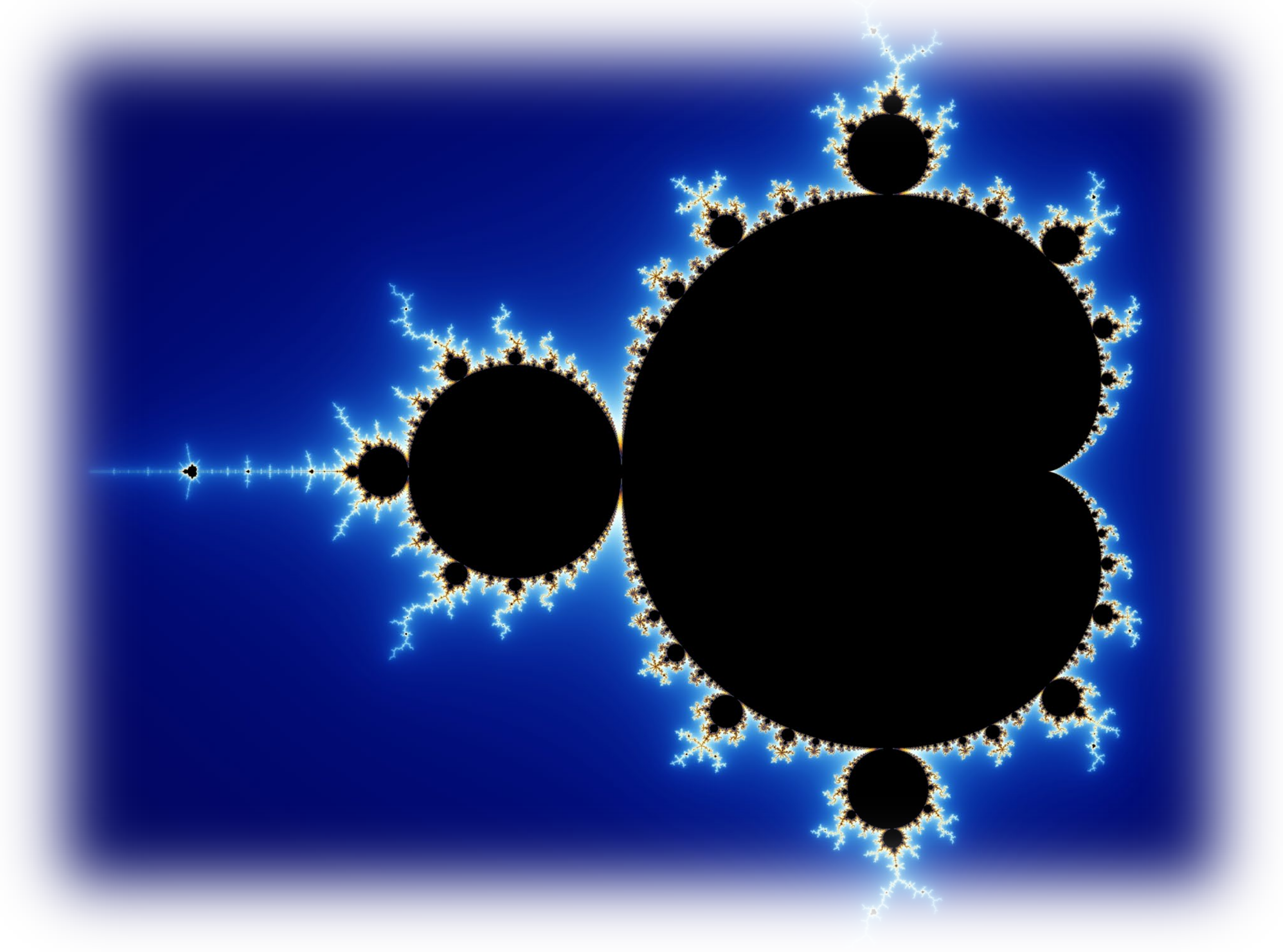
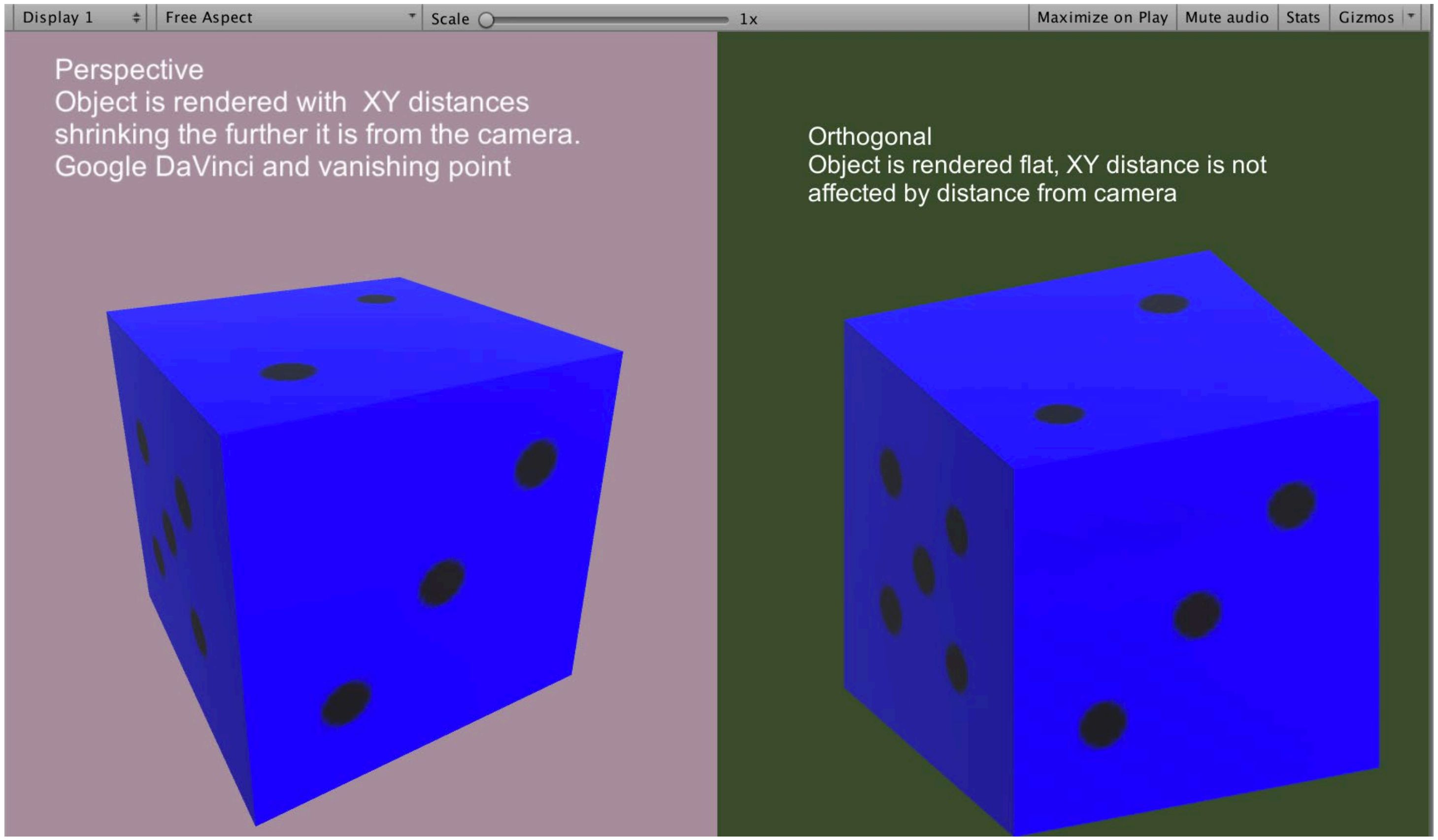


INTRODUCTION TO PROGRAMMING

Ci410

THE CAMERA



Perspective
Object is rendered with XY distances shrinking the further it is from the camera.
Google DaVinci and vanishing point

Orthogonal
Object is rendered flat, XY distance is not affected by distance from camera

VECTOR INTERLUDE

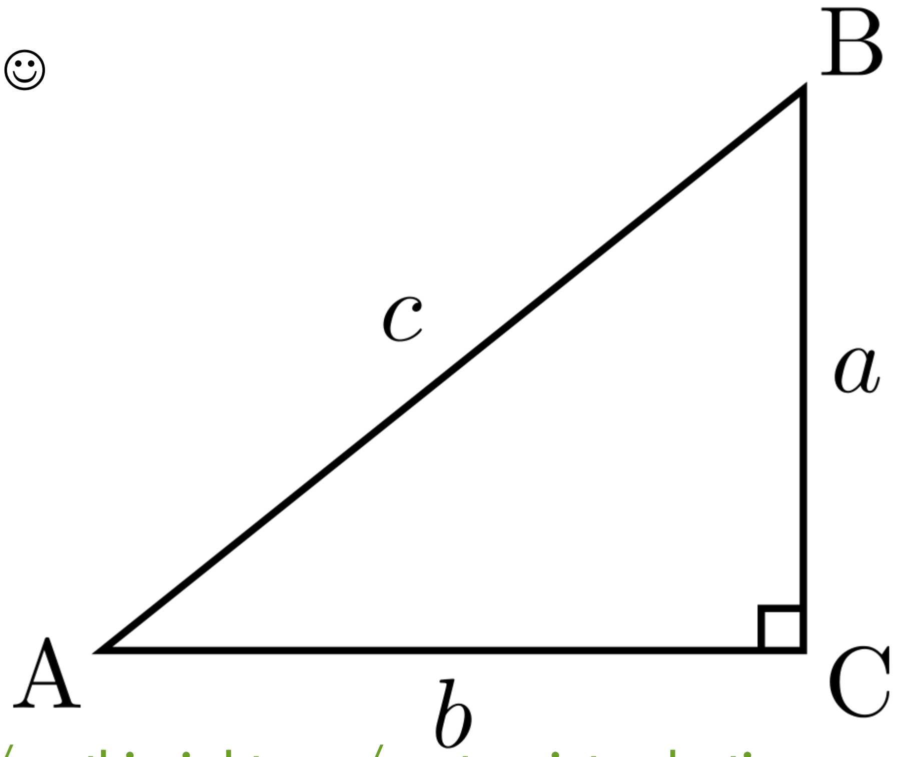
Bet you all loved Vectors at school?

