CODE NINJAS®

Bronze Belt Ninja Guide

Activity 04 Prove Yourself: TrickHead

PROVE YOURSELF: TRICKHEAD

Let's add some fake platforms to make playing SketchHead a little...trickier. Open your SketchHead file and make the following changes:

1. **Duplicate the Platform.**

2. **Rename the duplicated Platform(1) to FakePlatform.**

3. **Change the color of the FakePlatform and turn off its collision.**

4. **Add the FakePlatform to the scene.**

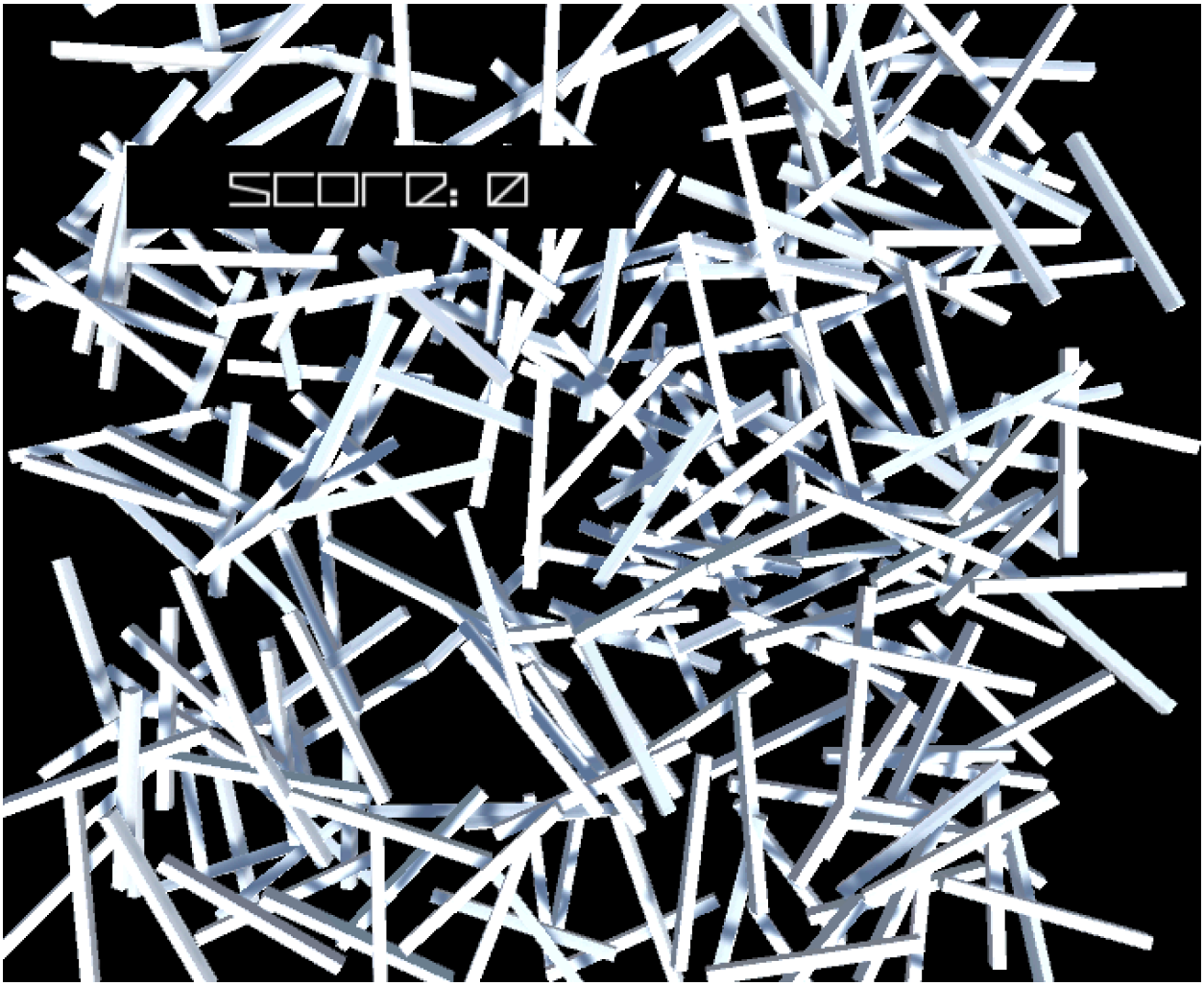
5. **Open the GameController Script and add this code inside the class before void Start:**
    a. *public GameObject fakePlatform;*

6. **Add the following code inside the void SpawnPlatforms:**
    a. *Instantiate(instance.fakePlatform, new Vector3(xPosition, instance.yPos, 0), Quaternion.identity);*

SCRIPTING

By now you probably know that scripting is an integral part of making games. Unity utilizes the **C-Sharp (C#)** coding language for all its scripts.

