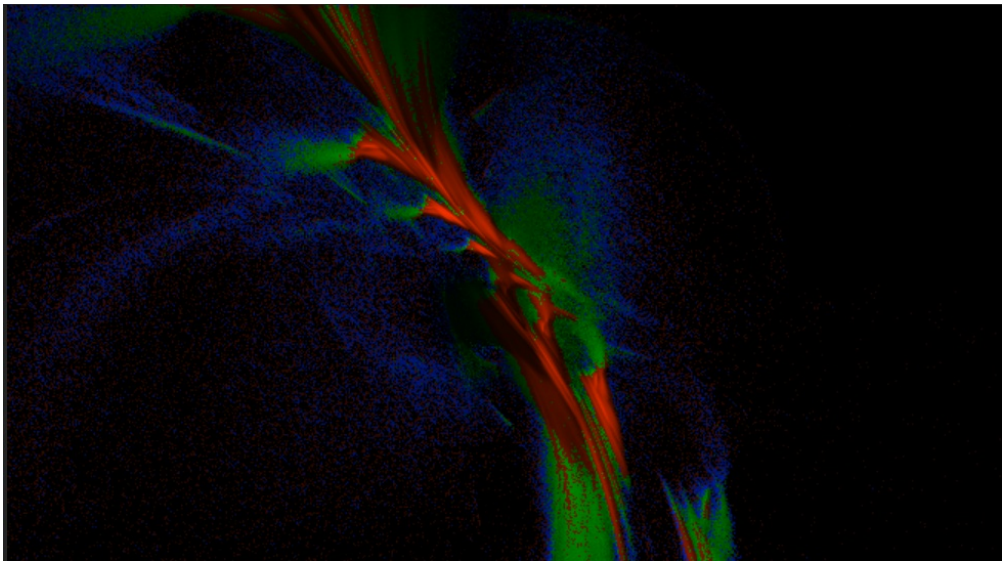
Displays a flat preview of the filter-kernel. In this type of view, negative values (which are typical for sharpening filters) are colored red.

3.3.4.9. *Adaptive filter preview*



By enabling the small checkbox labelled "Indicator" and re-rendering the preview (click the Render image preview button on the right of the preview area), you get a visualization of the different areas detected by the *Adaptive filter* .

- **red**: sharp areas
- **green**: smooth areas
- **blue**: areas with low density (which are also smoothed)



3.3.4.10. *Antialiasing amount*

Antialiasing is a technique to defend hard (or "jagged") lines or isolated pixels which seem too important. It works by adding a little "chaos" to the individual samples. The *Antialiasing amount* is a measure of this "chaos". The higher the value, the more "chaos" is applied, causing less hard lines, but possibly also causing fewer details. Set to 0 to disable antialiasing. See also *Antialiasing radius*

for another parameter to influence antialiasing.

3.3.4.11. *Antialiasing radius*

Radius of the added "chaos" to avoid jagged lines. See *Antialiasing amount* for a more detailed description.

3.3.4.12. *AI-based denoiser*

Type of AI-based denoiser applied to the final image:filtering_show_indicator.jpg

- **OptiX**: a very popular denoiser by NVIDIA, requires a modern graphics card from NVIDIA, see https://developer.nvidia.com/optix-denoiser
- **OIDN**: an open-source-denoiser by Intel, comes with fewer requirements than OptiX, see https://www.openimagedenoise.org/
- **None**: no post-denoiser

While not mandatory for rendering single images, these AI-based denoisers are extremely helpful when it comes to rendering animations. By using an AI-based denoiser you can cut down rendering times significantly, by factors of 10 or more.

This works by choosing a very low render quality in JWildfire (for example, lower than normal by a factor 10) and then "completing" those usually noisy images by using the AI-based denoiser. It is amazing how well this works. When you then finally assemble a compressed video sequence, you usually can see no difference when compared to rendering all the frames in high quality.

Please note, that AI-based denoisers are not available on all platforms. Therefore, JWildfire performs some self-test at startup and presents only the available options.

You can preview this feature using the *AI Denoiser preview* button in the preview area.

3.3.4.13. *OptiX filter blend*

This option is only available when choosing the OptiX denoiser. See *AI-based denoiser* for details. It defines a blend between the original (noisy) image and the denoised image.

- a value of 0 means that the original image is not blended with the denoised image at all, showing only the denoised image.
- a value of 0.5 means the original image is an average of the original and denoised images.

Usually, small values in the range between 0 and 0.2 are recommended.

You can preview this feature using the *AI Denoiser preview* button in the preview area.

3.3.4.14. *Denoise image*

Using this button you can denoise an external image using the currently selected AI-based denoiser. When using the OptiX-denoiser, the parameter *OptiX filter blend* is taken into account.

This feature is useful when you want to play with different denoising-settings or if you forgot to

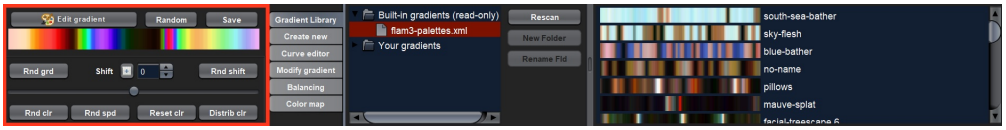
enable the denoiser or just want to denoise some non-Jwildfire-images.

Please note: It is recommended to use PNG as the file-format, other formats may work as well, but are not guaranteed to work. Eventually, you will need to convert the image before denoising it.

3.3.5. *Gradient* tab

At the gradient tab you will find numerous options to create and modify the gradients of your fractals. Note: gradients are sometimes also referred as "palette" or "color palette".

3.3.5.1. *General controls* subtab



The gradient-tab contains both some general controls and some sub-tabs. The former are described in this sub-chapter.

3.3.5.1.1. *Edit gradient* button

Opens/closes the classic gradient-editor of JWildfire. It is outdated and not described here. This gradient editor was superseeded by the curve-editor you find at the *Curve editor* subtab.

3.3.5.1.2. *Random* button

Shortcut for creating a new random gradient (using the current settings at the *Create new* subtab). Gradient changes are also part of the undo-history of the fractal flame, so you may switch through the randomly generated gradients by undoing/redoing afterwards.

3.3.5.1.3. *Save* button

Save the current gradient to the gradient-library (as *.map-file). See the sub-chapter *Gradient library* for futher details.

3.3.5.1.4. *Rnd grd* button

Select a random gradient from your gradients-library. Per default, the built-in library is selected. If you want to choose a random gradient from your own gradient-library or a sub-folder inside it, you must select the appropriate folder.

3.3.5.1.5. *Shift*

Modifies the gradient-shift, which moves all colors by the given amount. This is a non-destructive operation. Colors, which are "moving out" at one end of the gradient are "moving in" on the other side.

3.3.5.1.6. *Rnd shift* button

Applies a randomly chosen *Shift* value.

3.3.5.1.7. *Rnd clr* button

Sets the *Color*-parameters of all transformations of the fractal to a random value.

This does not change the gradient, but does change how it is applied to generate the final coloring of the fractal.

3.3.5.1.8. *Rnd spd* button

Sets the *Speed*-parameter of all transformations of the fractal to a random value.

This does not change the gradient, but does change how it is applied to generate the final coloring of the fractal.

3.3.5.1.9. *Reset clr* button

Sets both the *Color*- and the *Speed*-parameter of all transformations of the fractal to zero.

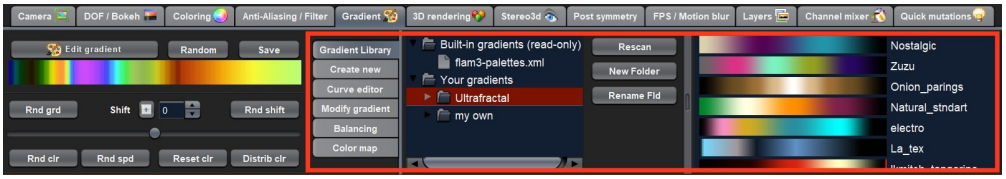
This does not change the gradient, but does change how it is applied to generate the final coloring of the fractal.

3.3.5.1.10. *Distribute clr* button

Distributes the *Color*-parameter of all transformations of the fractal evenly.

This does not change the gradient, but does change how it is applied to generate the final coloring of the fractal.

3.3.5.2. *Gradient library* subtab

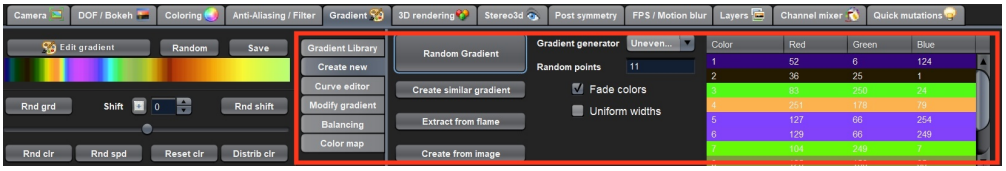


At this tab you can select pre-saved gradients and apply them to your fractal by just clicking on them. The gradient-library contains of two parts:

- in-built gradients: those are installed together with the program and can not be modified in any way
- your own gradients. They are located in a folder outside the JWildfire-installation-drawer, so that they are not lost when you update JWildfire. You must specify this path in the *Preferences* by setting the property **tinaGradientPath**.

User-defined gradients may be structured in sub-folders in order to make them more easily accessible.

3.3.5.3. *Create new* subtab



At this tab you can create gradients using various options. The most common use is to create random-gradients by selecting one of the sophisticated built-in random-gradient-generators.

3.3.5.3.1. *Random gradient* button

Create a new random gradient using the currently selected options, which are described in the following sub-sections. Each new gradient is a separate entry in the undo-history of the currently selected flame fractal, so you can cycle through all the generated gradients by using the *Undo* and *Redo* button.

3.3.5.3.2. *Gradient generator*

You may choose one of the following gradient generators:

- **(All)**: meta-random-gradient-generator which randomly chooses a random-gradient-generator to create the gradient
- **Strong hue**: creates a smooth gradient with strong colors
- **Stripes**: creates a gradient with many regular color changes giving a "stripy" look
- **Monochrome**: creates a smooth and monochrome gradient
- **Smooth**: creates a smooth gradient with primarily pastel colors
- **Bold**: creates a non-smooth gradient with large blocks of strong colors
- **Two colors**: creates a smooth gradient by fading two colors into each other
- **Uniform curves**: creates a random color curve with evenly distributed base colors and then derives the gradient from it. When using this generator, you have both a gradient and matching color curves.
- **Uneven curves**: creates a random color curve with unevenly distributed base colors. When using this generator, you have both a gradient and matching color curves. This setting usually creates the most interesting and versatile gradient together with the corresponding color curves and is the default.

3.3.5.3.3. *Random points*

Number of random base colors to derive a gradient from. The more base colors, the higher the number of colors in the final gradient.

Please note, that this setting is not respected by all random-gradient-generators (e. g. *Two colors* will always only use two colors).

3.3.5.3.4. *Fade colors*

Fade the base colors in order to create a smooth gradient, which is the default setting.

Please note, that this setting is not respected by all random-gradient-generators (e. g. *Bold* will

never fade colors).

3.3.5.3.5. *Uniform widths*

Distributes the base colors over the size of gradient in a uniform way.

3.3.5.3.6. *Base colors* **table**

In this table you can manually edit the base colors after a gradient is created.

Please note that these base colors are not saved, so you can only edit them at the time you create a gradient and not after saving and loading a flame.

3.3.5.3.7. *Create similar gradient* **button**

Creates another random gradient which has similar colors to the current gradient, but has a different distribution over the area of gradient.

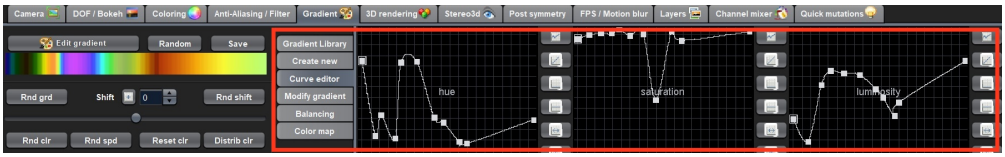
3.3.5.3.8. *Extract from flame* **button**

Load a gradient from a flame-file.

3.3.5.3.9. *Create from image* **button**

Imports an image and creates a gradient from the most frequent colors found in that image.

3.3.5.4. *Curve editor* **subtab**



Color curves are an alternative way to edit gradients. Using color curves usually gives much more control over the final result because it allows very interactive editing of both the gradient and the flame at the same time. Additionally, it allows more direct control over modifying the colors of a gradient, making it much easier to create "dramatic" effects.

Internally, each gradient consist both of the "classic" part of the gradient and three curves: **hue**, **saturation** and **luminosity**. Those curves belong to the flames, and are also stored within the flame-files.

When loading an older flame-file without these curves, or assigning a "classic" gradient to a flame, the color curves are approximated. Usually, this is only a very raw approximation, but this is absolutely intended. To reach a very accurate approximation, it would require many of curve-points, which would make the curve very hard to edit. Therefore, the goal of the raw approximation is to make the actual editing easier and not to have an exact synchronisation between gradient and color curves. Color curves are meant as a powerful additional tool which are useful in very many cases, but might not be useful in every case. For example, a gradient consisting of many small bands of color cannot be well described by a color curve.

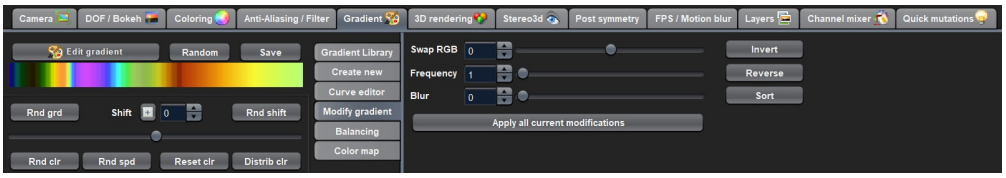
When you start to edit one of the curves, the gradient is synchronized to the shape of the curves.

When you save and re-load a flame, you can continue the editing of the curves.

Each curve has a toolbar with little buttons to help with editing:

- open a modal dialog for editing an enlarged version of a particular curve
- create ramp-shaped curve
- create line-shaped curve at the bottom (constant zero intensity)
- create line-shaped curve at the middle (constant half intensity)
- mirror the curve at the vertical axis
- mirror the curve at the horizontal axis (for example, turn a constant line of zero intensity into a line with maximum intensity)

3.3.5.5. *Modify gradient* **subtab**



At this tab you can modify the gradient globally. Please note that this does not take into account color curves, so any color curves will be out of sync.

3.3.5.6. *Swap RGB*

Swaps the color channels which changes the finally colors dramatically. Different values swap different channels.

3.3.5.7. *Frequency*

Repeats the gradient inside itself. The higher the frequency the more copies are made. Because the copies must fit in the place of one gradient, they are scaled down.

3.3.5.8. *Blur*

Blur smudges the colors of the gradient making it more smooth or "unsharp".

3.3.5.9. *Invert*

Turns the colors of the gradient into negative colors.

3.3.5.10. *Reverse*

Mirrors the positions of the colors (last color will be first and vice versa).

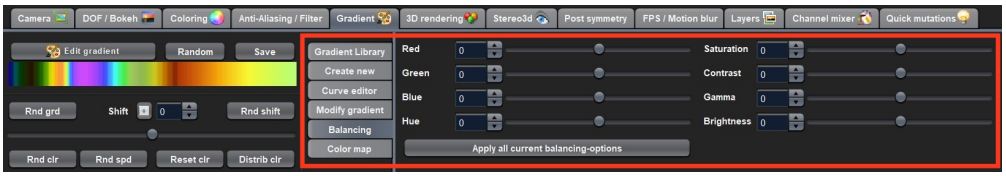
3.3.5.11. *Sort*

Sorts the colors by hue and brightness.

3.3.5.12. *Apply all current modifications* button

All of the above options are "non-destructive", which means they do not actually modify the gradient, but are applied dynamically. By using the *Apply all current modifications* button, you can finalize those changes by actually applying them to the gradient.

3.3.5.13. *Balancing* subtab



At this tab you can modify the colors of the gradient by applying typical color balancing operations. Please note that this does not take into account color-curves, so any color-curves will be out of sync.

3.3.5.14. *Red*

Change the red color balance.

3.3.5.15. *Green*

Change the green color balance.

3.3.5.16. *Blue*

Change the blue color balance.

3.3.5.17. *Hue*

Increase or decrease the hue of the colors of the gradient.

3.3.5.18. *Saturation*

Increase or decrease the saturation of the colors of the gradient.

3.3.5.19. *Contrast*

Increase or decrease the contrast of the colors of the gradient.

3.3.5.20. *Gamma*

Apply an overall gamma correction.

3.3.5.21. *Brightness*

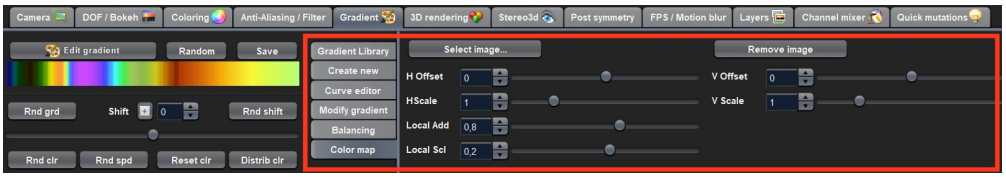
Change the overall brightness.

3.3.5.22. *Apply all current balancing-options* button

All of the above options are "non-destructive", which means they do not actually modify the gradient, but are applied dynamically. By using the *Apply all current balancing-options* button, you

can finalize these changes by actually applying them to the gradient.

3.3.5.23. *Color map* subtab



This tab has a new feature which lets you use an image as a gradient. It works well only for flat fractals. When coloring the fractal it takes into account both the position at the (x,y)-plane and the local color information from the iteration process.

3.3.5.24. *Select image*

Select an external image.

3.3.5.25. *H Offset*

Horizontal offset of the image map.

3.3.5.26. *H Scale*

Horizontal sizing of the image map.

3.3.5.27. *Local Add*

Measure of influence of local color information at the current position.

3.3.5.28. *Local Scl*

Measure of another type of influence of local color information at the current position.

3.3.5.29. *V Offset*

Vertical offset of the image map.

3.3.5.30. *V Scale*

Vertical sizing of the image map.

3.3.5.31. *Remove image*

Remove the selected color map (and revert to using a regular gradient for coloring).

3.3.6. *3D rendering* tab

At this tab you can influence the outcome of a fractal rendered in solid mode. This is also an experimental feature of JWildfire.

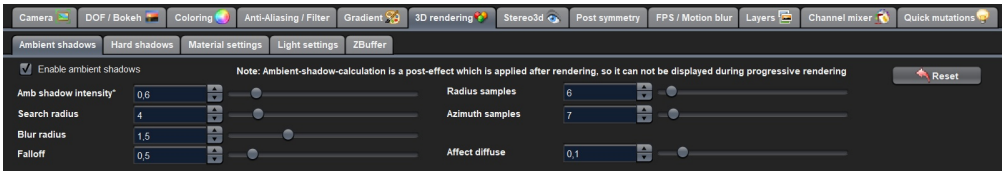
Solid rendering basically works by treating each sample of the fractal flame as a solid particle. The

sum of all of these particles makes up a larger solid shape. This feature works well for only certain types of flame fractals and needs more rendering time than "classic" fractal flames.