Well they are back with a vengeance ☺

Vector3(X,Y,Z)

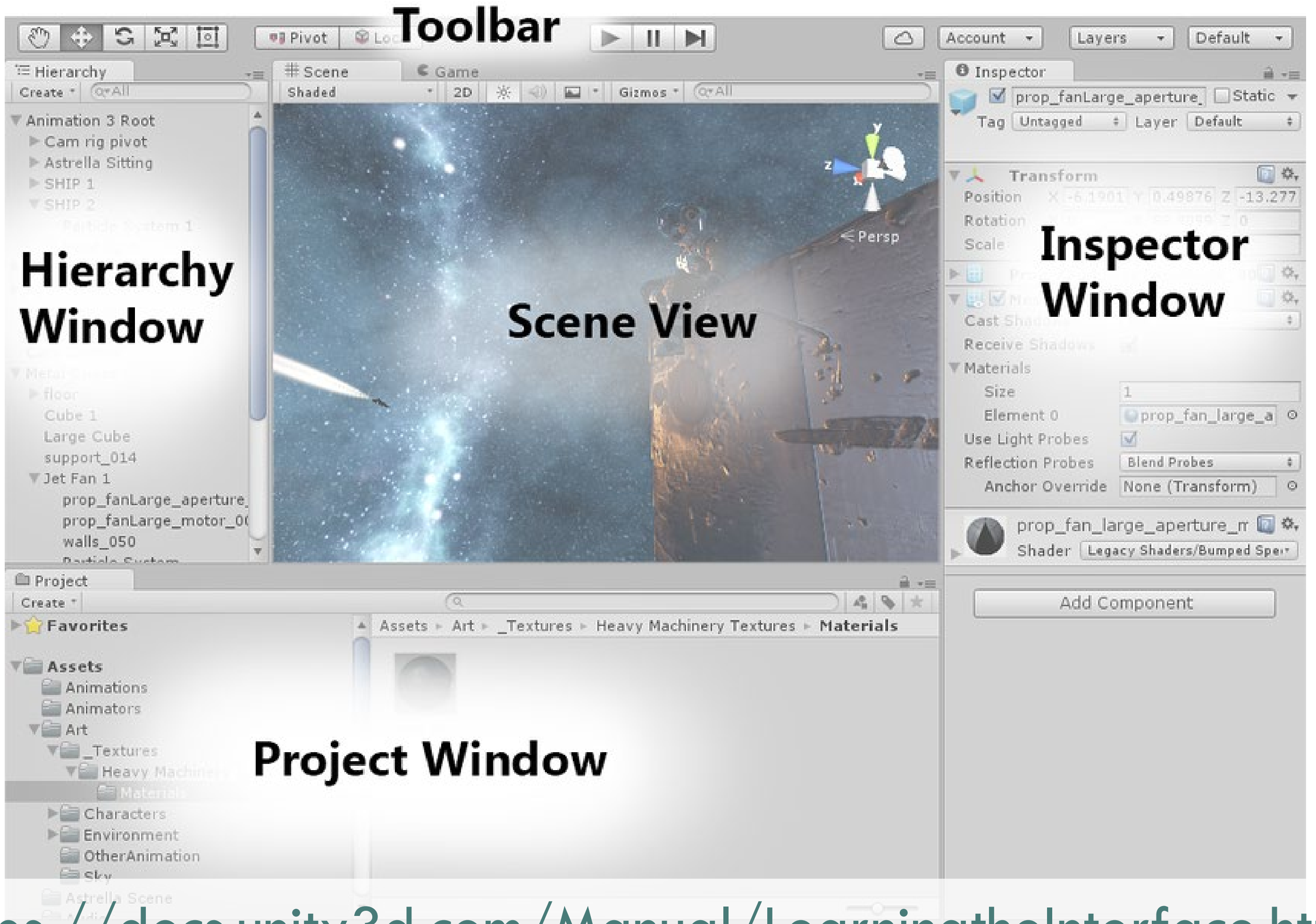
- Vector2(X,Y)
- Vector4(X,Y,Z,W)

position, force, velocity = Vector3()



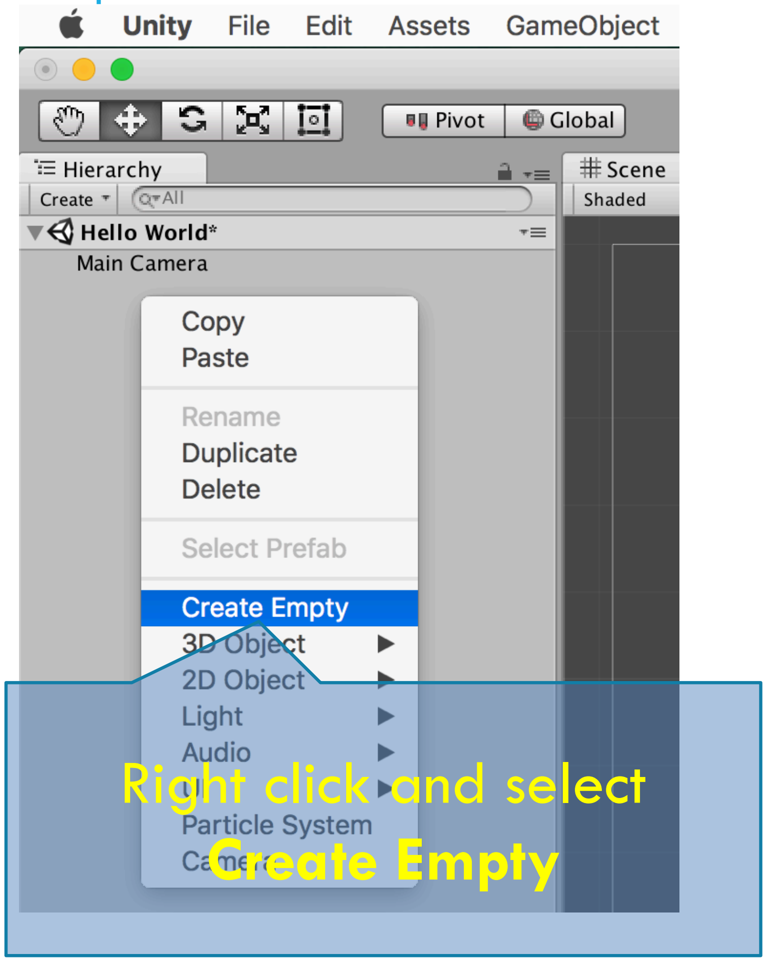
http://mathinsight.org/vector_introduction

