Greetings!
@codeliberation

Tara Collingwoode-Williams
taravreality@codeliberation.org

Charlie Ann Page
capage@codeliberation.org

Phoenix Perry
phoenix@codeliberation.org

Who are you?

Tell us about yourself!

Who we?

Tara / Charlie / Phoenix

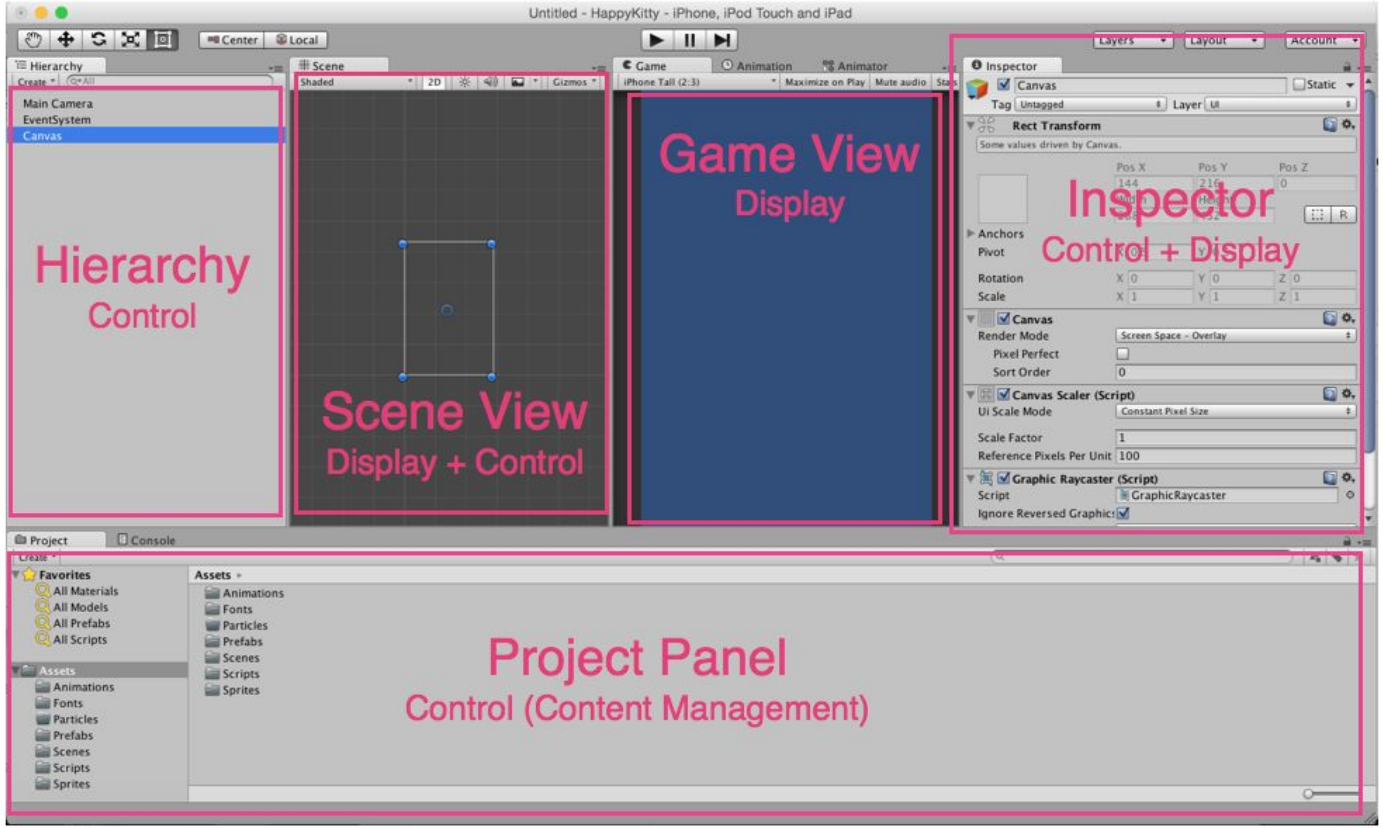
TODAY!

INTRO TO UNITY – 2D SPRITE GAME - POUNDGRAB!

That game you love…

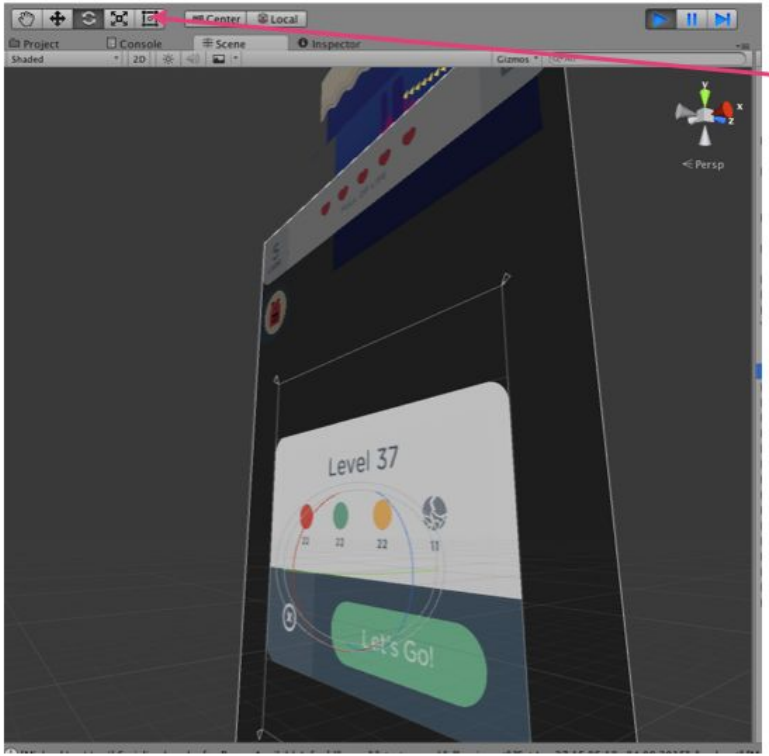
WHAT IS ...  ?

- Cross platform game engine
- Games can be made in 2D or 3D
- Suitable for creating AR and VR content
- Good for asset editing but not creating
- Can be programmed in C# or Javascript
- Free if you don't make more than $100,000
- Pro license $75 a month or $1,500 p/a



Let's take a closer look at each panel

First thing is to learn how to navigate the Scene View



1.Move in the view
2.Move an object
3. Rotate an object
4. Scale an object

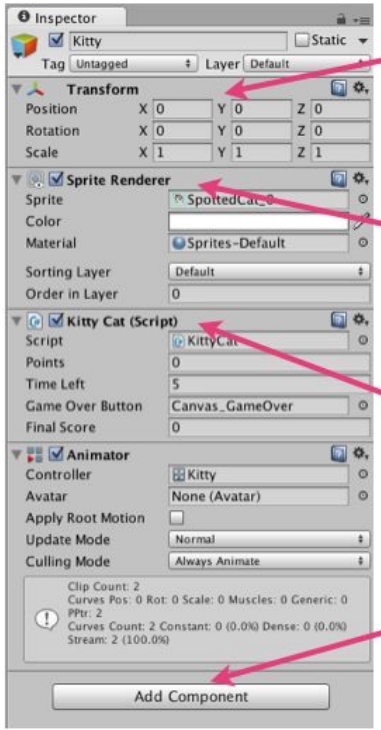
if Game = a full meal;

The inspector is where you prepare a dish (gameObjects)

E.g. In a salad...

