Project 1 for GAME-625

This project is about creating generative art in Unity3D. Your implementation should create a procedurally-generated branching structure akin to a plant or tree. Just like a real tree, your art grows from its tips. In a round of generation, each of the tips should do one of the following:

1. **Grow:** Replace the current tip with a new one in the direction of growth.
2. **Split:** Add another tip near the tip that is splitting.
3. **End:** Make the current tip no longer a tip.

The grow, split and end options should chosen from in a weighted manner. One weighting (as shown in the pseudo code outline below) is to grow 94% of the time, split 5% of the time, and end 1% of the time. This makes splitting more likely than ending. As a result, your structure is likely to blossom.

Psuedo code for one round of generation

```
  for each tip in tips
    get random number [0.0-1.0]
    if rand is greater than 0 and less than or equal to .94
      make new prefab
      give orientation of tip
      give position of tip + growth distance * rotation of tip
      remove old tip from tips
      add new tip from tips
    if rand is greater than 0.94 and less or equal to 0.99
      make new prefab
      give orientation of tip * rotation
      give position of tip + growth distance * new orientation
      add new tip to tips
    else
      remove tip from tips
```

Project Structure

Your project should have the following objects in the hierarchy:

- A camera.
- A directional light.
- An empty object to house your global scripts (a good name would be "ScriptHome").

You should have a prefab called *growth* in the Assets/resources/prefabs project directory. It should contain a cube with default size and scaling. You also need a C# script named *growing* in the Assets/scripts directory.

Structure of the *growing* script.

The script should have member variables and functions of this example:

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic; //allows us to use lists

public class growing : MonoBehaviour {

  List<GameObject> tips;
  Object growthPrefab;
  public float splitAngle = 30.0f; //the angle the 2nd tip grows from on a split

  //a MonoBehavior function
  void Start () {
```

```
    //initialize list of tips
    //get a reference to the growth prefab
    //create a new instance of the growth prefab
    //add the new growth to the tips list to make it the first tip
  }

  //also a MonoBehavior function
  void Update () {
    //if there is your desired keyboard or mouse event, call Generate.
  }

  //our lovingly hand-crafted artisinal function
  void Generate() {
    //implement the pseudo code ^^
  }
}
```

Useful code fragments

Handling lists of GameObjects

```
//defining a list of GameObjects called tips
List<GameObject> tips;

//initializing tips to an empty list
tips = new List<GameObject>();

//adding a game object to the list
GameObject myGameObject = new GameObject();
tips.Add(myGameObject);

//removing a game object from the list
tips.Remove(myGameObject);

//pulling out the first game object from the list (remember that counting is 0-based)
GameObject firstFromList;
firstFromList = tips[0];

//finding how many items are in the list
int numberOfItems;
numberOfItems = tips.Count;

//looping through a list once for each item from last to first
for(i=tips.Count-1; i >= 0; i--) {
  Debug.log("I'm at item " + i + ". It prints as this: " + tips[i]);
}
```

Getting references to and instantiating prefabs in code

```
Object growthPrefab;
//the Resources.Load function only looks in the Assets/resources folder!
growthPrefab = Resources.Load("prefabs/growth");

//creating a new instance of the growth prefab
GameObject myNewGrowth;
newGrowth = (GameObject)Instantiate(growthPrefab);

//creating a new instance of the growth prefab at the origin with no rotation
GameObject newGrowthAtOrigin;
newGrowthAtOrigin = (GameObject)Instantiate(growthPrefab, new Vector3(0,0,0), Quaternion.identity);
```