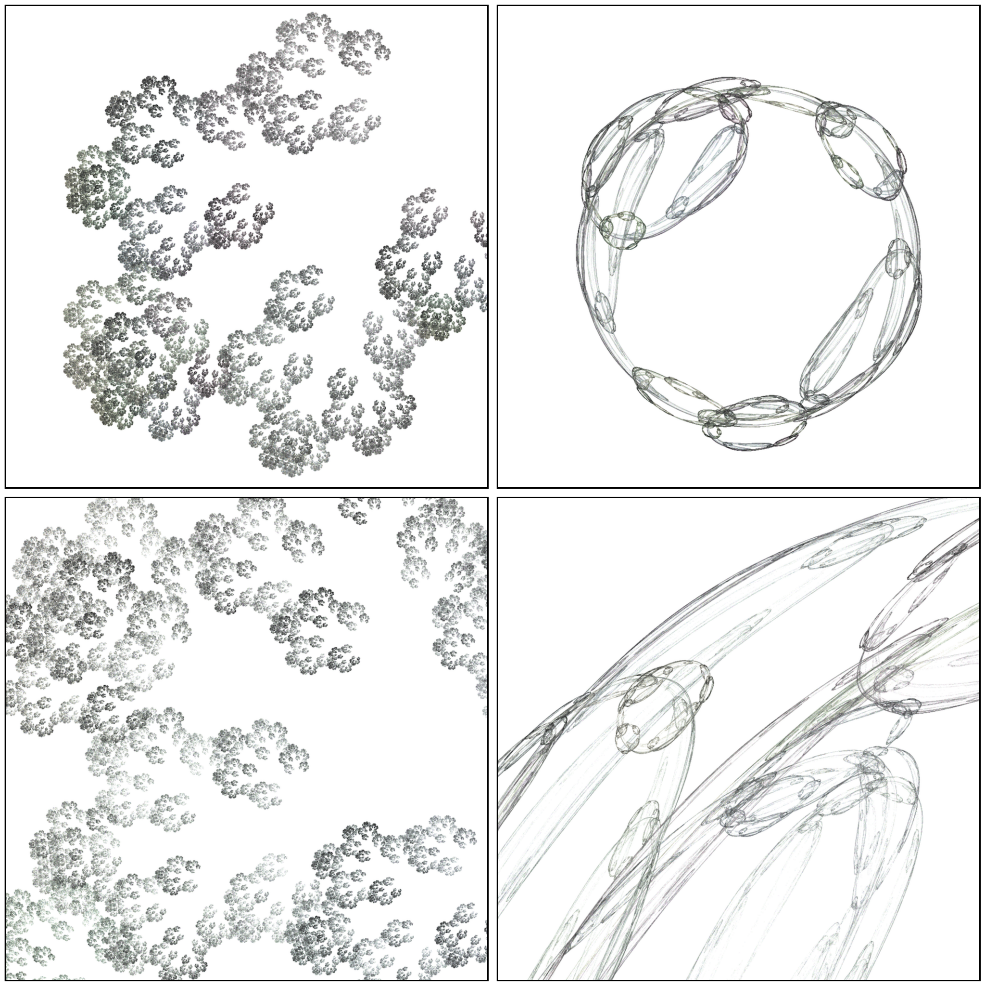
IFS Explorer

Version 1.4.1

An interactive **Iterated Function System** explorer. This program allows you to explore a vast world of complex and beautiful fractal images produced by simple mathematical rules, through a process of trial and error, experimentation and exploration. See the *References* section for links to books and papers describing the concepts with more detailed mathematics.



IFS Explorer provides an interactive UI to create and manipulate a set of affine transforms, which are then iterated randomly to produce an image. The Java `AffineTransform` class is used to represent and plot the transforms.

The systems can be saved and loaded as `.xml` files, and rendered images can be exported as `.png` files or printed.