- Some ingredients are pre-prepared
- Some ingredients you pre-prepare yourself
- You can choose the serving size of the dish, when you want to add ingredients and how much

As with the salad...
In the Inspector



- You can control the starting point, rotation and size of the gameObject
- Some components of the gameObject are pre-prepared by Unity
- Some components you pre-make yourself (through coding)
- Use the Add Component or specify in script button to add them

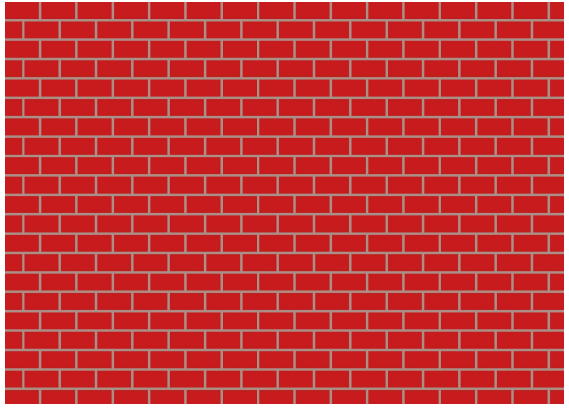
C#

Tonight we will be looking at C# It's a very powerful language. Windows runs it and many games use it also. But its not as scary as it seems…C# is intended to be a **simple, modern, general-purpose** language.

Sprites!

2D Game Objects

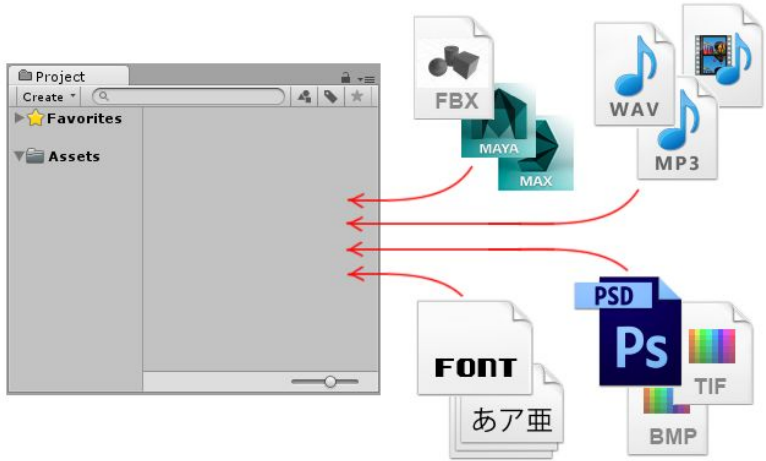
- Can animate
- Can respond to physics and light
- Can interact with only set sprites



Bring in Our Sprite Assets!

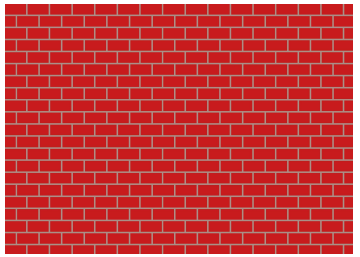
Assets are representations of any item that can be used in your project

- Drag folder into Assets



Set Up Screen

- Set screen size to 684x513
- Drag in Player controller sprite (0,0,0)
- Scale to 0.3
- Move Camera Y position (0,3,0)
- Add in Background sprite

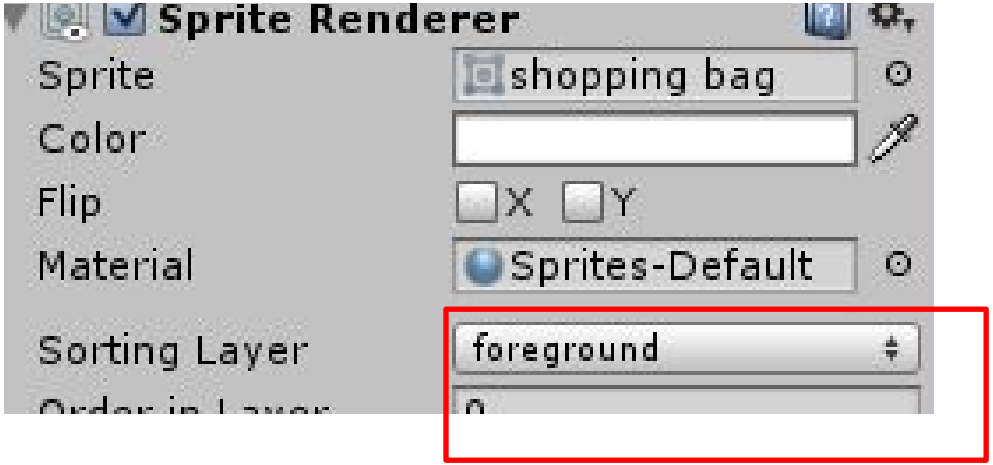


Uh Oh….

Where’s the image gone?

Sorting Layer

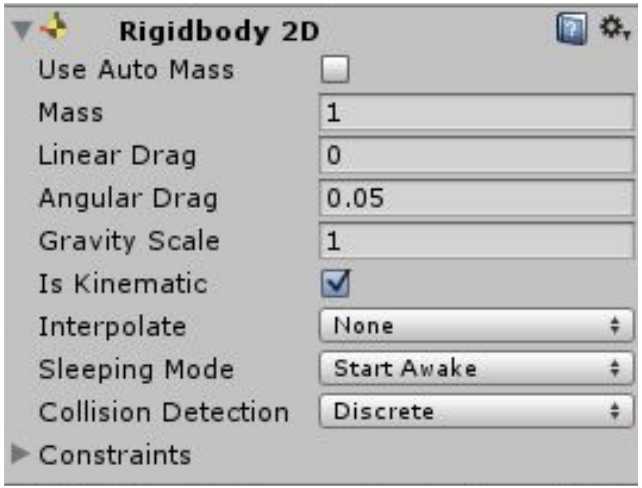
We want to sort our layers so they are visible



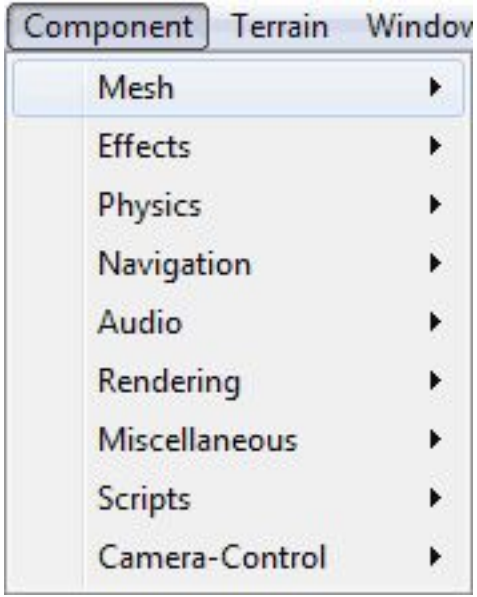
RigidBody2D

RigidBody2D = Component that makes your object respond to physics

- Add RigidBody2D Component to Player Controller Object
- Is Kinematic = Allows us to move sprite without Gravity



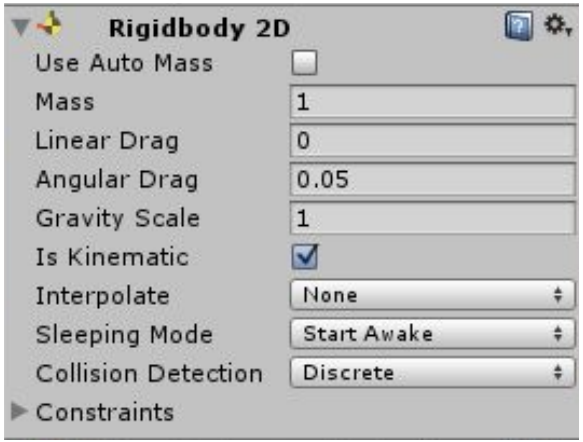
Unity is working with Components!!



RigidBody2D

RigidBody2D = Component that makes your object respond to physics

- Add RigidBody2D Component to Player Controller Object
- Affect the motion of other rigidbodies through collisions or joints.
- Is Kinematic = switches off the physical behaviour of the Rigidbody 2D so that it will not react to gravity and collision



Lets Move Our Player Controller!

Scripts are Components!

- Create and name C# Script

‘PlayerController.cs’ and add to the Player Controller Object

One Second…How?

