KINE 458: Virtual Interactive Worlds

Introduction to Unity Scripting

TOPICS FOR TODAY

- Programming Overview
  - Variables, Types, Methods
  - Conditional Statements
- Unity Programming Components
  - Game Objects
  - Rigid Body
  - Collider
  - Time

Virginia Commonwealth University
Gavin Alberghini
(alberghinig@vcu.edu)

Section One: Programming Overview

Variables

- ❑ Variables serve as ways of storing information in a program.
- ❑ Stored information is valuable because it can be reused multiple times when needed.
- ❑ Variables typically translate into some real world entity. (E.x. seconds passed, switch status, counts, lists, calculations etc.)

Types

- ❑ Variables have different types, these types are used to store different kinds of information.
- ❑ Typing is a key component in determining how variables can interact with each other.

Methods/Operators

- ❑ Operators are the actions you can perform on variables.
- ❑ Methods are subroutines of code and operators attached to complex variables.

Section One: Programming Overview

Variables

- ❑ Imagine a variable as a container
- ❑ There are three things we are concerned about with each variable:
  - ❑ The name of the container (variable name)
  - ❑ The type of container it is (variable type)
  - ❑ What the container is holding (variable value)

Var name: tomatoContainer
Var type: Square Food Container
Var value: Tomatoes

Var name: midSizedEmpty
Var type: Circle Food Container
Var value: Empty

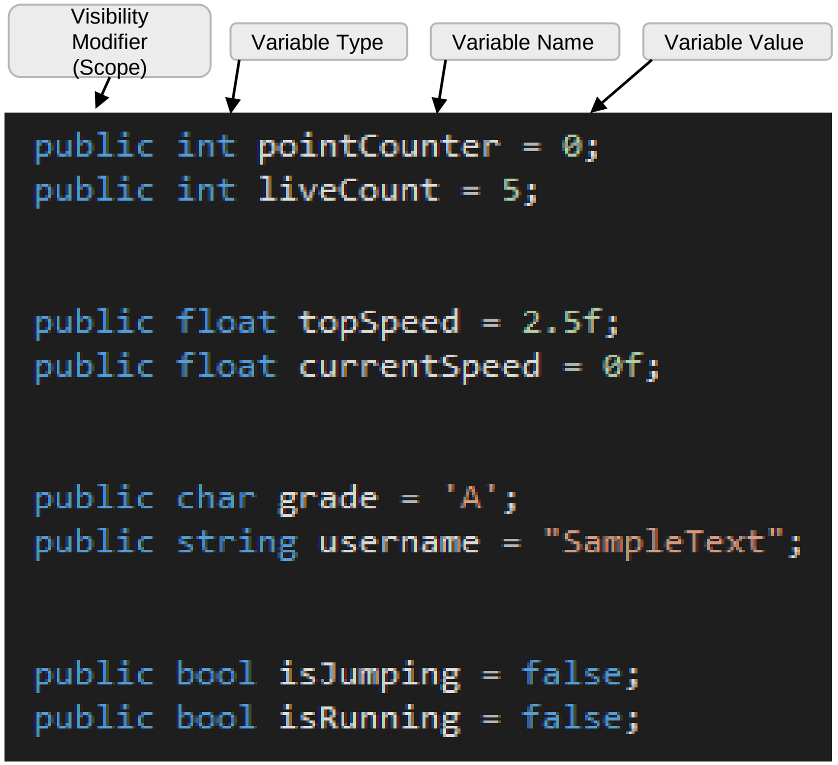
Section One: Programming Overview

Types

- ❑ The two main categories of variable types are primitive and object types. These both contain several other subtypes.
- ❑ Primitive Data Types
  - ❑ Int (E.x: 4, -5, 1298)
  - ❑ Float (E.x: 1.75, 2.15, -3245.556)
  - ❑ Boolean (E.x: true, false)
  - ❑ Char (E.x: ‘a’, ‘t’, ‘z’)
  - ❑ String (E.x: “hello”, “victory”, “yes”)
- ❑ Object Data Types
  - ❑ Vector3 (E.x: <2.5f, 10.2f, -3.2f>, <10.0f, 2.3f, 5.2f>)
    - ❑ Vector3 types represent a 3 dimensional vector within 3D space, they contain 3 float values corresponding to the x, y, and z axis
  - ❑ Rigidbody
  - ❑ GameObject
  - ❑ Collider
  - ❑ Transform
  - ❑ Time