Program requirements

- Java 1.8.0+ Runtime Environment
- Windows, Linux or OSX Operating System
- Maven and 1.8.0+ JDK for building *(Optional)*

Instructions

Either build the program using Maven or extract one of the packaged distributions. These can be downloaded from GitHub as either `.tar.gz` or `.zip` archives. Then run the relevant script for your operating system.

```
$ ./bin/explorer.sh [-f] [-c config] [-p palette] [ifs.xml]
% ./bin/explorer.command [-f] [-c config] [-p palette] [ifs.xml]
C> .\bin\explorer.cmd [-f] [-c config] [-p palette] [ifs.xml]
```

Alternatively, the `iterator-1.4.1-jar-with-dependencies.jar` .jar file can be downloaded and executed directly, with the `java -jar` command. This will execute using the default Java Swing platform look and feel.

```
$ java -jar iterator-1.4.1-jar-with-dependencies.jar
  ___ _____ ____  _____ ____    _  _____ ___  ____
 |_ _|_   _/ ___||_   _|  _ \  / \|_   _/ _ \|  _ \
  | |  | | \___ \  | | | |_) |/ _ \ | || | | | |_) |
  | |  | |  ___) | | | |  _ </ ___ \| || |_| |  _ <
 |___| |_| |____/  |_| |_| \_/_/   \_|_| \___/|_| \_\

    Iterated Function System Explorer 1.4.1

    Copyright 2012-2017 by Andrew Donald Kennedy
    Licensed under the Apache Software License, Version 2.0
    Documentation at https://grkvlt.github.io/iterator/

[-] Loading configuration resource explorer.properties
[-] Starting Explorer UI
```

The program can be controlled by setting properties (see the *Configuration* section) or by specifying flags as command-line arguments. To place the application into full-screen mode add the `-f` or `--fullscreen` flag to the command-line.

The default rendering mode is grayscale. To show a coloured image, choose one of the other rendering modes either by using the **Preferences** dialog, by editing the configuration properties or by setting the *explorer.mode* system property.

To use a colour palette taken from a sample image, specify `-p` or `--palette` on the command line, which will set the rendering mode to *palette*. The image used can also be set with the `explorer.palette.file` system property and can be one of several included named palettes, such as *abstract*, *wave* or *car*, or a URI pointing at a local or remote *.png* image to retrieve colours from.

Configuration

This list shows some of the available properties along with their default values.

- `explorer.mode` - Display mode - *palette*
- `explorer.render` - Rendering style and algorithm - *standard*
- `explorer.transform` - Co-ordinate transform function - *identity*
- `explorer.gamma` - Display gamma factor - *1.8*
- `explorer.vibrancy` - Colour vibrancy coefficient - *0.9*
- `explorer.blur` - Kernel size for blur density estimation- *4*
- `explorer.palette.file` - Colour palette image source - *abstract*
- `explorer.palette.size` - Number of colours in palette - *64*
- `explorer.palette.seed` - Random seed used for choosing colours - *0*
- `explorer.window.width` - Width of main window - *600*
- `explorer.window.height` - Height of main window - *600*
- `explorer.grid.min` - Minimum grid spacing - *10*
- `explorer.grid.max` - Maximum grid spacing - *50*
- `explorer.grid.snap` - Snap-to-grid spacing- *5*
- `explorer.threads` - Number of threads - *2*
- `explorer.debug` - Debugging mode - *false*

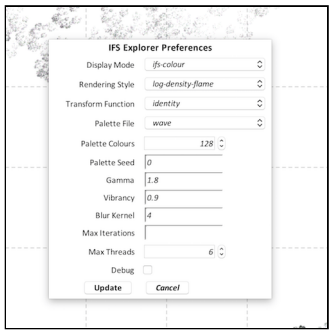
The configuration options above can be set in a *`.properties`* file, which can then be loaded by the application. The locations searched for configuration files are the users **`HOME`** directory, for a file named *`.explorer.properties`*, the current directory for a file named *`explorer.properties`* and finally the file specified on the command line using the `--config` *`config.properties`* option.

```
$ ./bin/explorer.sh --config debug.properties ./data/koch.xml
```