THE INTERFACE

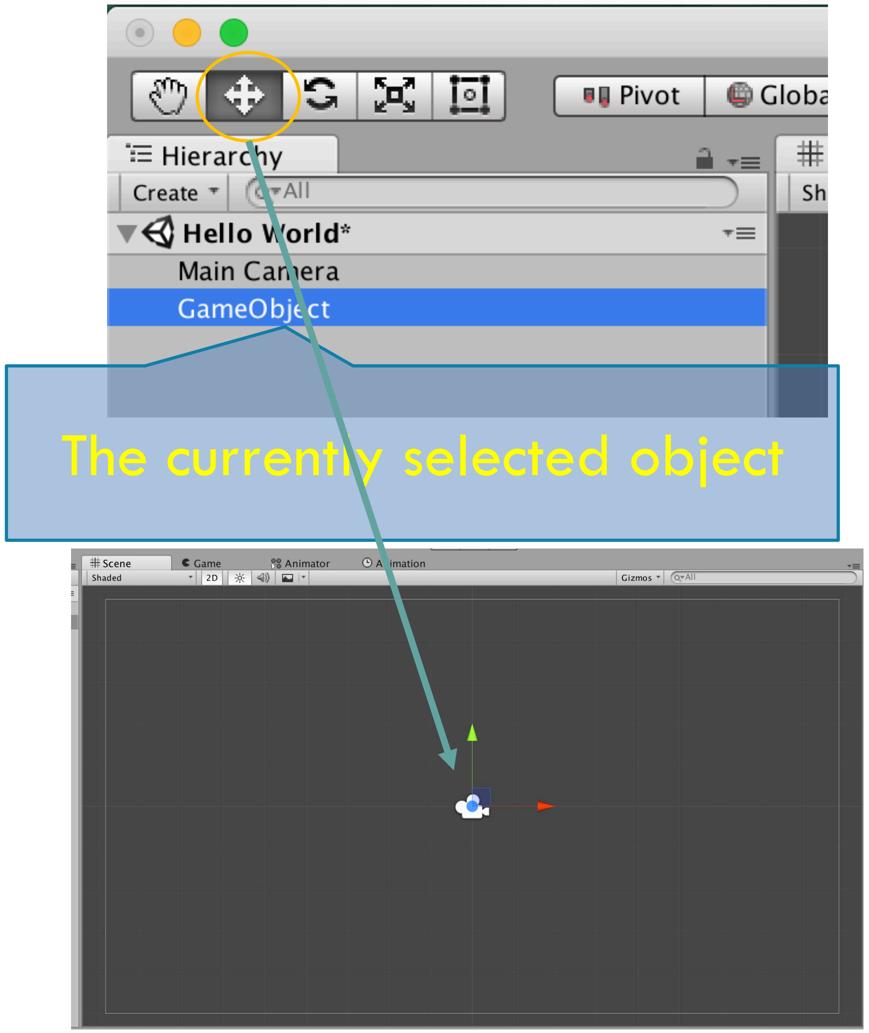


https://docs.unity3d.com/Manual/LearningtheInterface.html

EVERYTHING IS AN OBJECT



Right click and select **Create Empty**

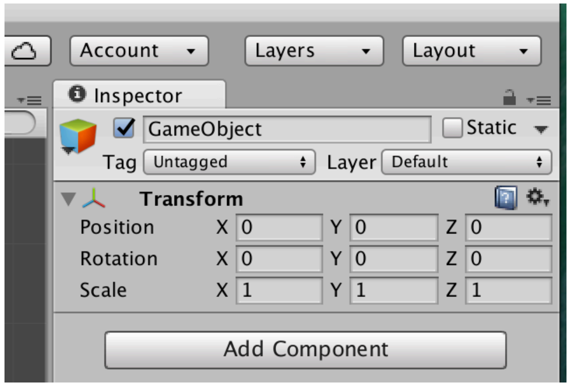


The currently selected object

THE GAMEOBJECT

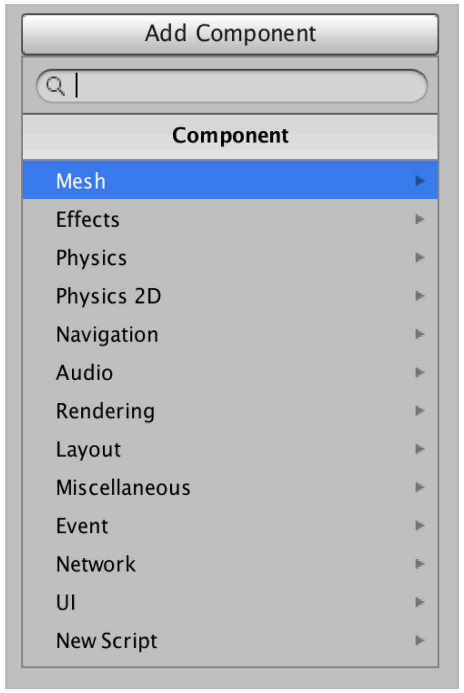
Mandatory

- Transform
  - Position, Rotation, Scale

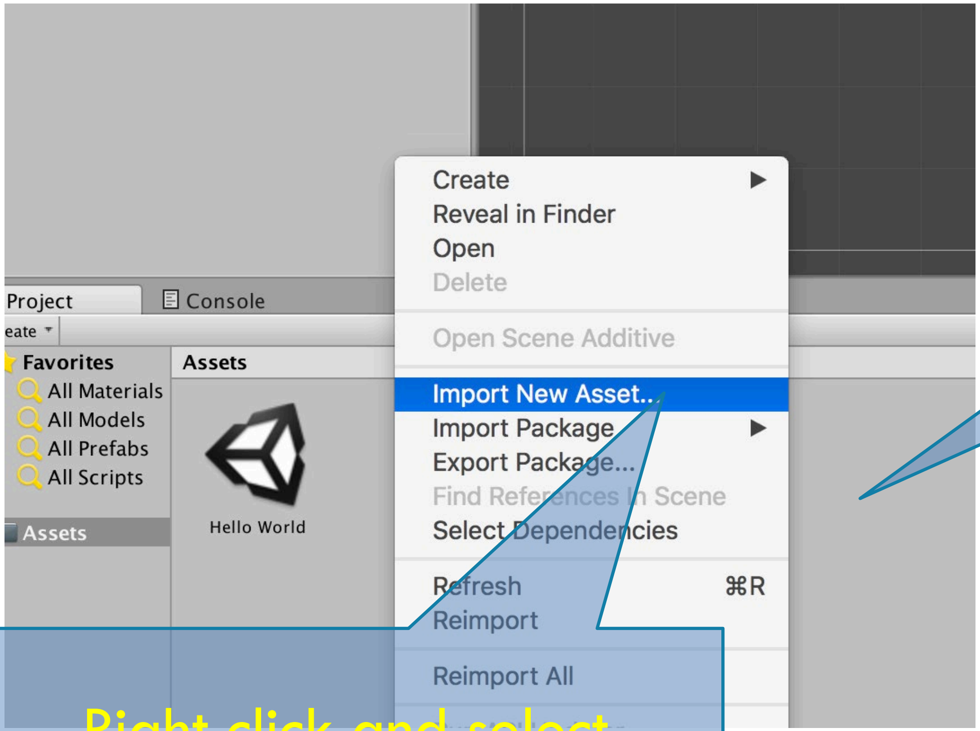


	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

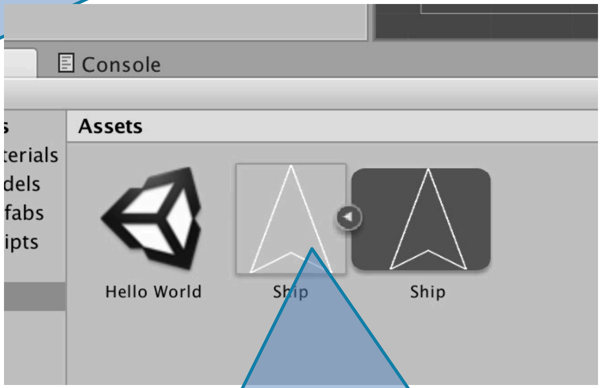
Optional



ADDING A SPRITE



Or drag image file to Asset area



Right click and select **Import New Asset**

Just added a ship and clicked the reveal button to show detail

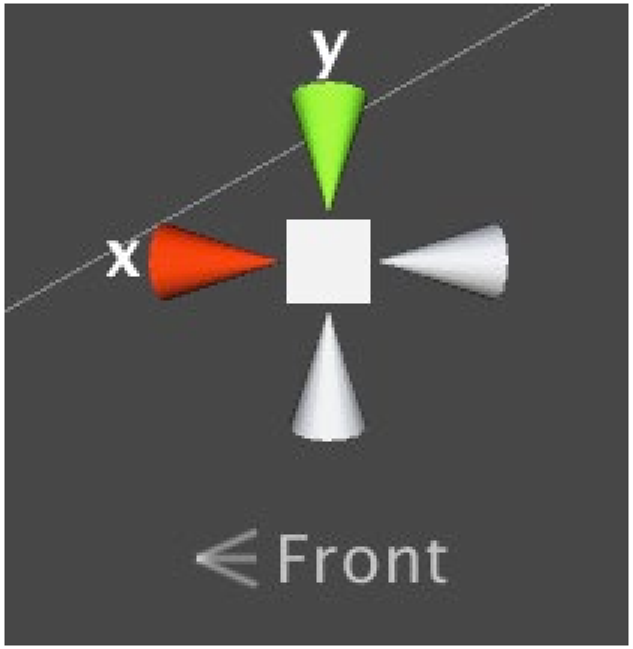
SCALE

Unity Default scales

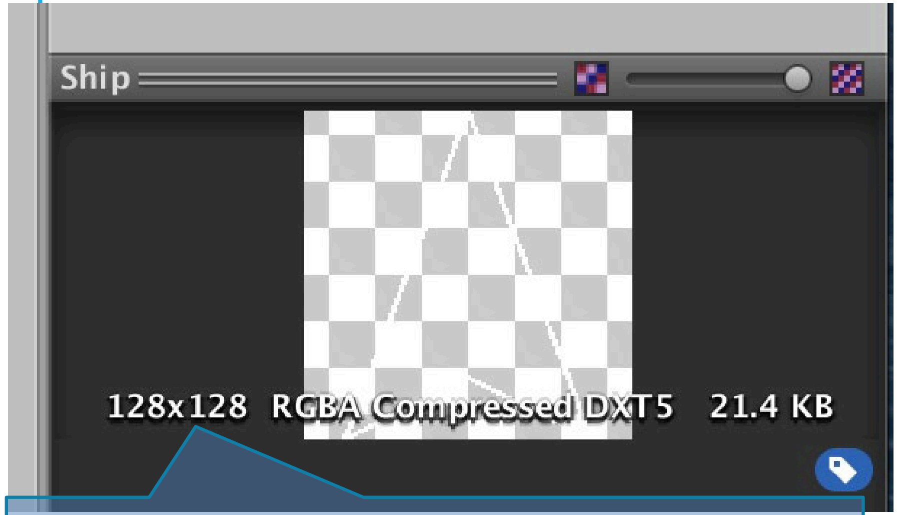
- Grid 1 x 1 x 1 meters
- Important when you import assets
- Critical for Physics

In 2D mode

- X Horizontal
- Y Vertical
- Z into the screen, without perspective

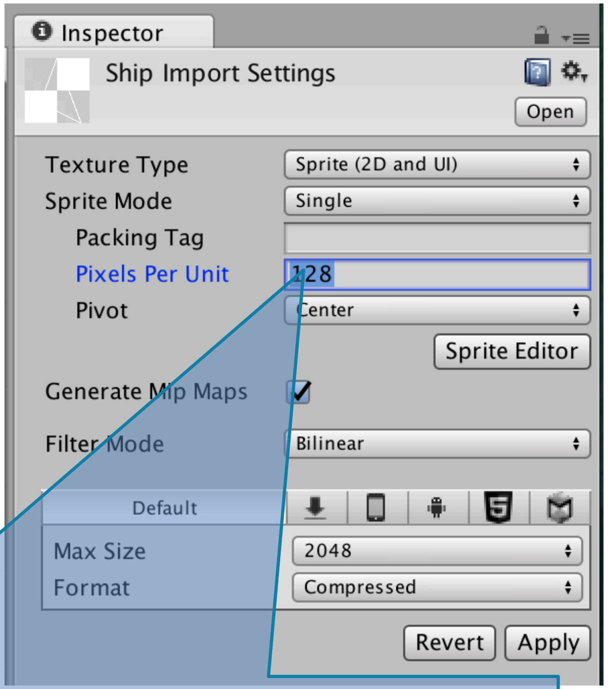


GETTING THE CORRECT SCALE


Ship

128x128 RGBA Compressed DXT5 21.4 KB

Imported image pixel size


Inspector

Ship Import Settings

Open

Texture Type	Sprite (2D and UI)
Sprite Mode	Single
Packing Tag	
Pixels Per Unit	128
Pivot	Center
Generate Mip Maps	✓
Filter Mode	Bilinear

