Notes and extra assignments for Unity beginner scripting tutorials 1-10

Software studies for game designers, Fall 2016.

The scripting tutorials can be found at: https://unity3d.com/learn/tutorials/topics/scripting

Classes and instances

The tutorials do not explain what is a *class* and what is an *instance*, which I'll briefly try to clarify here.

Consider the following code. As you learned in tutorial 3 (Conventions and Syntax), the green lines are comments that explain what the code does.

```
//Create a sphere and store a "handle" to it in a variable called myObj
GameObject myObj = GameObject.CreatePrimitive(PrimitiveType.Sphere);
//Set the created sphere's position
myObj.transform.position = new Vector3(0, 0, 0);
```

Here, GameObject is a type name, which in this case refers to a class. A real-world analogue for a class is "cat" when referring to the whole species. Games typically also use multiple instances of the same class. A real-world analogue for an instance is "cat" when referring to a specific cat. Instances are manipulated using "reference", here the myObj variable.

The Visual Studio syntax coloring shows classes in light blue, and Unity and my examples use a common naming convention where class names start with uppercase, and variable names are lowercase.

Note that classes and instances can have both code and data that you can access using the dot operator. Code and data in a class is common to all instances instead of applying just to a specific instance. Here, the GameObject class has a helper function called CreatePrimitive. The function creates a sphere instance in the current scene, and returns a reference, which we store in myObj. We can then manipulate the instance through the reference.

The "new" operator

In the example above, note that there is a "new" operator before the Vector3. The operator creates a new instance of a class and returns a reference to it – basically same as CreatePrimitive, and used for most classes other than GameObject. Unity GameObjects have some internal complexities because of which one typically does not "new" them directly.

In Javascript, one does not always need to type the "new", it's just a C# convention that you need to use "new" even for instances that are only created temporarily and then copied to other variables, as in the example above.

Extra questions/assignments

Note: in the following assignments you may need the following:

- Creating basic primitives, see the previous section
- Accessing the main camera in Unity, referenced through the Camera class as: Camera.main. For example, a Vector3 for the camera position can be accessed as Camera.main.transform.position

- Querying a vector that points to the direction of the mouse from the camera, which can be computed as: camera.ScreenPointToRay(Input.mousePosition).direction
- Adding physics to a game object using scripting:
  Rigidbody rb = myObj.AddComponent<Rigidbody>();

1 Scripts as components

Exercise: Make the cube change colour when you press the left mouse button. Try googling “unity script mouse input” to find out how. The Input class provides access to both keyboard and mouse

2 Variables and functions

Exercise: Create a new function MultiplyInts that multiplies two integers given as parameters. Test your new function using Debug.Log

6. Loops

Exercise: Create a for loop that computes the sum of numbers 2,3,4,…,20. Hint: you’ll need an extra variable for the sum

Exercise 2: Create a for loop that creates a tower of spheres

Exercise 3: Can you use loops inside loops to create a 10x10x10 stack of spheres or cubes?

Exercise 4: Same as Exercise 3, but with physics added to the objects.

Exercise 5: Create 100 randomly placed spheres. Google “Unity script random” to see how to create random numbers for the coordinates.

9. Vector math

Note: You can ignore the math in the video, just remember the helpers for magnitude, dot, and cross, as well as the left hand rules and the tank and plane examples.

Exercise: When the user clicks mouse button, create a spherical bullet that flies away from the camera to the mouse pointer direction.

Exercise 2: Use for-loops to create a wall of cubes that you can shoot with the bullets.