COMS 4115
Programming Languages and Translators

IpsoFracto: Fractal Generation Language
Whitepaper

Leo Gertsenshteyn (lpg2006@columbia.edu)
Sarah Gilman (srg2104@ columbia.edu)
Rob Notwicz (rcn15@ columbia.edu)
Anya Robertson (alr1@cs. columbia.edu)

Introduction

IpsoFracto will be a language for defining fractals from basic lines, polygons and kinks (bends in a line). It will produce bitmap images of a fractal from the code. This will be a handy tool allowing a user to alter the definition of their fractal, by changing the initial shape or the kinks applied to the lines, and quickly see the results.

Usability

IpsoFracto will be a simple language with very few syntactic restraints. The language will be easier to use due to the visual nature of the results; if a user is unsure of what the attributes of a kink do, they can always change them and see immediately what they control. In this way, IpsoFracto will be well suited to the beginner, even one relatively unfamiliar will programming or fractal math.

Portability

IpsoFracto will use ANLTR to initially parse the code, and a Java interpreter to produce bitmap images from the AST. This will allow IpsoFracto programs to be compiled on any platform which supports ANTLR and Java.

Error Handling

IpsoFracto will be a fairly simple language, with very few possible errors. The interpreter will be responsible for catching errors such as invalid points, or polygon specifications which contain crossed edges. It will obviously be possible for a user to get unexpected images, but the nature of the language will make it easy for them to return to the program and alter lines or kinks.

Library

Kinks will be defined in a library. They will take parameters indicating their size which relative to the line they are placed on. For instance, ‘trikink(5,5)’ produces a triangle shaped kink with a height and width roughly equal to a third of the line it is placed on. All kinks can be applied to lines with the key-word parameters for positions (‘left’, ‘right’ and ‘center’) and line orientation (‘pos’, and ‘neg’) as well as a parameter for specifying how many kinks should appear next to each other.

Data Types

IpsoFracto has three main data types: line, multi-line and polygon. These define the basic starting points for the fractal. All three are based on an implicit type, the point or 2-D position in the bitmap. They are all a set of points, linked with the operators −> and <−. These operators indicate the orientation of a line. For example, 'line myline (0, 10) −> (0, 30)!' defines a line from (0, 10) to (0, 30) while 'line myline (0, 10) <− (0, 30)!' defines a line from (0, 30) to (0, 10). These operators were chosen because it is intuitively clear which point is the start and which is the end of the line.

Multi-lines and polygons are collections of multiple points, with the polygon having an added implicit line from the last point to the first. For example, 'poly mysquare (0,10) −> (10,20) −> (10,10)!' defines a square. Polygons will be checked by the interpreter for crossed lines, while multi-lines will not.

It will be possible to name an instance of a kink and treat it as a variable throughout the program, allowing the user to define a kink's parameters once and use it repeatedly. Alternatively, it will possible to create anonymous kinks inside the call which applies a kink, as in our syntax sample below.

Scope

All lines and kinks defined in a program will be valid available throughout the program. Lines, multi-lines and polygons should be defined at the top of the file since they are the starting point for the fractal.

Control flow

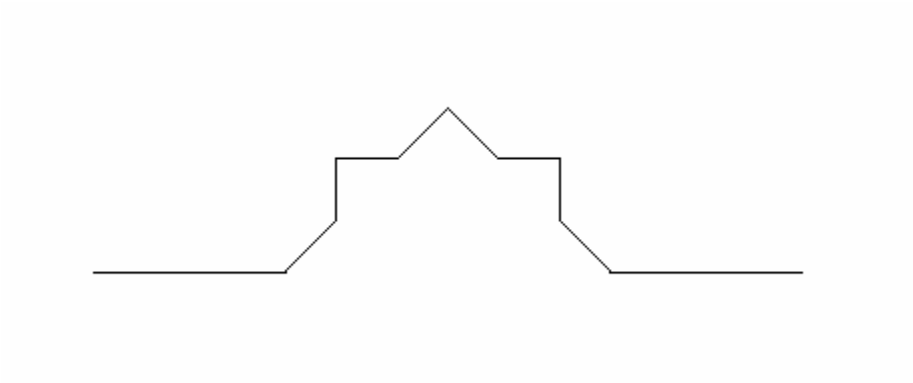
IpsoFracto supports variants of the while and for loops in order to allow the user to chose how many times they wish to apply a kink to a line. These can be controlled by the number of kink iterations, or time to add kinks, or the kink size (i.e. iterate until a kink is no longer visible).

IpsoFracto will also support a variation of the standard if statement in order to allow the user to alter the kink applied based on variables such as the current iteration count.

Sample syntax

```
line lineA (20, 0) → (50, 0)!
iter 2  {
   apply ( lineA, CENTER, POS, trikink(5,5), 1 )!
}
```

This syntax defines a single line, and then iteratively places a single triangular kink in the line at the center position and positive orientation. This syntax will produce a fractal which looks like this:



Extensions

It will be easy to extend IpsoFracto be defining libraries with new kinks. If time permits, we will do just this adding a library which defines some curved kinks.