Solid flames usually don’t work well if the structure is too thin or is in general too "chaotic" or too blurry.

Solid rendering can be used to create really unusual and interesting things.

3.3.6.1. *Ambient shadows* subtab



Ambient shadow calculation is an effective method for approximating more complex physical processes that cause the image to be darker in small corner ends or crevices.

There are several methods to implements this, JWildfire uses a method based on "Image-Space Horizon-Based Ambient Occlusion", which was developed by NVIDIA.

Ambient shadows are a global effect which is independent from any light sources.

3.3.6.1.1. *Enable ambient shadows* checkbox

Enable or disable Ambient shadow calculation (also known as ambient occlusion).

Please note that ambient shadows are applied after the main iteration process, so it will not be displayed in realtime during progressive rendering. Click the Render image preview button to preview the result.

3.3.6.1.2. *Amb shadow intensity*

Intensity of the ambient shadows. The higher the value the higher the darker the shadows in small corners. If changed after a preview render, the results are seen immediately in the preview. (This is not true of the other ambient shadow settings.)

3.3.6.1.3. *Search radius*

Distance to search for ambient occlusions. Affects both the quality and the overall appearance of the ambient shadows. Optimum values depends on the individual fractal. The default value should work not too badly in most cases, though.

3.3.6.1.4. *Blur radius*

Blur amount of ambient shadows. A value of 0 means no blur at all, which would look very unnatural.

3.3.6.1.5. *Falloff*

Falling off of the shadows. Higher values lead to higher falling off, making the shadows smaller.

3.3.6.1.6. *Radius samples*

Number of samples in radius-search-direction. Higher values increase both quality of the shadows and rendering time.

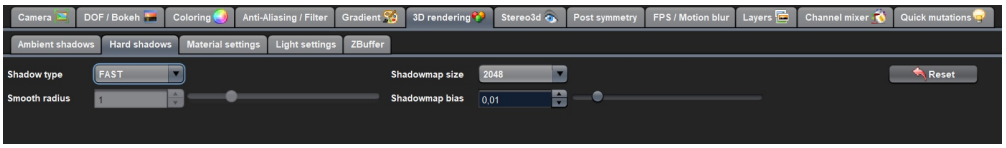
3.3.6.1.7. *Azimuth samples*

Number of samples in azimuth-search-direction. Higher values increase both quality of the shadows and rendering time.

3.3.6.1.8. *Affect diffuse*

An experimental option that allows ambient shadows to influence diffuse lighting. This is not physically correct, but can help to generate more dramatic effects.

3.3.6.2. *Hard shadows* subtab



At this tab you can configure classic hard shadows which are generated with the help of shadow maps.

Hard and ambient shadows can be used at the same time and work well together.

3.3.6.2.1. *Shadow type*

These are the options:

- **Off**: don't calculate hard shadows. This is the default setting.
- **Fast**: classic hard shadows
- **Smooth**: a modification of hard shadows that smooths the edges. Produces artifacts sometimes, in which case you should use **Fast** shadows.

3.3.6.2.2. *Smooth radius*

Radius for smoothing hard shadows.

3.3.6.2.3. *Shadowmap size*

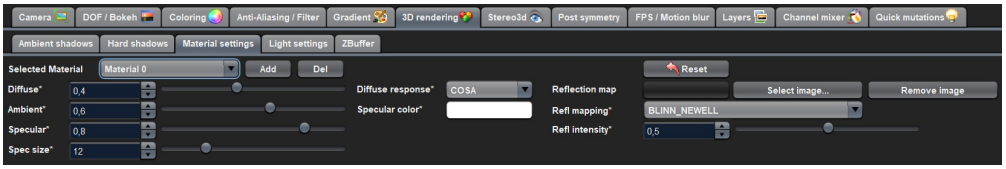
Resolution of the shadow map. The higher the resolution, the higher the quality of the shadows, but the higher the memory consumption.

Please note that a shadow map is required for each light source. So, having many light sources may increase memory consumption indirectly when using shadow maps.

3.3.6.2.4. *Shadowmap bias*

Sometimes shadow maps produce little stripy artifacts. By setting a small bias value those artifacts can be bypassed. The bias value is not necessarily high when there are a large number of artifacts.

3.3.6.3. *Material settings* subtab



Here you can set up the materials of your fractal. Fractals can have any number of materials, which are selected by a material-index similar to colors by the color-index.

3.3.6.3.1. *Selected Material* drop-down list

Here you select the material you want to edit.

3.3.6.3.2. *Add* button

Create a new material.

3.3.6.3.3. *Del* button

Delete the currently selected material. Use with caution as this will renumber the remaining materials but will not reference them.

3.3.6.3.4. *Diffuse*

Amount of diffuse reflection, which is light reflected from a rough surface. The color of this component is determined by the colors of the light sources.

3.3.6.3.5. *Ambient*

Amount of ambient light. The color of this component is determined by the fractal color.

3.3.6.3.6. *Specular*

Amount of specular light, which is light reflected from a smooth surface. The color of this component is determined by the *specular color*.

3.3.6.3.7. *Spec size*

Size of the specular reflections. Higher values produce smaller and more intense reflections.

3.3.6.3.8. *Diffuse response*

Controls how the material reflects diffuse light.

3.3.6.3.9. *Specular color*

Color of specular reflections.

3.3.6.3.10. *Reflection map*

Allows mapping of an image as an environment which is then reflected on the object's surface.

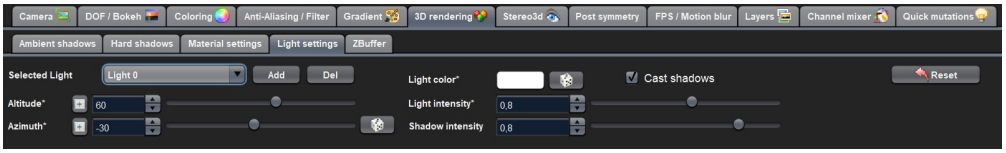
3.3.6.3.11. *Refl mapping*

The function used to map the image to the environment.

3.3.6.3.12. *Refl intensity*

Amount of light emitted by the reflection map.

3.3.6.4. *Light settings* subtab



At this subtab you may set up directional light sources for solid rendering.

You may have as many light sources as you want.

3.3.6.4.1. *Selected Light* drop-down list

Select the light source you want to edit.

3.3.6.4.2. *Add* button

Add a light source.

3.3.6.4.3. *Del* button

Remove the currently selected light source.

3.3.6.4.4. *Altitude*

Changes the light direction by altering the angle between the light and the z-axis.

3.3.6.4.5. *Azimuth*

Changes the light direction by altering the angle between the light and the yz-plane.

3.3.6.4.6. *Light color*

Color of the light, used for calculating the diffuse component of reflection.

3.3.6.4.7. *Cast shadows*

Choose if this light should cast shadows or not. Affects only hard shadows.

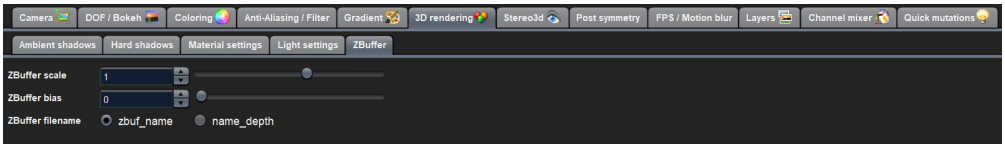
3.3.6.4.8. *Light intensity*

Intensity of the light source.

3.3.6.4.9. *Shadow intensity*

Intensity of the shadow cast by this light source.

3.3.6.5. *ZBuffer* subtab



Both, when rendering in solid and "classic" mode, you may also export an additional z-buffer-image. With these images you may achieve the popular "3d-photo"-effect you may find for example at Facebook. The settings on this tab helps to customize the creation of this z-buffer.

Important note: even when many fractal flames produce images which have some "pseudo-3d-look", many of these fractals are flat or have an actual different shape. Few of them may not even produce a valid z-buffer at all (they may be very noisy or may look totally different than one would suggest wehen looking at the regular render). This is due to the "ghostly" nature of fractal flames, and currently there are no tools in JWildfire to counteract these effect, but I'm working on it.

Hint: To export a z-buffer, you must activate the option *with Z-Buffer* in the *Quality*-profile you are using. See the sub-section about *Quality Profiles* for more details.

3.3.6.5.1. *ZBuffer scale*

Affects the amplitude of the z-shift/depth-effect caused by the z-buffer. The higher the value, the more dramatic the depth-effect caused by the z-buffer. If positive, close areas are light and far areas are dark. If negative, close areas are dark and far areas are light.

3.3.6.5.2. *ZBuffer offset*

Adjusts the center-position of the z-buffer in z-direction

3.3.6.5.3. *ZBuffer bias*

Moves background area of the z-buffer into z-direction. With a higher value the saturation is reached more quickly.

3.3.6.5.4. *ZBuffer filename*

There are two options to control how the filename of the z-buffer-image is determined:

- **zbuf_name**: use image filename prefixed with "zbuf_"
- **name_depth**: append "_depth" to the image filename

3.3.6.6. *Suggest params*

Renders an image using the currently selected render-settings to suggest settings for z-buffer which uses the available color-space in an optimal way. Does not work well for very noisy fractals.

Note: This option currently does not work well for GPU-rendering. This is due to the difference in implementation of CPU- and GPU-rendering, which also leads to a different implementation of rendering of the z-buffer. For GPU-rendering, it is recommended to use the *Render preview*-button at the *ZBuffer* subtab to adjust the z-buffer-settings.

3.3.6.7. *Render preview*

Renders a preview of the z-buffer using the currently selected z-buffer-parameters. Important notice: It is recommended to use a real render to evaluate the z-buffer-settings before rendering a longer sequence of images (with z-buffer), because the creation of the z-buffer may also depend on resolution and render quality.

3.3.7. *Stereo3D* tab

At this tab you can change options for rendering Stereo3D images. There are many options here, including settings to generate classic anaglyph images as well as options to generate interpolated images.

3.3.7.1. Stereo3d mode

Sets the type of Stereo3D mode:

- **None**: no Stereo3D rendering.
- **Anaglyph**: the classic way to generate a Stereo3D image: creates one image for each eye, turns them into gray, turns the gray into two different monochrome tones, and then merges those two images together. Creates one image per frame. Requires special 3d glasses with the matching colors to view the image.
- **Side by Side**: creates one image for each eye and puts them side by side. Creates one image per frame.
- **Interpolated images**: creates a number (N) of interpolated images for each eye. Creates 2 X N images per frame.

3.3.7.2. View angle

Angle between the eye-lines, also called Stereo angle.

3.3.7.3. Eye distance

Distance between the eyes.

3.3.7.4. Focal offset

Offset off the camera in z-direction.

3.3.7.5. Preview mode

Type of preview for Stereo3D-effect:

- **None**: no Stereo3D-preview. The fractal flame is displayed like a regular fractal flame.
- **Anaglyph**: preview as anaglyph image.
- **Side by Side**: preview in side-by-side mode.
- **Side by Side Full**: preview in enlarged side-by-side mode.

Please note, that the preview currently does not work correctly in Realtime mode. When creating Stereo3d-images, it is recommended to turn Realtime mode off

3.3.7.6. Left eye color

Color of the left-eye-image in anaglyph mode. These are the options:

- Red
- Cyan

3.3.7.7. Right eye color

Color of the left-eye-image in anaglyph mode. Same options as for the left eye.

3.3.7.8. Images per eye

Number of interpolated images which are created for each eye in *Interpolated Images* mode.

3.3.7.9. Swap sides

When checked, swaps the image for the left eye with the image for the right eye. Side-by-side stereo images are normally designed for parallel viewing. Checking this box creates stereo images for cross-eyed viewing. For some people is easier to see the 3D-effect this way, while for others it is easier the other way.

3.3.8. *Post symmetry* tab

The options at the *Post symmetry* tab are a very easy way to add symmetry to your fractal flames. You could also achieve this by adding certain transformation to your fractal, but this would require much more editing effort. Because symmetry is a very commonly used feature in fractal imagery, JWildfire offers this additional way to easily play with it.

3.3.9. Symmetry type

These are the symmetry types:

- **None**: no symmetry
- **X-axis**: mirror-symmetry along the x-axis (the mirror is the y-axis). Often used to create angel-like fractals.
- **Y-axis**: mirror-symmetry along the y-axis (the mirror is the x-axis) Often used to create portal-like fractals.
- **Point**: point-symmetry. Often used to create Mandala-like fractals.

3.3.10. Distance (only for X-axis- and Y-axis-symmetry)

Measure of the distance between the object and the mirror.

3.3.11. Rotation (only for X-axis- and Y-axis-symmetry)

Rotation angle of the mirror.

3.3.12. Symmetry order (only for Point-symmetry)

Order of the symmetry. Can be thought of as the number of copies which are placed around the symmetry center.

3.3.13. CentreX

Offset in the x-direction to place the symmetry-center.

3.3.14. CentreY

Offset in the y-direction to place the symmetry-center.

3.3.15. *FPS/Motion blur* tab

At this tab you can adjust options for motion-blur as well as the Frames Per Second setting for animations. While motion-blur is typical for animations (hence the name), it can also be used to create interesting effects for still images.

3.3.15.1. Blur length

The number of time steps needed for each frame. The more time steps, the smoother the effect but calculation time increases.

Recommended values are in the range of 16 to 48.

A value of 0 means no motion blur.

Please note that the calculation of motion blur may significantly reduce the realtime-animation-playback-speed. If you notice performance problems when previewing an animation, you should turn motion blur off temporarily.

3.3.15.2. Time step

The time difference for each computed blur step. Larger time steps produce a larger blur length, but also produces visual gaps between the individual steps.

3.3.15.3. Decay

Fall-off of the brightness of the generated time-steps. The visual difference increases with each time-step, which produces "trails" with diminishing intensity at the ends.

3.3.15.4. FPS

Number of generated frames per second. This setting is a general setting for animations and is not tied to motion-blur.

3.3.16. *Layers* tab

Layers allow you to create complex flames by merging multiple fractals. Unlike using traditional layers in image processing software, the layers in JWildfire are still fractals which are "alive" and can be modified as you would regular flames. Therefore, you can edit each layer before rendering the final image. Additionally, layers respect that flames have three coordinates, so each layer can be moved freely in space before rendering the image.

Flames are by nature translucent, so are always superimposed over each other. There are no "blend modes" and the order of layers doesn't matter.

Each layer has its own:

- set of transforms
- set of final transforms
- gradient

All other settings (camera, coloring, post symmetry, etc.) affect the whole flame, i.e. all layers as a whole.

3.3.16.1. Basic manipulations

There are basic functions for manipulating layers:

- **Add**: to create a new blank layer with a new randomly created gradient.
- **Duplicate**: to create a new layer and copy all transforms, final transforms and the gradient from the currently selected layer.
- **Delete**: to delete the currently selected layer. Please note, that the last layer cannot be deleted.
- **Extract**: will create a new flame in the thumbnail using only the layer that is currently selected.

3.3.16.2. Layer properties

Layers have the following properties which you can modify:

- **Caption**: a free text you can use to describe the layer
- **Visible**: this property controls if the currently selected layer is rendered or not
- **Weight**: this property controls the intensity/weight of the current layer. 1 is normal, lower values darkens the layer, higher values make it lighter.
- **Density**: this property controls the density of the layer: 1 (full density) to 0 (no density).

3.3.16.3. Hiding/Showing layers

It may be important to show/hide certain layers to help focus your editing. There are two useful functions to support this workflow:

- **Hide all others**: Hides all layers except the currently selected one.
- **Show all**: makes all layers visible.

3.3.16.4. Enabling _Layer append_mode

If you activate the button *Layer append mode* on the layers-tab, every fractal you load into the editor, will be appended as a layer. If the fractal to append has more than one layer, all layers are appended. Only the transforms, final transforms, and gradient are appended. The other settings are ignored.

3.3.16.5. Adjusting flame position and orientation inside a layer

You can move any fractal (layer) freely like a 3D-object in a 3D-modeling-software. All you need is a final transform. In flames which already have a final transform, just enable *Post transform*-editing -mode and you will be able to drag, rotate and scale the fractal by using the mouse.

If you work with 3D-fractals you may also want to modify the fractal along the z-axis. This is also easily done using the final transform, but you have to add variations which perform the desired transforms as:

- **post_rotate_x** and **post_rotate_y**
- **post_ztranslate_wf**
- **post_zscale_wf**
- **affine3D**

3.3.16.6. Layer indicators

There are two indicators to help you to work with layers:

- A red text "layer-append-mode active" on top of your preview if the append-mode is on.
- A small realtime-preview of the currently selected layer on top of the main preview. This preview is only visible if you have more than one layer and more than one layer is visible. If you hide all layers except one (for fine-tuning of this layer) the small preview is hidden and will not distract you. You can also turn off the preview by deactivating the Layer preview button on the left of the Layers tab.

3.3.17. *Channel mixer* tab

Using the channel-mixer you may influence the final coloring of your fractal by color-curves. This modifies the final colors by mapping input values to arbitrary output values.

The simplest case is to take into account brightness levels. The default curve is a linear ramp. Each brightness value outputs the same value. But, when you start to modify the brightness curve, you may change this behavior. For example, drag the curve to higher values at lower levels and vice versa. This will increase the brightness of dark areas and darken brighter areas.

You can do the same with color levels. Overall, these are the options:

- **Off**: no color curves
- **Brightness**: one curve which affects only the brightness levels:
  - new brightness as function of old brightness

- **RGB**: one curve for each color channel:
  - new red as function of old red
  - new green as function of old green
  - new blue as function of old blue
- **Full**: three curves for each color-channel:
  - new red as function of old red, old green and old blue
  - new green as function of old red, old green and old blue
  - new blue as function of old red, old green and old blue

3.3.18. *Quick mutations* tab

"Quick mutations" tab is an easy way of creating a batch of variations of the currently selected flame. You may load any of those mutations into the main editor by double-clicking at it. This creates a new flame in a new slot and does not alter the original flame in any way. You may set the default batch size for quick mutations in the *Preferences* (property **tinaQuickMutationBatchSize**). You may also set the default mutation-type for quick mutations in the *Preferences* (property **tinaQuickMutationDefaultMutationType**).

There is also an experimental strength-parameter which effects the stengths of the mutation. But, due to the complex overall nature of the mutation process, it is not very predictable (yet) and does not work well in many cases.

3.3.18.1. *Mutation type* drop-down list

Here you can choose the type of mutation to be executed:

- **Add transform**: add a random transformation
- **Add variation**: add a random variation to a randomly selected transformation
- **Affine**: randomly change parameters of the affine component of a randomly selected transform
- **Affine3D**: randomly change parameters of the affine component in all three dimensions of a randomly selected transform
- **Bokeh**: add a randomized bokeh-effect
- **Change weight**: randomly change the weight of randomly selected transformations
- **Color type**: randomly change the color-type of randomly selected transformations
- **Gradient position**: randomly change the color-parameter of randomly selected transformations
- **Local gamma**: add randomized local-gamma-effects
- **Random gradient**: create a new random gradient
- **Random parameter**: change some randomly selected parameters of randomly selected variations
- **Random ztransform**: randomly change parameters of the affine component primarily affecting the z-axis of a randomly selected transform

- **Similar gradient**: create a similar gradient
- **Weighting field**: add randomized weighting-field-effects
- **All**: randomly select one of the above mutations. This is the default option.
- **User1** to **User3**: user-defined lists of mutations. You may customize them in the *Preferences* by editing the properties **tinaMutaGenMutationTypesUser(N)**.