It is also possible to set individual configuration options as system properties, using the **`JAVA_OPTS`** environment variable. JVM memory options should be configured separately, using the **`JAVA_MEM`** variable.

```
$ export JAVA_OPTS="-Dexplorer.palette.seed=1234"
$ export JAVA_MEM="-Xms1g -Xmx2g"
$ ./bin/explorer.sh --palette wave ./data/spikes.xml
```

The *Preferences...* menu option displays a dialog with some of the available options. In particular there are drop-down menus to select the IFS display mode, such as *ifs-colour* or *gray*, and the rendering style, from *standard* to the range of density estimation based options such as *log-density-flame-inverse*. See the *Options* section for more details on these.



*The **Preferences** dialog.*

Settings changed using the **Preferences** dialog are applied to the current view, and can be saved as a *.properties* file using the *Save Preferences...* menu. This will write out all configuration options that have been explicitly set, either from the initial configuration file or by changes made using the **Preferences** dialog.

```
explorer.gamma = 0.5
explorer.mode = ifs-colour
explorer.render = log-density-flame-inverse
explorer.window.height = 2000
explorer.window.width = 2000
```

Example saved ***.properties*** *configuration file contents.*

> ***Note*** *The save dialog for the* ***.properties*** *file defaults to using the same name as the configuration file specified on the command line. If the name is not changed then its contents will be overwritten and any comments or formatting will be lost.*

Options

The most interesting configuration properties are the `explorer.mode`, `explorer.render` and `explorer.transform` which together define the way the IFS will be rendered and coloured. Additionally, the set of `explorer.palette.*` options help to determine the colours used, and `explorer.gamma` and `explorer.vibrancy` control the way on-screen colours are displayed.

> ***Note*** *Some of these options interact with each other, and also with the IFS transforms being displayed. Not all combinations of settings will make sense for a particular IFS configuration.*

The `explorer.mode` setting determines the way that colours will be chosen for points of the IFS. The options are *colour*, *gray*, *palette*, *stealing* and *ifscolour*. The *colour* option assigns different colours to each transform across the full spectrum, giving a rainbow coloured image. The *gray* option generates a black and white image, as differing levels of gray.

To produce a coloured image, with the colours taken from a reference image, use *palette* which picks a random set of colours from pixels in the source image. This mode uses `explorer.palette.file` to specify the image. If one of *abstract*, *autumn*, *car*, *car2*, *lego*, *night*, *forest*, *trees* or *wave* is specified then an embedded file from the application is used. Instead, a URL can be specified, either `file://` or `http://`, and that image will be loaded. The colours are assigned to each transform in the same way as with the *colour* mode. To change the colours used, the random number seed can be altered using `explorer.palette.seed`.

The *stealing* mode uses a colour stealing algorithm, taken from **Superfractals**. This also uses the palette file setting, and can produce some very appealing and natural images. Use *ifs-colour* to generate a brightly coloured image using the entitre spectrum mapped using the IFS.

The `explorer.render` option complements the display mode and can be set to one of *ifs*, *top*, *standard* or *measure*. The *ifs* option renders the IFS attractor set and is best combined with the *gray* display mode. The *standard* rendering mode is the default, and uses alpha blending to combine colours when rendering. The *top* rendering mode generates a fractal top, as defined in **Superfractals** and can only be used with the *colour* or *palette* display modes. Finally, *measure* generates the measure set, which displays on a black background and can be combined with any of the *colour*, *palette* or *stealing* modes to specify the colours.

