UI & Debug

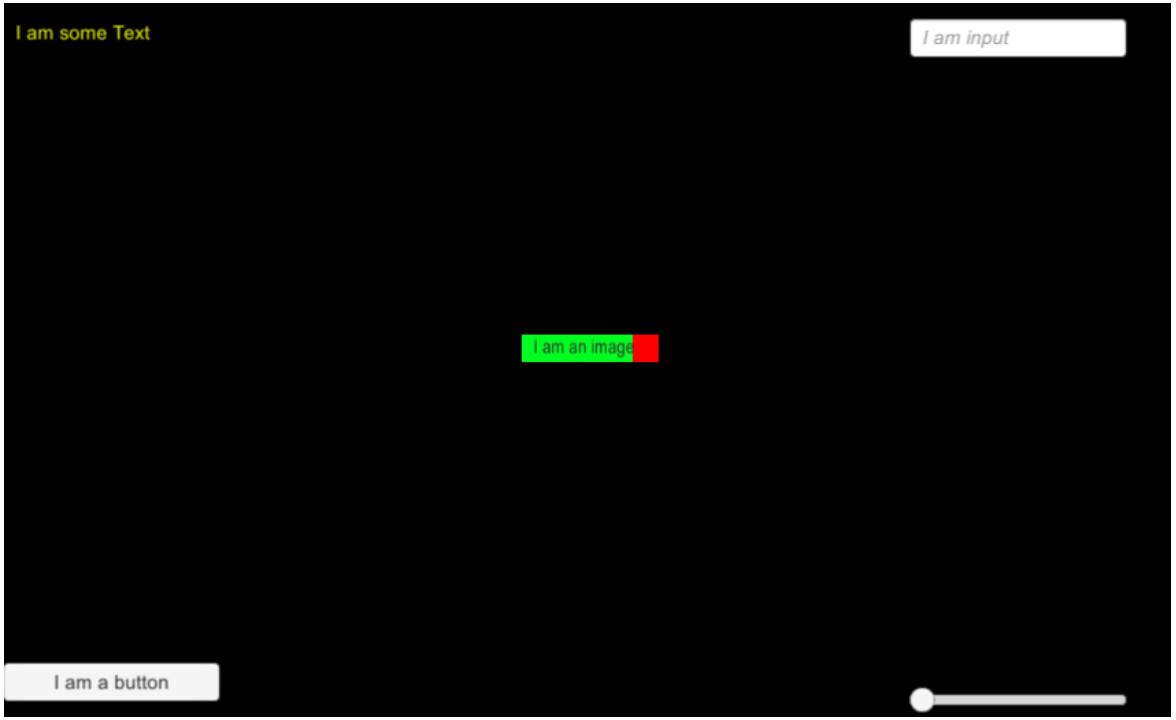
COMMUNICATING WITH THE PLAYER

Typical User Interface (UI)

- Keyboard & Mouse
  - Movement & Hotkeys
- Interacting with the player
  - Actions
  - Feedback
- Status
  - Score
  - Damage
- Communicating with the designer
  - Internal Status (sanity check)
  - Errors