TYPES: Primitive and Complex

Primitive Types

These primitive types are commonly used in C#

- 1 = Int = whole number
- 1.1 = Float = decimal
- “hello world” = Strings = words
- True = Bool = true or false

TYPES: Primitive and Complex

Complex Types

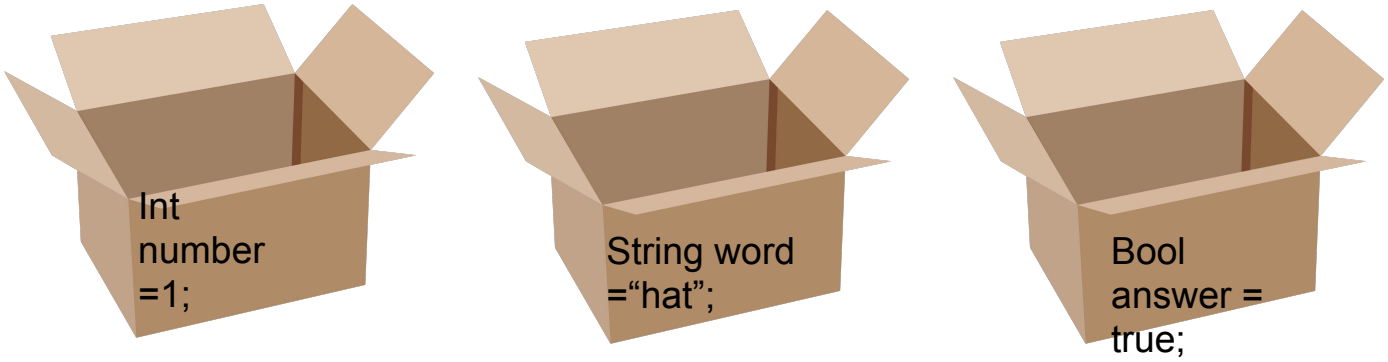
Unity has some built in types for in game good behaviour

- GameObject = Is a Object in your game
- RigidBody2D = Component that makes your object respond to physics
- Renderer2D = Component thats drawing your image
- Collision2D = Component that alerts you when your object is hit

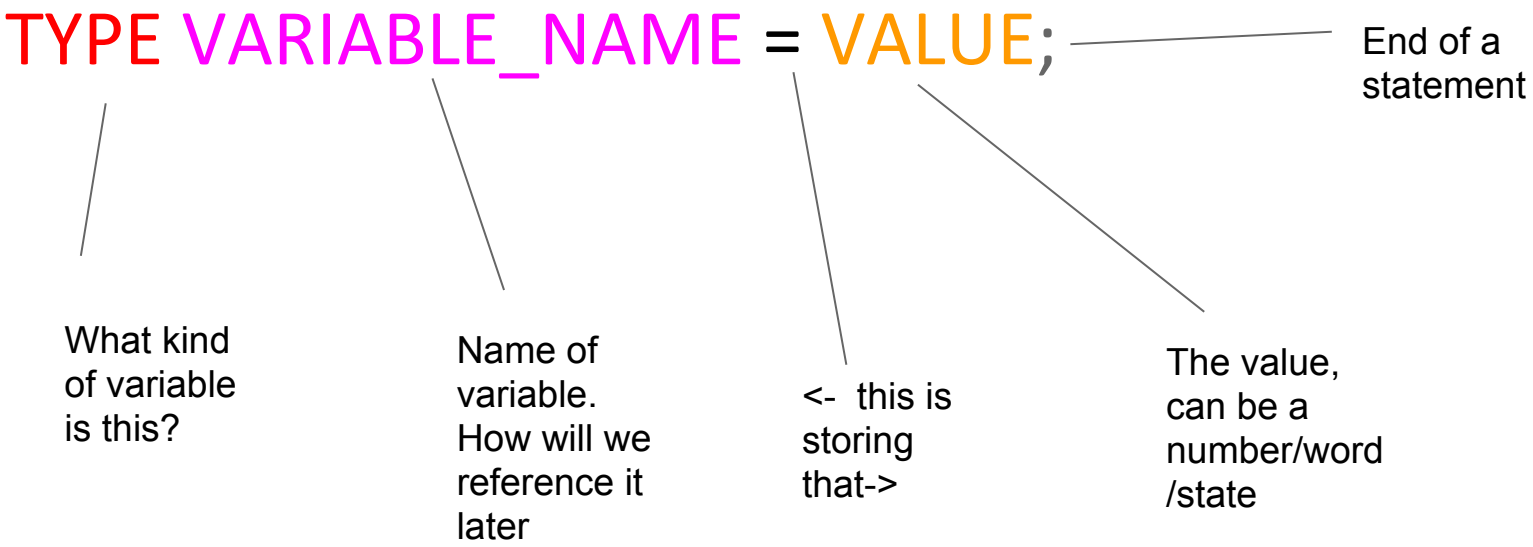
Variables

You can think of a variable as a box.

You can always open the box and change what's inside



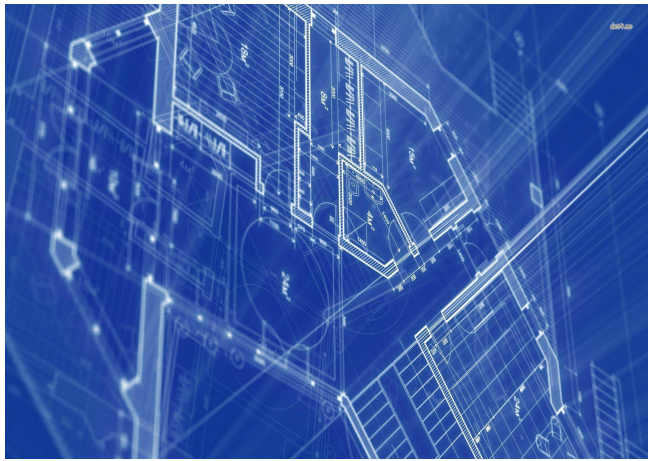
Variables


TYPE VARIABLE_NAME = VALUE;
End of a statement
What kind of variable is this?
Name of variable. How will we reference it later
<- this is storing that->
The value, can be a number/word /state

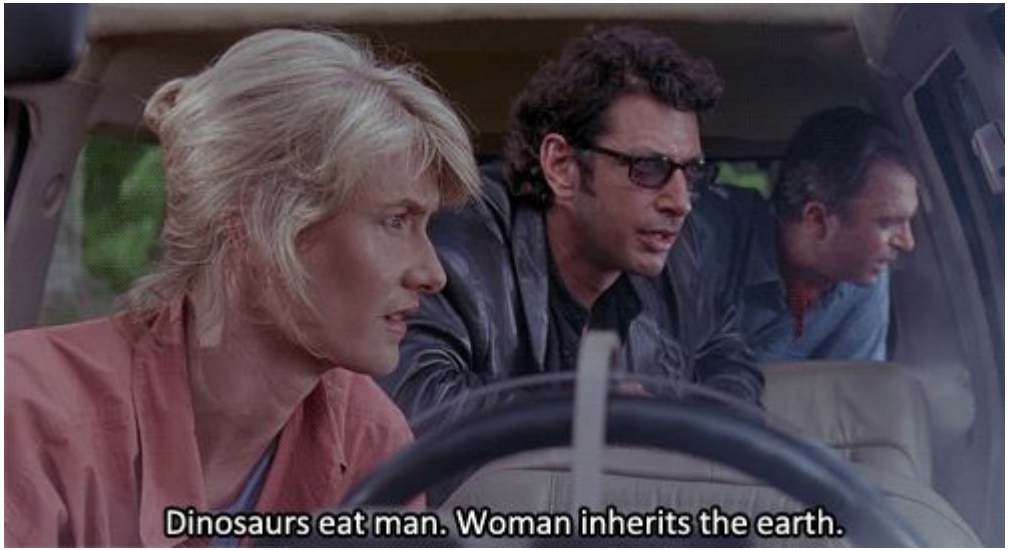
Classes

Classes are like blueprints. They allow you to customize and define data and the behaviour of a type.