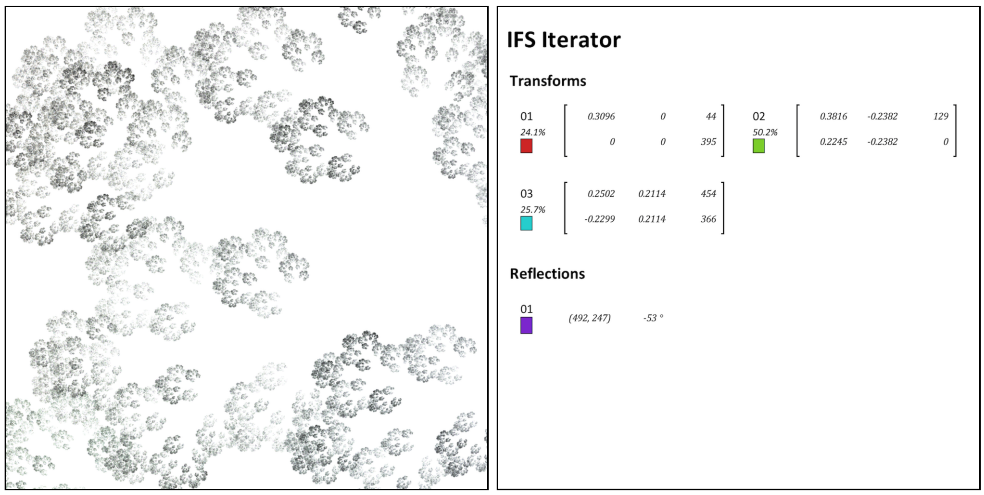
Modes *density*, *log-density*, *log-density-blur* and *log-density-flame* all use a similar density estimation mechanism when rendering. This gives a different behaviour in the viewer, with progressive renders showing more fine-grained detail at each pass. These modes can all be used with the various colour options to produce different styles of output. The **-inverse* modes use a black background with the highest desity rendering as white, but otherwise the same algorithm. The *log-density-blur* mode uses the value of `explorer.blur` as the size of a super-sampling blur kernel, which is mixed with the density value.

The `explorer.transform` option sets a final co-ordinate transformation function, to be executed after each iteration of the IFS rendering. The default is the *identity* function, which leaves the co-ordinates unchanged. The functions available are taken from the **Fractal Flame** paper listed in the *References* and include *spherical*, *exponential*, *fisheye* and *tangent*. These transform functions can be set from the **Preferences** dialog and the results viewed immediately.

Usage

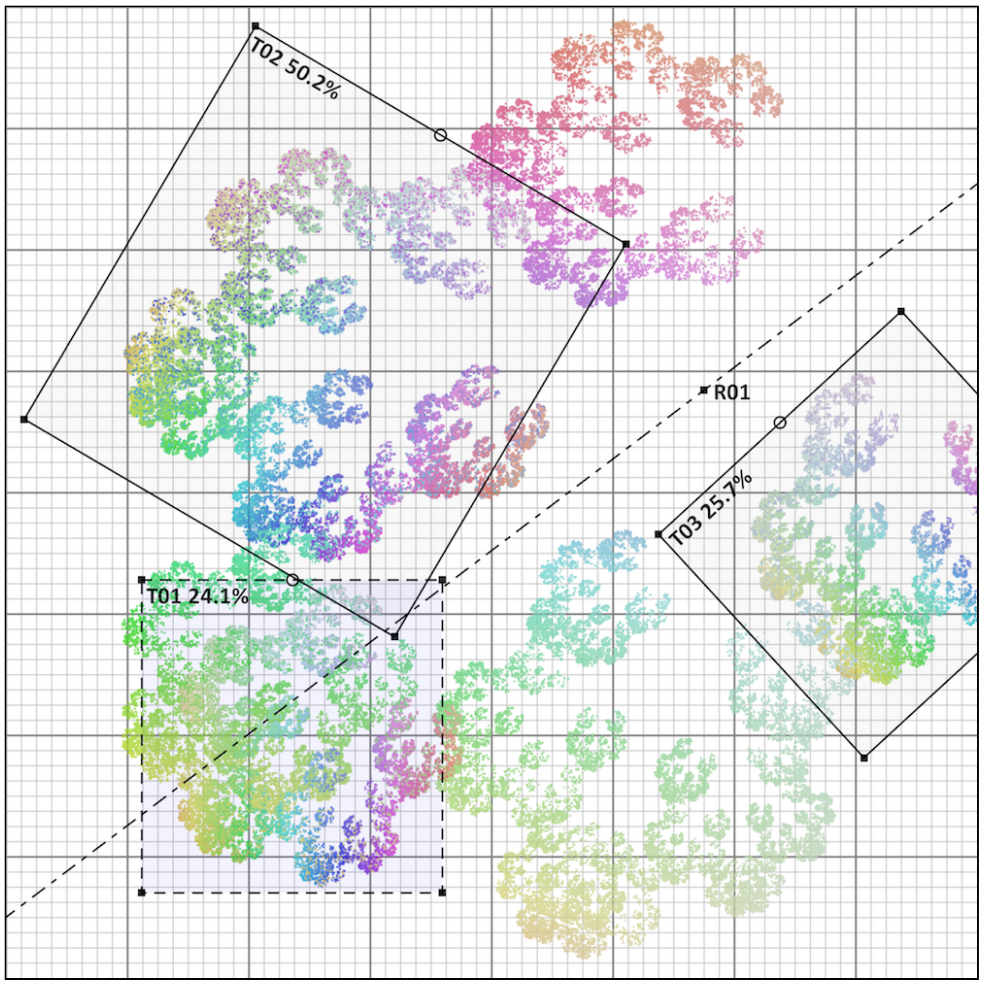
Once the program starts, it will diplay an empty grid in the **Editor** mode. The menus provide access to standard operations, including the ability to switch modes. The keyboard can also be used to toggle between modes using the *Tab* key. Press *h* or *?* to view help text listing all available keyboard actions.

