Charles Sturt University

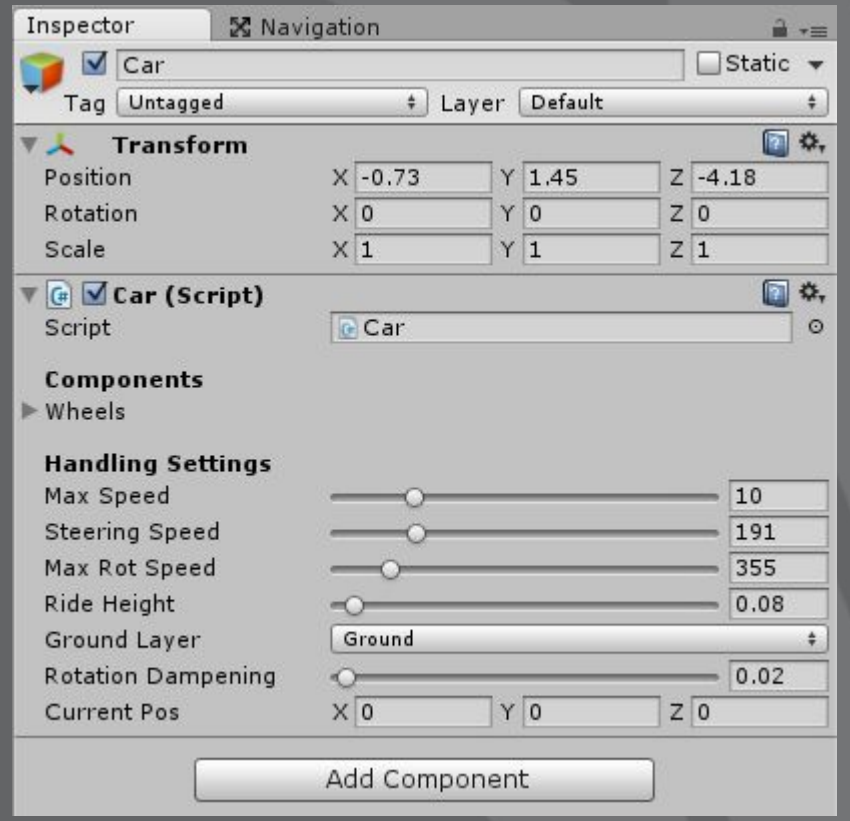
IT Masters
itmasters.edu.au

# Unleash your inner Game Dev
# ITE529 Free Online Course

## Week 2

# Coding Basics

1. Scripts are added to objects
2. Scripts can be used independently of objects
3. Scripts can talk to each other
4. Exposed variables can be a fast way to optimise and tweak gameplay



# Coding Basics - Variables

Variables are where we store data that we need.

Standard types include

- Integers (int)
- Floating Point Numbers (float)
- Vectors (multiple floats)
- Pointers
- Objects (gameObject)
- Many more..

```
4  public class Car : Photon.MonoBehaviour {
5
6      [Header("Components")]
7      [SerializeField]
8      public Wheel[] wheels;
9      CarMotor carMotor;
10     CarInput carInput;
11     PhotonView photonView;
12
13     [Header("Handling Settings")]
14     [Range(0, 50)]
15     public float maxSpeed = 5f;
16     [Range(50, 720)]
17     public float steeringSpeed = 10f;
18     [SerializeField]
19     [Range(250,1000)]
20     public float maxRotSpeed = 360.0f;
21     [SerializeField]
22     [Range(0,2f)]
23     public float rideHeight = 0.025f;
24     [SerializeField]
25     LayerMask groundLayer;
26     [Range(0,1f)]
27     public float rotationDampening = 1f;
```

# Coding Basics - Methods

Methods are where we perform actions. Methods can take input, provide output, etc.

Unity scripts by default inherit from MonoBehavior, and by default, it will automatically trigger its own methods..