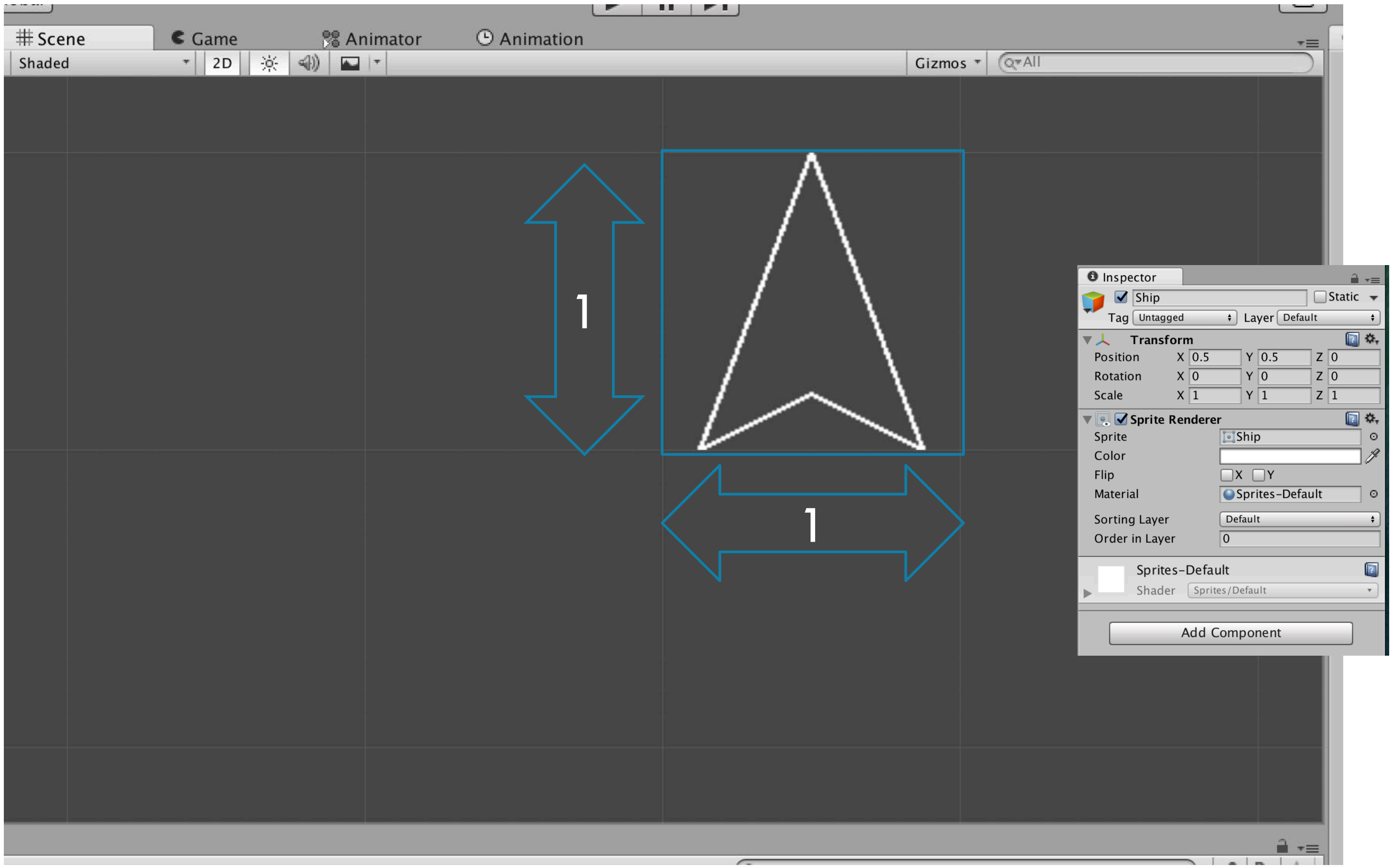
Max Size	2048
Format	Compressed

How many pixels to use per Unity "Unit"

If set to the image size the sprite will be show covering 1 x 1 unit

You will need to change the default from 100 in most cases to make it look right

THE GRID



Ship Static
Tag Untagged Layer Default

Transform

Position	X	0.5	Y	0.5	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

Sprite Renderer

Sprite	Ship
Color	
Flip	☐ X ☐ Y
Material	Sprites-Default
Sorting Layer	Default
Order in Layer	0

Sprites-Default
Shader Sprites/Default

WORLD SPACE

Every GameObject can be positioned using (X,Y,Z) position centred at the origin (0,0,0)

- Unity is a Cartesian coordinate system

Objects can be rotated and scaled

- Scale is (X,Y,Z) with (1,1,1) as the default
- Rotation is (X,Y,Z) with (0,0,0) as the default

Objects may have other GameObjects as parents to form a hierarchy of GameObjects

https://docs.unity3d.com/Manual/PositioningGameObjects.html

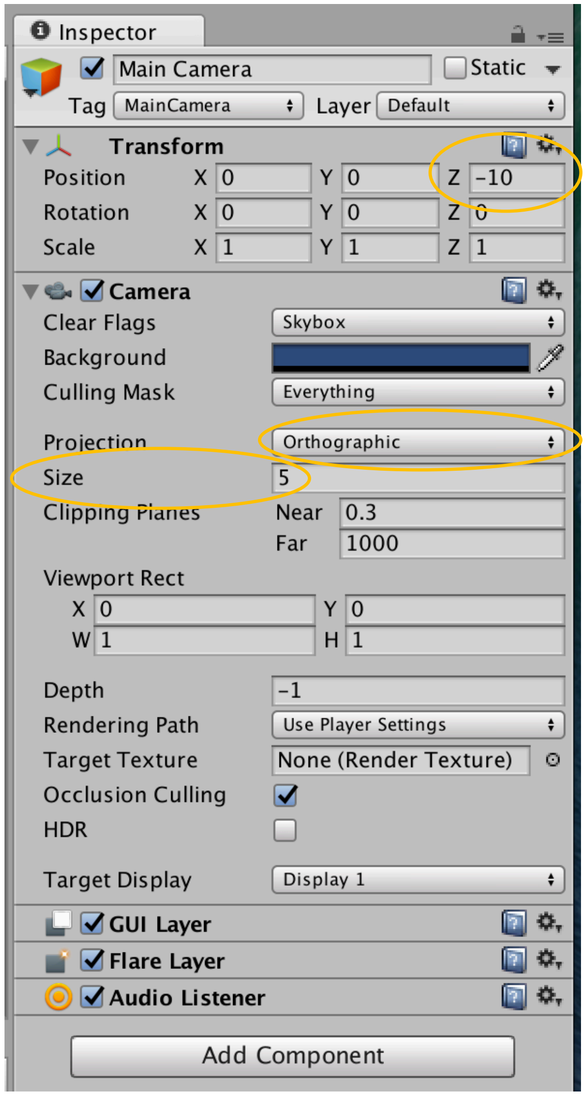
THE CAMERA

In 2D mode the default camera is Orthographic

Size = Height / 2

Its set at Z=-10 facing towards 0,0,0 to it can see the scene

If your objects don't show 9/10 times your camera is pointing the wrong way


Inspector

Main Camera ✓ — Static ☐
Tag MainCamera — Layer Default

Transform

Position	X	0	Y	0	Z	-10
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

Camera ✓

Clear Flags	Skybox
Background	🟦
Culling Mask	Everything
Projection	Orthographic
Size	5
Clipping Planes	Near 0.3
	Far 1000
Viewport Rect	X 0, Y 0
	W 1, H 1
Depth	-1
Rendering Path	Use Player Settings
Target Texture	None (Render Texture)
Occlusion Culling	✓
HDR	☐
Target Display	Display 1

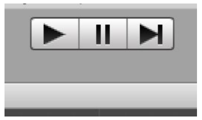
GUI Layer ✓
Flare Layer ✓
Audio Listener ✓

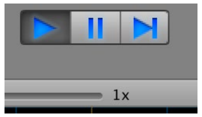
WARNING

Settings made in the IDE when not in game mode are saved in the scene file

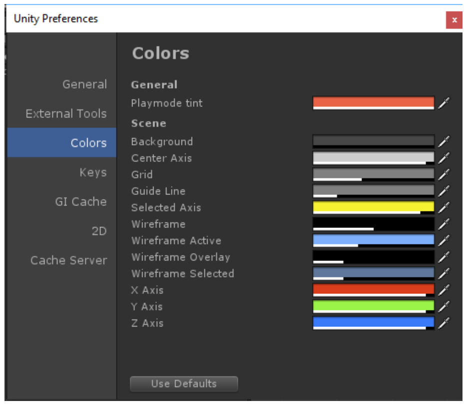
- If Unity crashes you loose any settings you did not save to a scene

Settings made while in game mode will **<u>NOT</u>** be saved, you cannot save the scene file while in game mode





Hint:Set Run tint


Unity Preferences

Colors

General
Playmode tint

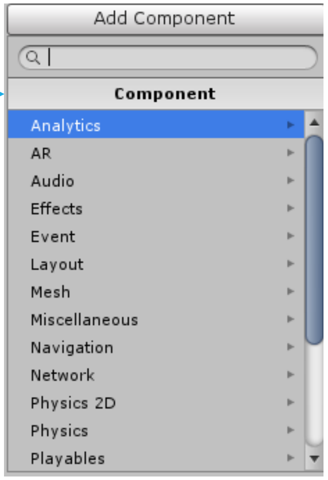
Scene
Background
Center Axis
Grid
Guide Line
Selected Axis
Wireframe
Wireframe Active
Wireframe Overlay
Wireframe Selected
X Axis
Y Axis
Z Axis

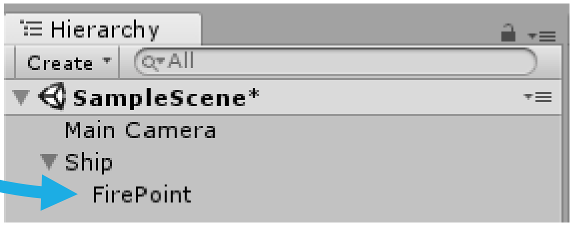
ADDING FUNCTIONALITY TO GAMEOBJECTS

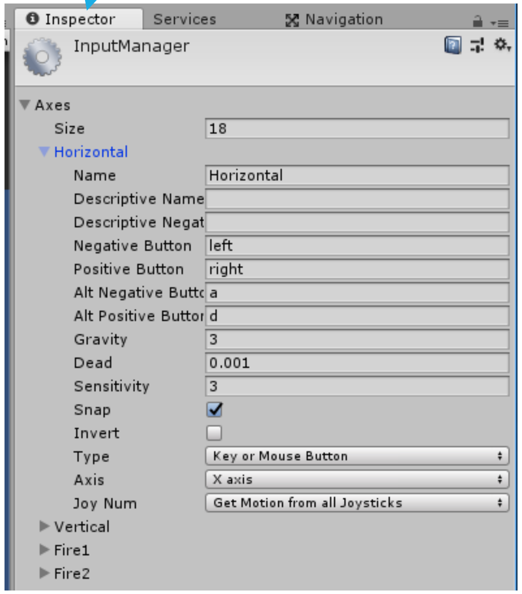
Prebuild components

Child objects

User Input







Axes

Size	18
Horizontal	
Name	Horizontal
Descriptive Name	
Descriptive Negat	
Negative Button	left
Positive Button	right
Alt Negative Buttc	a
Alt Positive Buttor	d
Gravity	3
Dead	0.001
Sensitivity	3
Snap	✓
Invert	
Type	Key or Mouse Button
Axis	X axis
Joy Num	Get Motion from all Joysticks

Vertical

Fire1

Fire2

CODING IN UNITY

Choice of Javascript or C#

We will use C# as its is a strongly typed language, based on C++ and Java.