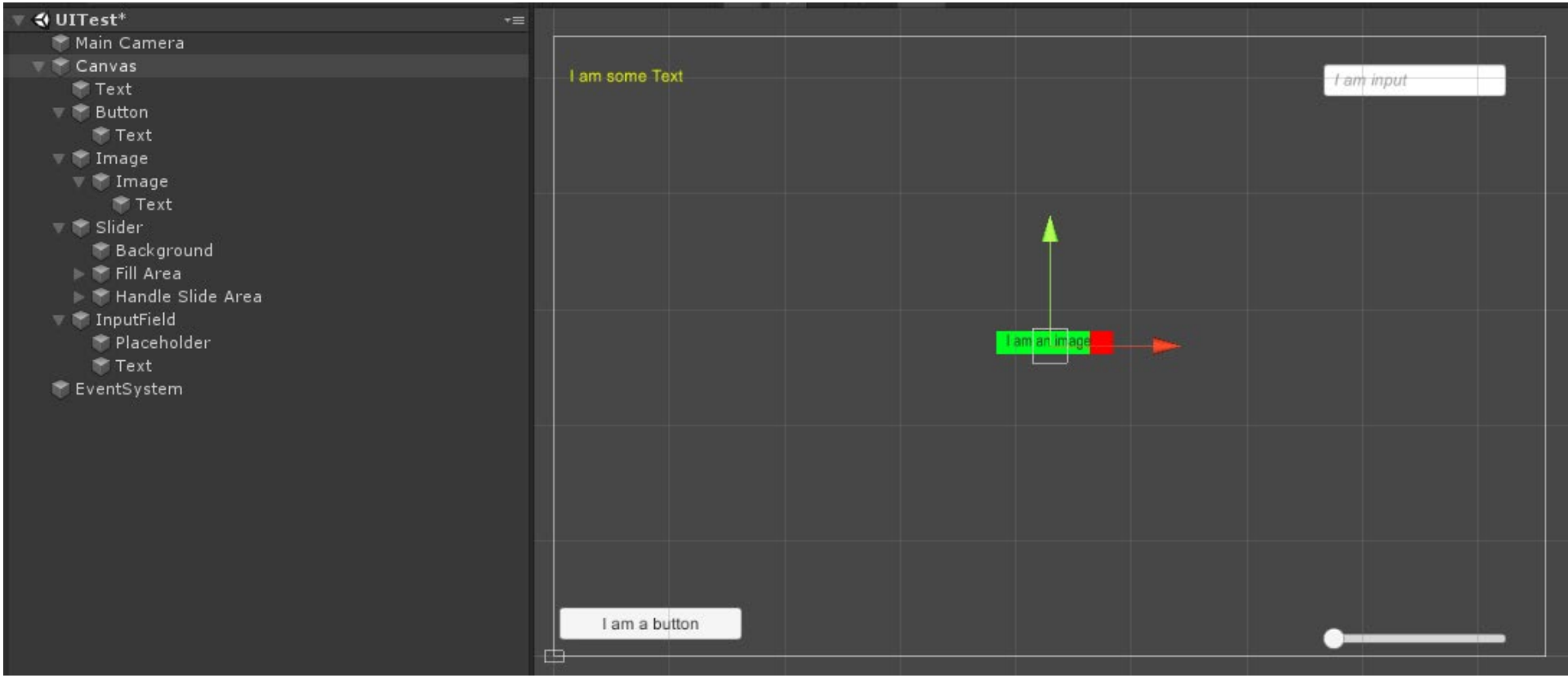
UI is typically 2D

1. Images
2. Text
3. Buttons
4. Sliders
5. Input field



UI Lives on a canvas

- The Canvas is the parent for all UI Elements



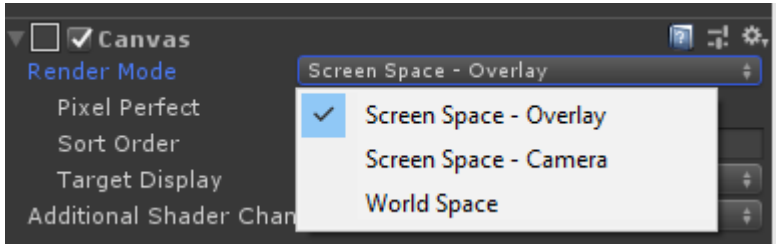
UI needs to get input

- When you add any UI element Unity will add

1. Canvas
   - This will scale the UI to fit the game screen
2. Event System
   - This takes input from the player

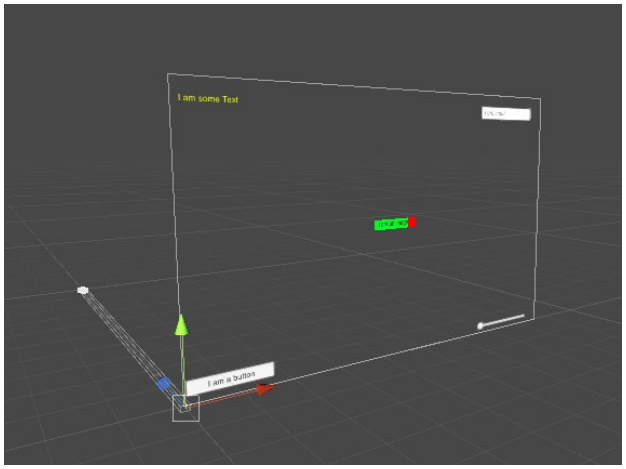
- **NB: Do not delete them as you cannot have a UI without a canvas and without the Event System you wont get input**
- The Event system will only be created once per scene

