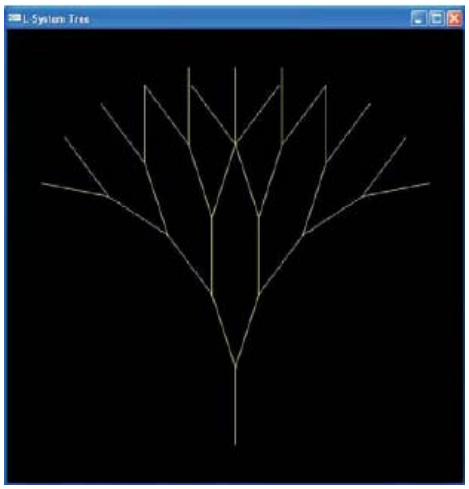
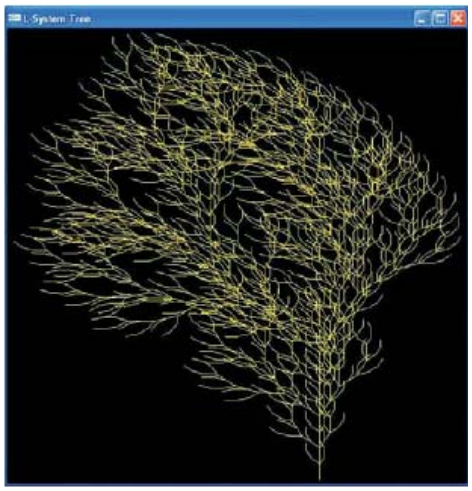
After basic code:



The following system is obtained by:

```
atom "FX"
F -> "FF-[-F+F+F]+[+F-F-F]+F-F+[+FF-]-"
X-> "X+F-[-FX]+"
angle = 27
```



Further randomization of the tree: angle=getrandom (15,22).

Challenge
Try and modify your code so the branches of the tree decrease in thickness from the trunk to the outermost branches. This may mean keeping some kind of thickness variable that is made smaller with each branch. One way to determine if you are drawing a smaller branch is to keep track of the '[' and ']' characters. For most tree strings the '[' denotes the start of a new branch and the ']' denotes the end.

Leaves

To make the tree even more interesting, we could add some leaves. Leaves usually occur at the end of branches. We can introduce a new character into our string to denote a leaf, 'L'. Whenever the produceString() function encounters an 'L' it can be instructed to draw a leaf at the current turtle location.

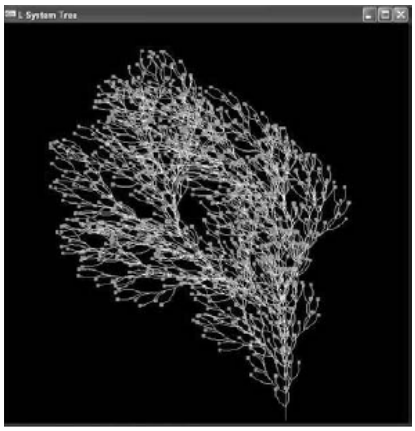
Modify your F production string to be this:
char Fstr[100] = "FF-[-F+F+F+L]+[+F-F-F+L]";

Notice that we have added a leaf to the end of each branch. Now to get the function to draw the leaf we need to add a new case to the switch statement, thus:

```
case 'L':   if(order <= 0)
            drawLeaf();
            break;
```

The drawing of leaf:

```
void drawLeaf()
{
      glColor3f(0,1,0);
      glPointSize(5);
      glBegin(GL_POINTS);
            glVertex2f(CP.x,CP.y);
      glEnd();
      glColor3f(1,1,0);
}
```

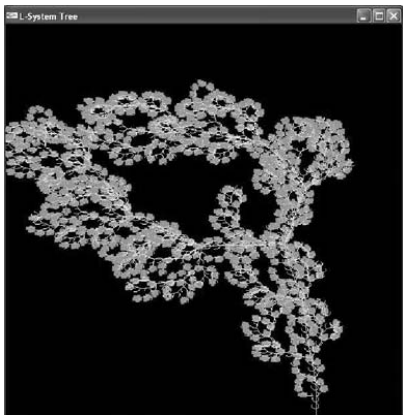


This is the result. To improve:

The tree below was created with random angles between 30 and 40 with:

```
atom "F"
F -> "FFF-[-FX+FX+FX]+[+FX-FX-FX]"
X-> "X-[FL+FL+FL]+[+FL-FL-FL]"
```

Notice that some of the branches (F) have been left without leaves.


L-System Tree