It uses classes & inheritance throughout

In C# each file is a class, it has to be named the same as the class

In Unity most classes you create will inherit from MonoBehaviour

https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

It contains the core code you need to allow you GameObjects (GO) to be displayed and moved, any additional functionality is provided by components & your own code

THE DEFAULT UNITY CLASS

At the start of the file we need to tell the compiler which other code files (libraries) we are using

If they are greyed out, it means that they are not yet needed, but DON'T remove them

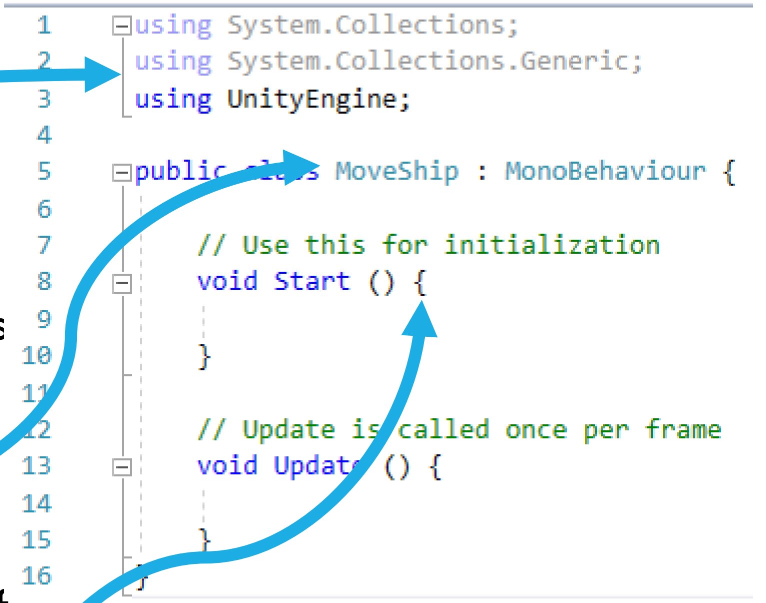
Next we have the class template, it encapsulates data & code, in this case there is just code.

public class MoveShip : MonoBehaviour

Public = any other class can access it

MoveShop : MonoBehaviour means it's a child of MonoBehaviour

{} signify a the start / end of a section of code. These can be nested



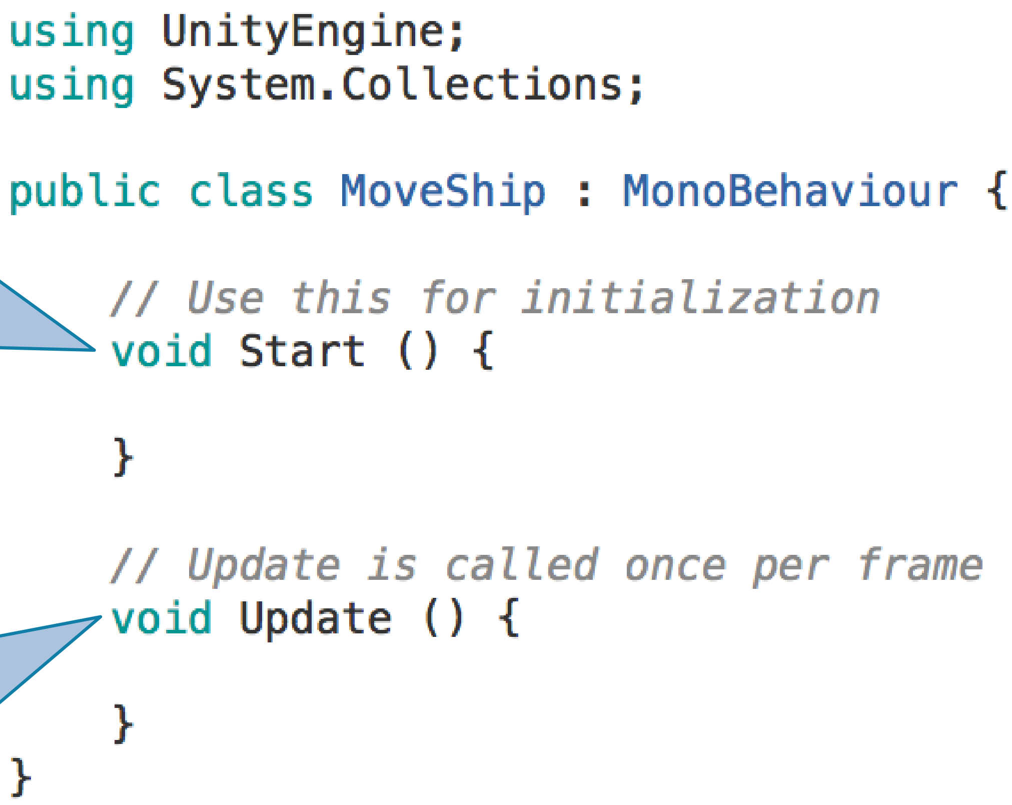
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MoveShip : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

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

    }
}
```

DON'T CALL US WE'LL CALL YOU

MonoBehaviour calls you when its ready for you to do your stuff.

Called **once** when this GameObject is ready to start, before its drawn

Called when the game object should be updated, **each time** before drawing it



```
using UnityEngine;
using System.Collections;

public class MoveShip : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

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

    }
}
```

READING MY CODE*

I tend to use the following conventions, which may differ from others

1. I don't prefix types
2. I do prefix scope

   tRotate //Temp variable has tXXX, I treat these as throw away

   mRotate // typically a member variable in a class

   vRotate // a variable passed to a function

   sRotate //Static variable with global scope

   Rotate //No prefix means I either forgot to prefix, or its an IDE editable variable

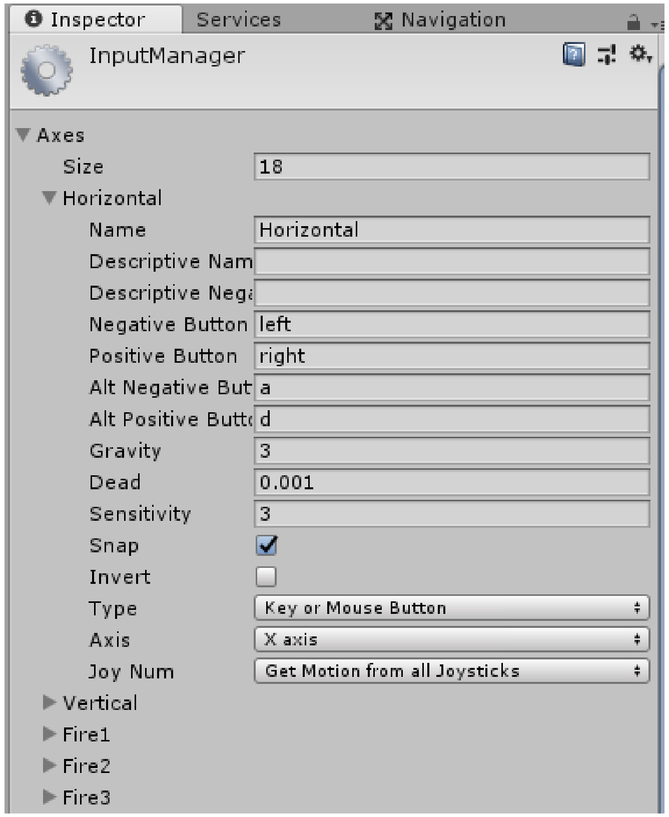
   *The good thing about standards is that there are so many to choose from ☺

USER INPUT

Raw events are mapped to input events, so multiple input options for each are supported

Project Settings ➔ Input allows you to map the mouse, joystick, pad, keys to named events

We will use Horizontal & Vertical in the workshop



▼ Axes

Size	18
▼ Horizontal	
Name	Horizontal
Descriptive Nam	
Descriptive Nega	
Negative Button	left
Positive Button	right
Alt Negative But	a
Alt Positive Butt	d
Gravity	3
Dead	0.001
Sensitivity	3
Snap	✓
Invert	
Type	Key or Mouse Button
Axis	X axis
Joy Num	Get Motion from all Joysticks

▶ Vertical
▶ Fire1
▶ Fire2
▶ Fire3

GAMES ARE MADE BY

1. Moving objects
2. Objects colliding & interacting
3. User input driving the player
4. AI driving the NPC’s

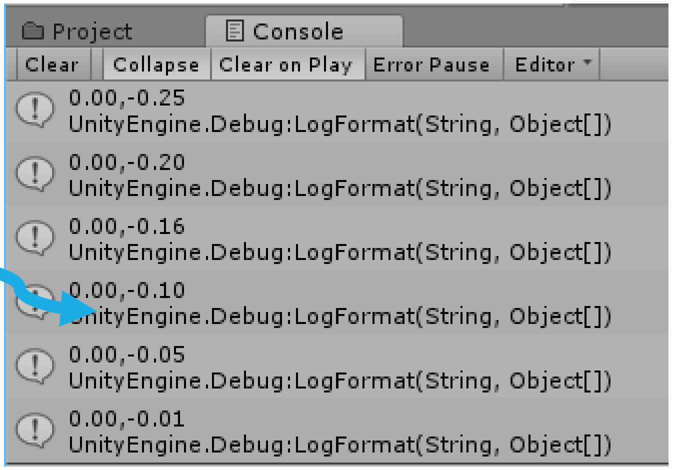
Its all about manipulating numbers in 2D or 3D

Code drives this, we will use a language called C#

READING INPUT & DEBUG

Debug.LogFormat(); is your main sanity check

```
// Move players ship
void Update () {
    float tHorizontal = Input.GetAxis("Horizontal");
    float tVertical = Input.GetAxis("Vertical");
    Debug.LogFormat("{0:f2},{1:f2}", tHorizontal, tVertical);
}
```



```
0.00,-0.25
UnityEngine.Debug:LogFormat(String, Object[])
0.00,-0.20
UnityEngine.Debug:LogFormat(String, Object[])
0.00,-0.16
UnityEngine.Debug:LogFormat(String, Object[])
0.00,-0.10
UnityEngine.Debug:LogFormat(String, Object[])
0.00,-0.05
UnityEngine.Debug:LogFormat(String, Object[])
0.00,-0.01
UnityEngine.Debug:LogFormat(String, Object[])
```

C# string formatting rules, there are 100's

http://www.cheat-sheets.org/saved-copy/msnet-formatting-strings.pdf

{0:f2} means, take the first parameter and print it as a floating point number with 2 decimal places

ROTATING THE SHIP

Rotate the ship