Public class Player_Controller : Monobehaviour{}



Inherit from our par-uhh-Monobehaviour



Access Modifier

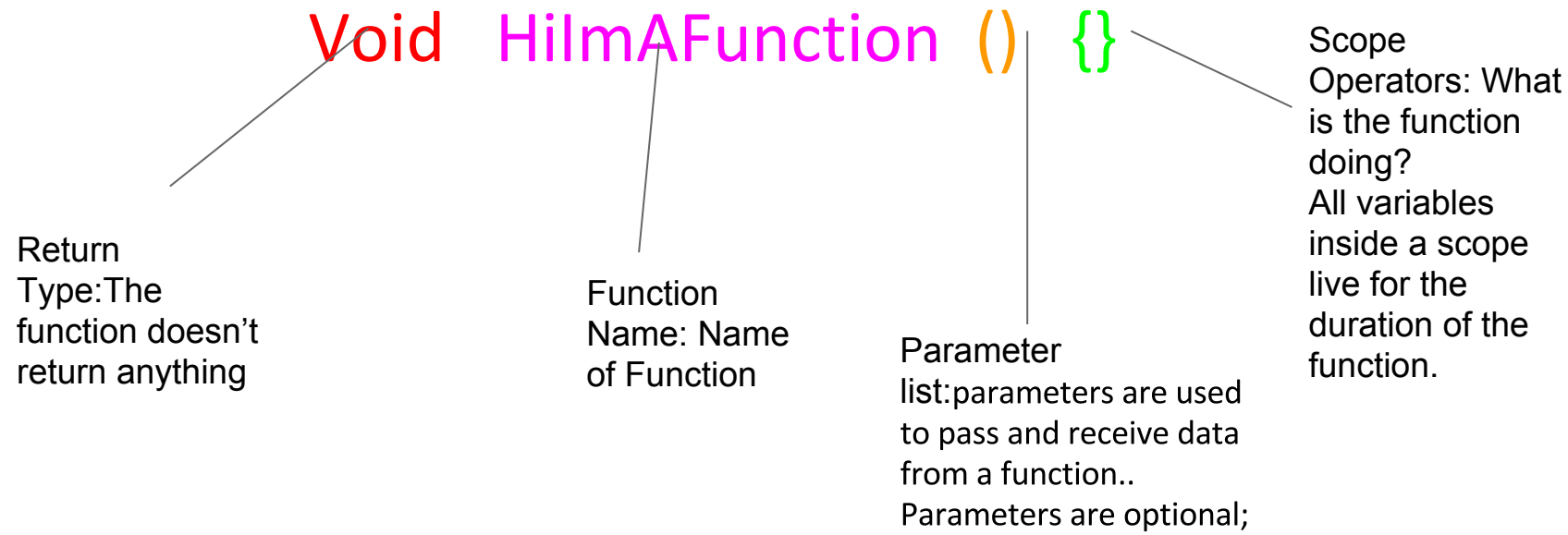
Allows other scripts to ‘see’ this class

Function

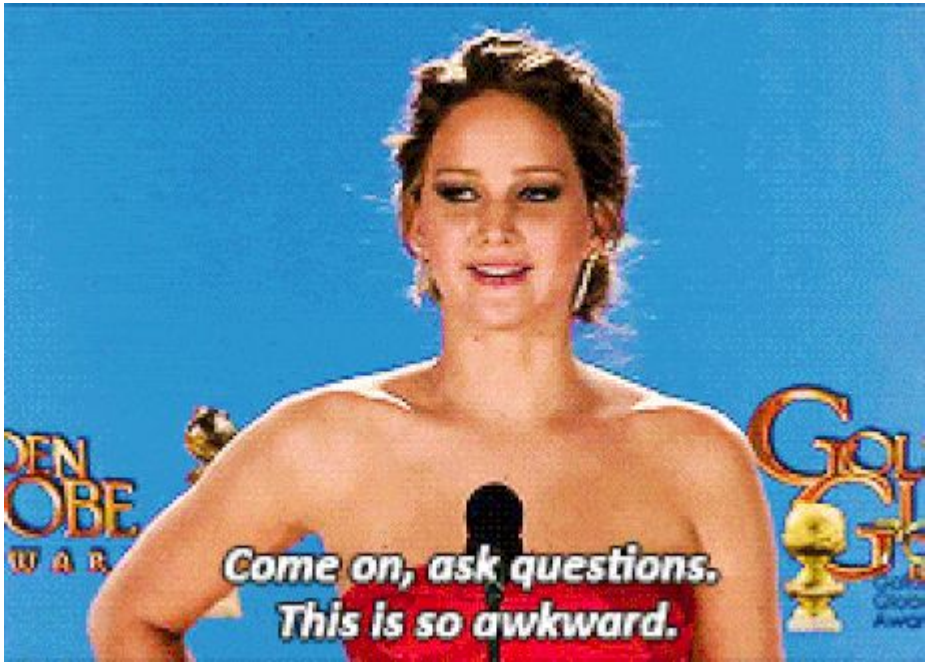
A function is a subroutine associated with a class. It executes the statements with its scope

```
Void HiImAFunction(){
    //Mega interesting stuff to happen
    }
```

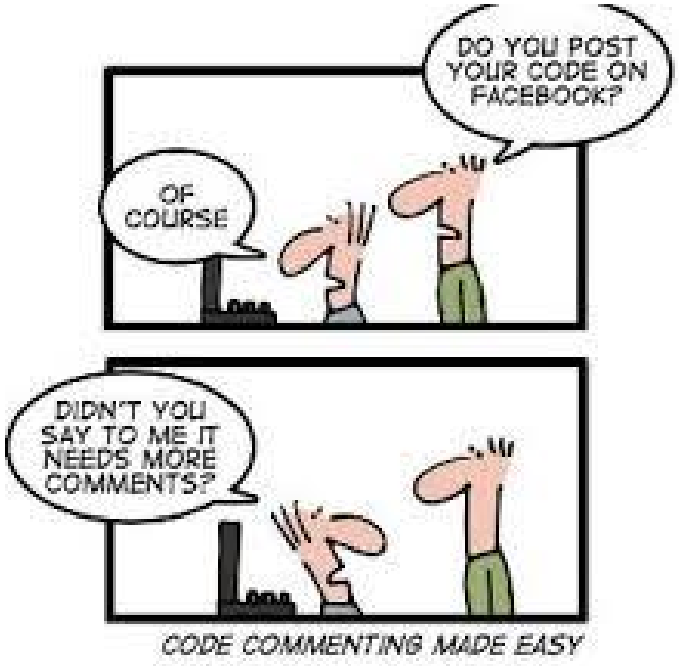
Function


Void HilmAFunction () {}
Return Type:The function doesn't return anything
Function Name: Name of Function
Parameter list:parameters are used to pass and receive data from a function..
Parameters are optional;
Scope Operators: What is the function doing?
All variables inside a scope live for the duration of the function.

With me so far? Questions?



Comments are our friends...