The other modes are the **Viewer** which renders the IFS with increasing detail as it executes multiple iterator thread, and **Details** which shows the transformcl matrices and reflection parameters that make up the IFS.



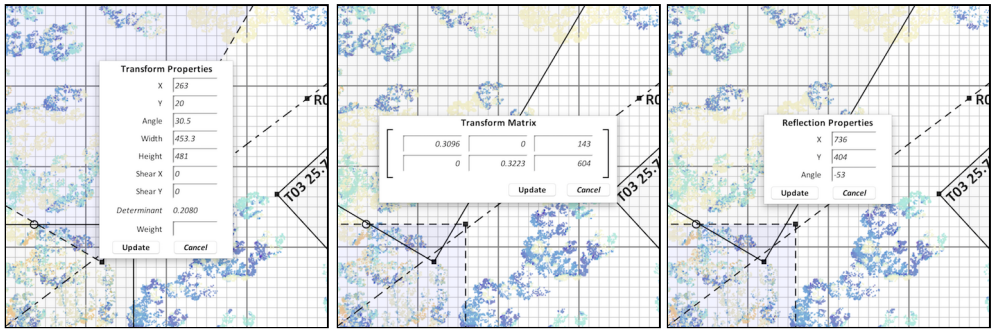
*Screenshots for the **Viewer** and **Details** modes.*

A live view of the generated IFS is displayed at low resolution to give an idea of the eventual rendered image of the fractal. **This capability is what makes the application so useful.** It allows much more creativity when designing an IFS. By showing the results of changes to a transform immediately it is possible to reach the desired final image much more quickly.



*Screenshot of the **Editor** mode with the live IFS view*

In the **Editor** mode, clicking and dragging the mouse will create a new transform which can then be moved, resised and rotated as desired. If ***Shift*** is held down while resizing the aspect ratio will be maintained. If ***Alt*** is held while clicking and dragging a transform, it will be duplicated. When a transform is selected, the *Delete* or *Backspace* key will delete it and the *+* or - keys will rotate it by ninety degrees. The arrow keys will move the transform in the appropriate direction by the minimum grid-snap increment, or by a single pixel if ***Shift*** is held down. On OSX the touchpad rotate and zoom gestures can be used as well on the selected transform.