```
float tHorizontal = Input.GetAxis("Horizontal"); //Get the inputs for Horz & Vert
float tVertical = Input.GetAxis("Vertical");
transform.Rotate(0, 0, tHorizontal);
```

NB:Update is not at a fixed rate, so to make it smoother we can use Time.deltaTime

Time.deltaTime = the number of seconds since last Update()

```
transform.Rotate(0, 0, tHorizontal*Time.deltaTime);
```

Now we rotate too slow as each degree is 1 second

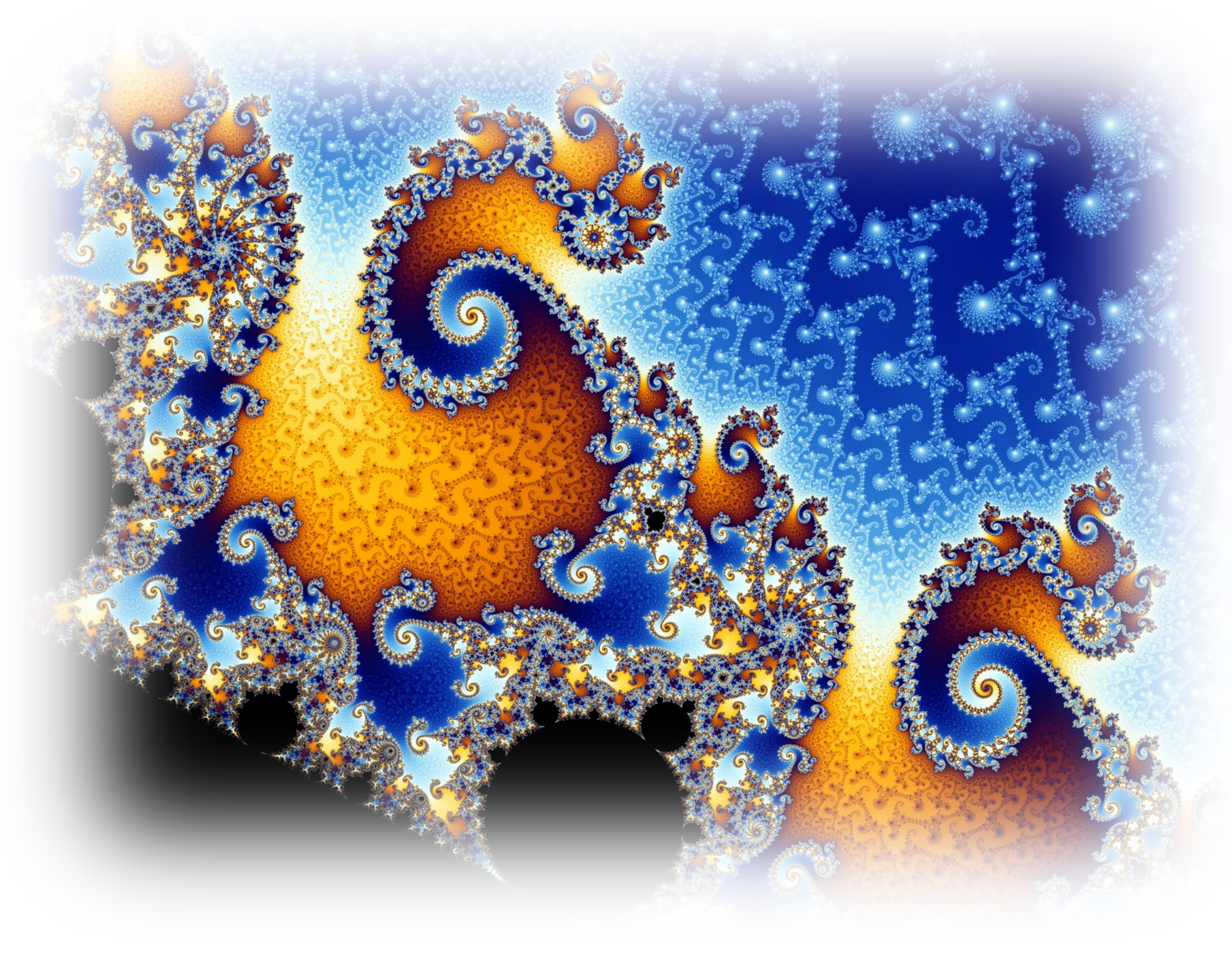
So we scale it by 360, now a full turn is 1 second