Make it Move!

```
Public class Player_Controller : Monobehaviour{

    //Our variables
    public Camera myCam;
    public Rigidbody2D rbBag;
    public Renderer rend;
    private float maxWidth;
```

Conditionals

Set of instructions that run is something is true

```
if(hasCats){
    ImHappy();
    Happyness = happyness+1;
}
```

Make it Move!

```
Void Start(){
    //put main camera in myCam variable

    if(myCam == null){
    myCam = Camera.main;
    }
```

Vectors2D

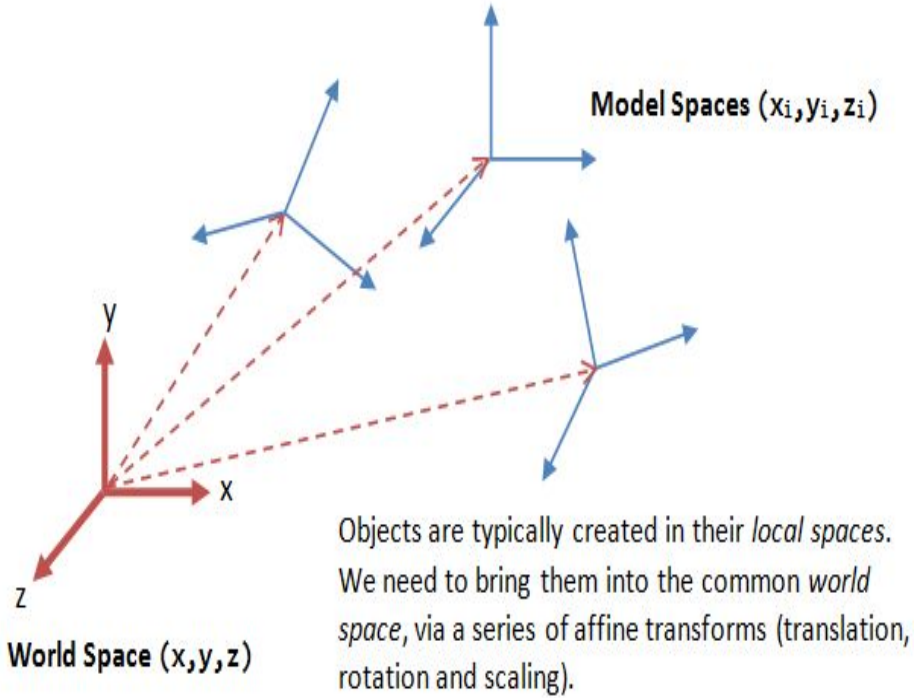
Points in either 2D 3D space contained one object.
To create
Vector3 myVect = new Vector3(1.0f, 0.0f, 1.0f);

Position, rotation and scale in unity is stored as a Vector3 variables
transform.position;

To get access to the x, y, or z positions separately? Use dot syntax
transform.position.x
tranform.rotation.x

World space

Objects have their own axis and that is different than the world axis, which is always fixed in one direction. Be warned, forward for an object might not be forward for the world.



Make it Move!

```
//Grab RigidBody2D component from inspector
rbBag = GetComponent<Rigidbody2D>();
//Store the size of the screen
Vector2 uppercorner = new Vector2(Screen.width, Screen.height, 0.0f);
//transforms position from screen space to world space
Vector2 targetwidth = myCam.ScreenToWorldpont(uppercorner);
```

Make it Move!

```
        //Grab Renderer component from inspector
        rend = GetComponent<Renderer>();
        //The renderer knows the size of the Graphics, ask it for the size and store
        //Extents gets half the size
        float bagWidth = rend.bounds.extents.x;
        //Store the max with of screen minus the size of our object
        maxWidth = targetwidth.x - bagWidth;

}
```

Breathe…….Aw A Pomsky!

Make it Move!

```
// Update is called once per physics timestamp
void FixedUpdate () {
        //Create a Vector to store mouse position
         Vector2 rawMousePosition =
myCam.ScreenToWorldPoint(Input.mousePosition);
        //Create Vector to move player in mouse direction
         Vector2 targetPosition = new Vector3(rawMousePosition.x, 0.0f);
```

Make it Move!

```
        //Clamp playerController to width of screen, left and right.
        float targetWidth = Mathf.Clamp(targetPosition.x, -maxWidth, maxWidth);
        //update target Position
        targetPosition = new Vector3(targetWidth, targetPosition.y,      targetPosition.z);

        //move playerController
        rbBag.MovePosition(targetPosition);
}
```

Don’t forget to fill in the Components in Inspector!

Ouu…It moves….

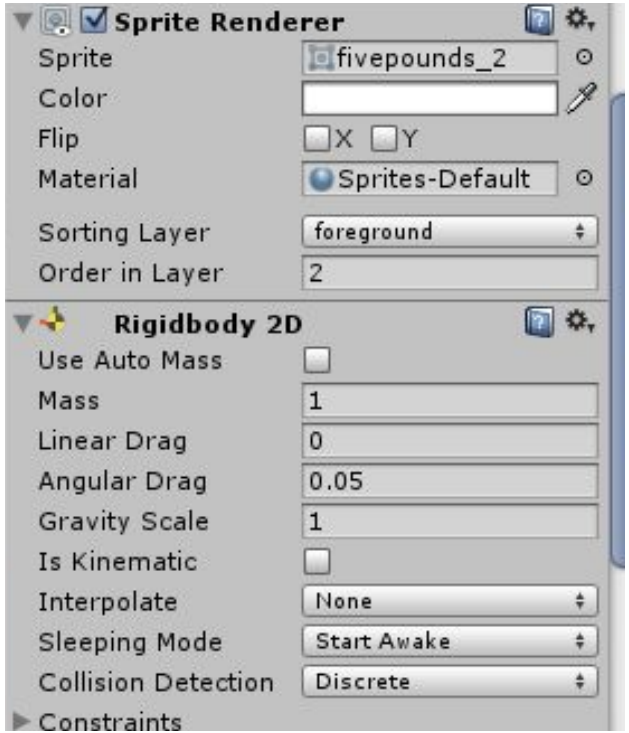
You can do it!

Next...Time to Collect!

Add Money!! :D

Spawn Money

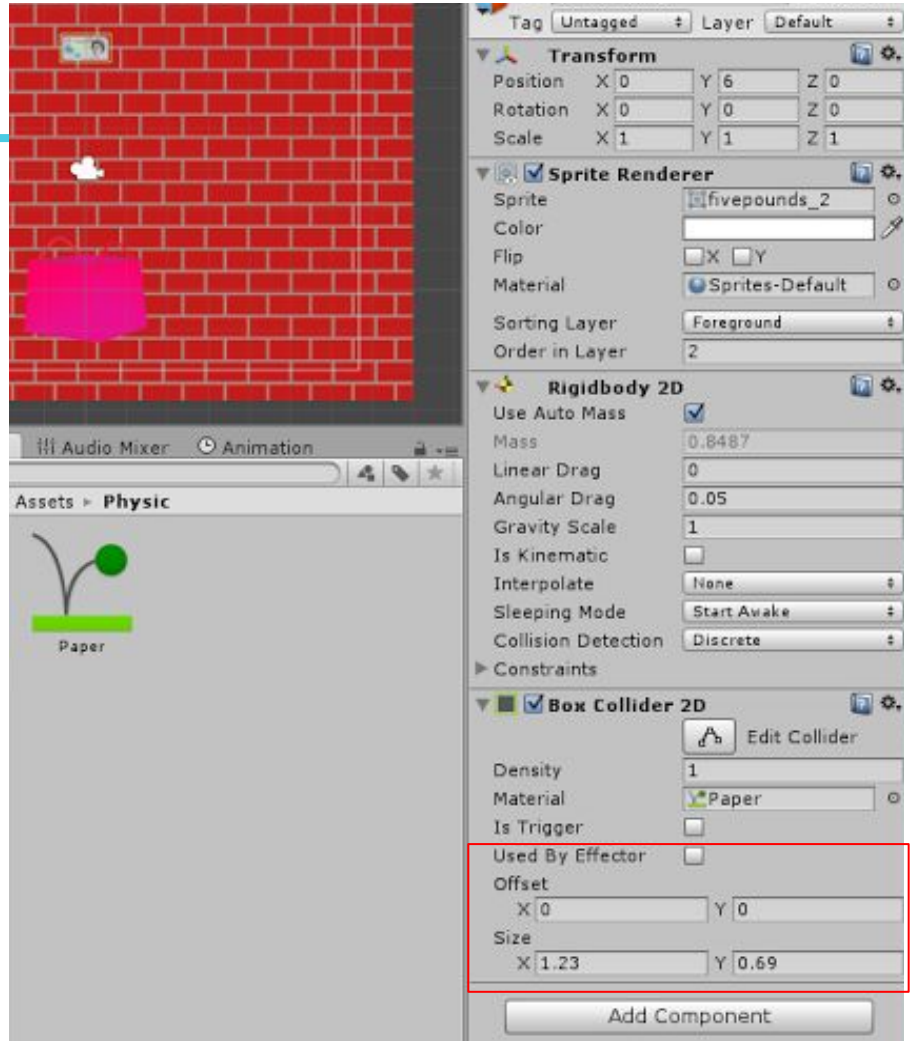
- Add Money Sprite to Scene and sort Layer
- Give it a RigidBody2D
- No Kinematics!
- Make a Prefab!