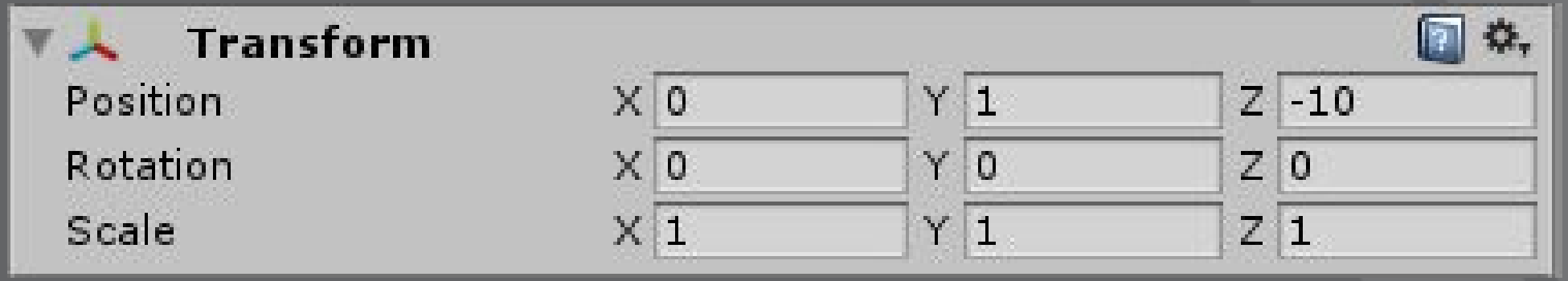
1. Awake
2. Start
3. Update
4. LateUpdate
5. OnEnable..

```
// Use this for initialization
void Start () {
    // Component Setup
    carMotor = gameObject.AddComponent<CarMotor>();
    carMotor.InitMotor(rideHeight, wheels, groundLayer);
    carInput = gameObject.AddComponent<CarInput>();
    carInput.InitCameraOffset(Camera.main.transform.localEulerAngles.y - 90f);
    photonView = GetComponent<PhotonView>();
}
```

# Coding Basics - Movement

All Unity objects exist in euclidean space. To represent this, all game objects within Unity have a *Transform* component.

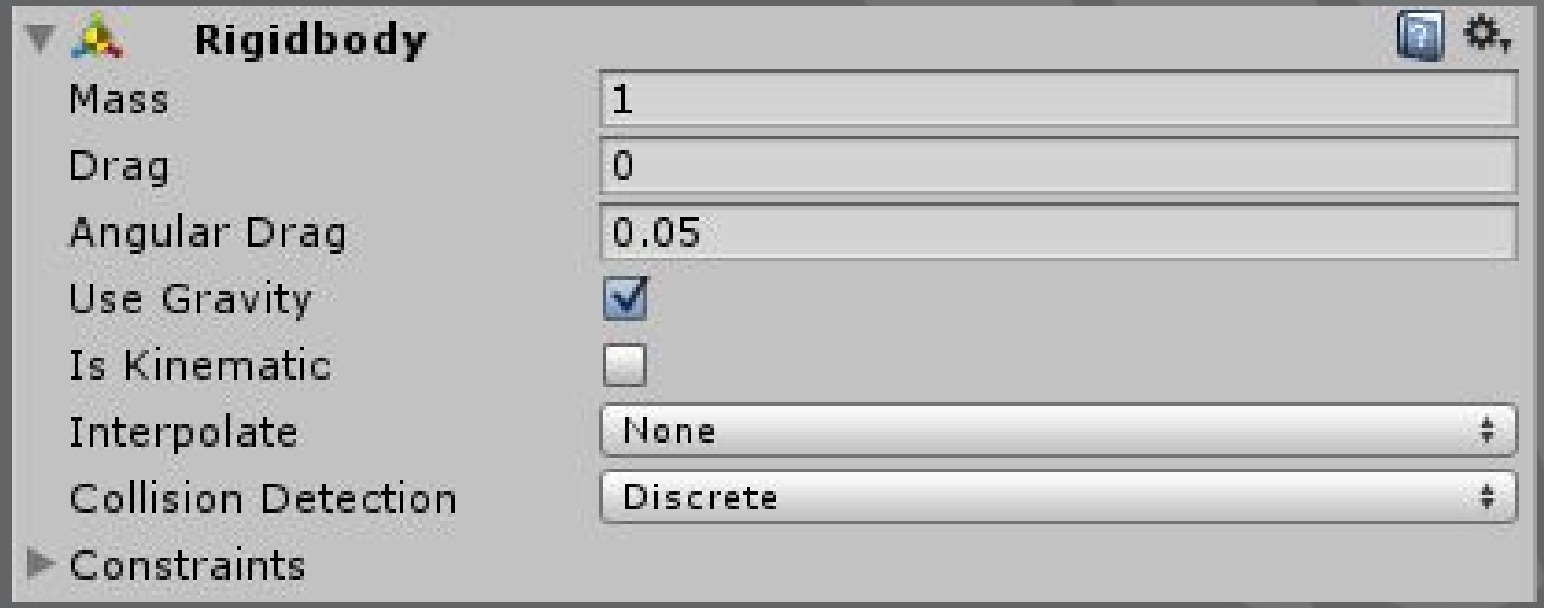
Moving an object in through space means modifying it's x/y/z position.



