フラクタル手法によるコンピュータグラフィックス

研究者:杉山凱哉

1　はじめに

今までのプログラミングの中で、たくさんの処理の手順について学んできた。今回の研究に使う Java は、C 言語とは少し違うプログラミング言語なので、どんな構造をしているのかを学びたいと思った。また、数学が好きな私には適したテーマであると思ったので、この研究を行った。

2　研究内容

フラクタルとは、全体の図形と相似な図形の繰り返しをしているものである。これを作成するために使った Java は、サンマイクロシステムズがリリースしたプログラミング言語である。

Web ページに表示されている、C 言語などのプログラミング言語で描かれたフラクタルを参考にして、それを Java に置き換えながら作成した。

3　研究過程

6月　Java バイブルテキストの黙読
　　フラクタル CG コレクションの黙読
　　計算、代入、出力のプログラムの練習
7月　配列、判別処理、繰り返し処理の練習
　　メソッドの処理の練習
8 月　引数の処理の練習
　　入力の処理の練習
　　GUI の、フレームとパネルを出力させるプログラムの練習
9月　ボタンとテキストを表示させるプログラムの練習
　　アクションとアイテムを使った、ボタンを押したときの処理の練習
10月　グラフィックのプログラムの練習
　　コッホ曲線のプログラムの作成
11月　2種類の木のフラクタルのプログラムの作成
　　ドラゴン曲線のプログラムの作成
　　シェルピンスキーのプログラムの作成
　　入れ子のプログラム作成
　　マンデルブロ集合のプログラムの作成
12月　ヒルベルト曲線のプログラムの作成
1月　カントール集合のプログラム

4　研究の成果

(1) Java の文法

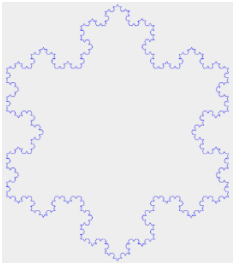
Java のプログラミングを行い、C 言語とは違う書き方をすることが分かった。

```
public class ファイル名{
   public static void main(String args[ ]){
      (変数や処理など);
   }
}
```

Java のプログラム構造

Java を使って、GUI を表示させるなどの練習や、グラフィックを使って図形や模様を描く処理を書く練習を行った。これらの練習を行って、Java の文法をたくさん覚えることができた。

(2) コッホ曲線



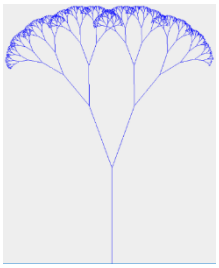
コッホ曲線

六光星を作るようにするため、初めの正三角形のすべての辺を3等分に分ける。その真ん中を新たな正三角形の底辺とする。そして底辺の長さに等しい2つの辺を作り、底辺を消す。そのプログラムを書いた。その再帰処理を使ってコッホ曲線を描いた。

```
c.x=(2*a.x+b.x)/3;
c.y=(2*a.y+b.y)/3;
d.x=(a.x+2*b.x)/3;
d.y=(a.y+2*b.y)/3;
xx=b.x-a.x;
yy=-(b.y-a.y);
distance=Math.sqrt(xx*xx+yy*yy)/Math.sqrt(3);
if(xx>=0){
    angle1=Math.atan((double)yy/xx)+Math.PI/6;
    e.x=a.x+(int)(distance*Math.cos(angle1));
    e.y=a.y-(int)(distance*Math.sin(angle1));
}
else{
    angle2=Math.atan((double)yy/xx)-Math.PI/6;
    e.x=b.x+(int)(distance*Math.cos(angle2));
    e.y=b.y-(int)(distance*Math.sin(angle2));
}
```

プログラムの例

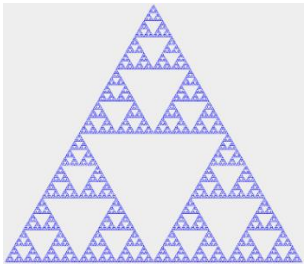
(3)樹木曲線



樹木曲線

最初の線の終点から、2本に枝分かれさせるために、縦のベクトルと横のベクトル、角度のラジアンを使った式をたてた。そして、傾けた線の終点を求める。この再帰処理を繰り返し、樹木曲線を描いた。

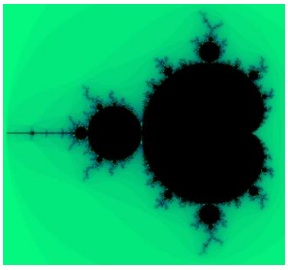
(4)シェルピンスキーのギャスケット



シェルピンスキーのギャスケット

最初の正三角形を作り、そのすべての辺の中点を求め、それらを使って新たな正三角形を作る。その再帰処理を使って、シェルピンスキーのギャスケットを描いた。

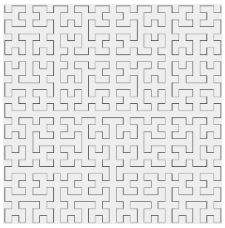
(5)マンデルブロ集合



マンデルブロ集合

実数の値と虚数の値を使った式を使って、値の入れ替えを行い、色の変化をつけた。また、上下左右の移動や拡大縮小のプログラムを書き、マンデルブロ集合の拡大画像を調べることができるようにした。

(6)ヒルベルト曲線



ヒルベルト曲線

指定された大きさに合わせてコの字の図形を描く時、始点と終点を準備して、線の長さ分を伸ばして描き、終点を始点に変えて、新たに線を描く。これを2回繰り返し、コの字の図形を作成する。その再帰処理を使って、ヒルベルト曲線を描いた。

5 考察

ヒルベルト曲線において、Math.pow を使った計算で、int 型で計算をしていたがエラーが発生してしまった。これは累乗の計算であるため値が非常に大きくなる場合があるとわかった。そのため、double 型で計算したところ、エラーは解消された。

他のフラクタルにおいて、種類によって、角度を求めるものがあったり、いくつかに等分するものがあったりする。

6 まとめ

プログラミングが得意である私にとって、最初は難しそうだと感じたが、この課題研究をやっているうちに、これは自分でもできるという自覚をもった。C 言語との違いを比較しながら練習を進めていった。また、フラクタル図形の制作において様々な計算式や処理を学ぶことができた。

これにより、将来に役立つために必要なプログラミングの知識をある程度身に付けることができた。これからもプログラミングの様々な知識を身に付けていきたいと思う。