Box Collider

Add Box Collider Component to Money

Resize if necessary using Size

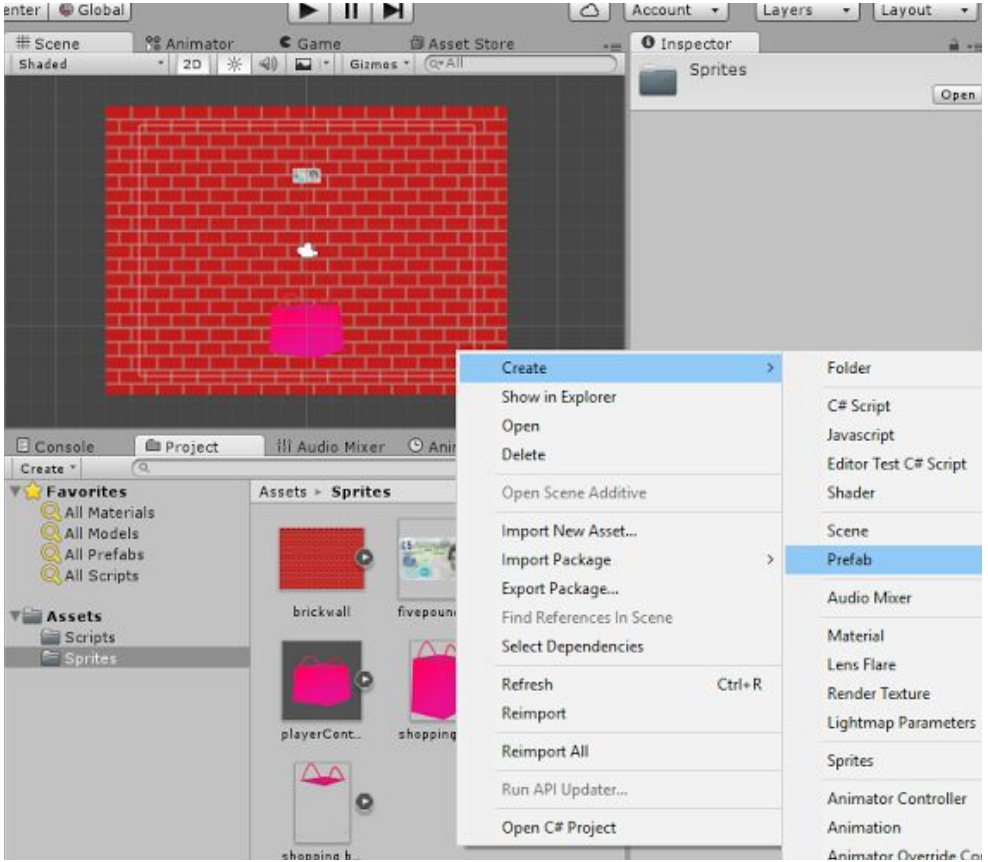


Prefabs

- Saved game objects. You save these for creation later. You can create these objects (Instantiate) them dynamically with code.

Make Prefab!

- Drag Object onto Prefab
- Put into Prefab folder

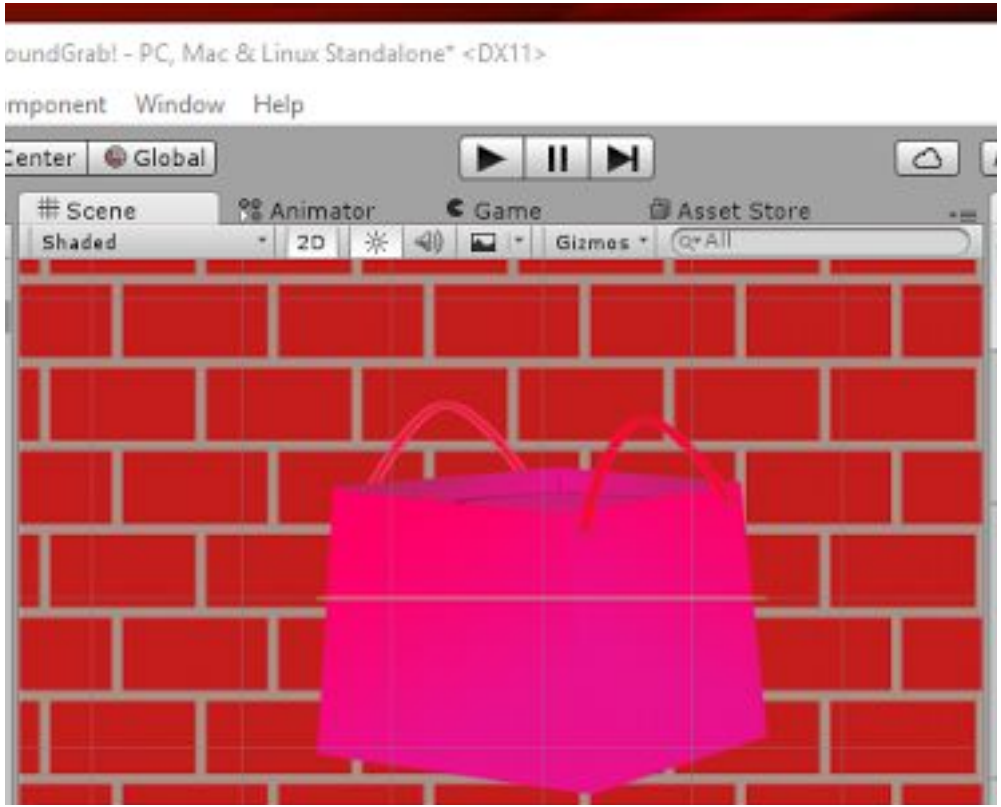


Play it..Our Money Falls! Let's Catch it! >=]

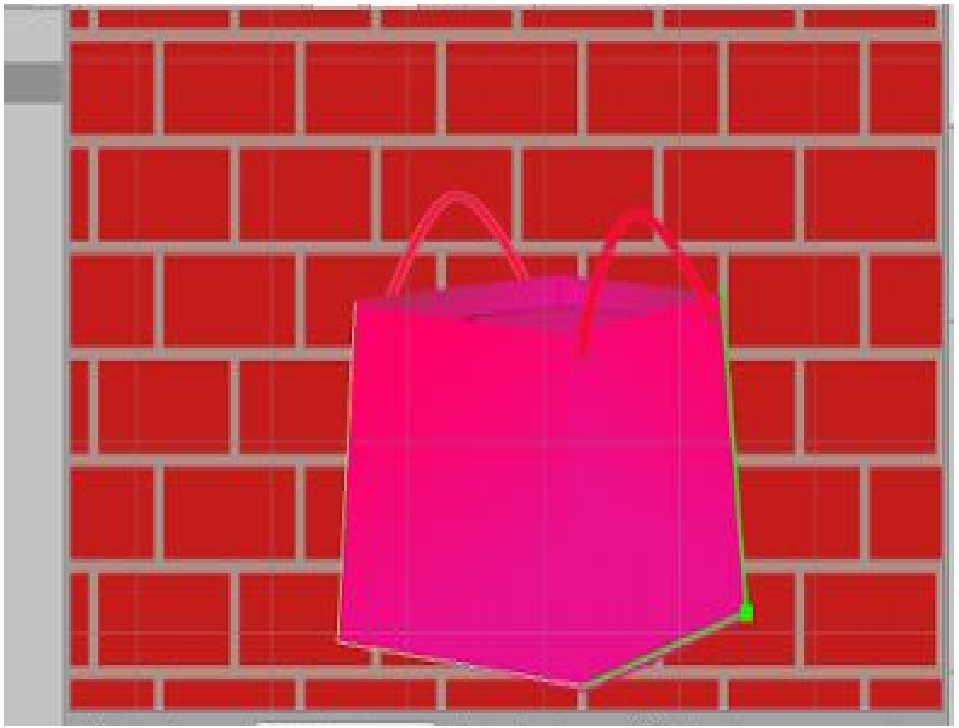
Box Collider & Edge Collider

- Another Component in the inspector
- A Collider
- Freeform Edge – Can be transformed!
- Used in Script to trigger a event of function
- Add an Edge Collider to the Player Controller

Edge Collider



Edge Collider



Caught it! I'm Rich!