# Coding Basics - Movement

There are a few conventions for moving objects..

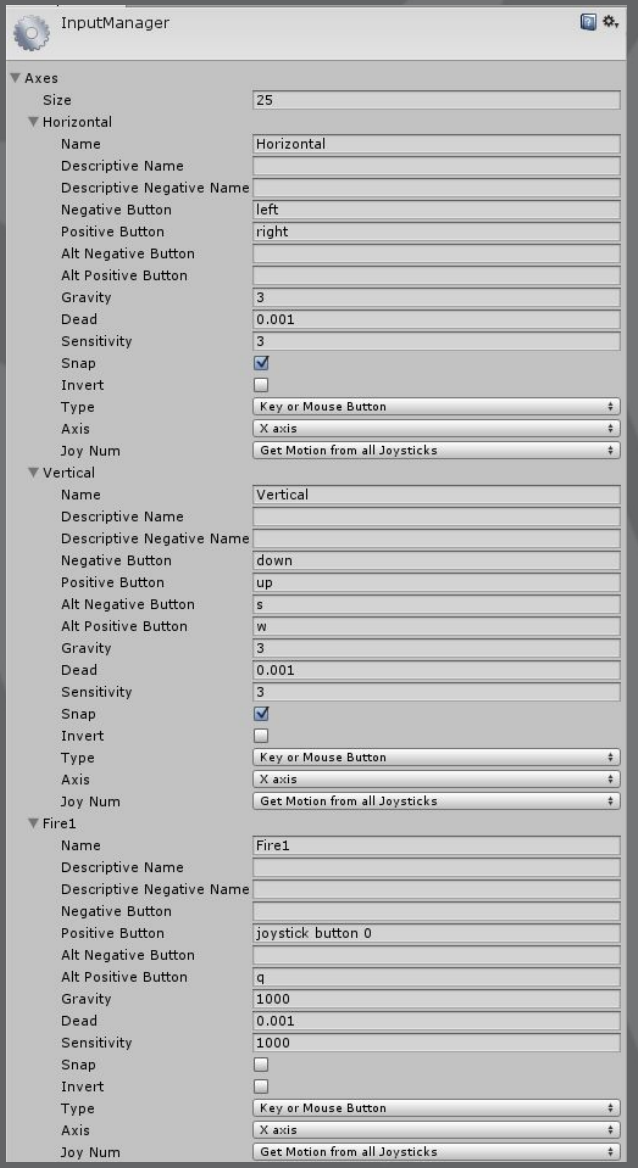
1. Manually altering the position in Update()
2. Using a Rigidbody
3. UI Positioning
4. Animation
5. Bones



# Coding Basics - Input

There are many ways to get input..