```
transform.Rotate(0, 0, tHorizontal*Time.deltaTime * 360);
```

MOVING THE SHIP

transform.position = Vector3(), we only use X & Y because its 2D

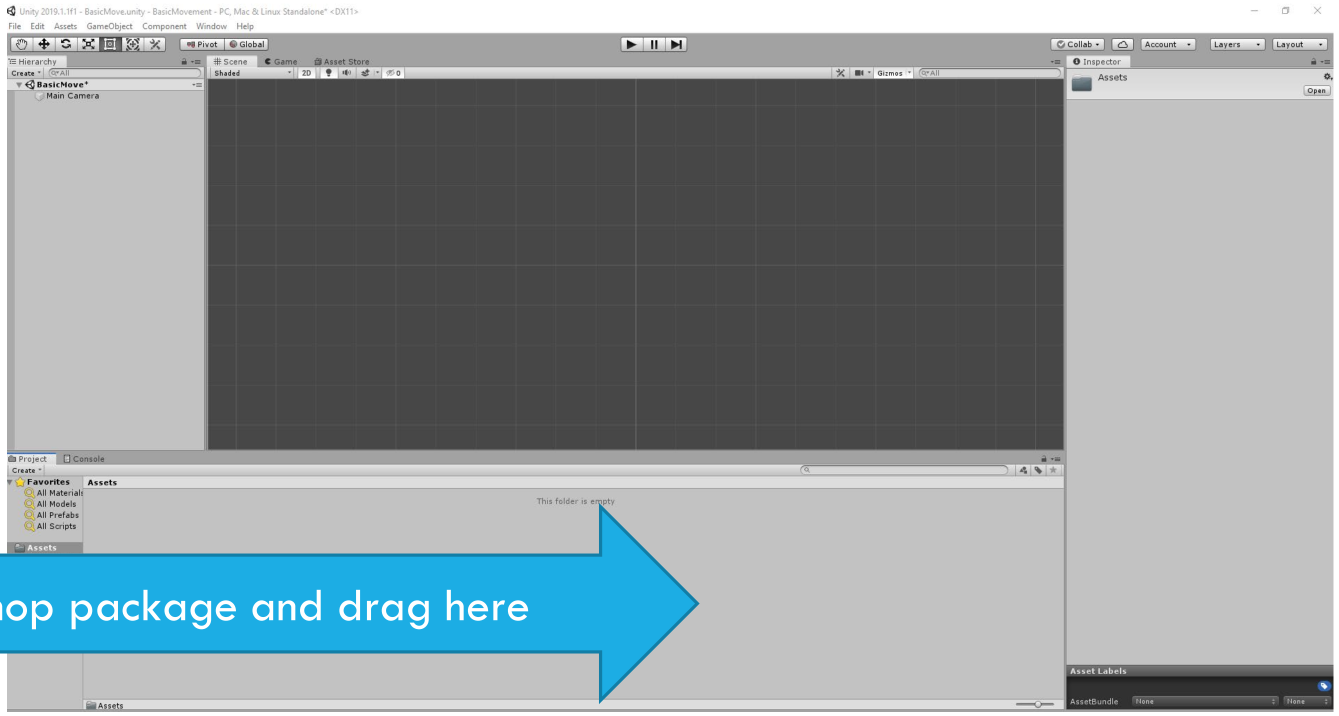
Basically you take the current position and add a movement to make the next one



WORKSHOP

OPEN UNITY

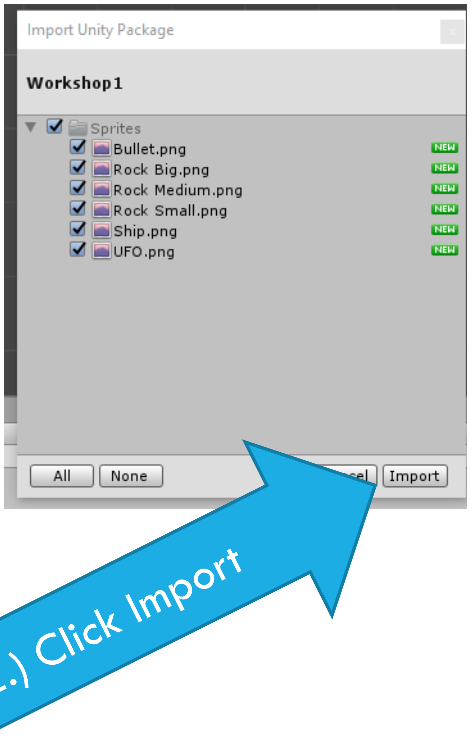
Check the controls and make sure you can identify the key screen areas



1.) Download workshop package and drag here

https://docs.unity3d.com/Manual/LearningtheInterface.html

CONFIRM THE IMPORT


Import Unity Package

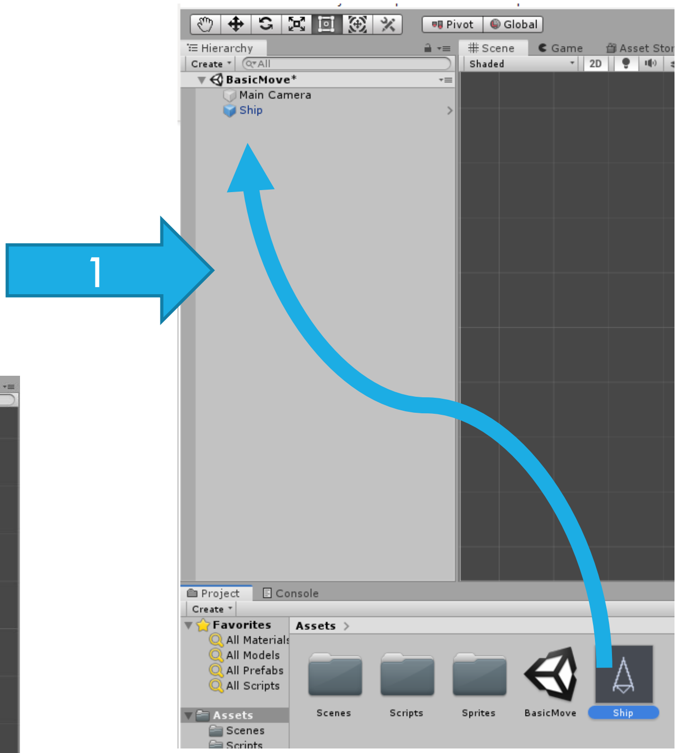
Workshop1

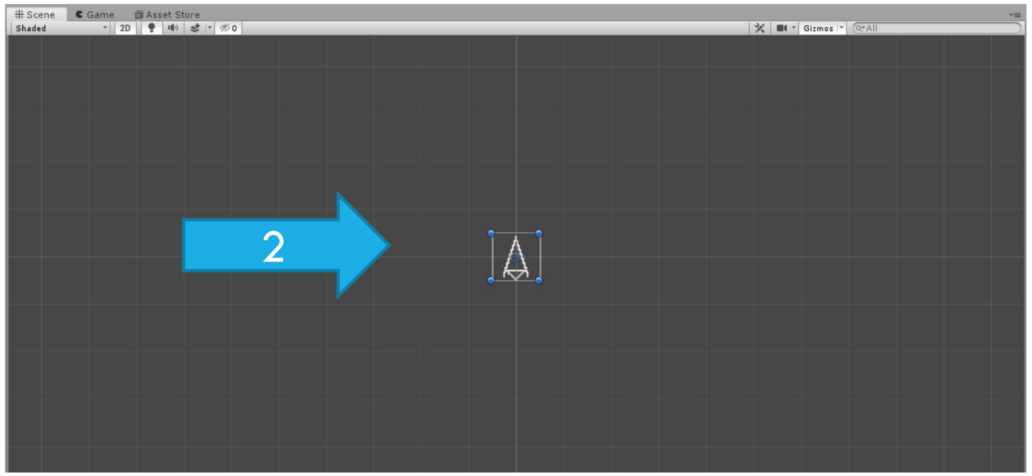
- Sprites
  - Bullet.png NEW
  - Rock Big.png NEW
  - Rock Medium.png NEW
  - Rock Small.png NEW
  - Ship.png NEW
  - UFO.png NEW

FIND THE SHIP PREFAB

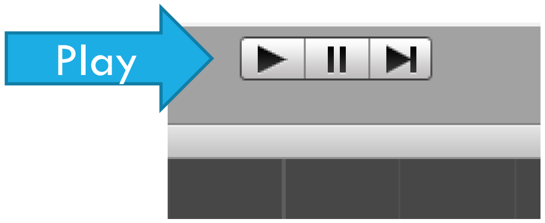
Drag the ship from folder sprites to the Scene

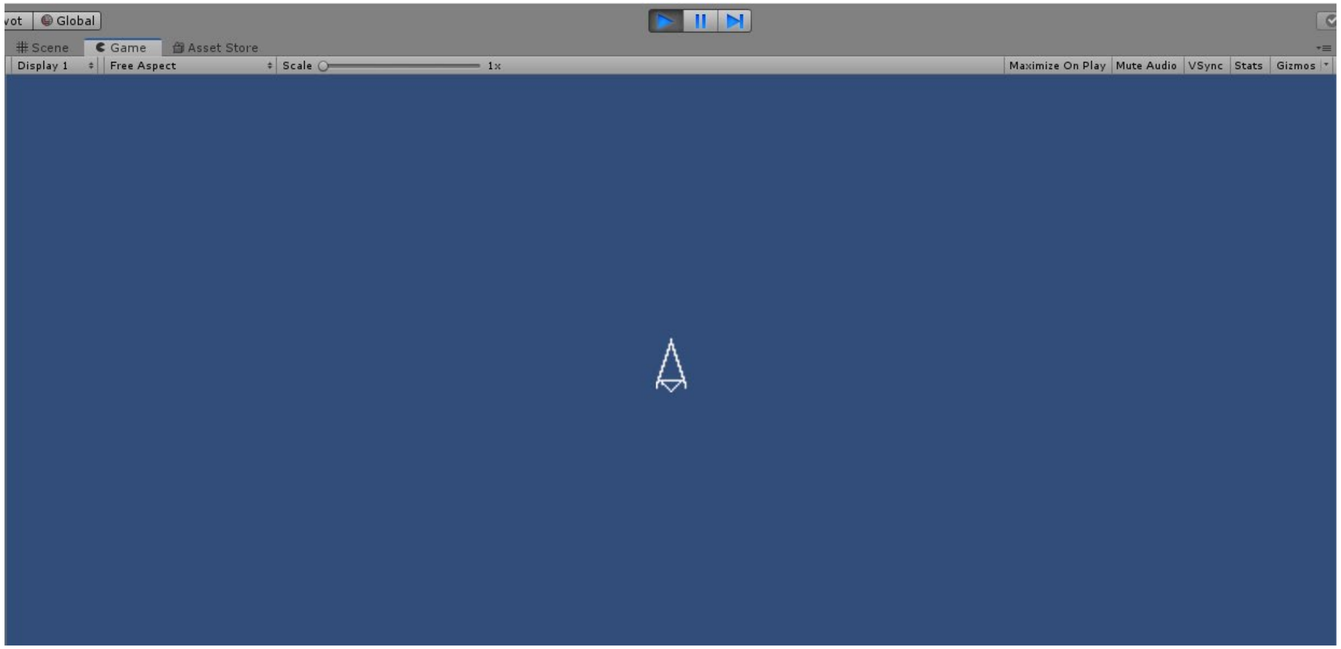
You should see the ship onscreen





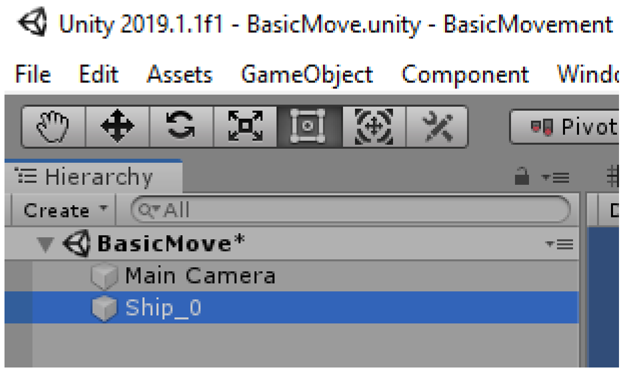
HIT PLAY


Play


vot
Global
Scene
Game
Asset Store
Display 1
Free Aspect
Scale
1x
Maximize On Play
Mute Audio
VSync
Stats
Gizmos

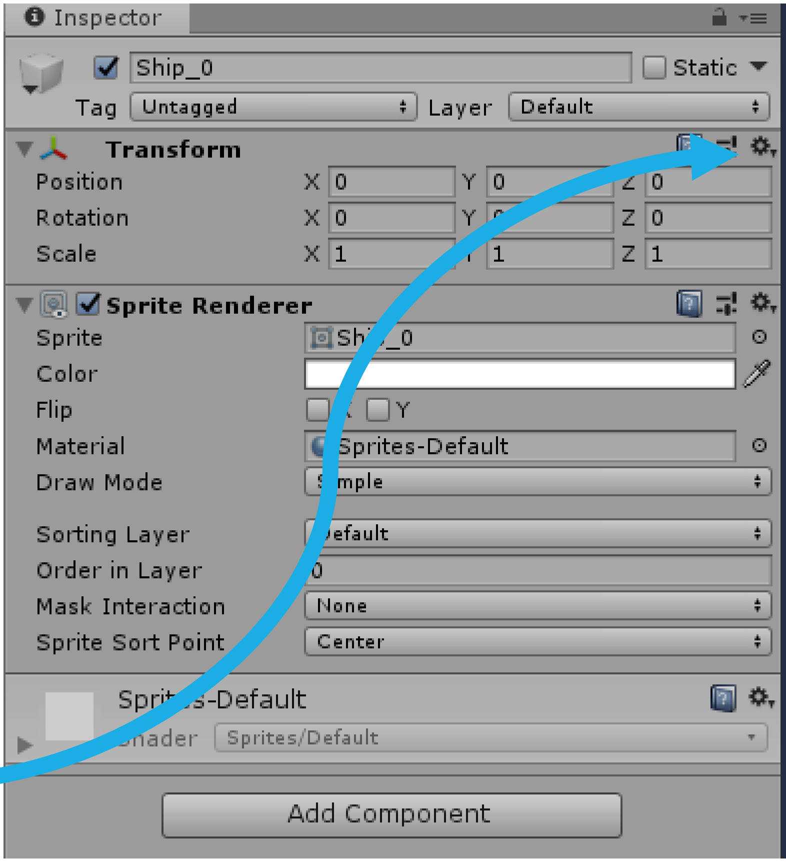
CHECK OUT THE INSPECTOR

With ship is selected



Unity 2019.1.1f1 - BasicMove.unity - BasicMovement

Look at inspector



Inspector

Position	X	0	Y	0	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

Ship_0 — Static
Tag: Untagged — Layer: Default

Transform

Sprite Renderer
Sprite: Ship_0
Color:
Flip: X Y
Material: Sprites-Default
Draw Mode: Simple
Sorting Layer: Default
Order in Layer: 0
Mask Interaction: None
Sprite Sort Point: Center

Sprites-Default
Shader: Sprites/Default

Change Position to -1,-1,0

Change rotation to 0,0,90

Try other values, if ship disappears reset transform

REMINDER: GAMES ARE MADE BY

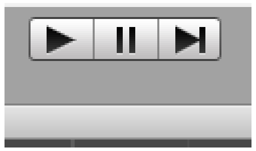
1. Moving objects
2. Objects colliding & interacting
3. User input driving the player
4. AI driving the NPC’s

Its all about manipulating numbers in 2D or 3D

Code drives this, we will use a language called C#

ADDING CODE TO A GAMEOBJECT

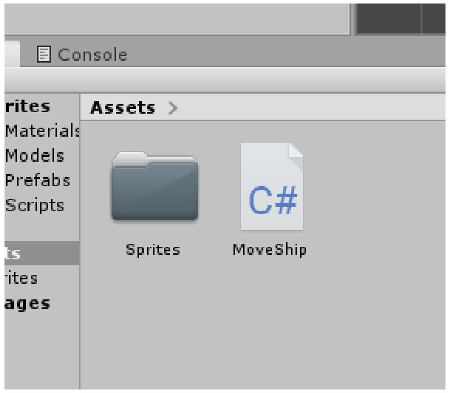
Make sure you are no longer in play mode

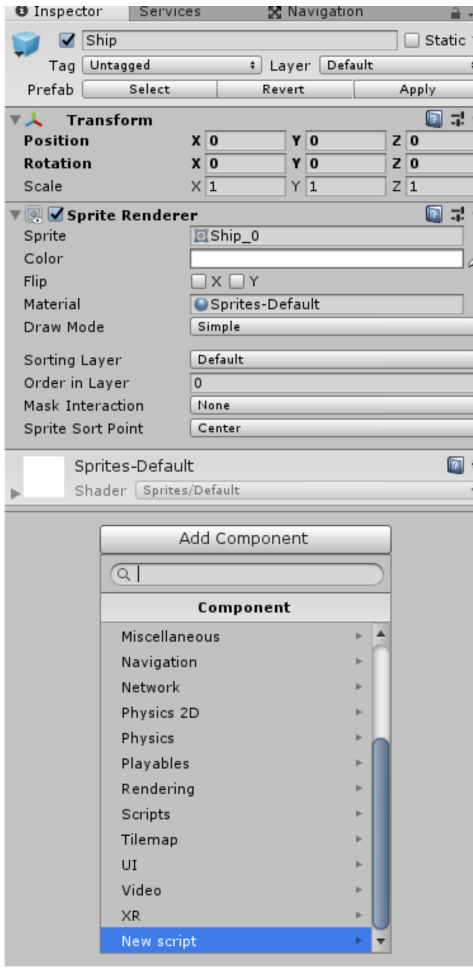


With the Ship as before selected use AddComponent➔NewScript

Name the Script MoveShip

Unity will create a new script and call it MoveShip.cs & place it in assets

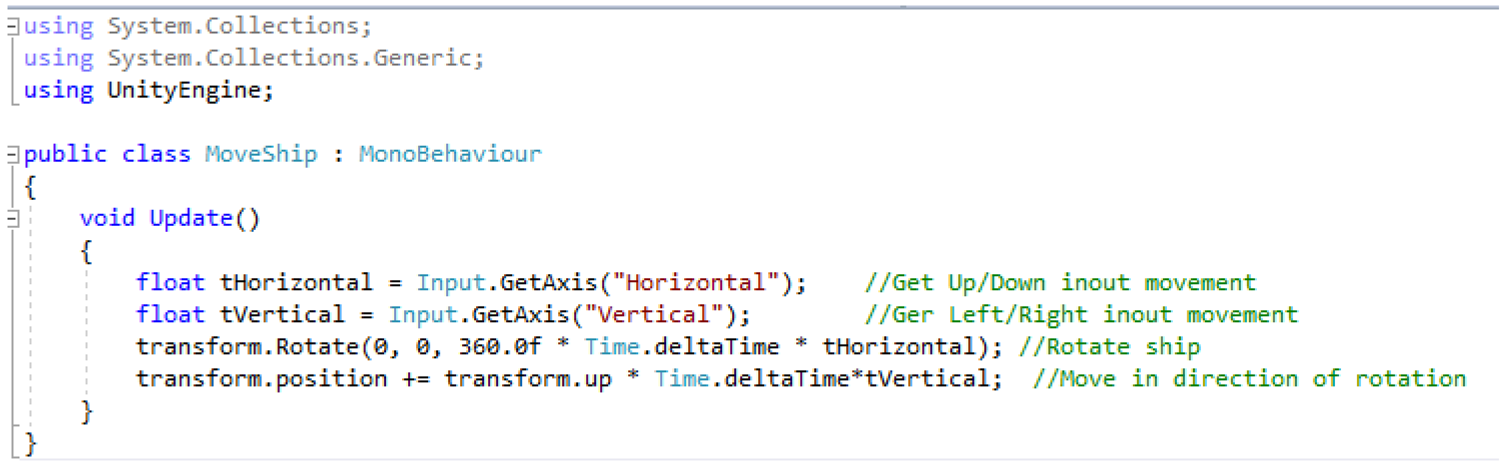




DOUBLE CLICK ON MoveShip.cs

This will open Visual Studio, please amend the file to look like this, you can remove anything else for clarity

All the code is case sensitive and the punctuation matters



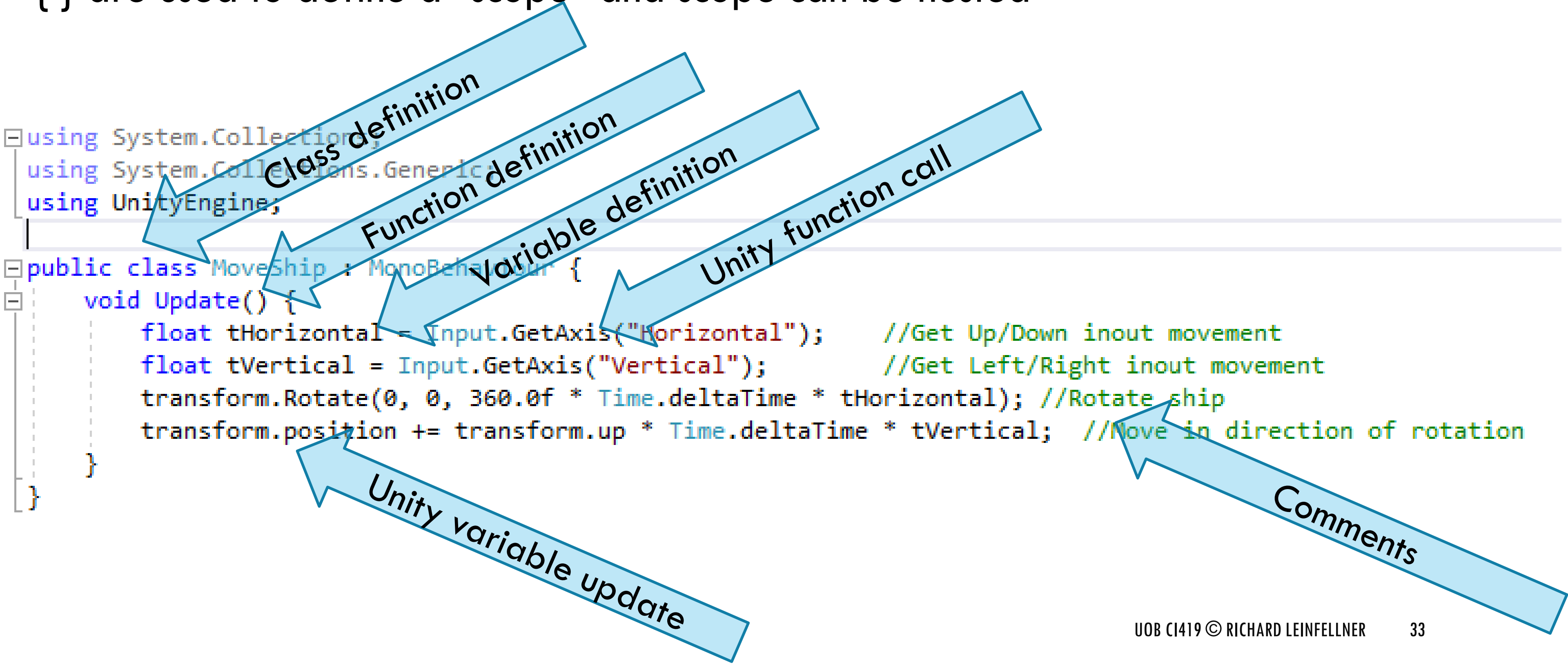
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MoveShip : MonoBehaviour
{
    void Update()
    {
        float tHorizontal = Input.GetAxis("Horizontal");    //Get Up/Down inout movement
        float tVertical = Input.GetAxis("Vertical");        //Ger Left/Right inout movement
        transform.Rotate(0, 0, 360.0f * Time.deltaTime * tHorizontal); //Rotate ship
        transform.position += transform.up * Time.deltaTime*tVertical;  //Move in direction of rotation
    }
}
```