We Need to Destroy it…Sigh

- Add another Edge Collider to PlayerController
- Is Trigger
- Add Script onto PlayerController – “DestroyMoney.cs”
- Drop in Scripts folder

We Need to Destroy it…Sigh

```
public class DestroyMoney : MonoBehaviour
{
    //This tells us when the collider is hit, then completes function.
//Collider2D is the collider component of the Gameobject and that is a
parameter for this function
        void OnTriggerEnter2D(Collider2D other)
    {
        //Destroys gameobject that it collided with after a delay
        Destroy(other.gameObject, 1.0f);
    }
}
```

Bye Money…

Keep trying little guy!

BREAK!

We Still have things to do!

1. SPAWN MONEY
2. CREATE POINT SYSTEM
3. TIMER

GameManager

- Create an Empty Game Object – “GameController”
- Place it where the money is
- Create a Script on it called “GameManager.cs”

GameManager

We want to achieve two things with this script

- Spawn Money
- Create Timer

GameManager – Spawn Money

First we must once again get the size of the sprite and store its location in WorldSpace

We can copy and paste code from ‘PlayerController.cs’ and make minor changes

GameManager – Spawn Money

```
//variables
    public Camera myCam;
    private float maxWidth;
    public GameObject money;
```

Start Coroutine/WaitForSeconds/IEnumerator

- Yield return new Wait for seconds: Suspends the coroutine execution for the given amount of seconds using scaled time.
- IEnumerator _function(): Supports iteration and the yield function
- StartCoroutine(_function()): starts coroutine

GameManager – Spawn Money

```
void Start () {
        if (myCam == null)
        {
            myCam = Camera.main;
        }
        Vector2 uppercorner = new Vector2(Screen.width, Screen.height, 0.0f);
        Vector2 targetwidth = myCam.ScreenToWorldPoint(uppercorner);
//instead of using renderer I used a small figure as it's not likely to leave the screen.
        maxWidth = targetwidth.x - 5;
//Starts coroutine
        StartCoroutine(Spawn());
         }
```

GameManager – Spawn Money

```
IEnumerator Spawn(){
//yield/wait causes a delay before completing the function
      yield return new WaitForSeconds(2.0f);
//This randomly picks a position on the x-axis and stores it in a variable
       Vector2 spawnPosition = new Vector2(
            Random.Range (-maxWidth, maxWidth), transform.position.y, 0.0f);
//Quaternion is what unity uses to represent rotations. Quaternion.identity suggests
no rotation
         Quaternion spawnRotation = Quaternion.identity;
```

GameManager – Spawn Money

```
//Instantiate creates prefab in position set with no difference in rotation
        Instantiate(money, spawnPosition, spawnRotation);
        yield return new WaitForSeconds (Random.Range(1.0f, 2.0f));

    }

    }
```

Still with me?

GameManager – Timer

```
//Variables to store time details
    public float timeLeft;
```

GameManager – Timer

```
void FixedUpdate () {
//Creates count down
        timeLeft -= Time.deltaTime;
//when timer hits 0 stop countdown.
        if (timeLeft < 0)
        {
            timeLeft = 0;

        }
}
```

Don’t forget to fill in the Components in Inspector!

Hmm….What are we missing?

GameManager – Spawn Money

```
IEnumerator Spawn(){
//yield/wait causes a delay before completing the function
        yield return new WaitForSeconds(2.0f);

        while ( timeLeft > 0){

//This randomly picks a position on the x-axis and stores it in a variable
        Vector3 spawnPosition = new Vector3(
            Random.Range (-maxWidth, maxWidth), transform.position.y, 0.0f);
//Quaternion is what unity uses to represent rotations. Quaternion.identity suggests
no rotation
          Quaternion spawnRotation = Quaternion.identity;
```

Don’t forget to fill in the Components in Inspector!

GUI Text

- Flat on screen text you can change with Script
- Can animate but cannot interact with game objects

GUI TEXT- Timer & Counter

Introducing….CANVAS

GUI TEXT- Timer & Counter

- Add Canvas UI Gameobject to script

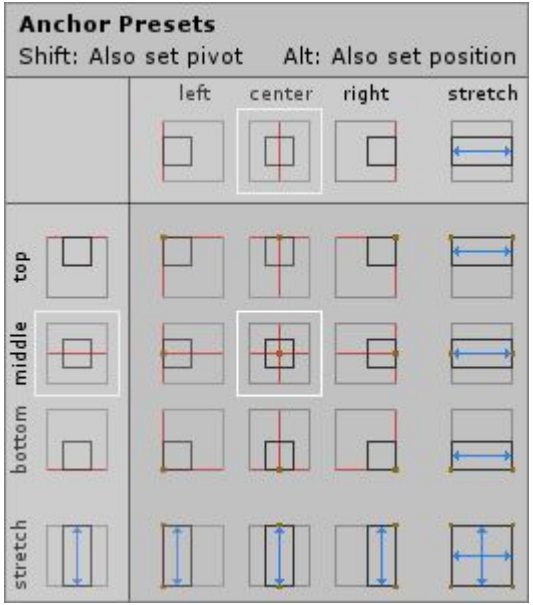
  Create->UI->Canvas
- Add two Text GameObjects and NEST it in Canvas. ‘Timer’ and ‘Counter’

  Create->UI-> Text

Anchor it Right…

Anchor the TextBox to a corner (or center!) of the Canvas

- Click on Anchor
- Hold ‘Alt’ and choose a position



GUI TEXT- Timer

In GameManger.cs…

```
Using UnityEngine.UI;
```

GUI TEXT- Timer

In GameManger.cs…

```
public Text timerText;
```

GUI TEXT- Timer & Counter

In GameManger.cs…

```
void FixedUpdate()
   {
      timeLeft -= Time.deltaTime;
      if (timeLeft < 0)
      {
         timeLeft = 0;
}
 //this will replace the text in the textbox in the inspector
      timerText.text = "Time Left: " + Mathf.RoundToInt(timeLeft);

   }
```

Don’t forget to fill in the Components in Inspector!

GUI TEXT- Counter

- Add a new Script to PlayerController ‘Score.cs’

GUI TEXT- Timer

In Score.cs…

```
Using UnityEngine.UI;
```

GUI TEXT- Counter

```
public class Score : MonoBehaviour {
    //Create variables to hold Text component and score
    public Text counter;
    private int score;

void Start () {
//Score starts at 0
        score = 0;
//Create a function that runs at the start of game. Updates score.
        UpdateScore();

}
```

We need to know when the money is in the bag...

How do we do this?

