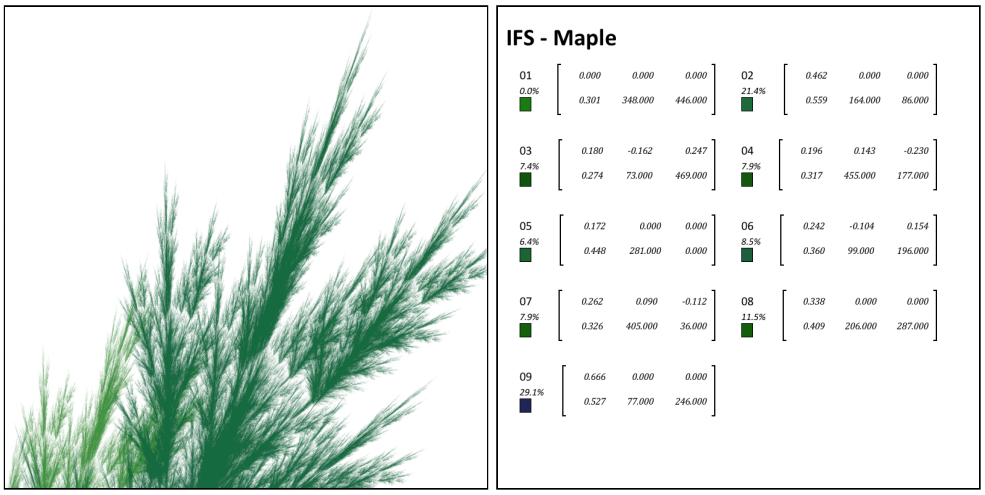
# IFS Explorer

## Version 1.0.11

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 below for more details on the mathematics and concepts behind these systems.

**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 graphics files or printed.




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

## Program requirements

- Java 1.8.x Runtime Environment
- Windows, Linux or OSX Operating System
- Maven and 1.8.x 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] [-p palette] [ifs.xml]
$ ./bin/explorer.command [-f] [-c] [-p palette] [ifs.xml]
C> .\bin\explorer.cmd [-f] [-c] [-p palette] [ifs.xml]
```

Alternatively, the `iterator-1.0.11-jar-with-dependencies.jar` .jar file can simply be downloaded and executed directly, with the `java -jar` command.

```
$ java -jar iterator-1.0.11-jar-with-dependencies.jar
   ___ _____ ____    _____                _
  |_ _|  ___/ ___|  | ____|_  ___ __ | | ___  _ __ ___ _ __
   | || |_  \___ \  |  _| \ \/ / '_ \| |/ _ \| '__/ _ \ '__|
   | ||  _|  ___) | | |___ >  <| |_) | | (_) | | |  __/ |
  |___|_|   |____/  |_____/_/\_\ .__/|_|\___/|_|  \___|_|
                               |_|