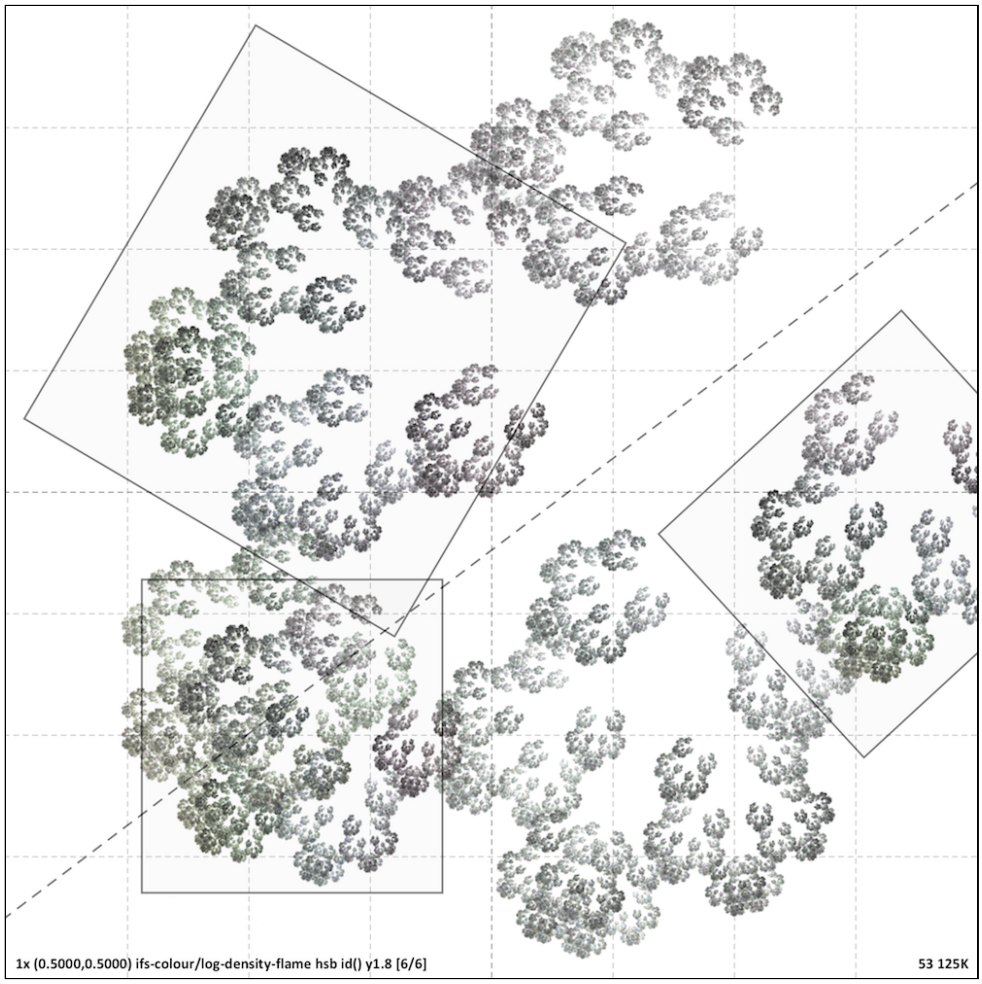
*The **Transform Properties**, **Transform Matrix** and **Reflection Properties** dialogs.*

The context menu displayed by right-clicking on a transform allows the ***z-index*** to be raised or lowered, and the selected transform can be moved to the front or back to allow selecting overlapping transforms. A transform can be removed using the *Delete* menu item, or its properties can be altered by chooing the *Properties...* option. This will display the **Transform Properties** dialog, allowing the position, size, rotation, shear and weight values to be editied. The *Matrix...* menu item allows the six affine transform matrix coefficients to be altered directly.

> ***Note*** *Once the affine transform matrix has been edited directly, it will no longer be possible to edit the transform visually or change its properties, although it can still be moved using the mouse or keyboard and rotated through 90 degrees.*

The default menu displayed when right-clicking on the background has three options. *New IFS* will delete all existing transforms and reflections and create a new, empty IFS. *New Transform* and *New Reflection* create new functions, respectively an affine transform and a planar reflection.