3.3.19. *Batch size*

Here you can specify the number of quick mutations to create. In some cases it may make sense to use a large number so you can create a huge batch of quick mutations with one click and then view them later.

3.4. *Transformations and Scripts*

Transformations and *scripts* are the creative heart of JWildfire. Using *transformations* you actually create and modify the fractal structure of a fractal flame, and using *scripts* you can automate this.

3.4.1. *Transformations*

At this tab you can modify the structure or shape of the fractal itself. You do this by adding and modifying transformations.

3.4.1.1. *Theory*

It is beyond the scope of this book to explain exactly how transformations work and how fractal flames are generated in general. If you are interested in the math behind fractal flames, a very good resource is the dissertation by Scott Draves, the inventor of the fractal flame itself: https://flam3.com/flame_draves.pdf

Many of the actions described here are very intuitive to use, and allow you to create fractal artwork without knowing all of the processes involved (which are not too intuitive from an artist's view.)

Besides this users manual, I'm planning to write another book which will bring the two worlds of theory and practice together. I realize this will not happen right away, but sometime in the future.

3.4.1.2. *Transformations* **table**

This table displays all transforms of the currently selected fractal flame. There are two types of transforms:

- (regular) **Transforms**: they build the actual structure of the fractal.
- **Final transforms**: they are applied after the regular transforms and only affect the shape after the fractal is created, but not the inner details.

You may think of **transforms** as being the **polygonal mesh** in a classic 3D-software, and **final transforms** being deformations of this mesh.

3.4.1.3. Transformation *weights*

Each transform has a numerical weight-value attached. This value influences the priority of this transform in the iteration process of the fractal.

3.4.1.4. Transformation *order*

Please note that there is *no* transformation order. Swapping the positions of two transforms would not change the fractal, hence there are no buttons to change the positions of the transformation inside the table.

Importance/influence of a transform can be affected by:

- the **weight** of the transform itself
- **relative weights**, also known as **Xaos**-values

3.4.1.5. *Add* button

Adds a new transform.

3.4.1.6. *L (Linked transform)* button

Adds a linked transform. A linked transform is a regular transform which has a special setup of *relative* weights, so that it is executed after the transform it is linked to, hence the name.

3.4.1.7. *T (Rename transform)* button

Opens a popup to enter a name which is displayed in the transformations table. Useful for complex fractals.

3.4.1.8. *Dupl* button

Duplicate the currently selected transform.

3.4.1.9. *Delete* button

Delete the currently selected transform.

3.4.1.10. *Add final* button

Add a final transform. You can add multiple final transforms. **Please note that all final transforms are applied at each iteration step. Therefore final transforms massively affect render performance.** This does not hold for regular transforms.

3.4.1.11. *Affine* tab

At this tab you can modify the affine component of a transform. This is very intuitive because an affine transformation is a possible combination of

- movement
- rotation

- scale
- shear
- reflection

Affine transformations are represented by the controls you see in the preview area. So, on the *affine* tab, you can make the same changes that preview control dragging does, but in a more precise way.

3.4.1.12. *Post TF* button

Switches between editing the regular component of a transform the post-transform-component.

3.4.1.13. *Reset TF* button

Resets the selected component (regular or post) of the current transform. This means setting the transformation coefficients so that there is no movement, rotation and scale.

3.4.2. *Preserve Z* button

Adds a linear transform in z-direction. It is helpful when creating 3d-fractal flames which should contain nonlinear-transforms which only support x- and y-axis. Without activating the *Preserve Z* option, these fractals would be flat (because any displacement in the z-direction would be lost).

3.4.3. *X1*, *X2*, *Y1*, *Y2*, *O1*, *O2* fields

These are called affine transformation coefficients, and they build the actual value of the transform. Sometimes they are also called **c00**, **c01**, **c10**, **c11**, **c20**, **c21** or **a**, **b**, **c**, **d**, **e**, **f**. If the YZ or ZX edit plane is selected, Z1 and Z2 will replace one of the other pairs.

There are two sets of these coefficients, one set for the regular component and one set for the post component.

3.4.3.1. *Nonlinear* tab

On this tab you can add up to 12 nonlinear transformations (with numerous parameters) to each of the transformations of your fractal.

3.4.3.1.1. *Linear/affine transformations*

Please note that you can also create a lot of very beautiful fractals without any nonlinear transformations. This works by using *linear* as "nonlinear" transformation (for each transformation) and has the advantage that is very fast to compute. But, these are not true fractal flames.

3.4.3.1.2. *Nonlinear transformations*

The primary invention of Scott Draves, when creating the idea of fractal flames, was not only to allow "classic" affine transformations, but also to allow arbitrary transformations. These can be mathematically complex, but the basics are easy to understand.

- do not only allow transformations of the type rotation, translation, shear or reflections

- but do allow more generic transformations, which will still allow that the basic algorithm to work. For example, a transformation which sets any value to 0, will probably not lead to any interesting results. A transformation that undulates a point could be just as valid as a transformation that moves a point along a line, but lead to much more interesting results.

So, a good way of thinking about nonlinear transforms is that they add small details to otherwise more "straight" transformations like a rotation.

3.4.3.1.3. *Different names for nonlinear transformations*

There are different names for nonlinear transformations:

- **nonlinear transformation**: this is the preferred name
- **variations**: this is shorter and also common
- **plugins**: because in some programs they are supplied as plugins which can be loaded at runtime. JWildfire also supports the creation of some kind of plugins by using the *wf_custom* or *wf_custom_full*-variation. But this is intended only for exploration of some new ideas. After exploration, they are included in the main code base, hence in JWildfire you usually do not find the term "plugin".

3.4.3.1.4. *Superimposition of nonlinear transformations*

You can add up to 12 nonlinear transformations on this tab. These will all be overlaid. Most of the nonlinear transformations have many parameters; some really have a lot; and some can be thought of as complete independent simulating programs.

The possibility of changing these parameters together with the superposition of different nonlinear transformations, together with the possibility of changing parameters of the affine transformations, leads to an infinite number of possibilities for each flame.

3.4.3.1.5. Types of nonlinear transformations_

Currently, there are about 800 nonlinear transformations included in JWildfire.

It is beyond the scope of this manual to describe them. In many cases, the description would be very technical and would not help much (in contrast to just trying them out).

But it has turned out that there are some categories of nonlinear transformations which are well understood. You can restrict the selection of nonlinear transformations by using these categories. Each nonlinear transformation can have one or more of the following categories:

- **Blur**: creates a blur-effect, for example a *gaussian_blur*.
- **2D**: a transformation which is restricted to the x- and y-axis.
- **ZTransform**: a transformation which is specialized to transform to the z-axis, for example *zblur*.
- **3D**: a transformation which transforms all three coordinates.
- **DC**: a transformation which also changes the color-index (and overrides the default algorithm to calculate colors).

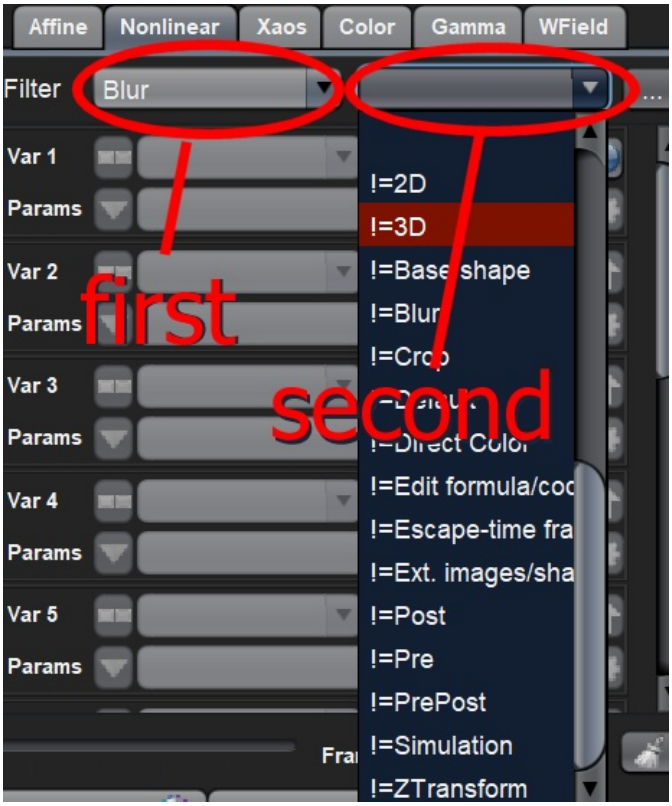
- **Simulation**: a more complex simulation, for example a snowflake-generator.
- **Base shape**: a transformation which creates a base-shape, for example *square*.
- **Pre**: a transformation which has the **Pre**-priority.
- **PrePost**: a transformation which has the **PrePost**-priority.
- **Post**: a transformation which has the **Post**-priority.
- **Crop**: a cropping transformation.
- **Edit formula/code**: a transformation where you can actually enter code or a formula to customize it.
- **Escape-time fractal**: a transformation which computes an escape-time fractal, for example: *fract_mandelbrot*.
- **Supports external shapes**: a transformation which can be customized by using external resources such as images, .svg-files or meshes.

3.4.3.1.6. Filtering nonlinear transformations

Currently, there are about 800 nonlinear transformations included in JWildfire. Sometimes this is just too much, even just to play around with. You can reduce this list by using the *Filter* option.

The *Filter* option consists of two drop-down lists where you can choose up two *Variation profiles*:

- when the **first** *Variation profiles* is selected, only the variations which are inside the selected profile are displayed.
- when **two** *Variation profiles* are selected, only the variations which are inside the both profiles are displayed.
- the **second** drop-down list alternatively allows you to select an ***inverted variation profile*** (prefixed by "!=" in the list). When you select an ***inverted variation profile*** in the **second** option, only variations which are inside the first selected *Variation profile*, but not in the *inverted variation profile*, are displayed.

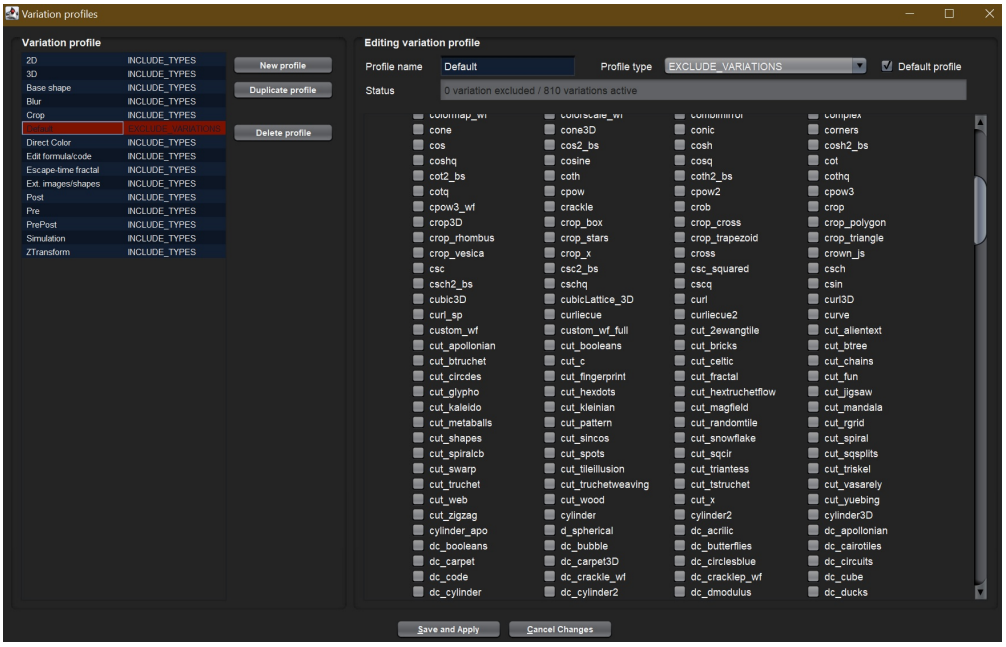


Examples:

- if no *variation profile* is selected: all variations are displayed.
- if the *Blur variation profile* is selected as the **first** option and the **second** option is empty, then only blur-variations are displayed.
- if the *Blur variation profile* is selected as the **first** option and the *3D variation profile* is selected as the **second** option, then only blur-variations which are also 3D-variations are displayed.
- if the *Blur variation profile* is selected as the **first** option and the *!=3D variation profile* is selected as the **second** option, then only blur-variations which are *not* 3D-variations are displayed.

The *variation profiles* may be customized freely. See the next sub-section.

3.4.3.1.7. Editing *Variation profiles*



You may edit the *Variation profiles* freely.

For each profile, these are the options to specify the *Profile type*:

- **Include variations**: specify which variations you want to *include*.
- **Exclude variations**: specify which variations you want to *exclude*.
- **Include types**: specify a list of *variation types*. Each variation that satisfies this selection will be included.
- **Exclude types**: specify a list of *variation types*. Each variation thar satisfies this selection will be excluded.

3.4.3.1.8. Adding nonlinear transformations

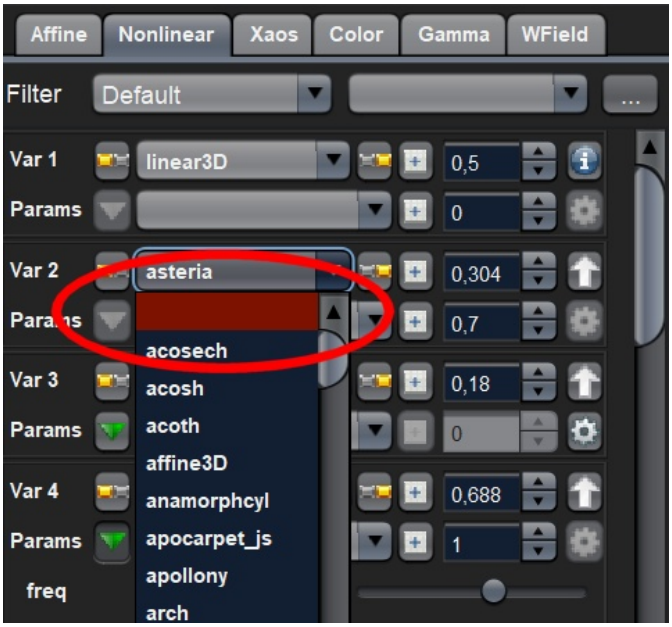
You can add up to 12 nonlinear transformations to each of the transformations of your fractal. Therefore, there are 12 fixed slots on the *Nonlinear* tab where you can select the name of the desired nonlinear transformation from a drop-down list.

Hint: If you know the beginning of the name, you can enter it in the selection list. The corresponding entry will then be selected. For example: typing s-u-b-f will select *subflame_wf* -variation.

3.4.3.1.9. Removing nonlinear transformations

There is a special empty entry in the drop-down list. It is the top entry and should be used if you want to remove a specific nonlinear transformation from your fractal.

When you select this entry the corresponding variations will be removed and the following variations (if any) will move up.



3.4.3.1.10. Editing nonlinear transformations

At each variation-slow there are the following options to edit the nonlinear transformation:

- a *numeric field* right of the name: this is the **amount** of this nonlinear transformation.

- a *drop-down list* below the name: here you can select a **parameter** of this nonlinear transformation you want to edit. Some nonlinear transformations have no parameters. In this case the *drop-down list* is empty.
- a *numeric field* right of the **parameter** *drop-down list*: this is the amount of the currently selected parameter.
- a small *yellow/gray* toggle button left of the name: use this button to make this transformation a *pre*-transformation. Any non-linear transformation having the prefix "pre_" in the name usually is a *pre*-transformation by convention. In this case the button will be active. You may change it to use this transformation as a *non-_pre*-transformation.
- a small *yellow/gray* toggle button right of the name: use this button to make this transformation a *post*-transformation. Any non-linear transformation having the prefix "post_" in the name usually is a *post*-transformation by convention. In this case the button will be active. You may change it to use this transformation as a *non-_post*-transformation.
- a small *green arrow* button left of the **parameter** *drop-down list*: Here you can show/hide an optional parameter panel, with which you can control all parameters of the currently selected variation with separate sliders. Since this takes up a lot of space, it can be switched on/off separately for each variant.

- an *arrow* button right of the **amount** field: here you can move this variation up one position.
- small *cog* button right of the **parameter *amount**: some variations have not only numeric parameters. In this case you will need to click at this button to open a popup which allows to edit the corresponding parameter. For example, the *svg_wf* variation has a *svg* parameter that points to an external **.svg** file. If you click the *cog* button with the parameter selected, you get a file selection dialog.

3.4.3.2. *Xaos* tab

Xaos is a synonym for *relative weights*. Since "Xaos" is much shorter, it was used in the user interface. But the term *relative weights* is much easier to understand:

- Each transformation has an attached *weight*. This is an numerical value and controls how often

the transformation is invoked in the iteration process. This weight is called an *absolute weight*.

    - When using only *absolute weights*, no attention is paid to the execution order of the transformations.

- It has been observed that the execution order of certain transformations can lead to important and interesting changes of the structure of the fractal.
    - The basic idea behind *relative weights* is to promote certain execution orders and prevent other execution orders.

3.4.3.2.1. *Relative weights* table

The *relative weights* are basically a matrix. Using this tab you can edit the part of this matrix that concerns the currently selected transformation.

For each combination of the chosen transformation with all of the other transformations of the fractal, this table describes the probability that a certain combination will be executed in the iteration process:

- the first column contains the order of execution of two of these transformations.
- the second column contains the corresponding probability that the order of execution actually can happen:
    - a value of 0 means that it does not happen at all.
    - a value of 1 is the default value. The actual probability is not 100%, but depends on the distribution of the *absolute weights*.

So, a relative weight of 0 is much easier to understand than the other values. Luckily, finding the right combinations where it makes sense to use 0 as the *relative weight* has the biggest impact on the fractal images. All the other possibilities are also very interesting, but lead to much more subtle differences (in most of the cases).

3.4.3.2.2. *View as "to"*

Controls which values of the *relative weights* matrix are displayed in the table. The *View as "to"* option shows the transitions from the currently selected transformation to any other selected transformation. This includes the transformation itself, which can be a bit confusing at first. But calling the same transformation again after this transformation has already been called is a perfectly valid and common behavior.

For example, having three transformations, **1** and **2** and **3**, the table will show:

- **1 to 1** and **1 to 2** and **1 to 3** when transform **1** is selected,
- **2 to 1** and **2 to 2** and **2 to 3** when transform **2** is selected,
- **3 to 1** and **3 to 2** and **3 to 3** when transform **3** is selected.

Note: changing the view mode does not affect the actual *relative weights*.

3.4.3.2.3. *View as "from"*

Controls which values of the *relative weights* matrix are displayed in the table. The *View as "from"* option shows the transitions from any other transformation to the currently selected transformation. This includes the transformation itself, which can be a bit confusing at first. But calling the same transformation again after this transformation has already been called is a perfectly valid and common behavior.