WHAT DOES IT ALL MEAN?

Keywords, variables, functions (aka methods), comments

This is a class, it contains code & variables to make out ship move, its called MoveShip

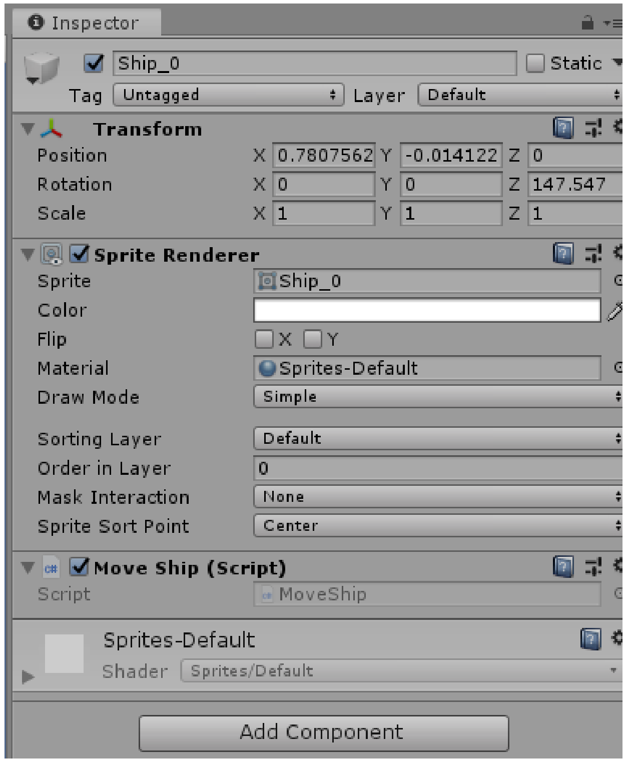
{ } are used to define a “scope” and scope can be nested



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MoveShip : MonoBehaviour {
    void Update() {
        float tHorizontal = Input.GetAxis("Horizontal");    //Get Up/Down inout movement
        float tVertical = Input.GetAxis("Vertical");        //Get Left/Right inout movement
        transform.Rotate(0, 0, 360.0f * Time.deltaTime * tHorizontal); //Rotate ship
        transform.position += transform.up * Time.deltaTime * tVertical;  //Move in direction of rotation
    }
}
```

PLAY & TEST

See how the code is changing the values in inspector


Inspector

Ship_0 Static

Tag Untagged Layer Default

Transform

Position	X	0.7807562	Y	-0.014122	Z	0
Rotation	X	0	Y	0	Z	147.547
Scale	X	1	Y	1	Z	1

Sprite Renderer

Sprite	Ship_0
Color	
Flip	X Y
Material	Sprites-Default
Draw Mode	Simple
Sorting Layer	Default
Order in Layer	0
Mask Interaction	None
Sprite Sort Point	Center

Move Ship (Script)

Script MoveShip

Sprites-Default

Shader Sprites/Default