Reflections can also be created by *Ctrl*–clicking on the background or duplicated by holding down *Alt* and clicking and dragging an exsting reflection. Reflections can be modified by dragging them by their square handle or by editing their position and rotation values directly using the **Reflection Properties** dialog, available via their context menu.

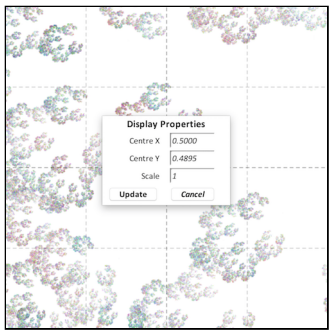


*Screenshot of the **Viewer** mode with IFS overlay and grid*

When in **Viewer** mode the *Space* key will pause and continue the rendering process. To zoom into the fractal select a specific area by dragging the left mouse button, or use the keyboard *+* and - keys to to zoom in and out of the centre of the image by a factor of two. The = key will reset the view to the original unmagnified and centered settings. To display the outline of the transforms being rendered, use the *o* key, and the *i* key will toggle display of information about the current view.

The view information includes: the scale factor and viewpoint center co-ordinates; the display and rendering modes and the active co-ordinate transform function name; the gamma factor; current and maximum thread counts; and the number of iterations. This information will be included in a printout of the view (as well as the grid and overlay, if enabled) but not in exported ***.png*** images.

The thread count can be adjusted using the *Up* and *Down* arrow keys, as well as through the **Preferences** dialog, and pressing `t` will print out a list of the active threads and their current state.



*The **Display Properties** dialog for editing scale factor and viewpoint centre.*

> ***Note*** *When zoomed in at high magnification the iterative process can take a very long time until the rendered fractal becomes visible.*

The **Details** mode shows the actual affine transforms and their matrix coefficients, and the co-ordinates and angle of the reflection planes. The scrollable list can also be printed, and will be resized to fit on a single page.

> ***Important*** *Editing and other actions can also be accessed using the right-click context menu on most screens.*

Animation

You can animate a sequence of changes to the individual transformations using the animator scripts in the `bin` directory. These use a configuration file that describes the set of changes, such as the example provided in `data/leaf.animation`, which is rendered as follows.

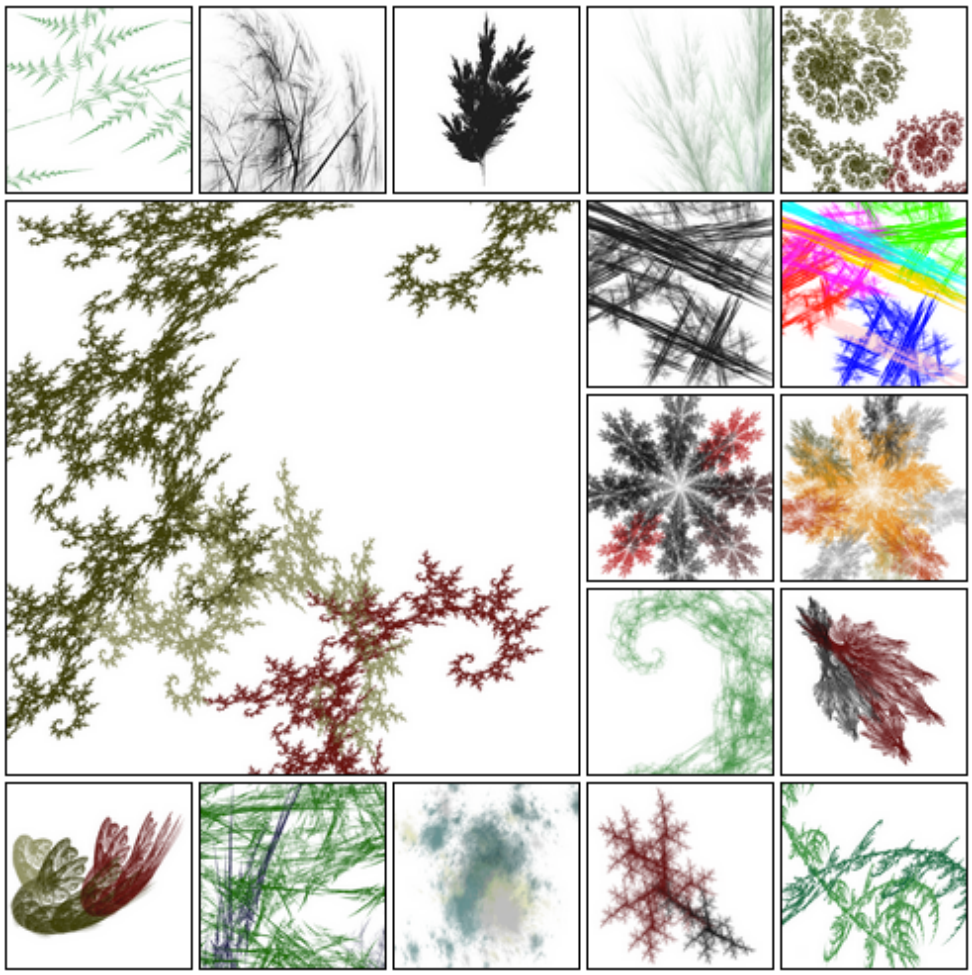
```
$ ./bin/animator.sh ./data/leaf.animation
Configured colour: abstract
File 0000.png: 3 transforms: 1.0x scale at (400.00, 400.00)
with 1,165K iterations
File 0001.png: 3 transforms: 1.0x scale at (400.00, 400.00)
with 1,310K iterations
```