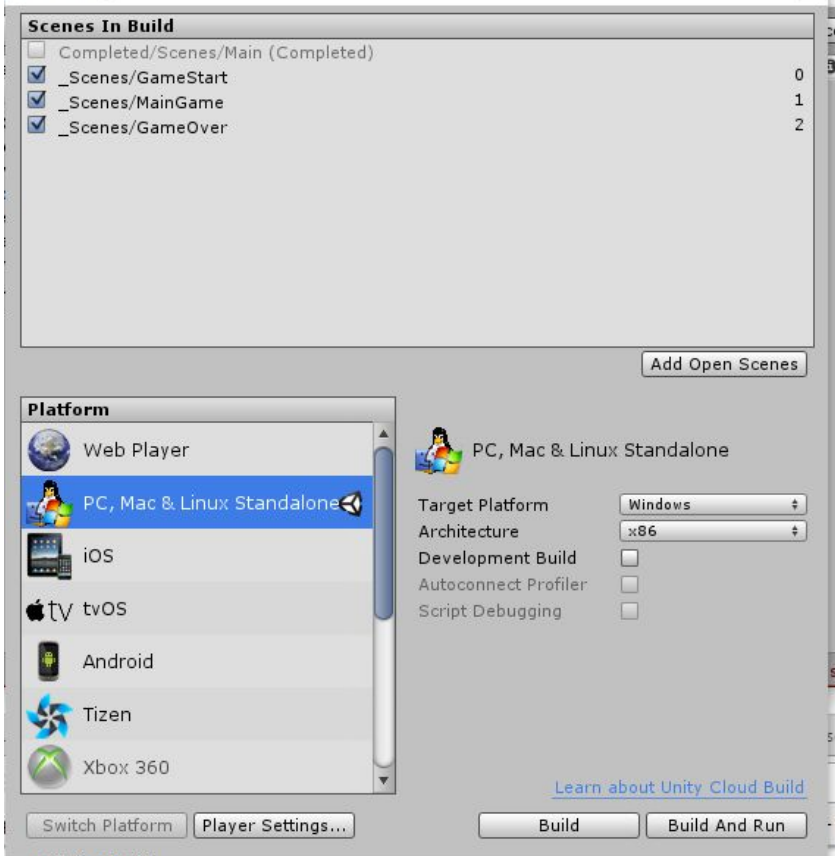
GUI TEXT- Counter

```
 //When money hits the bag, score increases by 5
void OnTriggerEnter2D(Collider2D other)
   {
       score += 5;
//call this function to update score
       UpdateScore();
   }
//this function overwrites the text of score
void UpdateScore () {
       counter.text = "Money: £" + score;
}
}
```

Don’t forget to fill in the Components in Inspector!

NOW BUILD IT!

- Go to File -> Build Settings
- Add OpenScenes
- Build!



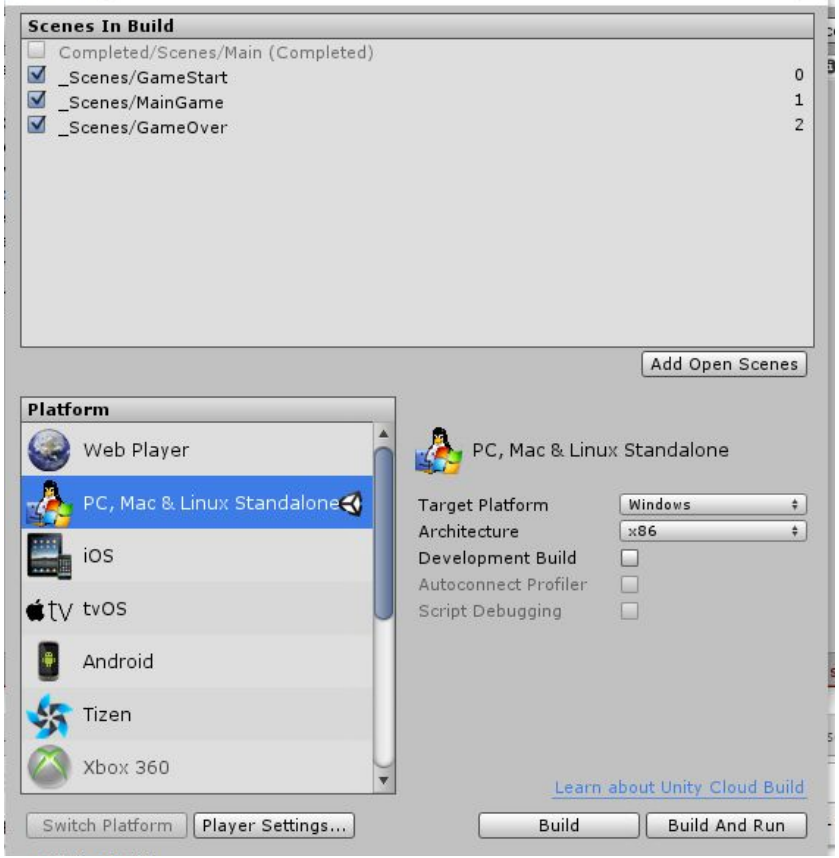
EXTRA! START GAME

- New Scene! Create a Canvas with Text.
- Create a script, ‘StartGame.cs’ on an Empty GameObject

```
//another library that looks after scene changes
using UnityEngine.SceneManagement;
void Update()
   {//Getkeydown(keyCode.Return) acknowledges when Enter is pressed
      if (Input.GetKeyDown(KeyCode.Return))
      {
         Debug.Log("pressed");
         SceneManager.LoadScene("MainGame", LoadSceneMode.Single);
      }
   }
```

NOW BUILD IT!

- Go to File -> Build Settings
- Add OpenScenes
- Build!



EXTRA! GAME OVER

- Create another Scene!

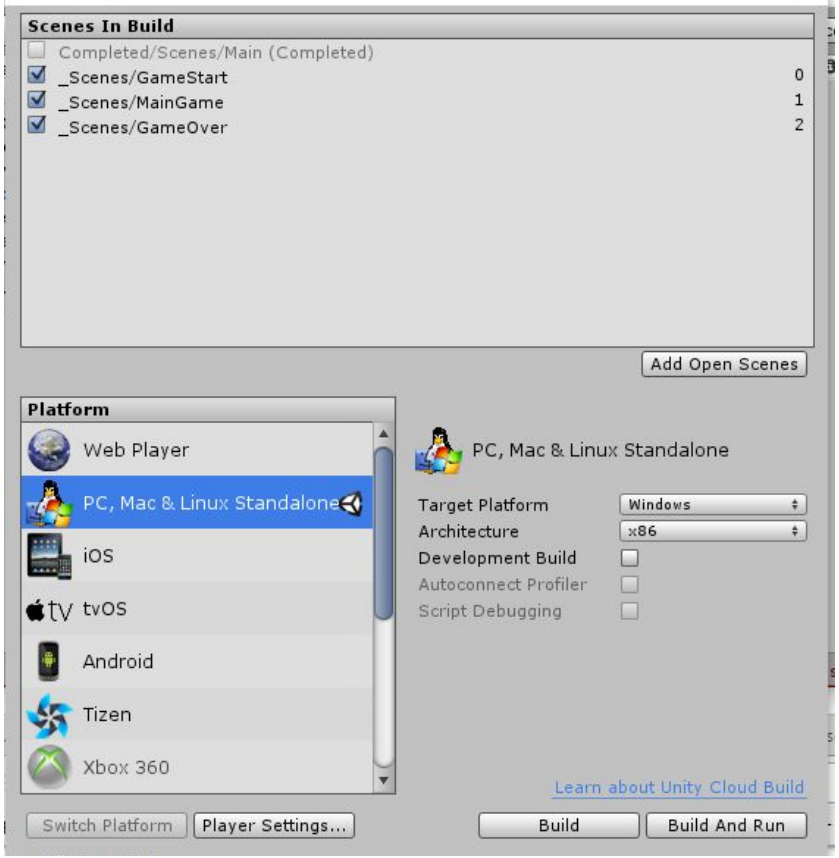
In ‘GameManager.cs’…

```
//another library that looks after scene changes
using UnityEngine.SceneManagement;

if (timeLeft < 0)
      {
         timeLeft = 0;
SceneManager.LoadScene("GameOver", LoadSceneMode.Single);
}
```

NOW BUILD IT!

- Go to File -> Build Settings
- Add OpenScenes
- Build!



Even more you can do…

- Add different amounts of money
- Add something to avoid
- Change the concept, make your own sprites!

Resources

- https://docs.unity3d.com/Manual/index.html
- https://docs.unity3d.com/ScriptReference/index.html
- https://docs.unity3d.com/352/Documentation/ScriptReference/KeyCode.html