For example, having three transformations, **1** and **2** and **3**, the table will show:

- **1 from 1** and **1 from 2** and **1 from 3** when transform **1** is selected,
- **2 from 1** and **2 from 2** and **2 from 3** when transform **2** is selected,
- **3 from 1** and **3 from 2** and **3 from 3** when transform **3** is selected.

Note: changing the view mode does not affect the actual *relative weights*.

3.4.3.2.4. *View as "to"* **vs** *View as "from"*

The values displayed in these two views are in some way inverses of each other.

For example, having again three transformations, **1** and **2** and **3**:

- select transformation **1**
- in *View as "to"* select the row **1 to 2** and press the *0* button (setting this *relative weight* to 0)
- switch to *View as "from"* mode and selected transformation **2**
- you should now see that the value at the first row **2 from 1** is 0

Please note that switching the view mode does not affect the order of execution of transformations (relative weights) in any way. It's just different views of the same thing.

3.4.3.2.5. *0* **button**

Sets the currently selected *relative weight* to 0. Entering a numerical value would have the same effect, but clicking the button is easier.

3.4.3.2.6. *1* **button**

Sets the currently selected *relative weight* to 1. Entering a numerical value would have the same effect, but clicking the button is easier.

3.4.3.2.7. *Reset* **button**

Resets all the *relative weights* of the currently selected transformation (to the numerical value 1).

3.4.3.2.8. *Reset all* **button**

Resets all the *relative weights* of all transformations (to the numerical value 1).

3.4.3.3. *Color* tab

On this tab you will find important options for the coloring options. These options do not primarily determine the final colors, but how many color details the fractal has and how they are distributed over the entire shape of the fractal. During the iteration process, each point has an associated color. When a transformation maps a point to a new position, it also gets a new color. The options on this tab give you control over exactly how these colors are calculated.

3.4.3.3.1. *Coloring type*/*Color*/*Speed* (by Rick Sidwell)

The *Coloring type* controls how the color "moves" through the fractal during the iteration process. These are the options:

- **None**: does not change the point's color; it will be the same as the input point. This is the default for final transforms since using a different type would reduce the color range.
- **Diffusion**: is the default flame coloring type invented by Scott Draves and used in all flame fractal programs and is the default for normal transforms. It is based on a gradient, which can be selected and edited in the *Gradient* tab, and uses an index to the gradient between 0 (leftmost color) and 1 (rightmost color) to set the flame colors. There are two controls: *Color* and *Speed*.
  - *Color* specifies the target gradient index for the transform. With each iteration, the index for the point will move towards the target index.
  - *Speed* controls how fast it moves; 0 will move the index halfway between the current and target indices. Higher values will move it less than halfway (more slowly); the highest value of 1 means it won't move at all; the color won't change. Negative Speed values will move it more than halfway (more quickly); the extreme of -1 means it moves all the way; the new color will be the gradient color matching the Color value.
- **Target**: ignores the gradient. The Color setting is a target color instead of a target gradient index. On each iteration, the point's color will be interpolated between the current color and the target color. *Speed* works in the same way as DIFFUSION:
  - 0 will make the color halfway between the current and target colors,
  - 1 will not change the color,
  - and -1 will set the color to the target color.
- **TargetG**: works like **Target**, but uses the gradient to set the target color. So the target color will change when a new gradient is selected, but the color is more likely to be consistent with other gradient colors. The point's color is interpolated with the target color as with **Target** rather than moving along the gradient.
- **Distance**: uses the gradient indexed by the distance the transform moved the point. *Color* determines the starting point and *Speed* controls how far along the gradient each unit of distance moves from there. Like other coloring types, a Speed of -1 sets the color to the specified color. But the input color is ignored, so unlike the other coloring types, a Speed of 1 will still change the color. **Distance** is especially effective for coloring flames made with a single non-base-shape transform, which are otherwise quite difficult to color.
- **Cyclic**: uses the gradient, but simply shifts the gradient index right by the amount specified by Speed. If it goes off the end of the gradient, it wraps around to the other side. **Cyclic** will not converge to a color when iterated, therefore it doesn't work as the only coloring type in a flame;

some other iterated transform needs to use **Diffusion** to get a stable color that **Cyclic** can use as a starting point. Normally, Speed is set to m/n, where m and n are small integers; this makes the colors cycle through n gradient colors, and is especially useful for coloring tilings.

3.4.3.3.2. *Draw mode/Opacity*

Controls the visual contribution of the currently selected transformation to the fractal.

These are the options:

- **Normal**: each iteration point contributes to the fractal without any modification.
- **Hidden**: the contribution of the currently selected transformation is invisible (not drawn).
- **Opaque**: the contribution of the currently selected transformation is faded out by taking into account the *Opacity* parameter. Any value between 0 and 1 is possible, including the two extremes:
  - 0 means minimum Opacity, this is equal to using the **Hidden** setting.
  - 1 means maximum Opacity, this is equal to using the **Normal** setting.

3.4.3.3.3. *Material/Mat Spd*

Works similar to *Color*/*Speed*, but controls the material index used in Solid rendering rather than the color index.

3.4.3.4. *Gamma* tab

On this tab you will find some advanced color change options. They are an extension of the original fractal flame algorithm.

The original fractal flame algorithm uses a gradient together with a pointer to a certain position within this gradient to color the fractal. This gradient pointer changes during the iteration process, i.e. it points to different colors, which eventually causes the fractal to contain different colors. The change of the gradient pointer is controlled by the *Color* and *Speed* parameters, which you can find on the *Color* tab.

The basic idea behind the extension of this algorithm was to change similar color-related properties with a similar process:

- **Gamma**
- **Contrast**
- **Saturation**
- **Hue**

Therefore, on this tab you will find pairs of parameters: the parameter itself and its rate of change, e.g. *Gamma* and *Gamma Spd*. The parameters *Gamma* and *Gamma Spd* were the first, hence the name of the tab.

And because this kind of effect is local (changes with each iteration), it is called a **local color change**. Again, because the change in *gamma* was the first change introduced, these effects are

called **local gamma** effects. (But there was not enough space in the user interface, hence only the name *Gamma* of the tab.)

3.4.3.4.1. *Gamma/Gamma Spd*

These parameters control local changes in gamma correction. A higher gamma correction brightens the fractal (locally) and vice versa.

3.4.3.4.2. *Contrast/Contrast Spd*

These parameters control local changes in contrast.

3.4.3.4.3. *Saturation/Saturation Spd*

These parameters control local changes in saturation. Can help create very dramatic effects.

3.4.3.4.4. *Hue/Hue Spd*

These parameters control local changes of the hue. Can lead to very unusual effects, as the hue can change drastically.

3.4.3.5. *Randomize all* **button**

Creates a random local gamma effect:

- when the *Whole fractal* checkbox is checked, each transformation of the fractal might be affected.
- when the *Whole fractal* checkbox is not checked, a random local gamma effect is only applied to the current selected transformation.

3.4.3.6. *Reset all* **button**

Removes the local gamma effect:

- when the *Whole fractal* checkbox is checked, the local gamma effect for each transformation of the fractal is removed.
- when the *Whole fractal* checkbox is not checked, only the local gamma effect for the current selected transformation is removed.

3.4.3.7. *Whole fractal* **checkbox**

Affects how the *Randomize all* button and the *Reset all* button will work, please see above.

3.4.3.8. *WField* **tab**

Weighting fields are a nice artistic tool that can be used to create very organic-looking fractals,

The basic idea is to add some kind of "natural disturbance" at the transform-level to the flame-fractals. This is done in a non-destructive way, i.e. without changing the actual flame. This makes it very easy and intuitive to try out different styles and parameter combinations. It is possible to choose between different types of this kind of disturbance, for example the classic *Perlin Fractal*

Noise for a very organic style. You can also load image-maps (photographs) for more individual styles. Each weighting field type offers various options.

Currently, the following fractal-parameters may be affected by a weighting-field:

- variation-amount
- variation-parameter-amount (for up to three parameters per transformation, for example, *frequency* parameter of a *waves*-transform)
- color-index (allows subtle color-changing effects)

3.4.3.9. *Weighting field* drop-down list

Type of the weighting field to apply to the currently selected transformation. These are the choices:

- **None**: no weighting field
- **Cellular Noise**: apply a weighting field using the *cellular noise algorithm*. This produces structures which look like cells.
- **Cubic Noise**/**Perlin Noise**/**Simplex Noise**/**Value Noise**: apply a weighting field using the selected *noise algorithm*. This gives a nice organic look.
- **Cubic Fractal Noise**/**Perlin Fractal Noise**/**Simplex Fractal Noise**/**Value Fractal Noise**: apply a weighting field using the selected *fractal noise algorithm*. This also gives a nice organic look. The word "fractal" here means applying the same type of noise multiple times at different scales.
- **Image Map**: apply a weighting field using the levels of red color channel of the given image map.

3.4.3.10. *Input coordinate* drop-down list

Sets the position where the weight field is calculated. These are the options:

- **Position**: Position of the current iteration point before applying the affine transform
- **Affine**: Position of the current iteration point after applying the affine transform

3.4.3.11. *Var amounts*

Strength by which the magnitude (also called *amount*) of the nonlinear variations is influenced by the weighting field. A value of 0 means no influence by a weighting field.

3.4.3.12. *Color amount*

Strength by which the color-calculation is influenced by the weighting field. A value of 0 means no influence by a weighting field.

3.4.3.13. *Var param 1*/*Var param 2*/*Var param 3*

You can define up three parameters of nonlinear transformations which may be affected by the weighting field. The numerical field specifies the magnitude, in the drop-down list you select the name of the parameter. A value of 0 means no influence by a weighting field.

3.4.3.14. *Jitter amount*

Strength of noise to add after the transformation. A value of 0 means don't add additional noise.

3.4.3.15. *Seed*

Parameter to randomize the shape of the noise. Different values lead to different distributions.

3.4.3.16. *Frequency*

Frequency of the noise. Larger values lead to more details, smaller values lead to a more smooth shape.

3.4.3.17. *Gain*/*Octaves*/*Lacunarity*/*Noise Type*

These options for fractal noise control how to apply the same type of noise multiple times at different scales.

3.4.3.18. *Return Type*/*Distance Function*

These options control the shape of *cellular noise*.

3.4.3.19. *CentreX*/*CentreY*/*SizeX*/*SizeY*

These options control how to place the image map when using the option **Image map** as Weighting field type.

3.4.3.20. *Randomize all* **button**

Creates a random weighting field:

- when the *Whole fractal* checkbox is checked, each transformation of the fractal might be affected.
- when the *Whole fractal* checkbox is not checked, a random weighting field is only applied to the current selected transformation.

Hint: On small monitors this button might be hidden per default. You can resize the whole window or the transformations table to make it visible.

3.4.3.21. *Reset all* **button**

Removes the weighting field:

- when the *Whole fractal* checkbox is checked, the weighting field for each transformation of the fractal is removed.
- when the *Whole fractal* checkbox is not checked, only the weighting field for the currently selected transformation is removed.

3.4.3.22. *Whole fractal* **checkbox**

Affects how the *Randomize all* button and the *Reset all* button work, please see above.

3.4.4. *Scripts and Custom-Buttons*

At this tab you can manage JWF-scripts and organize custom buttons which allow easier access certain scripts.

Custom-buttons are completely optional. You can also execute scripts directly from the *Scripts* library by double-clicking at a script name.

JWF-scripts are written in the Java-language and can use most of the language-features offered by Java 1.4. Therefore you could also read external files, access resources from the internet, open custom-windows etc. There are also several possibilities to interact with the Main Editor, for example, to modify the currently edited flame and to refresh the preview.

3.4.4.1. *Scripts* library

The scripts-library contains of two parts:

- built-in scripts: these are installed together with the program and can not be modified in any way.
- your own scripts: these are located in a folder outside the JWildfire-installation-drawer, so that they are not lost when you update JWildfire. You must specify this path in the *Preferences* by setting the property **tinaJWFScriptPath**.

User-defined scripts may be structured in sub-folders in order to make them easier to find.

3.4.4.2. *Import script* button

Import a script from an external file. The following file-types are supported:

- a single **.java**-file containing the script-code
- a **.zip**-archive containing the script-code together with additional files or folders.

When you download scripts from the internet, for example from https://www.jwfsanctuary.club, they usually come as .zip-files.

3.4.4.3. *New* button

Create a new script from scratch. This is rarely used; it is usually easier to duplicate an existing script or to create a script from a flame and then modify it.

3.4.4.4. *Dupl* button

Duplicates the currently selected script.

3.4.4.5. *From flame* button

This is a very useful function to get started with creating scripts. It creates a script which contains all the Java-code to create the currently selected flame.

This has the following purposes:

- you have a working script which actually produces something with a single button-click, and
- by modifying certain parts you can programmatically create similar flames.

3.4.4.6. Creating flame-randomizers

This function also creates code, which randomizes certain parameters at the level of each transform. This code is inactive by default. The generated flames of this script will always look the same until you modify the script. This is intentional, because randomizing a random selection of parameters usually does not lead to interesting results. Therefore you must find out for yourself, which parameters are good for randomizing a flame and which are not. This always depends on the type of fractal, and small differences between flames may affect this behavior in a very strong way.

You can find the inactive code by looking for comments like this:

```
// random affine transforms (uncomment to play around)
//   XFormTransformService.scale(xForm, 1.25-Math.random()*0.5, true, true, false);
//   XFormTransformService.rotate(xForm, 360.0*Math.random(), false);
//   XFormTransformService.localTranslate(xForm, 1.0-2.0*Math.random(), 1.0-
2.0*Math.random(), false);
// random affine post transforms (uncomment to play around)
//   XFormTransformService.scale(xForm, 1.25-Math.random()*0.5, true, true, true);
//   XFormTransformService.rotate(xForm, 360.0*Math.random(), true);
//   XFormTransformService.localTranslate(xForm, 1.0-2.0*Math.random(), 1.0-
2.0*Math.random(), true);
```

The first line is an actual comment (not code). But the 2nd line is code, which was commented out.

- *Commenting in* means to remove the characters // at the start of line, this activates the code.
- *Commenting out* means to add the characters // at the start of a line, this inactivates the code.

If you now *comment in* the 3rd line, you apply a rotation by an angle which is randomized.

```
XFormTransformService.rotate(xForm, 360.0*Math.random(), false);
```

The expression **360.0*Math.random()** calculates a random value between 0 and 360, which differs each time you execute the script. So you will get a different fractal flame each time you execute the script.

If you see that this leads to interesting effects, you may leave it that way and work at another transformation or, otherwise you could uncomment another line of this pre-generated code. You can also play with the values, for example, to replace the **360** by some smaller value when the rotations are too large. Of course, you can combine all of these code-fragments freely and also add your own code. A rule of thumb is that many large changes at a fractal flame at the same time do not lead to interesting results. In many cases you will not have any visible result at all.

3.4.4.7. *Btn* button

Create a *Macro button* in the bottom area of the preview to execute the currently selected script. You can remove this button using the *Delete*-button at the *Macro buttons* subtab.

3.4.4.8. *Scan* button

Rescans the script-library. This is only necessary when you modified files from outside JWildfire or changed the library-path in the *Preferences*.

3.4.4.9. *Run* button

Runs the currently selected script.

3.4.4.10. *Edit* button

Opens a pop-up window to edit the currently selected script.

3.4.4.11. *Description*

Shows a description of the script, which might contain hints for usage or copyright information. Use the Edit button to edit the text.

3.4.4.12. *Code preview*

Shows the code of the script. Because this area is too small for making changes there is a separate *Edit* button to edit the script.

3.4.4.13. *Macro buttons*

At this tab you can control so-called *Macro buttons*. A *Macro button* is a button which executes a script. Scripts are also called *Macros*, hence the name.

Macro buttons appear either below or to the right side of the Preview/Editor area, depending on the **tinaMacroButtonsVertical** setting in *Preferences*.

Add a new Macro button by selecting a script and clicking the *Btn* button as described above.

3.4.4.13.1. *Macro buttons* table

In this table all *Macro Buttons* are displayed and you can also edit the label and short hint of each button.

3.4.4.13.2. *Up* button

Moves the button in the user-interface one position to the left.

3.4.4.13.3. *Down* button

Moves the button in the user-interface one position to the right.

3.4.4.13.4. *Delete* button

Deletes the currently selected *Macro button*.

3.5. Global functions

In this section you will find global functions that are used very often. For example, loading and saving flame files or generating random flames.

3.5.1. *Random batch* button

Creates a new batch of random flames and displays them in the *thumbnail ribbon*. This is probably one of the most popular features of JWildfire because using this function allows you to create endless beauty in a very relaxing way. It is also very unlikely that you will ever generate two fractals which look exactly the same, so it's likely that you will be surprised by new and interesting results even after years of using the program.

Please note that this will, per default, remove any previously loaded flames together with their undo-history from memory. So, any changes you did not save will be lost. You can change this behavior by editing the parameter **tinaRandomBatchRefreshType** in the *Preferences*.

The random-flame-generating-process can be influenced by changing the following parameters:

- **Rnd flame-generator**
- **Rnd Symmetry**
- **Rnd Gradient**
- **Rnd WField**
- **tinaRandomBatchSize** (in the *Preferences*)
- **tinaRandomBatchBGColorRed**, **tinaRandomBatchBGColorGreen**, **tinaRandomBatchBGColorBlue** (also in the *Preferences*)

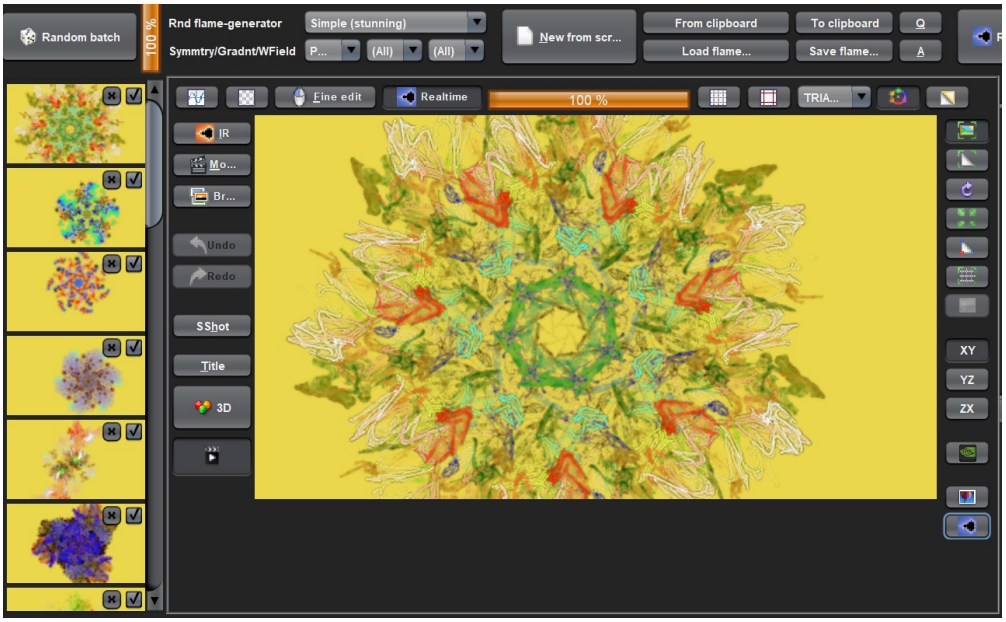
All of these parameters may be combined freely to create customized random flames. See the following sub-sections for more details about these parameters.