Using a utility such as *ffmpeg*, the generated images can then be combined into an `.avi` MPEG video file.

> ***TODO*** *Further documentation on the `.animation` file format and generating video files is required.*

Examples

Some example *.xml* data files are provided with the distribution. Some of these are taken from the books **The Computational Beauty of Nature** and **Superfractals**. The **Fractal Flame** paper describes the density estimation methods used, and illustrates the various co-ordinate transformation functions available. See the *References* for more details.



Gallery of IFS fractal images.

This selection of images is included in the distribution archive. To view it, open the `data/gallery/index.html` file in a web browser. Images of the example *.xml* files are also available as *.png* files in the same directory.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<IFS name="example">
    <Transforms>
        <Transform id="1" zIndex="0" x="100" y="100"
                w="500.0" h="500.0"
                r="1.5707963267948966" shx="0.0" shy="0.0"
                sw="1000" sh="1000"/>
    </Transforms>
    <Reflections>
        <Reflection id="1" x="500" y="600"
                r="2.3781689744890073" sw="1000" sh="1000"/>
    </Reflections>
</IFS>
```

*Example saved **.xml** file contents.*

References

1. **Iterated Function System**; *http://en.wikipedia.org/wiki/Iterated_function_system*; *Wikipedia*
2. **Affine Transform**; *http://en.wikipedia.org/wiki/Affine_transformation*; *Wikipedia*
3. **Construction of fractal objects with iterated function systems**; *http://www-users.cs.umn.edu/%7Ebstanton/pdf/p271-demko.pdf*; *Demko, Stephen and Hodges, Laurie and Naylor, Bruce; SIGGRAPH Computer Graphics, Volume 19, Number 3, 1985*
4. **The Fractal Flame Algorithm**; *http://flam3.com/flame.pdf*; *Draves, Scott and Reckase, Eric; 2008*
5. **Superfractals: patterns of Nature**; *http://www.amazon.co.uk/SuperFractals-Michael-Fielding-Barnsley/dp/0521844932*; *Barnsley, Michael F; Cambridge University Press; 7 Sep 2006; ISBN 978-0521844932*
6. **The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems and Adaptation**; *http://www.amazon.co.uk/The-Computational-Beauty-Nature-Explorations/dp/0262561271*; *Flake, Gary W; MIT Press; 1 Mar 2000; ISBN 978-0262561273*