    Iterated Function System Explorer Version 1.0.11
    Copyright 2012-2017 by Andrew Donald Kennedy
    Licensed under the Apache Software License, Version 2.0
    https://grkvlt.github.io/iterator/
```

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.

Enable colour rendering with the `-c` or `--colour` flag. This will use a fixed list of colours for the transforms. To use a colour palette taken from a sample image, specify `-p` or `--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.grid.min` - Minimum grid spacing - *10*
- `explorer.grid.max` - Maximum grid spacing - *50*
- `explorer.grid.snap` - Snap-to-grid spacing- *5*
- `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.debug` - Enable debugging mode - *false*
- `explorer.mode` - Colour display mode - *palette*
- `explorer.render` - IFS rendering style - *standard*
- `explorer.threads` - Number of threads - *2*

The configuration options above can be set in a property 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.

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

## Options

The most interesting configuration properties are the `explorer.mode`, `explorer.render` and `explorer.palette.*` options. These determine the colour and style that will be used to display the IFS on screen.

> ***Note** that these options interact with each other and not all combinations of settings will make sense.*

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`.

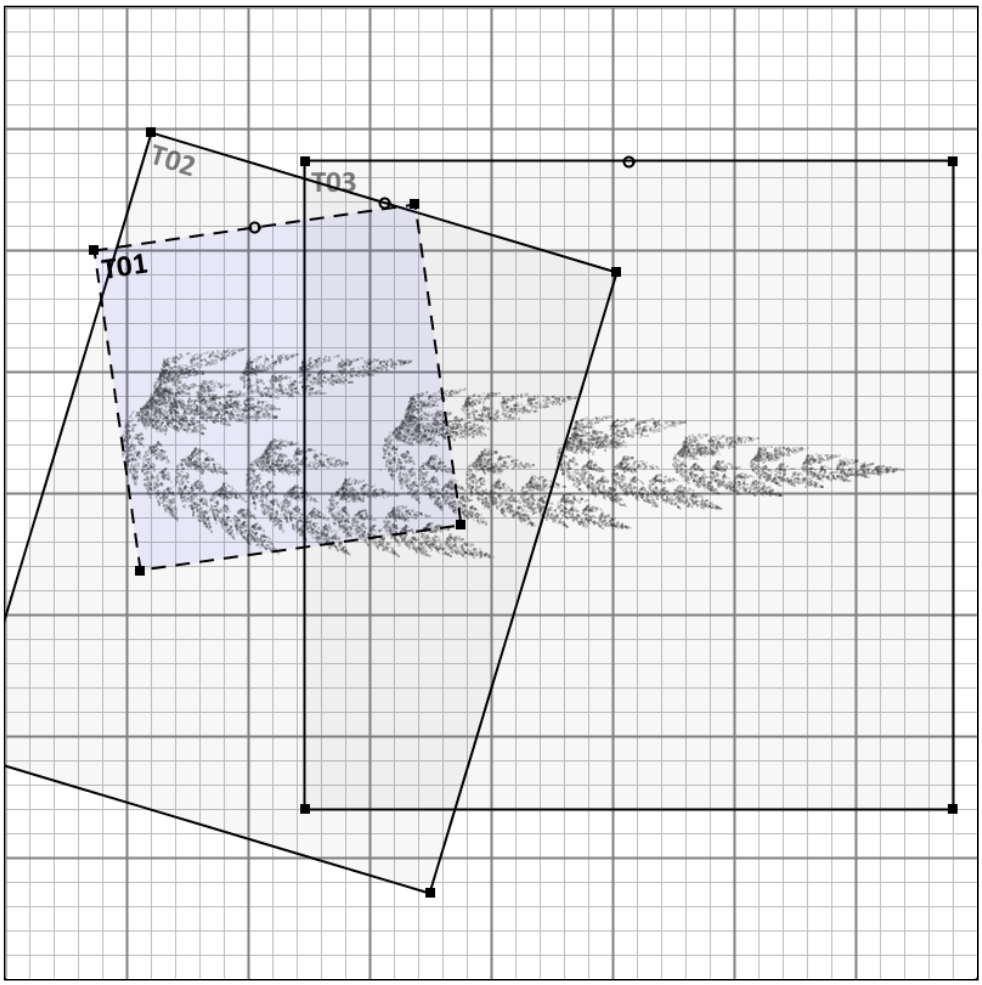
The *stealing* mode uses a colour stealing algorithm, taken from **Superfractals**. This also uses the palette file setting, and can produce some very appealing and natural images. Finally, *ifscolour* generates a colour image using a spectrum of colours generated using the IFS.

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.

## 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.

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 (with *Shift* held down) as desired. In this mode, the *Delete* or *Backspace* key will delete the selected transform and the *Left* or *Right* arrow keys will rotate it by ninety degrees.

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 a dialog allowing the position, size or rotation 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.*

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.

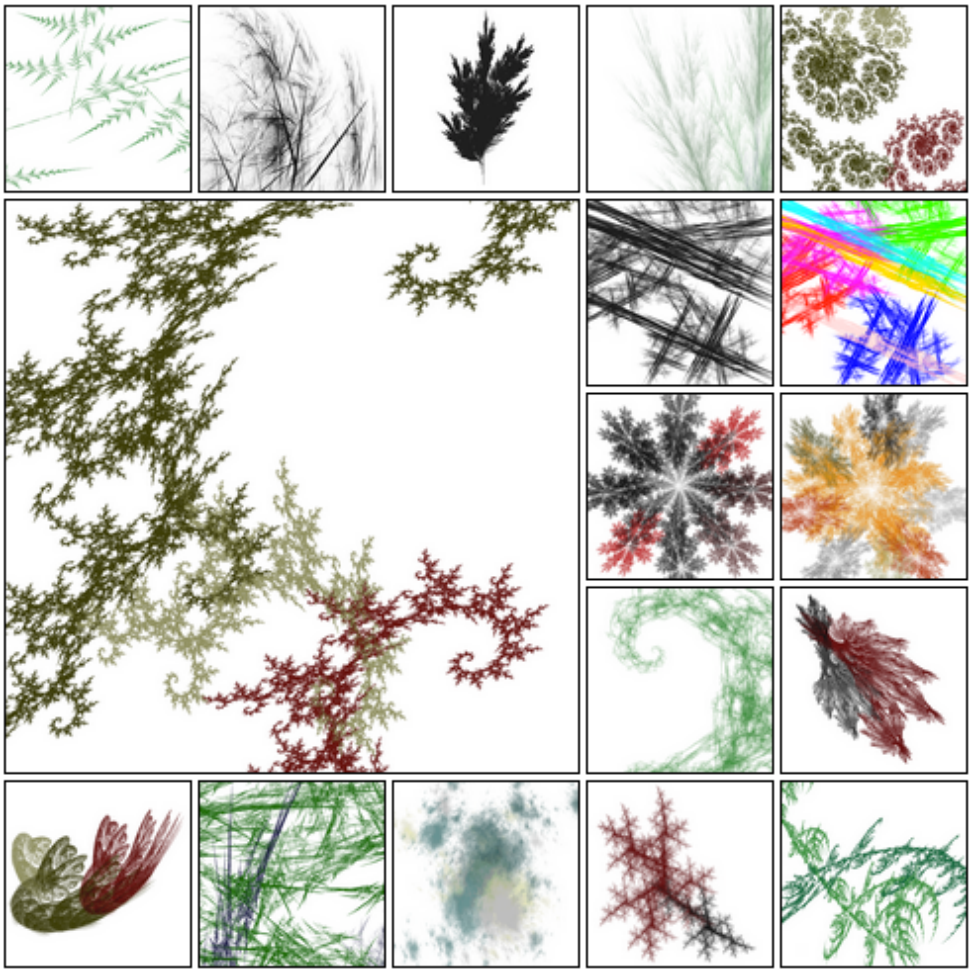
> ***Note** that when zoomed in at high magnification the iterative process can take a very long time until the rendered fractal becomes visible.*

The **Details** mode shows the actual affine transforms and their matrix coefficients for the system as a scrollable list.

## Examples

A series of example XML data files for IFS transforms are provided in the `data` directory of the distribution. Some of these can be found in the books **The Computational Beauty of Nature** and **Superfractals**, details of which are given in the *References* section.

To load the examples, specify the path to the file as the last argument on the command-line. Sample images for these transforms have also been provided, so be sure to use the `.xml` file, not the `.png`.



*Gallery of example fractal images.*

A more extensive gallery of IFS images is included in the distribution archive. To view it, open the `data/images/gallery/index.html` file in a web browser.

## 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.

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

## 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. **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*
5. **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*


---