3.5.1.1. Example of combination of parameters for creating specific random flames

For example, to generate mandala-like-structures which look very organic and have a yellowish background, you could use the following setup:

- **Rnd flame-generator**: **Simple (stunning)**
- **Rnd Symmetry**: **Point**
- **Rnd Gradient**: any
- **Rnd WField**: **(All)**
- **tinaRandomBatchBGColorRed**: **235**
- **tinaRandomBatchBGColorGreen**: **215**

- **tinaRandomBatchBGColorBlue**: 75



3.5.2. *Rnd flame-generator*

Here you can choose the random-flame-generator that will be used to generate a batch of random flames. A random-flame-generator is a set of instructions to generate a skeleton of a certain type of fractal and fill this skeleton with random values or random sub-structures. What works well and what does not work well depends on the type of fractals. Sometimes the result is just the random change of a single number, or sometimes a random transformation or formula is inserted.

Typically, the JWildfire random-flame-generators employ the strategy of using some well explained ranges of parameters together with a large degree of freedom. This way you can often get a very pleasing result, and you may encounter some new variations/shapes which you have never seen or tried before.

There are two types of random-flame-generators:

- **meta-random-flame-generator**: uses randomly-chosen random-flame-generators internally
- **random-flame-generator**: actually creates a random flame

3.5.2.1. The special "All"-random-flame-generator

The most common random-flame-generator is the default generator named "All". It is a meta-random-flame-generator and contains all the other random-flame-generators. When it generates a random flame, it randomly selects one of those internal generators to do the actual work. By choosing "(All)" you will get the most diverse results.

3.5.3. Rnd Symmetry

Sets the random-symmetry-generator for generating random flames. This generator randomly sets post-symmetry-options of the random flames.

These are the options:

- **None**: the flame will have no post-symmetry
- **XAxis**: a random x-axis-symmetry
- **YAxis**: a random y-axis-symmetry
- **Point**: a random point-axis-symmetry
- **(All)**: randomly selects a random-symmetry-generator of generating post-symmetry
- **(All, sparse)**: like **(All)**, but with reduced probability to generate post-symmetry at all

The default value is **(All, sparse)**, which means it will randomly create random symmetry of all types, but only a few random flames.

See the section about *Post symmetry* for more details about the types of post-symmetry and related parameters.

3.5.4. Rnd Gradient

Set the random-gradient-generator for generating random flames.

There are the following options:

- **(All)**: meta-random-gradient-generator which randomly chooses a random-gradient-generator to create the gradient
- **Strong hue**: creates a smooth gradient with strong colors
- **Stripes**: creates a gradient with many regular color changes, having a "stripy" look
- **Monochrome**: creates a smooth and monochrome gradient
- **Smooth**: creates a smooth gradient with primarily pastel colors
- **Bold**: creates a non-smooth gradient with large blocks of strong colors
- **Two colors**: creates a smooth gradient by fading two colors into each other
- **Uniform curves**: creates a random color curve with evenly distributed base colors and then derives the gradient from it. Using this generator will give both a gradient and matching color curves.
- **Uneven curves**: creates a random color curve with unevenly distributed base colors. Using this generator will give both a gradient and matching color curves.

The default setting is *Uneven curves*, which creates usually the most interesting and versatile gradients together with corresponding color curves.

3.5.5. Rnd WField

Set the random-weighting-field-generator for generating random flames.

These are the options:

- **Cellular Noise**: create random weighting-fields using Cellular Noise
- **Basic Noise**: create random weighting-fields using basic noise, like Perlin noise

- **Fractal Noise**: create random weighting-fields using fractal noise, like Perlin fractal noise. Fractal noise means to apply one type of noise multiple times at different scales.
- **Image Map**: create random weighting-fields using an image map. By default, the same fixed internal map is always used. You can change it by editing the flame afterwards.
- **(All)**: randomly selects a random-weighting-field-generator to generate weighting-fields
- **(All, sparse)**: like **(All)**, but with reduced probability of generating weighting-fields at all

3.5.6. New from scratch

Create a new and empty flame with a random gradient. This is usually the first step when creating a fractal manually "from scratch", hence the name.

3.5.7. From clipboard

Load a flame which was stored in the clipboard.

In social-media it is common to exchange flame-files in text-form by posting them. When loading such a flame you usually copy the text and the use the *From clipboard* button to load it into JWildfire.

3.5.8. Load flame

Load a flame from a file and display it in the *Thumbnail ribbon*.

Flame files can also contain a batch of flames. In this case, each flame from the flame-batch-file will be imported as an individual flame which is displayed as an individual thumbnail in the *Thumbnail ribbon*. Each individual flame of such a batch also has its own undo-history.

3.5.9. To clipboard

Stores the currently selected flame in text-form in the Clipboard.

In social-media it is common to exchange flame-files in text-form by posting them. Using this function you may export your flame to paste it somewhere, for example, Facebook.

3.5.10. Save flame

Save the currently selected flame.

3.5.11. Q (Quicksave)

Quicksaves the currently selected flame by automatically generating a filename in the form **"qsave_"** + <**Date**> + <**Counter**> +**".flame"** in the folder where the last flame was saved (the default flame folder if this is the first). You may set the default flame folder in the *Preferences* by changing the property **tinaFlamePath** .

3.5.12. A (Save all)

Save all selected flames together into one flame. Selected flames are indicated by a small checkmark at the right upper edge of the thumbnail. See the sub-section *Selecting flames in the thumbnail ribbon* for more information about selecting flames.

3.5.13. *Render image/movie* button

Render the current flame using the selected *quality* and *resolution profile*.

Rendering means to create a final image and consists of the following steps:

- perform the fractal iteration process, possibly using an increased internal resolution (compared to the desired image resolution). See *Spatial oversampling* for details.
- perform tone-mapping by turning the raw fractal information into colored pixels. See the section about the *Coloring tab* for more details.
- optionally perform AI-based denoising. See *AI-based denoiser* for more details.

When you click the Render image/movie button, you will be prompted for the filename to use. The extension determines whether to render an image or movie. Use .png or .jpg to render an image, or .mp4 to render a movie.

The rendering runs in the background and can be cancelled at anytime. But, you can not render more than one flame at a time here. For rendering multiple images see the section about *Batch renderer*.

3.5.14. *Render resolution* drop-down list

Here you choose the desired render resolution by choosing a *Resolution profile*. JWildfire comes with some predefined *resolution profiles*, but you can also customize them.

Even when you do not render the flame in the main editor, the correct render resolution should be specified because of the aspect ratio. After changing a *resolution profile*, JWildfire will change the size of the preview-area accordingly to the aspect ratio of the selected *resolution profile*. For example, when you select a profile with same width and height, you will have a square shaped preview area.

3.5.15. Edit *resolution profiles*

Edit the list of resolution profiles by clicking the button to the right of the render resolution drop down list. In this pop-up window you may alter existing *resolution profiles* as well as create your own.

These are the parameters:

- **Width** x **Height**: image resolution
- **Default profile**: one profile can be defined as the default profile. The profile is chosen when you start JWildfire.

3.5.16. *Render quality* drop-down list

Here you choose the desired render quality as well as some output options by choosing a *Quality profile*. JWildfire comes with some predefined *quality profiles*, but you can also customize them.

Especially when using an AI-based post-denoiser, you may want to revisit your quality profiles. Such a denoiser can help to create visually appealing results with much lower quality settings in order to reduce render times. The default settings were designed to be used without such a denoiser. See the sub-chapter *AI-based denoiser* for more details.

Quality profiles also contain the information about which type of images should be generated. For example, you may specify inside a *quality profile* that JWildfire shall generate an additional z-buffer-image when using solid rendering.

3.5.17. Edit *quality profiles*

Edit the list of quality profiles by clicking the button to the right of the render quality drop down list. In this pop-up window you may alter existing *quality profiles* as well as create your own.

There are the following parameters:

- **Quality**: quality level
- **with HDR**: create an additional *High Dynamic Range Image* in .hdr-format. Such an image contains raw colors in a much more precise way than conventional images. But, it can usually not be displayed without special software. Is is similar to an RAW-image in photography.
- **with Z-Buffer**: create an additional z-buffer-image. Together with the actual image of the fractal you can create "3d wiggle images" which are/were popular on some social media platforms like Facebook. Currently, this is only implemented for fractals rendered in solid mode.
- **Default profile**: one profile can be defined as the default profile. The profile is chosen when you start JWildfire.

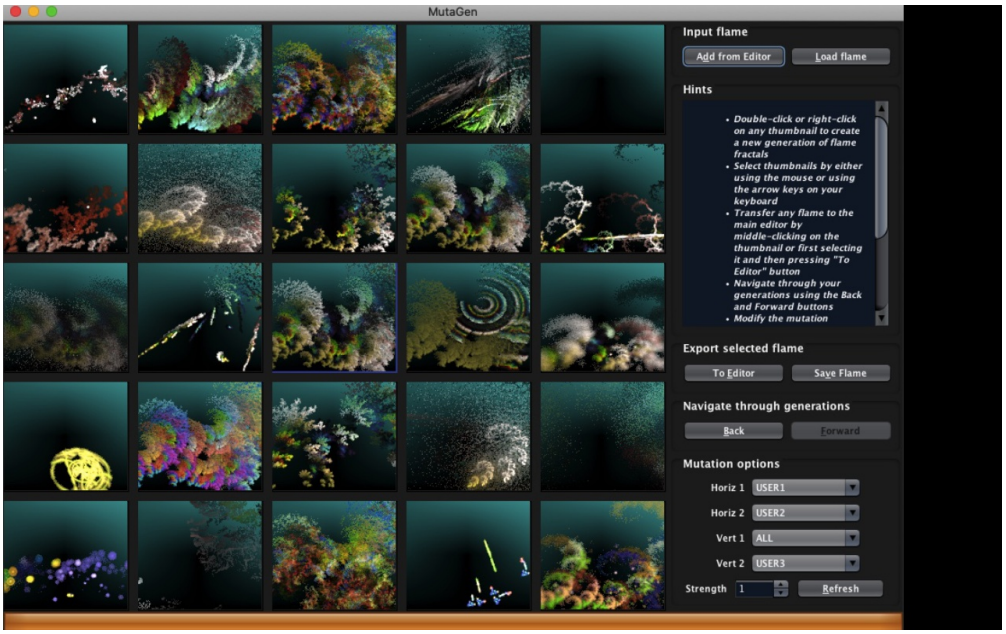
3.5.18. Batch renderer

Quick-saves the currently selected fractal flame and sends it to the *Batch renderer*.

Please note, that this does not actually render the fractal flame. You must later open the *Batch renderer* and invoke the rendering process. Therefore, it is just a shortcut to simplify adding fractal flames to the rendering batch.

Hint: when you continue to edit the fractal flame and create a quicksave again, the *Batch renderer* will still render the earlier quicksave you sent to the *Batch renderer*.

Chapter 4. *MutaGen* module



In this module you may create mutations of a certain flame in an interactive way. Open the MutaGen module by selecting *Fractal flames: MutaGen* from the *Windows* menu.

There are several mutation types affecting different "genes" (types of flame properties). Those mutations are applied at two levels: generation A (8 new fractals from the base flame) and generation B (16 new fractals from generation A), generating a total of 24 new flames at each mutation step.

You can easily exchange fractals between the main editor and the new module at any time.

There is also a history, i.e. you can navigate back to earlier generations at any time.

See the section about *Quick mutation types* for a description of available mutation types.

Chapter 5. *Interactive Renderer* module

The *Interactive Renderer* is an alternative rendering module which focuses on rendering a single image in contrast to the *Batch Renderer* which renders multiple images. Open the Interactive Renderer module by clicking the Interactive Renderer button on the left of the preview area or by selecting *Fractal flames: Interactive renderer* from the *Windows* menu.

The *Interactive Renderer* has three main purposes:

- allows the user to watch the rendering process for important artwork to let it render "until it is done".
- explore random fractal flames and watch them get rendered and become clearer and clearer. This may be very relaxing.
- explore the fractal rendering process in general.

5.1. Saving the image while rendering

While rendering is in progress you may save a snapshot of the current image at any time. This will not influence or even interrupt the ongoing rendering process in any way and is an intended workflow.

5.2. Stats

Statistics about the current quality level and predictions when new quality levels will be reached are shown in the top left of the Interactive Renderer window.

You may toggle viewing of Stats off when you are not interested in them by clicking the Stats button. This will slightly improve render performance.

5.3. Preview

While previewing the image during rendering is a key-feature of the *Interactive Renderer*, it also slows down rendering. For most of the flames the slowdown will be negligible, but there can be a significant slowdown for some flames. So you may want to alternate between faster rendering for a short period and monitoring the render progress using the Preview button.

5.4. From Editor

Imports the currently edited fractal flame from the *Main Editor* and starts rendering it.

Please note that this will cancel any ongoing render and you will lose any unsaved results.

5.5. From Clipboard

Imports a fractal flame from the clipboard and starts rendering it.

Please note that this will cancel any ongoing rendering and you will lose any unsaved results.

5.6. Load flame

Loads a fractal flame from a flame and starts rendering it.

Please note that this will cancel any ongoing rendering and you will lose any unsaved results.

5.7. Resume render

Loads a previously saved *Render State* and starts rendering it.

Please note that this will cancel any ongoing rendering and you will lose any unsaved results.

5.8. Stop

Stops the current ongoing rendering process.

Please note that the render can not be resumed. You must save the *Render State* before stopping the render when you want to be able to continue a render.

5.9. 1:2

Sets the render size to one half of the currently selected render size. Can help to get a quick preview when rendering at very large resolutions.

5.10. 1:4

Sets the render size to one quarter of the currently selected render size. Can help to get a quick

preview when rendering at very large resolutions.

5.11. Full

Uses the full render resolution of the selected *Resolution Profile*. This is the default setting.

5.12. Resolution

Desired render resolution. See the sub-section about *Resolution Profiles* for defining custom resolutions.

5.13. Save render state

Save the current state in a proprietary format. You may load this file later to continue rendering at exactly the same step.

5.14. Save image

Create a snapshot of the current render and save it as an image.

5.15. Z (Save Z-Buffer)

Create a snapshot of the current render and save it as a z-buffer. Currently, z-buffers are only supported when rendering in solid mode.

5.16. Autoload saved image

When this option is checked, JWildfire will automatically load and display images created by *Save image* and the *Z (Save Z-Buffer)* function. Viewing the final image may help to better decide when rendering is complete rather than viewing the render-preview.

5.17. To Editor

Transfer the currently rendered flame to the Main editor.

5.18. To Clipboard

Transfer the currently rendered flame to the clipboard.

5.19. Save Flame

Save the currently rendered flame to file.

5.20. Next (random flame)

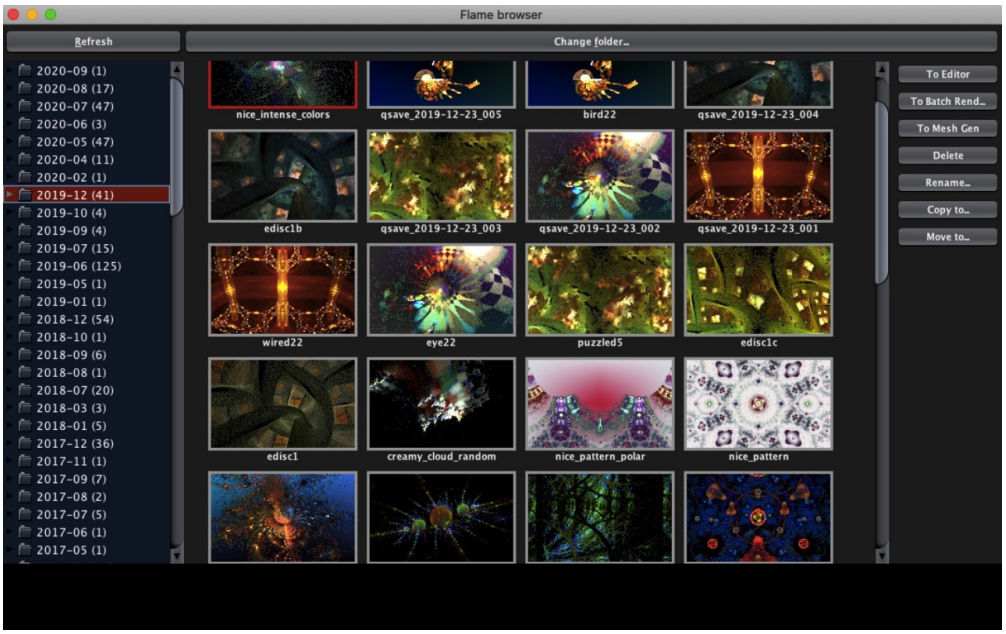
Creates a random flame and starts rendering it.

Please note that this will cancel any ongoing rendering and you will lose any unsaved results.

5.21. Random generator

Select the random-flame-generator used to generate a random-fractal-flame when using the *Next* button.

Chapter 6. Flame Browser

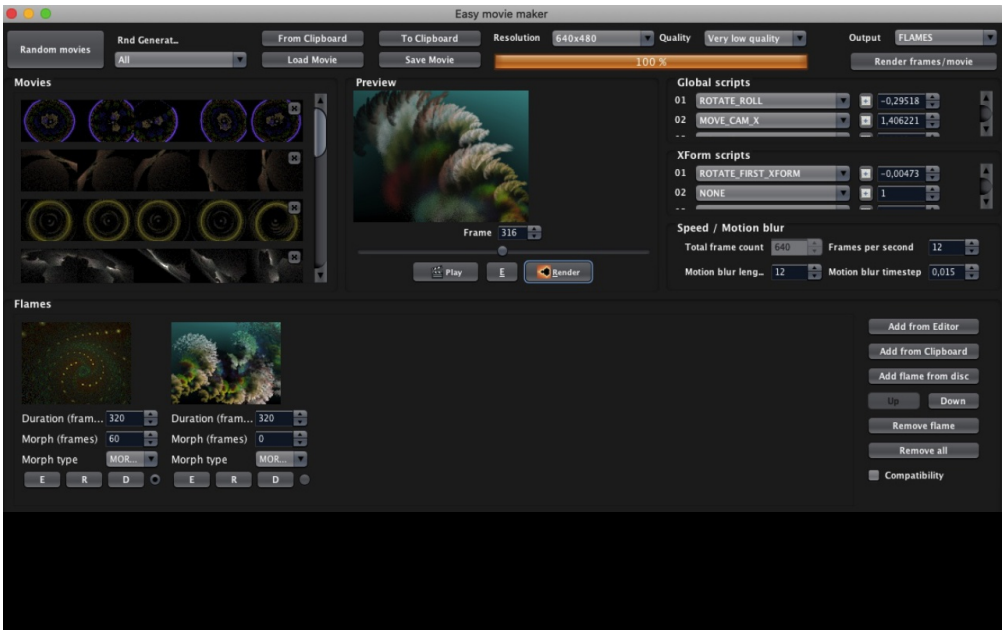


The flame-browser can help you to get an overview of the fractal flames you have created over the time. It can scan a base-folder for flame-files and will display them as thumbnails in reverse chronological order. For a cleaner structure, for each month a separate virtual folder is created and flames are distributed over this folder-structure. Open the Flame Browser by clicking the Flame Browser button on the left side of the preview area or by choosing *Fractal flames: Flame browser* from the *Windows* menu.

