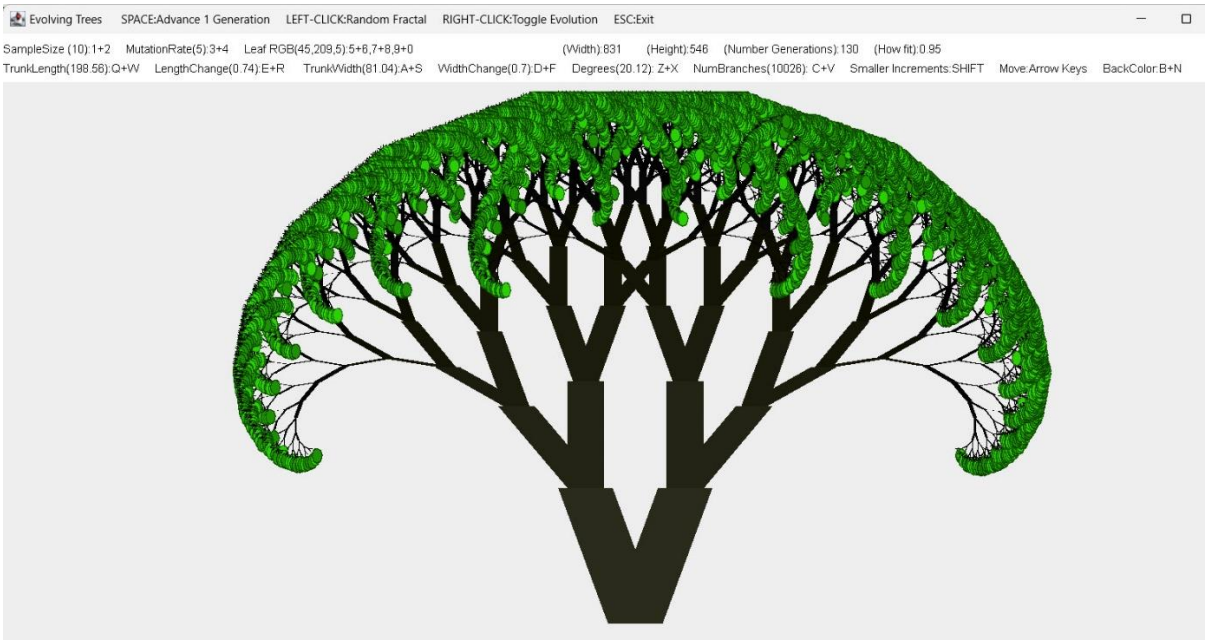
Genetic Algorithms Lab

The aim of this lab is to learn about genetic algorithms as a subset of artificial intelligence and implement a simple algorithm with the intention of evolving trees to be more fit.



`Tree.java` contains the DNA for what it means to be a Tree (graphic fractals in this case, but they look treeish). You only need to know about the existence of the following Tree methods:

- **`public Tree makeRandomTree()`** `//returns a random Tree fractal`
- **`public Tree mutate(int mutationRate)`**
```
//given the current tree, return an
//almost-clone of the tree where
//each field can vary by a
//mutationRate
```
- **`public int compareTo(Object other)`**
```
//Trees compare to one another by
//fitness: more fit > less fit
```

In `TreePanel.java`, you need to know of the following data-fields:

- **`private static Tree [] forest;`** `//most fit is at forest[0]`
- **`private static int SAMPLE_SIZE = 10;`** `//number of trees to mutate (2-25)`
- **`private static int MUTATION_RATE = 5;`** `//percentage a tree can mutate`
- **`private static int numGenerations;`** `//count the number of generations`

A constructor for the Panel will populate the forest with random trees.

```
public TreePanel()
{
   numGenerations = 0;
   evolve = false;
   forest = new Tree[SAMPLE_SIZE];
   forest[0] = forest[0].makeRandomTree();
   for(int i=1; i<forest.length; i++)
      forest[i] = forest[0];
}
```

If done correctly, the fitness of the trees should always increase when you start evolution. You can see where the fitness level is tracked at the top of the screen.

Code needs to be completed in `TreePanel.java` to do the following:

- Complete the main brains of the AI to advance the evolution of trees by one generation:

```
private static void advanceGeneration()
{
     numGenerations++;
     //***Complete the code here***/
}
```

This method should put the most-fit tree at the front of the list. The fitness of trees can be compared by the `compareTo` method. Then, you can repopulate the rest of the forest with children of the most-fit tree, where each child can differ from the most-fit by the `MUTATION_RATE`.

The client will have the option of pausing evolution (right mouse click) to adjust the sample size (# of children in forest) or the mutation rate.

- Complete the code to allow the client to make the forest one larger or one smaller. The more children there are, the more likely it will be that a child will be more fit than the parent, but will require more processing to find the most fit.

```
private void keyPressed(KeyEvent e)
{
      if(e.getKeyCode()==KeyEvent.VK_1)      //Sample Size (2-25)
      {
            if(SAMPLE_SIZE <= 2)
               return;
            SAMPLE_SIZE--;
            //***COMPLETE THE CODE HERE***/
            repaint();
            return;
      }
     if(e.getKeyCode()==KeyEvent.VK_2)      //Sample Size (2-25)
     {
            if(SAMPLE_SIZE >= 25)
               return;
            SAMPLE_SIZE++;
            //***COMPLETE THE CODE HERE***/
```

These blocks of code will increase or decrease the size of the forest by 1, maintaining the integrity of all of the trees you are allowed to keep.

- If we are going to decrease the size of the forest by 1, we will lose the genetics of one tree: make sure that it is not the one that is the most-fit.

- If we are going to increase the forest size by one, we will want to populate the new last index with a new tree of some kind.

The controls for running the simulation are as follows:

- SPACE: advance one generation
- Left-click: start with a new random fractal
- Right-click: toggle starting/stopping evolution
- 1,2: decrease/increase sample size (forest)
- 3,4: decrease/increase mutation rate
- 5,6: alter RED value of leaves
- 7,8: alter GREEN value of leaves
- 9,0: alter BLUE value of leaves
- Q,W: adjust trunk length
- E,R: adjust the delta-length in going from branch to sub-branch
- A,S: adjust trunk width
- D,F: adjust the delta-width in going from branch to sub-branch
- Z,X: adjust the branch angle
- C,V: adjust the number of branches
- SHIFT: make for smaller increment changes
- ARROWS: move the fractal on the screen
- B,N: change the background color

Douglas Oberle, 2024