Section One: Programming Overview

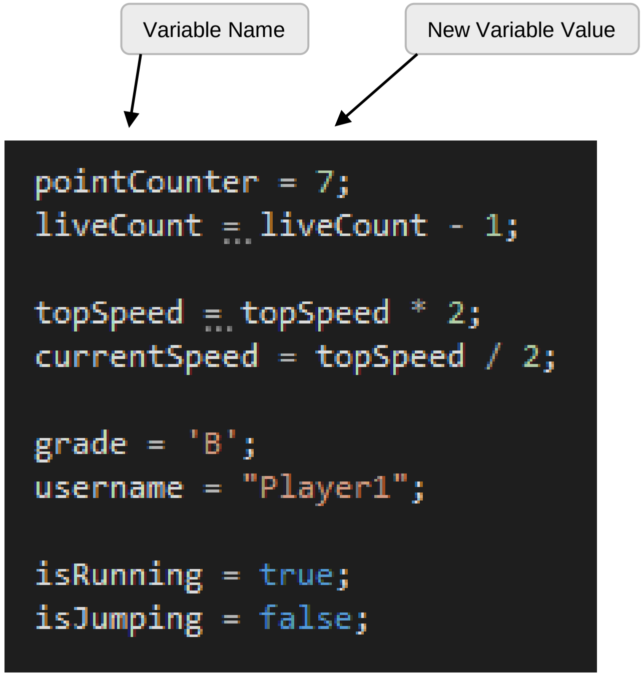
Declaring Variables & Types



```
public int pointCounter = 0;
public int liveCount = 5;


public float topSpeed = 2.5f;
public float currentSpeed = 0f;


public char grade = 'A';
public string username = "SampleText";


public bool isJumping = false;
public bool isRunning = false;
```

Section One: Programming Overview

Updating Variables

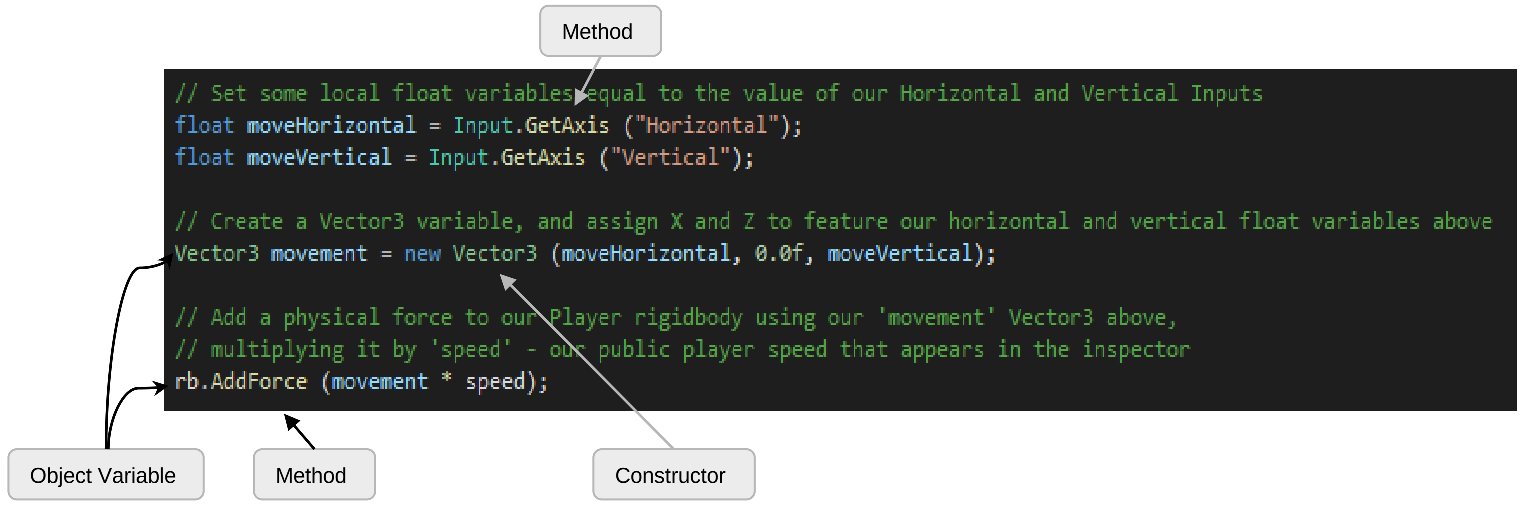


```
pointCounter = 7;
liveCount = liveCount - 1;

topSpeed = topSpeed * 2;
currentSpeed = topSpeed / 2;

grade = 'B';
username = "Player1";

isRunning = true;
isJumping = false;
```

Section One: Programming Overview

Methods/Constructors & Parameters

- Methods are associated with Object type variables. They offer subroutines of code that often assist in manipulating the variable.
- Constructors can be thought of as special methods that only get run once when an Object type variable is created.
- Some methods require parameters. These are variables and values that are used as input to a method.