On the right side of the window you can execute one of the following actions on the selected flame:

- **To Editor**: load the selected flame into the *Main Editor*
- **To Batch Renderer**: send the selected flame into the *Batch renderer*
- **To Mesh Gen**: send the selected flame into the *Mesh generator*
- **Delete**: delete the selected flame
- **Rename**: rename the selected flame
- **Copy to**: copy the selected flame into another folder
- **Move to**: move the selected flame into another folder

Chapter 7. *Easy Movie Maker* module



With the *Easy Movie Maker* you can create animations consisting of multiple flames. Open the Easy Movie Maker module by clicking the Easy Movie Maker button on the left of the preview area or by selecting *Flame fractals: Easy movie maker* from the *Windows* menu.

It combines three types of animation:

- the keyframe-based animation of a fractal flame itself. You set up these kinds of animations in the Main Editor
- transitions between two fractal flames
- global animation scripts, for example, a movement of the camera

You can even generate random movies, which is an experimental feature.

7.1. *Thumbnail ribbon*

There is a thumbnail ribbon on the left, which holds all currently loaded movies. (This is very similar to the thumbnail ribbon inside Main Editor, which hold all currently loaded flames.) You may select a movie by double-clicking at its thumbnail.

7.2. Functions at the main button bar (at the top)

Here you find global functions at the movie level.

7.2.1. *Random movies*

Using this button you can generate random movies using the selected random-movie generator. This works by generating random-flames and then applying random motions to them. This is currently limited to creating movies containing only one animated flame.

7.2.1.1. *Rnd generator*

These are the options:

- **Transforming bubbles**: generates a random fractal of type *bubbles* and applies random motions at the transformation level to it
- **Transforming duckies**: generates a random fractal of type *duckies* and applies random motions at the transformation level to it
- **Rotating Mandelbrots**: generates a random fractal of type *Mandelbrot* and applies a random rotation-motion to it
- **All**: randomly selects one of the random-movie-generators mentioned above to generate a random movie

7.2.2. *From clipboard*

Loads a movie-file from the clipboard and appends it to the *thumbnail ribbon*.

7.2.3. *From movie*

Loads a movie from a file and appends it to the *thumbnail ribbon*.

7.2.4. *To clipboard*

Transfers the currently selected movie in text-form to the Clipboard.

7.2.5. *Save Movie*

Saves the currently selected movie to a file.

7.2.6. *Resolution*

Here you select the *resolution profile* for rendering the movie.

7.2.7. *Quality*

Here you select the *quality profile* for rendering the movie.

7.2.8. *Output*

These are the output-options for generating a movie:

- **Flames**: Produces a batch of flame-files (one flame file for each frame position), which are rendered later using the Batch renderer. This is the recommended setting for larger projects, because using the Batch renderer for rendering images is the most comfortable way in JWildfire. It supports pausing/resuming and GPU-rendering.
- **PNG-Images**: Produces a batch of .png-images by taking into account the selected resolution- and quality-settings. You need some external software (like *VirtualDub*) to actually create a video-file from this batch of images. Recommended for simple projects, which can be

completely rendered in two hours or less.

- **MP4**: Directly produces an .mp4-movie by also taking into account the selected resolution- and quality-settings. This is recommended for very simple projects or quick test-renders, because there are no options to fine-tune the video-compression-settings (yet).
- **ANB**: Produces an *Animated Brush* in *.anb-format, which is used by painting software *Project Dogwaffle Howler*. Because the output is a brush you use for painting, it is only recommended for small animations. Takes the selected resolution- and quality-settings into account.

7.3. The timeline

At the bottom there is a timeline where you can place individual flames. Each animated fractal flame has the following attributes:

- **Duration (frames)**: number of frames the flame is part of the whole movie
- **Morph (frames)**: number of frames of the transition to the next flame. These sub-frames are part of the whole duration. For example, when you have a duration of 120 frames, and a transition of 60 frames, the transition will start at frame 61
- **Morph type**: a type of transition. These are the options:
  - **Fade**: fades the first flame out, and fades the second flame in. Works for any flame.
  - **Morphs**: tries to actually morph between the flames at the transformation level. Works best on flames that do not differ very much from each other. Often produces interesting results, even if they were not what was expected.

These are the actions you can perform at the animated flame level:

- ***E*** (Edit): transfers the current flame to the Main Editor
- ***R*** (Replace): replaces the current flame with the currently loaded flame in the Main Editor. In combination with the *E* button this allows easy post-editing of a flame.
- ***D*** (Delete): Removes the current flame from the movie

7.3.1. *Add from Editor* button

Appends the currently loaded flame in the Main Editor to the movie.

7.3.2. *Add from Clipboard* button

Appends a flame from the Clipboard to the movie.

7.3.3. *Add flame from disc* button

Loads a flame from a file and appends it to the movie.

7.3.4. *Up* button

Moves the currently selected animated flame one position up (to the left). The currently selected flame is indicated by a filled circular control to the right of the *D* button.

7.3.5. *Down* button

Moves the currently selected animated flame one position down (to the right). The currently selected flame is indicated by a filled circular control to the right of the *D* button.

7.3.6. *Remove flame* button

Removes the currently selected animated flame from the movie. The currently selected flame is indicated by a filled circular control to the right of the *D* button.

7.3.7. *Remove all* button

Removes all flames from the movie.

7.4. *Global (animation) scripts*

Lets you apply up to 12 global animations scripts to the whole movie, i.e. those scripts are applied to each individual flame in the form of a post-effect. They override motions of the same type, in case they were are already applied at the flame-level.

These are the options (which may be combined freely):

- **None**: no motion applied to the slot
- **Rotate Roll/Pitch/Yaw/Bank**: constant rotation of the camera, by modifying the *Roll*- or *Pitch*- or *Yaw*- or *Bank*-parameter of the flame
- **Move CamX/CamY/CamZ**: a movement of the camera into the x-, y- or z-direction in the form of a sine-wave (moving gently back and forth)

By changing the numerical value (to the right of each script slot) you change the strength of the effect.

7.5. *XForm (animation) scripts*

Lets you apply up to 12 flame transformations to the whole movie, i.e. those scripts are applied to each individual flame in the form of a post-effect. They override motions of the same type, in case they were are already applied at the flame-level.

These are the options (which may be combined freely):

- **None**: no motion applied to the slot
- **Rotate Full**: constant rotation of all affine-transforms
- **Rotate First/2nd/3rd/4th/Last XForm**: constant rotation of the specified affine-transform
- **Rotate Final XForm**: constant rotation of the final-affine-transform
- **Rotate Post Full**: constant rotation of all post-affine-transforms
- **Rotate Post First/2nd/3rd/4th/Last XForm**: constant rotation of the specified post-affine-transform

- **Rotate Post Final XForm**: constant rotation of the post-final-affine-transform

These animations are only applied when a flame has the corresponding transformations. For example, selecting the animation script *Rotate Final XForm* on a flame without a final transform will have no effect, but will also not cause an error.

By changing the numerical value (to the right of each script slot) you may change the strength of the effect.

7.6. (animation)Speed/Motion blur

Here you can specify the overall animation speed (*frames per second*) and set up motion blur for the whole movie.

7.6.1. Total frame count

Displays the total frame count which is calculated according to the individual settings of the animated flames.

7.6.2. Frames per second

Global fps-setting of the movie.

7.6.3. Motion blur length

Global motion blur length of the whole movie.

7.6.4. Motion blur time step

Global Motion blur time step of the whole movie.

7.7. Preview

You can preview the animation at any time by hitting the *Play* button in the middle of the window.

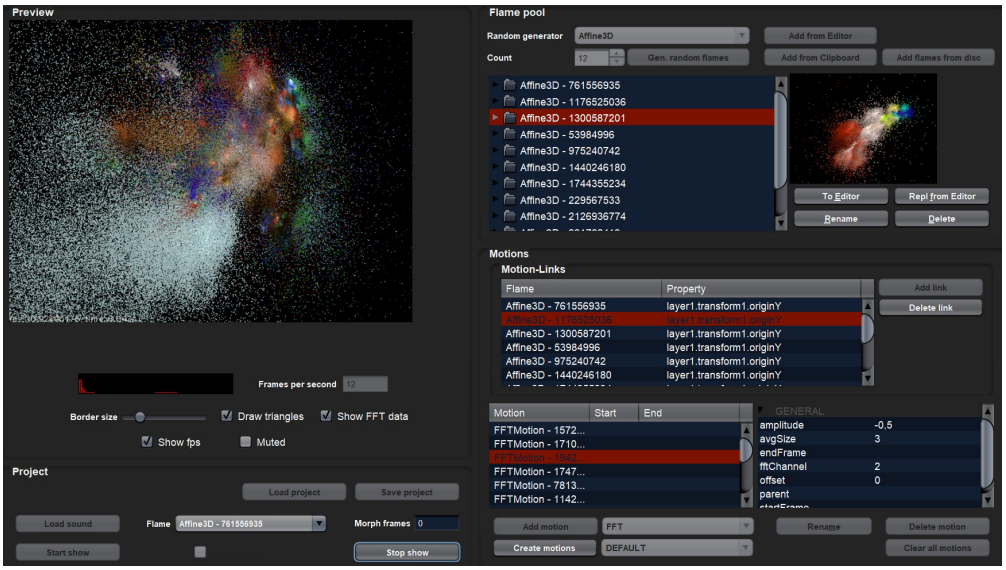
7.7.1. *Frame*

Displays the frame position at which the preview is displayed.

7.7.2. *E* (Edit) button

Sends the currently displayed flame (at the selected frame position) to the editor. This might be especially useful when you are morphing between different flames and like the result at a certain frame position and want to work with the fractal flame at the current morphing-state.

Chapter 8. *Dancing flames* module



The dancing-flames-module is another experimental feature of JWildfire. Open the Dancing flames module by selecting *Fractal flames: Dancing flames* from the *Windows* menu.

It allows you to create sound-synchronized animations of fractal flames in a very sophisticated way.

What can you do:

- animate anything! (i.e., any property of a flame which can be accessed in the flame-editor can be animated)
- synchronize motion with sound (a frequency spectrum computed by a Fast Fourier Transform (FFT), a computer algorithm commonly used to analyze sounds, can be used to link motion with "beat")
- create motion curves (amplitude over time) for certain properties by using a spline editor or apply a predefined motion (basing on a certain formula, like "Sawtooth")
- group motions together (for example, have a spline-motion-curve which is slightly altered by beat")
- watch the motion in realtime and record it to be processed again to create frames at accurate frame rate
- use as many fractal flames as you like in your movie and switch between them in real time during recording
- load and save projects

8.1. The basic concepts

The basic concept of the animations in the dancing-flame-module is both very versatile and very simple and contains four building blocks:

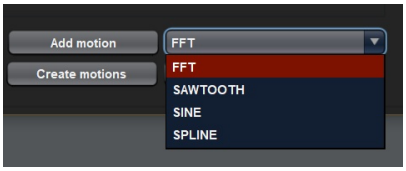
- **motions**
- **flame properties**

- **motion links**
- **motion hierarchies**.

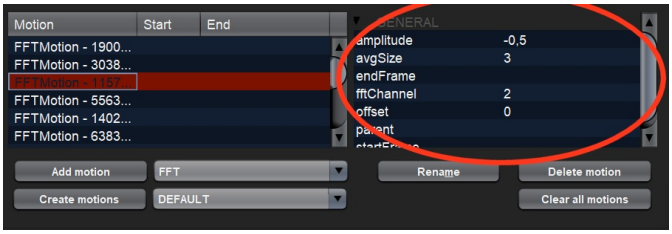
8.1.1. Motions

You can define any number of motions, for example:

- a rotation with frequency x
- "beat" data of a certain channel from the FFT
- a manually created motion curve using the spline-editor



Each motion has special parameters which can be accessed through a property table. There are some parameters which can found by all motion-types.

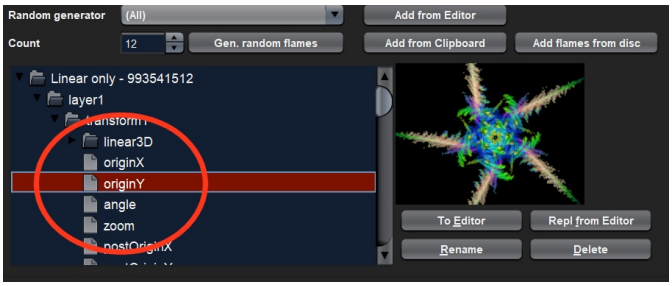


8.1.2. Flame properties

Each flame has many properties. Most of them can be easily be animated, for example:

- camera angle
- zoom factor of transform 2
- Julia index of the Julian variation in the final transform.

You don't need to know all of these, they can be accessed through a tree-view in the graphical interface.



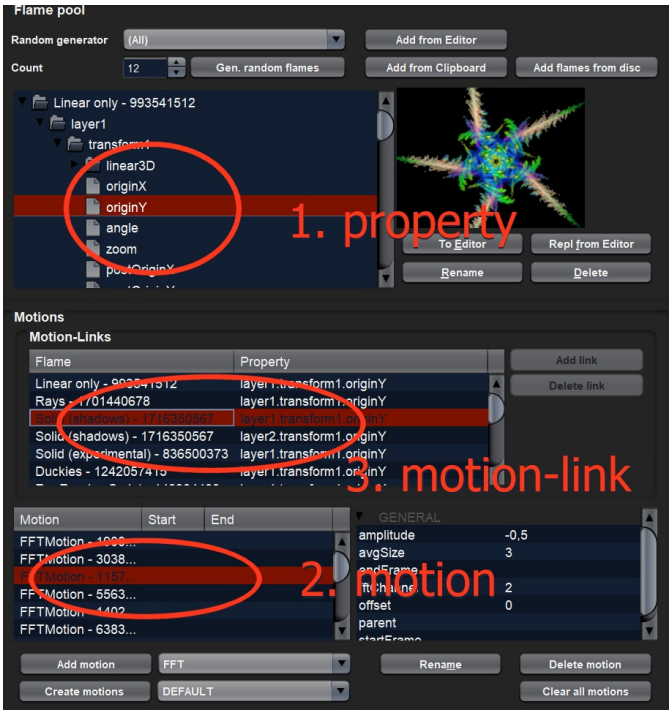
Each fractal flame has its own top-level-folder in this view. There are sub-folders for:

- each **transform**
- each **final transform**

- **gradient** parameters.

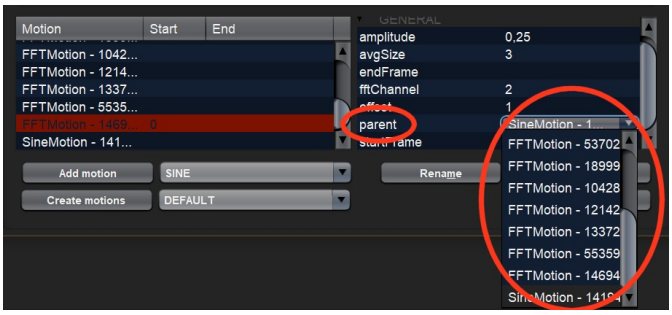
8.1.3. Motion links

To actually animate a fractal flame we must connect properties to motions. Any motion can be connected with a number of fractal properties from any flame of the project.



8.1.4. Motion hierarchies

In certain cases it may make sense to group motions together. For example, to have a "rougher" motion (like a motion curve which describes a basic rotation) where a "small" motion (like "beat") is added. To achieve this, you may choose a parent for any motion.



A motion having a parent can not explicitly be linked to a flame property (because it is already linked implicitly through its parent).

8.2. Basic workflow

In the following sub-section a basic workflow is outlined.

8.2.1. Creating a flame-pool

The flame-pool is a collection of flames which belong to a dancing-flames-project and can all be animated into this project. Simple projects start with only one flame in the pool, but there is no limit and even if you have many flames in the pool, not all have to be used in the video.

There a several ways to add flames to the pool:

- Generate a batch using a random generator,
- import the currently load flame from the main editor,
- add a flame from the clipboard or
- load a batch from hard-disc.

8.2.2. Adding sound (mp3)

Sound isn't actually required to create a dancing flame project, but most projects will benefit from the use of sound. Currently, there can only be one mp3-file for the whole project (if you want to use more than one sound file, you have to merge them and load them as single file). To add a sound file, just press the *Load sound* button and choose a mp3 file.

Please note that the loading may take a while as sound is actually processed and analyzed in the background.

8.2.3. Adjusting the project's FPS (frames per second)

The FPS setting of the project is one of the most important parameters; it affects how smooth your animation will be and how many frames (single images) you will have to generate in order to create the final animation. The smoother your animation, the more data that needs to be generated and the larger the movie will become. In my personal experience a fps value of 25 (i.e., 25 images shown per 1 second of the animation) is a very good compromise.

8.2.4. Previewing the show

To preview the current animation press the *Start Show* button. This will display a preview animation and play back the sound.

In many cases the preview will not be able to follow your FPS setting. This is not a problem because the speed of the realtime-preview is not the speed at which the final animation will be created. The final animation is always created at the true fps setting, even for slow machines.

8.2.5. Adjusting the speed of the realtime-preview

You can improve the speed of the realtime-preview by decreasing its size. You can do this in real-time by modifying the *border size* slider.

8.2.6. Switching between flames during the show

You may switch between the flames of your flame pool during the show at any time. This is

currently the only interaction which is actually required to be done during the show.

This kind of interaction is absolutely intended, and will give your animation a more "spontaneous" or "looking-alive" behavior. Just swap flames when you feel it. To switch between flames just select them from the list-box in the project area during the show.

8.2.7. Recording a show

To record a show just enable the *Record show* checkbox and start the show. Now any user action (currently only switching between flames) is recorded according to your fps setting.

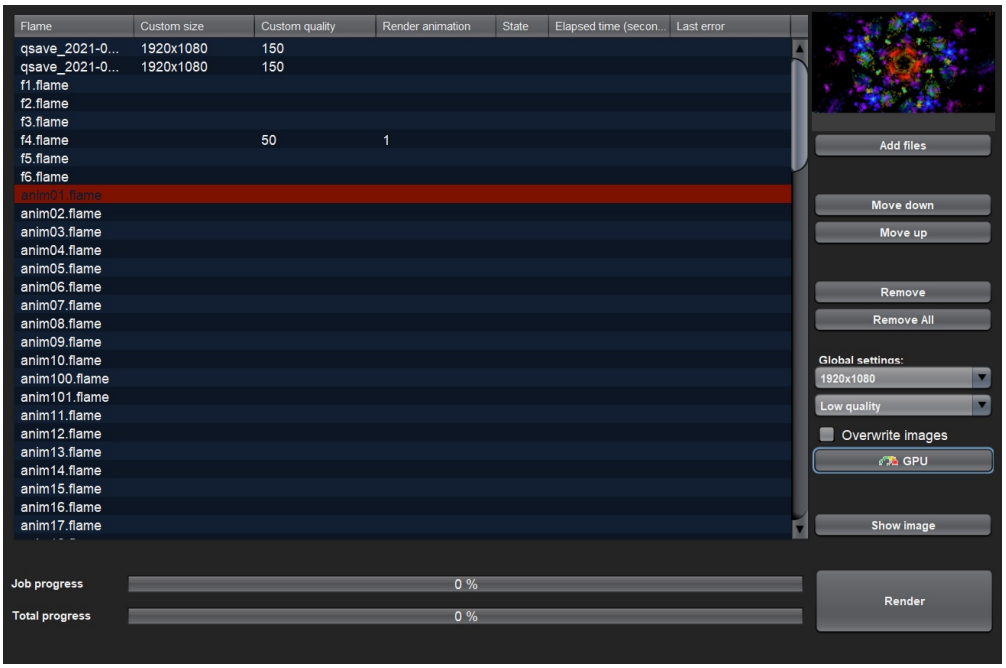
After finishing the show (by pressing the *Stop show* button) you will be prompted for an output path. Just choose a directory and specify a base-name for the flame files to be generated.