1. Touch
2. Mouse
3. Joystick
4. Steering Wheels
5. Keyboard(s)
6. Other..

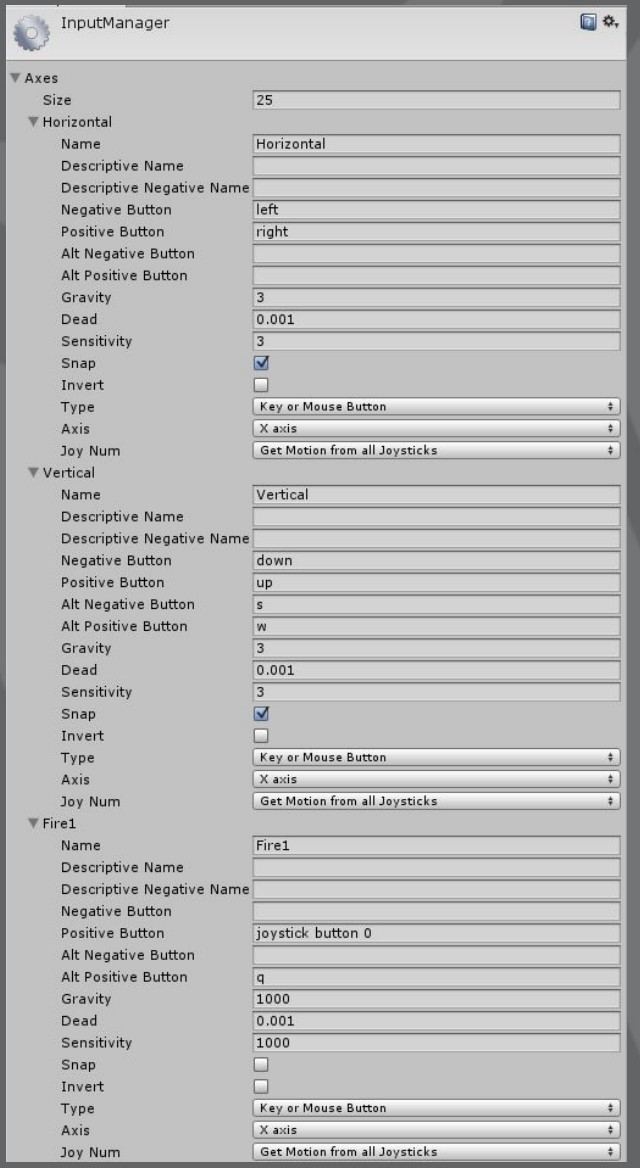


# Coding Basics - Input

The InputManager is designed to simplify multiple forms on input and allow for simple customisation.

The resulting triggers and axis are easily called from code.

The input manager does not work for pointers (touch screen, light guns, mouse control).



# Coding Basics - Input

An example of handling both keyboard/customised input with hardware specific input..

```
if (CrossPlatformInputManager.GetButton("Wheel_DriftTrigger") || CrossPlatformInputManager.GetButton("Fire1"))
{
    if (!hasSelected)
    {
        hasSelected = true;
        Debug.Log("Selection locked in on car " + currentCar);
        transform.Find("CharacterCarouselPanel").gameObject.SetActive(false);
    }
}
```

# Coding Basics - Input

Let’s find out what input the user is touching..

# Coding Basics - Movement

Let’s make our character move!

# Prefabs

Let’s flesh out our scene with modular pieces.

# Decoration

Let’s flesh out our level.

# Homework

Using the sprites provided (or your own), create enemies!

# Further Reading

More on the Input Manager
http://docs.unity3d.com/Manual/class-InputManager.html

More on using Sprites
https://unity3d.com/learn/tutorials/modules/beginner/2d/sprite-editor

More on Rigidbody 2D
http://docs.unity3d.com/ScriptReference/Rigidbody2D.MovePosition.html