unity C# Overview

Variables and Data Types

```
int i = 5;
float x = 0.1f;
bool isDone = false;
string name = "Brian" ;
int[] array1;
int[] array2 = { 1, 3, 5, 7, 9 };
Vector3 loc = new Vector3(0,0,0);
Transform target; // or any other component type
GameObject projectile; // can be set to a gameObject or Prefab
```

Variables and Scope

```
int p; // private member variable
private int p; // also private member variable
public int p; // public member variable
public static int p; // global member variable
```

Assignment

Expression	Shortcut
x=x+1	x++
y=y-1	y--
x=x+y	x+=y
x=x-y	x-=y

Logic

Logic	Symbol
Equal	==
Not Equal	!=
Greater Than	>
Greater Than or Equal To	>=
Less Than	<
Less Than or Equal To	<=
And	&&
Or	||

Arrays

```
string[] family;
family = new string[] {"Homer","Marge","Bart","Lisa","Maggie"};
Debug.Log("Family members="+family.Length);
Debug.Log("First members="+family[0]);
foreach (string name in family) {
        Debug.Log(name);
}
```

Classes

```
public class ClassName : ClassNameExtends {
}
```

Functions

```
int functionName(int x, int y) {
  // local variable
   int result;
  // actions
  result = x + y;
  // return from function
  return result;
}
```

Conditionals

```
if (condition1) {
   // do this
} else if (condition2) {
   // do that
} else {
  // do default
}

switch(name) {
   case "brian":
      Debug.Log("Welcome Brian");
      break;
  case "will":
      Debug.Log("Welcome Will");
      break;
 default:
      Debug.Log("I don't know you");
      break;
  }
```

Loops

```
for (initialization; condition; increment) {
    // actions
  }

while (condition) {
    // actions
  }
```

unity C# and Unity API

Typical C# Script Template

```
using UnityEngine;
using System.Collections;

public class BasicTargetMover : MonoBehaviour {

	// Use this for initialization
	void Start () {

	}

	// Update is called once per frame
	void Update () {

	}
}
```

Important concepts in Unity

- MonoBehavior is the base class that all Unity scripts are derived from.
- Public variables are exposed in the editor
- Public variables and functions can be accessed when you have a reference to the component
- You can make an Object global by creating a public static variable

Referencing GameObjects

You can reference a gameObject in code in the following ways:

If the script is attached to the gameObject, you can reference the gameObject as:

`this.gameObject` or simply `gameObject`

If we want to reference a gameObject that the script is NOT attached to, we can do this in a number of ways:

We can have a public variable of type GameObject that we attach to another gameObject or prefab in the Unity Editor.

```
public GameObject target;
```

We can search for the game Object using GameObject.Find to search for a GameObject by name:

```
GameObject target = GameObject.Find("Enemy");
```

We can search for the game Object using GameObject.FindWithTag to search for a GameObject with a particular tag:

```
GameObject target = GameObject.FindWithTag("Player");
```

Referencing Components

You can get a reference to a component in the following ways:

Setup a public variable that holds the reference that you set in the Editor, such as:

```
public AudioSource backgroundMusic;
```

Get the component through a reference to a gameObect, such as:

```
gameObject.GetComponent<AudioSource>();
```

Instantiating Prefabs

You can dynamically create, or instantiate gameObjects in a scene from other gameObjects or Prefabs:

```
GameObject spawnedObject = Instantiate
(prefabObject, spawnPosition, spawnRotation) as
GameObject;
```

Common Game Events
Awake & Start - called once. This is where you can set things up.

Update - is called every game cycle before rendering a frame. This is where most game behaviour code goes, except physics code.

FixedUpdate - is called once every physics time step. This is the place to do physics-based game behaviour.

OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.

OnTriggerEnter is called when this collider has touched another collider that is tagged as a trigger.

Useful API Classes
GameObject, Time, Transform, RigidBody, AudioSource
(look in documentation for details and other UnityEngine classes)