Now JWildfire recalculates the show again at the desired FPS setting and produces the flame-files which have to be rendered later to create the movie. Depending on the length of your show and the FPS setting, this may take a while and produce lots of files. Please note that it is very important not to interrupt this process in order to create very accurate timing. It’s best to leave your computer alone, especially if you have a rather slow system.

8.2.8. Rendering the flames

The project, until now, was free from any output resolution settings and only fractal flames were generated. You can render them using any resolution and quality settings later. The recommended way is to use the Batch renderer to perform final rendering.

Chapter 9. *Batch renderer* module



The batch renderer is designed to render fractal flames in the background (for example overnight). Therefore it is well suited to render frames for animations or images with large resolution, which take a while to render. Open the Batch renderer module by selecting *Fractal flames: Batch renderer* from the *Windows* menu.

A special feature of the *Batch renderer* is the area where it places the finished images. **All images rendered by the *Batch renderer* are placed in the same folder as the corresponding flame-file**. This makes it easy for the *Batch renderer* to recognize images that have already been rendered (they are then skipped by default). This also makes it easy to pause and resume rendering.

9.1. *Render jobs* table

All flames are displayed in a large table which has some editable columns:

- **custom size**: here you can enter a custom resolution when it differs from the main resolution selected in this window. The format is **<width>x<height>**, for example: **1920x1080**.
- **custom quality**: here you can enter a custom quality when it differs from the main quality selected in this window. The format is a numerical value, for example: **500** for a decent quality level.
- **render animation**: if you enter "1" here, JWildfire creates a whole .mp4-animation instead of a single .png-image. You can cancel and continue this at the frame level of the animation. When cancelling the rendering of an mp4-animation, you can continue it later at the frame you cancelled it.

9.2. *Add files* button

Here you can add a single flame or any number of flame files at once to the *render jobs* table.

9.3. *Move down* button

Flames are processed in the order they are displayed in the *render jobs* table. By using this button you can move the currently selected flame down one position.

9.4. *Move up* button

By using this button you can move the currently selected flame up one position.

9.5. *Remove* button

Remove the currently selected flame from the *render jobs* table. This does not delete the flame file.

9.6. *Remove All* button

Removes all flames from the *render jobs* table. This does not delete any flame files.

9.7. *Settings: Resolution profile*

Select the Resolution profile to be used for rendering each entry the of *render jobs* table. If you want to use a different resolution for certain flames, you may override this setting by editing the column **custom size** for the individual flames.

9.8. *Settings: Quality profile*

Select the Quality profile to be used for rendering each entry the of *render jobs* table. If you want to use a different quality setting for certain flames, you may override this setting by editing the column **custom quality** for the individual flames.

9.9. *Overwrite images* checkbox

Per default, the *Batch renderer* will not re-render already rendered images. You can change this by activating this checkbox.

9.10. *GPU* toggle button

This toggle is only enabled when *GPU rendering* is configured on your system. When you activate this button, all images are rendered using the *GPU renderer*. But please be aware that this may not work well on some kind of flames and does not support all possible features. For example, sub-flames are not supported. So, you should test it with a single flame before GPU-rendering a longer batch.

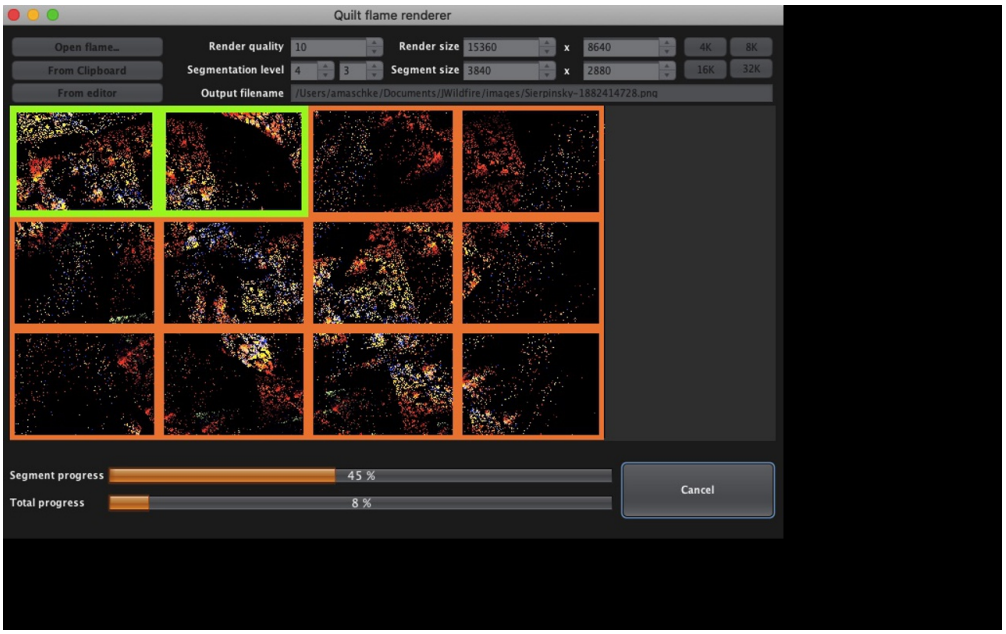
9.11. *Show image* button

Loads and displays the corresponding image of the currently selected flame, if already rendered.

9.12. *Render* button

Starts the rendering batch.

Chapter 10. *Quilt Renderer* module



The *Quilt Renderer* allows you to render flames of nearly unlimited size. This works by splitting the whole image into tiles. Open the Quilt renderer module by selecting *Fractal Flames: Quilt renderer* from the *Windows* menu.

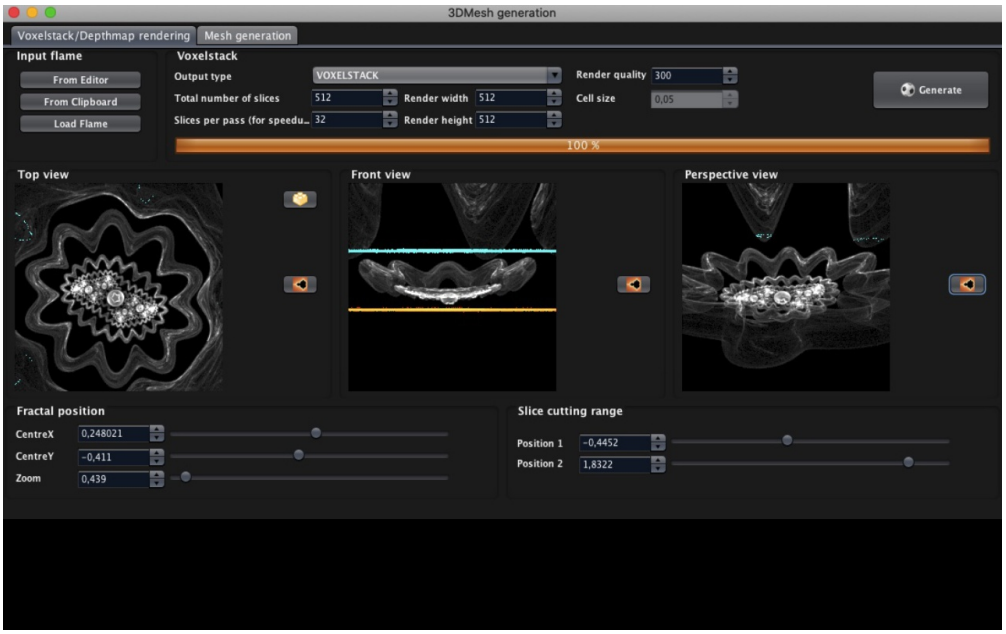
You may freely specify how the tiles are generated, for example, splitting the image in 3x2 tiles or 8 vertical stripes.

Each of the rendered tiles is a valid image. So, even if JWildfire failed to assemble the tiles into one final image (due to memory restrictions), you could perform this step using external image-processing-software, which is more optimized to handle large images. Images with a size up to 32K (30720x17280) were tested without any problems so far.

You can cancel/resume the rendering-process at tile-level.

Please note: Due to the nature of the fractal-flame-algorithm, the rendering of each tile may take as long as rendering the entire image. So, this method is not efficient, but it is effective. A 16K render on a modern computer can take about 12 hours, but you will usually get an awesome result with endless details.

Chapter 11. *Mesh generator* module



Using the *mesh generator* you can turn your fractals into *polygonal meshes* or *point clouds*. Open the Mesh generator module by selecting *Fractal flames: Mesh generator* from the *Windows* menu.

11.1. Generating *Polygonal meshes* by slicing the fractal in *Voxel stacks*

The creation of solid meshes will not work well on all types of fractals and will require some patience and experience to get nice results. But once you have generated a nice mesh object, you may use it in any 3D software or even try to 3D print it.

In JWildfire the process of generating *Polygonal meshes* requires two steps:

- creating a number of slices through the fractal. (These slices are called *Voxel stacks* because they have both a fixed resolution (like picture elements or pixels in a image) and a certain depth. So each slice contains pixels with a depth, which are three-dimensional pixels, called Volume elements, or *Voxels*. And these slices stacked together build a representation of the object, hence the name *Voxel stack*.)
- building a *polygonal mesh* from all these slices using the popular *Marching Cubes Algorithm*.

11.1.1. *Total number of slices*

Sets the number of slices to be rendered which represents the voxel resolution in the z-direction.

11.1.2. *Render width*

Sets the render resolution in the x-direction which represents the voxel resolution in the x-direction.

11.1.3. *Render height*

Sets the render resolution in the y-direction which represents the voxel resolution in the y-direction.

11.1.4. *Slices per pass*

With this setting you can improve the speed of the scanning process in a significant way but at a cost to consumed memory. If you have enough memory you can even set this to the *Total number of slices* to create all slices at once.

Please note that these three values should match the shape of the desired object. For example, if you have a very thin but very tall object, you may want a much higher resolution into z-direction.

11.1.5. *Generate* (*Voxel stack*)

When the *Output type* is set to **Voxelstack**, this will generate a batch of grayscale images which represent the slices through the fractal (similar to the scans made by a computer tomography).

11.1.6. *Mesh generation*

Once you have finished creating a *Voxel stack*, you may create the final mesh with the *Mesh generation* tab.

11.1.6.1. *From renderer*

Loads the previously rendered *Voxel stack* without the need to select it in a file dialog.

11.1.6.2. *Load sequence*

Loads a previously rendered *Voxel stack* from hard disc.

11.1.6.3. *Image downsample*

Downsample the slices by this factor to reduce noise. Reduces resolution, but increases quality of the mesh.

11.1.6.4. *Spatial filter radius*

Filter-radius used for downsampling.

11.1.6.5. *Brightness threshold*

Brightness level at which the object is actually seen. Increasing this value may also help to reduce background noise, but can also lead to loss of details.

11.1.6.6. *Image step*

With values higher than 1 you can skip slices.

11.1.6.7. *Post smoothing*

When you enable this checkbox you can apply a post-smoothing effect to the generated mesh. This is an experimental feature and is only useful on a certain types of meshes.

11.1.6.8. *Generate* (*Polygonal mesh*)

Using this button you generate the final *polygonal mesh* from the currently loaded *Voxel stack*.

11.2. Generating *Point clouds*

Generating *Point clouds* it a little bit more complex internally, but much simpler from the user's point of view. You have to only set up two parameters (*Cell size* and *Render quality*) and then the *Point cloud* can be generated in one step.

11.2.1. Cell size

This is the most important parameter for generating *Point clouds*. It affects both memory consumption and resolution of the final result. It represents the minimum absolute voxel size which can be filled with information. The smaller the *cell size*, the more voxels are generated and the more processing time is required. Voxels are generated at two levels:

- **raw voxels**: represent a result of a fractal iteration. Are generated at a scale of billions.
- **final voxels**: represent a kind of *average* of all *raw voxels* in a cell of *cell size*.

The smaller the *cell size*, the more calculations are needed to create final voxels, and the more memory is required to temporarily hold *raw voxels*.

Please note that the influence of this parameter is cubic. For example, halving it causes an increase of effort by a factor 2 in 3 dimensions, reaching a total factor of 2 X 2 X 2 = 8. As a comparison: it is like increasing screen resolution, but into three directions instead of two.

11.2.2. *Generate* (*Point cloud*)

When the *Output type* is set to **Point cloud**, this will generate a *Point cloud* in **.ply**-format. You may process/view such files for example in *Houdini* or *Meshlab*.

11.3. General options

11.3.1. *Output type*

There are two options:

- **Voxel stack**: create a *Voxel stack* as the first step of generating a *polygonal mesh*
- **Pointcloud**: create a *point cloud* in one step

11.3.2. *From Editor*

Transfers the currently edited fractal flame from the *Main Editor* to the *Mesh generator*.

11.3.3. *From Clipboard*

Loads a fractal flame from the *Clipboard* into the *Mesh generator*.

11.3.4. *Load Flame*

Loads a fractal flame from an external file into the *Mesh generator*.

11.3.5. *Fractal position*

This area contains controls to set the position and size of the fractal flame for the scanning process. Any portions which are outside of the preview will not be scanned and therefore will not be part of the generated mesh.

11.3.5.1. *CentreX*

Moves the fractal in the x-direction.

11.3.5.2. *CentreY*

Moves the fractal in the y-direction.

11.3.5.3. *Zoom*

Scales the fractal.

11.3.6. *Slice cutting range*

Here you set the start- and end-position of the scanning process in the z-direction. Any portions of the fractal which are outside of this range will not be scanned and therefore will not be part of the generated mesh.

11.3.6.1. *Position 1*

Sets the start position in the z-direction for the scanning/slicing process.

11.3.6.2. *Position 2*

Sets the end position in the z-direction for the scanning/slicing process.

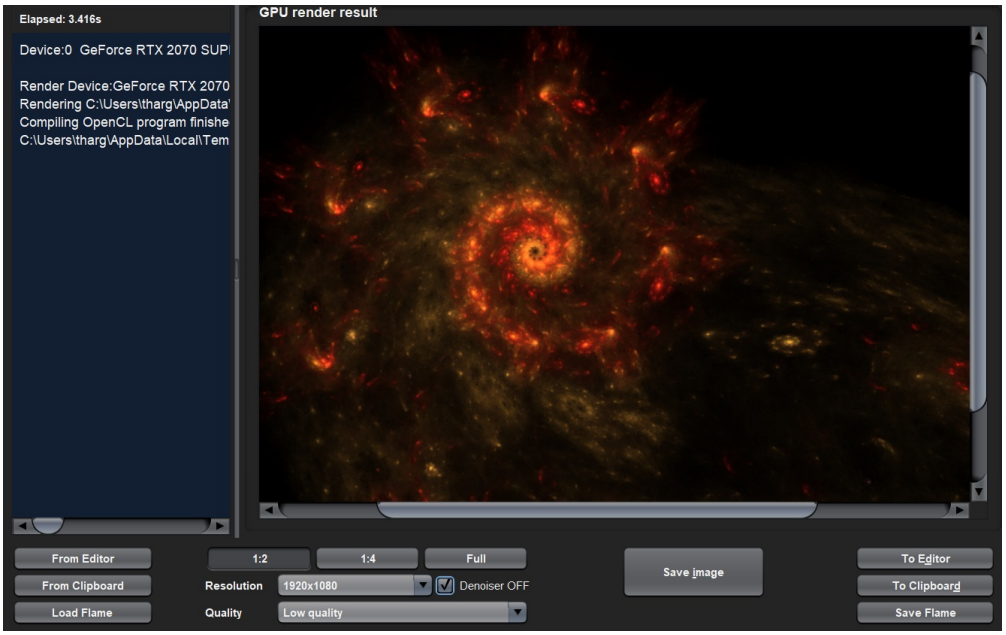
Hint: the order of *Position 1* and *Position 2* does not matter, you may have *Position 1* at the top or at the bottom depending on your personal preference.

11.3.7. *Render quality*

This sets the render quality for scanning the fractal. Optimum values highly depend on the type of

fractal. Solid fractals usually need a high number of samples to be really clean. Values in the range of 300 to 2000 are typical.

Chapter 12. *GPU render* module



GPU rendering in JWildfire is enabled by integrating FAEngine, a GPU-rendering-engine created by Steven Brodhead. In JWildfire 7.0, GPU-rendering was re-implemented in a fundamental way:

- It is still using FAEngine as GPU-gateway, but most of the GPU-code is actually JWildfire-specific
- Most of the actual GPU code is provided by JWildfire during runtime and compiled and executed during FAEngine on the GPU It would be impossible to create a variation set which contains all GPU-supported variations. They would be too many, and compilation at the device would fail or even crash the whole system. Therefore, JWildfire always sends only the code to FAEngine which is actually required. This makes the rendering startup for each flame a little bit slower (and does not allow for a realtime-display), but actually works very good and reliable)
- uses a special binary (FARenderJWF) and and custom kernel which already provides many of JWildfire's features
- only this gives actual control over the behaviour of variations and allows us to implement features which are not supported naturally by FAEngine

12.1. Currently supported features:

- support for about 660 variations (of a total of 850)
- motion blur
- some additional color balancing options
- 3d-affine-transformations ("XY"-, "YZ"-, "ZX"-buttons)
- weighting fields
- custom variations

12.2. Currently not supported features:

- layers
- local gamma effects
- solid rendering

12.3. Supported platform

The new solution is using CUDA instead of OpenCl, so AMD devices are no more supported. I had to choose between supporting more devices or supporting more possible features (and a better developer speed), therefore I switched to CUDA.

So, GPU-rendering is currently only supported on Windows 10 and requires a CUDA-capable graphics card.

For example: NVIDIA 1060 or better will work very well. This card is also the minimum requirement which is supported from my side. So, when you have an card which is older, it may also work, but when not, I can not help you with it.

12.4. Note about Performance

Please note: the more features you use in your flame, the more code is created, compiled and executed. Especially, weighting-fields are very expensive in comparison to other features. At higher resolution the GPU render will always be faster - MUCH faster. But, it is possible, that a small preview on GPU takes more time than a CPU render. This is due to the complicated process to invoke a GPU render (while a CPU render can start instantly at every time).

12.5. Installation/Automatic Self-test/Troubleshooting

In comparison to previous versions there is no separate installation required nor possible. Everything, which is required from the side of JWildfire, is already included and installed. (What is not included, is for example the driver for your graphics card.)

A self-test is performed at the startup of JWildfire. When the self-test passed, you will see two "GPU"-buttons in the left of the preview area inside the main editor.

When you do not see the buttons, but meet the system requirements, please check if you have the latest NVIDIA drivers for your device. This actually seems to solve about 90% of the problems I have seen so far.