By now, you also probably know that while scripts have slightly different ways of using them, they all use variables, conditionals, functions, and loops. **C#** is no different from other coding languages in that aspect.



Bits and Pieces of Code

In Unity, scripts are attached to objects to tell these objects what to do. The behavior can be very specific (put the object at a specific position) or general (keep track of the time). Scripts in Unity are modular, meaning that the same script can be attached to several objects and a single object can have several scripts attached to it. The advantage of this is that a simple script (such as for movement) can be used in a variety of projects and if a project needs something more, then an additional script can be written to support the original one.

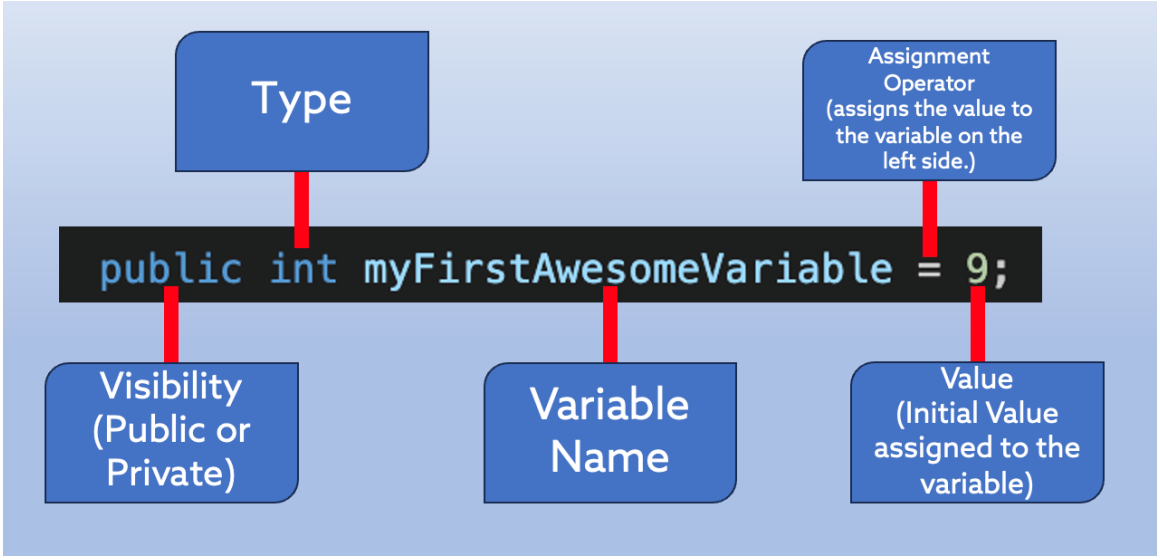
Anatomy of a Script

The main parts of a C# script are variables, functions, and classes. You will also notice that at the top of a script is something called a using directive. These specify elements of C# that the script will be using most frequently and are included by default in every new script you create. There are other directives for other things (such as the user interface) that will be explained later. For now, we will talk about variables and functions.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class Demo : MonoBehaviour
6   {
7       public int myFirstAwesomeVariable;
8       private bool mySecondCoolVariable;
9
10      // Start is called before the first frame update
11      void Start()
12      {
13
14      }
15
16      //Update is called once per frame
17      void Update()
18      {
19
20      }
21  }
```

Variables

Variables in C# have 3 main parts: **visibility** (usually public or private), **type**, and a **name**. When declaring a variable, you'll decide if the variable needs to be seen outside of the script. If you are declaring a speed variable and you want to adjust it without having to edit the script, you would make it public. Else, make it private. The type is what kind of variable. It could be an int (integer), float (floating), bool (boolean) or even a GameObject or another script. The final part is the name of the variable and, optionally, the value of that variable.



Functions

Functions are where things happen in a script. When a new script is created, two functions are automatically created, **Start()** and **Update()**. Start is called when the game starts. Update is being called constantly. All other functions must be called by name before they do anything. Like variables, functions can be public or private, depending on whether or not you need other scripts to call them. Most of the time, a script starts with a **void** type, meaning that the function is just running and not returning any data.