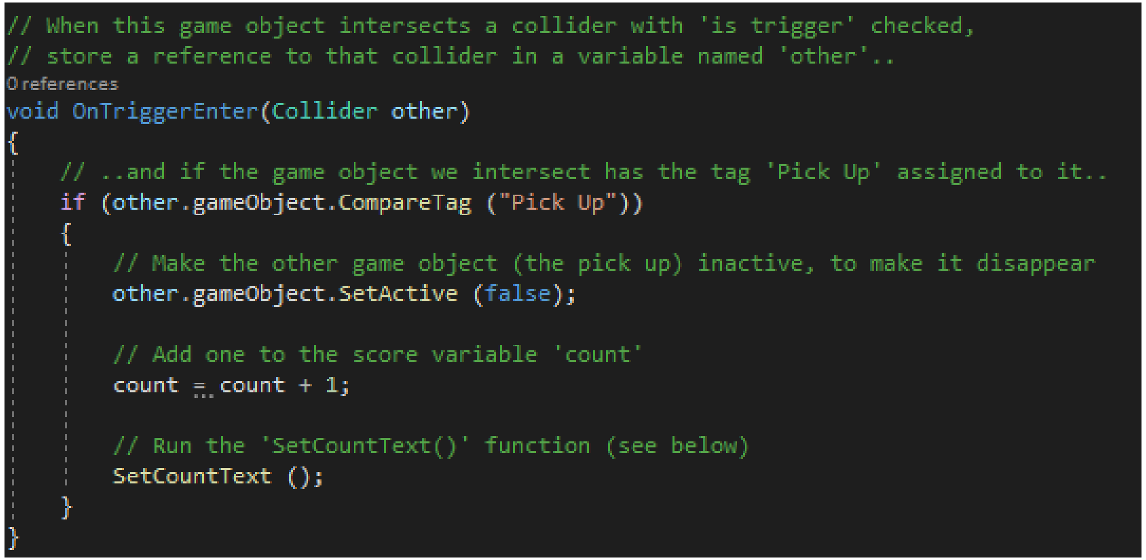


```
// Set some local float variables equal to the value of our Horizontal and Vertical Inputs
float moveHorizontal = Input.GetAxis ("Horizontal");
float moveVertical = Input.GetAxis ("Vertical");

// Create a Vector3 variable, and assign X and Z to feature our horizontal and vertical float variables above
Vector3 movement = new Vector3 (moveHorizontal, 0.0f, moveVertical);

// Add a physical force to our Player rigidbody using our 'movement' Vector3 above,
// multiplying it by 'speed' - our public player speed that appears in the inspector
rb.AddForce (movement * speed);
```

Section One: Programming Overview

Conditional Statements

- Conditional statements are tools for establishing flow control in programs. This is a fancy way of saying that we can dynamically make decisions about what portions of code to run. (E.x: If condition one is true do this, otherwise do that)



```
// When this game object intersects a collider with 'is trigger' checked,
// store a reference to that collider in a variable named 'other'..
0 references
void OnTriggerEnter(Collider other)
{
    // ..and if the game object we intersect has the tag 'Pick Up' assigned to it..
    if (other.gameObject.CompareTag ("Pick Up"))
    {
        // Make the other game object (the pick up) inactive, to make it disappear
        other.gameObject.SetActive (false);

        // Add one to the score variable 'count'
        count = count + 1;

        // Run the 'SetCountText()' function (see below)
        SetCountText ();
    }
}
```

Section Two: Roll-a-ball Live Demo

Additional Resources

Roll-a-ball Walkthrough Tutorial: https://learn.unity.com/project/roll-a-ball-tutorial?signup=true

C# For Beginners: https://www.youtube.com/playlist?list=PLPV2KyIb3jR6ZkG8gZwJYSjnXxmfPAl51

Complete Roll-a-ball Project: https://assetstore.unity.com/packages/essentials/tutorial-projects/roll-a-ball-tutorial-complete-77198

Unity hub download: https://unity3d.com/get-unity/download

Questions?