12.6. Rendering flames using the GPU

12.6.1. GPU renderer module

There is a separate "GPU renderer"-window to explore GPU rendering. This window works similar to the Interactive renderer; you may import flames from the editor or load them from your hard

disk.

In this window, you will be able to see the exported GPU-code. And you will get the most detailled information messages, compared to all other places where also GPU-rendering is supported

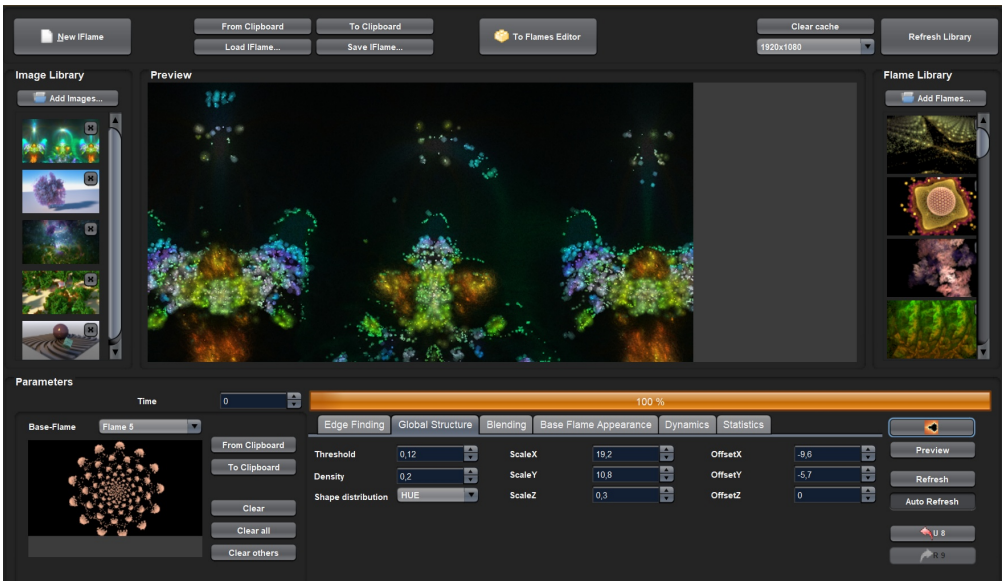
12.6.2. Main editor

When you have activated the GPU-mode inside the main-editor, all rendering is done automatically using GPU. This applies also to the rendered preview (the button at the lower right corner of the preview-area), but not to the real-time-preview which is used for editing.

12.6.3. Batch rendering

When GPU-rendering is available at your system, a "GPU" button will appear in the Batch-renderer. You may use it to render whole batches using the GPU. But, before you render a huge sequence, you should be sure that the flames do not use features which are not supported by GPU-rendering (for example: sub-flames). You may quickly do so by transferring one of the flames to the "GPU renderer" window and see if it works.

Chapter 13. *IFlames* module



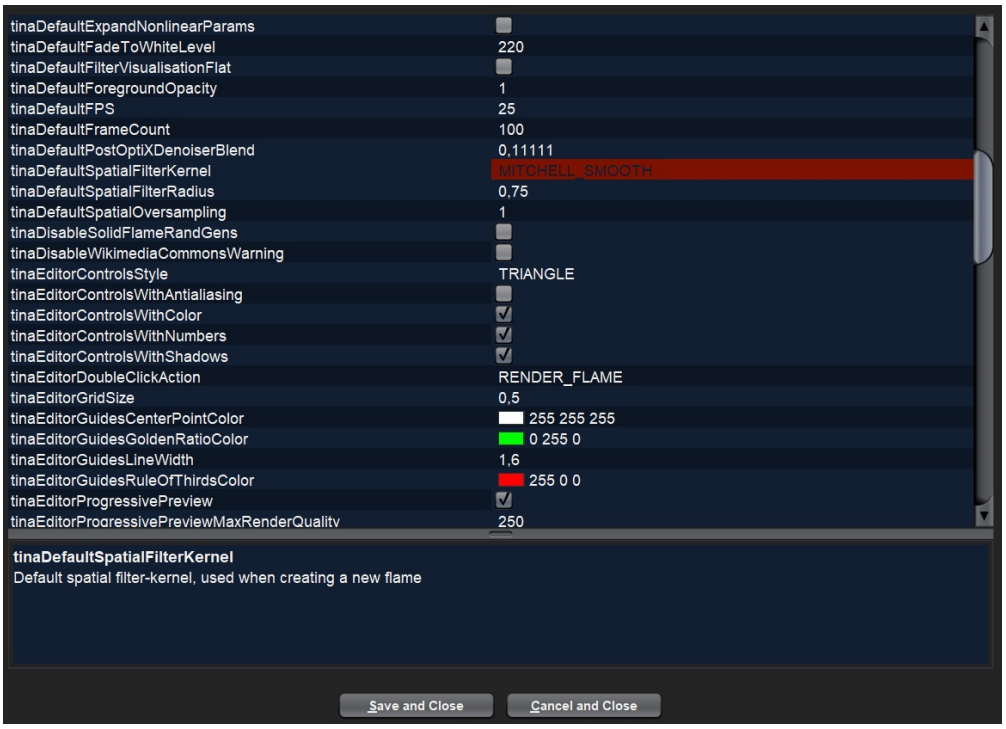
This is an experimental application which combines particle simulation and fractal flames.

It requires a rather powerful computer, especially when you want to animate the iflames.

There are about 350 parameters to change, many of them may be animated:

- **Edge-finding**: controls where to place base-shapes
- **Global structure**: controls the global structure of the resulting iflame
- **Blending**: controls coloring
- **Dynamics**: controls global settings affecting dynamics aspects, like gravity
- **Base Flame Appearance**: controls the base-shapes inside an iflame (you may have up to 6 different base-shapes)
- **Size and orientation**: The size and orientation of the base-shapes
- **Distribution**: how to place which base-shape
- **Blending**: local coloring
- **Mutations**: controls how to create mutations from the base-shapes
- **Dynamics**: dynamic properties (like speed in a certain direction) of base-shapes

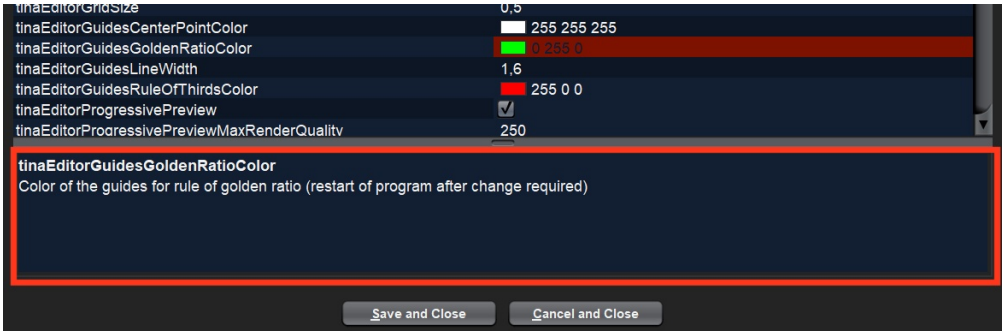
Chapter 14. *Preferences* window



In the *Preferences*-window you may customize JWildfire to your needs. Open it by choosing *Preferences* from the *Settings* menu.

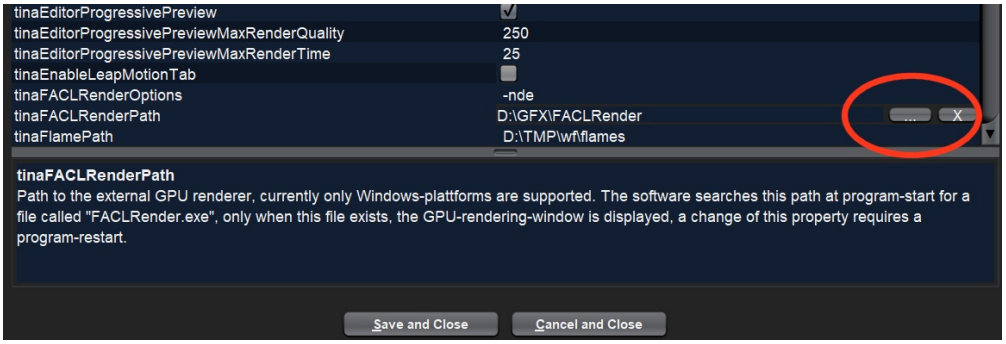
The customizable properties are key-value pairs which are displayed in one large table. You may edit a property by double-clicking the value in the 2nd column of the table.

Each property has a small description which is displayed at the bottom of the window.



Depending on the type of parameters, there are special parameter-editors available. You reach them by double-clicking the value in the 2nd column of the table. For example:

- for color-parameters a color-selection-dialog is available.
- for folder-parameters (for example, to select the gradient-folder) you can both enter the folder manually or select the folder using folder-dialog.



Some properties may be changed "on the fly", while others will need a restart of JWildfire in order to show an effect. This is usually mentioned in the description of the property.

Chapter 15. The JWildfire CLI (command-line-interface)

Sometimes it is much more convenient to automate certain tasks and let them run in the background or overnight, rather than performing them manually. This is where the JWildfire CLI steps in. The CLI allows to execute certain tasks from a command line or script, without any user interaction.

15.1. How to set up the CLI

To use the CLI you need a valid JWildfire installation and a JDK (Java development kit) in version 8 or higher.

Please note: The complete JWildfire-installation is required as soon as more advanced function like GPU-rendering are involved. When you only use basic features, it may be sufficient to place the j-wildfire.jar in the Java classpath.

In any case, you should remember/locate the path where the j-wildfire.jar of your JWildfire installation is located. You will need this location when executing the CLI.

15.1.1. Basic example: create a random flame

Open a terminal and enter the folder of a typical JWildfire installation. In this example the folder is called "jwildfire-7.30". Execute the command:

```
java -cp ./jwildfire-7.30/lib/j-wildfire.jar org.jwildfire.cli.RandomFlameGenerator
```

It should both generate a flame and a rendered preview of this flame in the current folder.

15.1.2. Avalailable EndPoints

In the example above you have called the EndPoint to generate random-flames. Currently, there are the following EndPoints:

- FlameRenderer
- RandomFlameGenerator

15.1.3. EndPoint Options

To get a list of options, you can call the End Point with the --help -Parameter

```
java -cp ./jwildfire-7.30/lib/j-wildfire.jar org.jwildfire.cli.RandomFlameGenerator
--help
```

15.2. FlameRenderer-EndPoint: Rendering flames using the CLI

This EndPoint supports both CPU- and GPU-rendering. Example:

```
java -cp ./jwildfire-7.30/lib/j-wildfire.jar org.jwildfire.cli.FlameRenderer -f
./aacae305-d08a-4515-abaa-ca5c668f452b.flame -w 800 -h 600 -q 50 -gpu -dn OPTIX
```

Parameters:

```
usage: FlameRenderer
 -dn,--denoiser <arg>       ai-based post-denoiser
 -f,--flame <arg>           flame path
 -fdn,--forceDenoiser       force ai-based post-denoiser
 -gpu,--useGPU              render using GPU
 -h,--renderHeight <arg>    render height
 -q,--renderQuality <arg>   render quality
 -w,--renderWidth <arg>     render width
```

15.3. RandomFlameGenerator-EndPoint: Creating random flames using the CLI

EndPoint to generate batches of random flames. Creates both a random flame and a preview image. This preview could be used by AI for further processing, like measuring the uniquess or quality of the image. This EndPoint supports both CPU- and GPU-rendering. Example:

```
java -cp ./jwildfire-7.30/lib/j-wildfire.jar org.jwildfire.cli.RandomFlameGenerator -w
800 -h 600 -q 40 -bc 3 -rgflame "Tile Ball" -rgwfield None -rgsymm None
```

Parameters:

```
usage: RandomFlameGenerator
 -bc,--batchCount <arg>           number of generated flames
 -dn,--denoiser <arg>             ai-based post-denoiser
 -fdn,--forceDenoiser             force ai-based post-denoiser
 -gpu,--useGPU                    render using GPU
 -h,--renderHeight <arg>          render height
 -lrgflame,--listRndGenFlame      list the available
                                  random-flame-generators
 -lrggrad,--listRndGenGradient    list the available
                                  random-gradient-generators
 -lrgsymm,--listRndGenSymmetry    list the available
                                  random-symmetry-generators
 -lrgwfield,--listRndGenWField    list the available
                                  random-weighting-field-generators
 -q,--renderQuality <arg>         render quality
 -rgflame,--rndGenFlame <arg>     random-flame-generator to use
 -rggrad,--rndGenGradient <arg>   random-gradient-generator to use
 -rgsymm,--rndGenSymmetry <arg>   random-symmetry-generator to use
 -rgwfield,--rndGenWField <arg>   random-weighting-field-generator to use
 -w,--renderWidth <arg>           render width
```

Chapter 16. Platform-specific topics (Windows, Mac, Steam, ...)

In this section some platform-specific topics are covered. Even though JWildfire is widely platform-independent, there are features which are not available on certain platforms. Also discussed are options for customizing the startup of the application which must be specified in different ways, depending on the platform.

16.1. Startup options

In some cases it is necessary to set up some startup parameters for the application. Some common examples are:

- increase the maximum amount of usable memory (to be able to render in higher resolutions)
- increase the overall UI-scale (which may be too small on 4K monitors))
- fix some UI-glitches

The location of these parameters differs depending on the platform.

16.1.1. Steam Edition

Currently, you must edit these parameters after each update. To simplify this, there is a shortcut inside JWildfire to open the correct file. It is located under *Settings → Startup settings (Steam)*.

When you click at this menu-item, it should open the file **<SteamApps>\common\JWildfire\app\JWildfire.cfg** (When this does not work due to missing permissions, you may also open and edit this file manually).

Please see the sub-section *Example (for both _Steam* and *Mac* version)_ for an example.

16.1.2. App Store Edition

Currently, you must edit these parameters after each update. These parameters are inside a file named JWildfire.cfg inside the application folder.

To edit this file you must open the *Finder* and locate the application *JWildfire* inside the *programs* folder. Right-click on the entry *JWildfire* and select the option **Show Contents**. This will show the files which belong to the application. Navigate to the sub-folder *Contents/app* and you should see the file *JWildfire.cfg* inside this folder. You must edit this file in order to change the startup-parameters.

Please see the next sub-section *Example (for both _Steam* and *Mac* version)_ for an example.

16.1.2.1. Example (for both *Steam* and *Mac* version)

Once you edit the file *JWildfire.cfg*, it works the same for both the *Steam* and the *Mac* platform.

The file contains different sections, the *startup JVM parameters* for JWildfire must be placed under the section **[JavaOptions]**.

Example file:

```
[Application]
app.name=JWildfire
app.version=6.40
...

[JavaOptions]
-splash:app/splash.png
-Xmx8G

[ArgOptions]
```

Here you see that under **[JavaOptions]** there is an entry **-Xmx8G**. This specifies the maximum usable memory. The default is **8G**, which means 8 gigabytes of memory. If you want to increase this to 32 GB, you just would change the line to:

```
...
[JavaOptions]
-splash:app/splash.png
-Xmx32G
...
```

Please note that JWildfire will not start when this value exceeds the actual free memory at the time you start JWildfire

16.2. Restrictions

16.2.1. Steam Edition (Windows only)

There are no restrictions. The Windows-based version of JWildfire currently has the richest feature set.

16.2.2. App Store Edition (Mac only)

On the Mac platform there are the following restrictions:

- the JWildfire-specific file-dialogs can not be used due to security-related requirements, instead the macOS-native dialogs are used. This is not bad, because it provides the typical behavior under macOS, but there are some features that you will only find in JWildfire-specific file dialogs. These features are not available on the Mac platform.
- AI-based denoisers are not supported.
- GPU rendering is not supported.

- LeapMotion-controller-support is not available.

16.2.3. Linux Version

There is no special Linux version, but the free version from the official website https://jwildfire.overwhale.com does also work on Linux. To make it work, you must install a Java Virtual Machine (JVM).

Then there are the following restrictions:

- AI-based denoisers are not supported.
- GPU rendering is not supported.
- LeapMotion-controller-support has not been tested, but probably will not work.

Chapter 17. FAQ (frequently asked questions)

Here are answers to some popular questions which I get very often. They did not fit well into the other chapters, so I collected them in this section.

17.1. JWildfire does not use all my memory to render images at large resolutions

Due to its nature, JWildfire uses a fixed setting for the maximum memory it can use. The default setting for this value is 8 GB. When this limit is too low for you to render images at your desired resolution, you can increase this value. To achieve this, you have to modify the JVM parameter **Xmx**.

For example, to increase the value to 32 GB, you should change the **Xmx**-value in the *JWildfire.cfg* like this:

```
--Xmx32G
```

You can also decrease the value, if you want to use less memory.

Please note, that JWildfire will not start when this value exceeds the actual memory available.

It is not recommended to use higher than necessary values as this will degrade performance slightly.

Please see the sub-section Changing JVM startup parameters to see how this file can be edited.

17.2. The UI scale is microscopic (after an update)

Currently, the UI-scale must be set after each update. You can increase the UI scale by changing the startup parameter **sun.java2d.uiScale**, for example:

```
-Dsun.java2d.uiScale=1.65
```

Please see the sub-section Changing JVM startup parameters to see how this can be achieved.

17.3. Are 4k renders possible?

Yes.

You can literally render in any resolution, also 8k:

- by rendering a single image at once. The maximum render size is limited only by your available memory. (For really huge images, you may need patience too.)
- by using the *Quilt-renderer*-module you may render really huge images in smaller parts, which are then assembled together. The maximum render size is also limited by memory. But often, you will get other problems before reaching the memory limit in JWildfire. For example, some image files can be very big (I have tested with sizes up to 1 GB), and it is hard to find software that can process such large images.

17.4. Can I render images with removed/transparent background?

Yes, you can.

See the properties *Background transparency* and *Fg opacity* at the *Coloring*-tab.

You can also preview the rendering with transparent background. Set the *Display/hide transparency* - button at the preview-area.

17.5. How to save a flame as an image?

You can not directly save a flame as an image. You must *render* a flame in order to generate an image. There are multiple options you can specify for rendering (for example: the output resolution) which may allow you to get different images out of one fractal flame.

17.6. Is it possible to render flames at higher resolutions than inside JWildfire?

There are no restrictions regarding render size which must be resolved by an external application for rendering. The only limitation is memory. See Are 4k renders possible? for more information.

17.7. How to fix glitches/weird behavior of the UI

Some users have reported weird behavior of the UI:

- for example, one window overwriting the content of another
- or windows following the mouse pointer

It seems that this is caused by some misconfiguration of *DirectDraw*-acceleration of Java2D (maybe a driver issue).

The known solution is to switch to *OpenGL*-acceleration or turn *DirectDraw* off.

17.7.1. Option 1: Enable *OpenGL*

set the following *JVM parameter*:

```
-Dsun.java2d.opengl=True
```

(the uppercase T is correct)

17.7.2. Option 2 (if option 1 does not work): Disable *DirectDraw*

set the following *JVM parameter*:

```
-Dsun.java2d.noddraw=true
```

(the lowercase t is correct)

Please note that only one of these *JVM parameters* should be set at the same time.

The location of these *JVM Parameters* is platform-specific. Please see the section *Platform-specific topics* for more information about changing *JVM Parameters*.