Types of Canvas



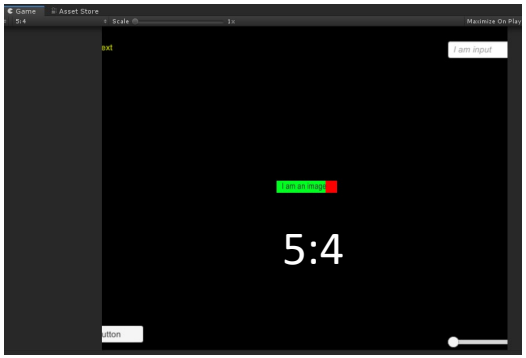
1. Screen Space – Overlay
   - You can have one of these per Scene, it overlays the scene
2. Screen Space –Camera
   - You can have one of these per Camera, it overlays the Camera
3. World Space
   - You can have lots of these as they live inside the scene as GameObjects
4. It may help to consider the Canvas as a 2D overlay in a 3D space

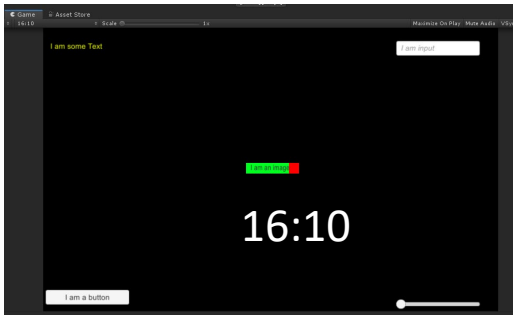
NB: UI is very complex
Research unity3d UI & watch unity tutorials to get insights

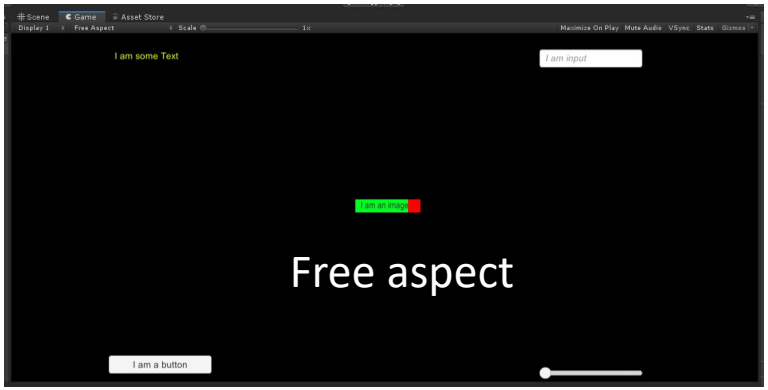


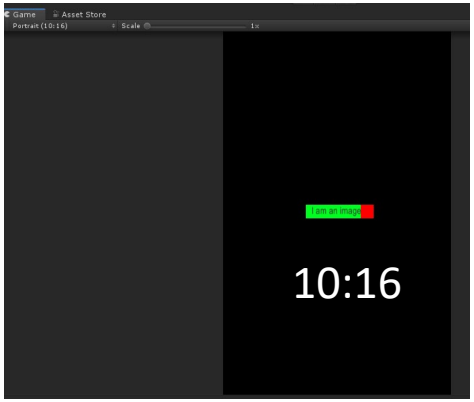
UI is device dependant

- Unity is always 3D, and 3D objects can scale in all directions, you may see more or less depending on the Game view



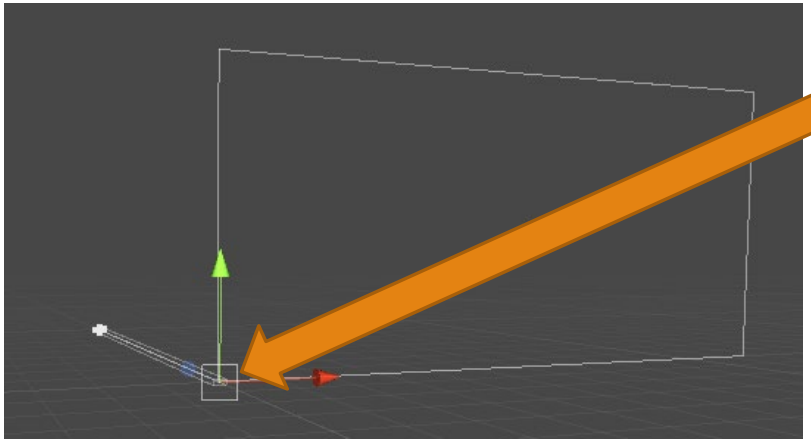


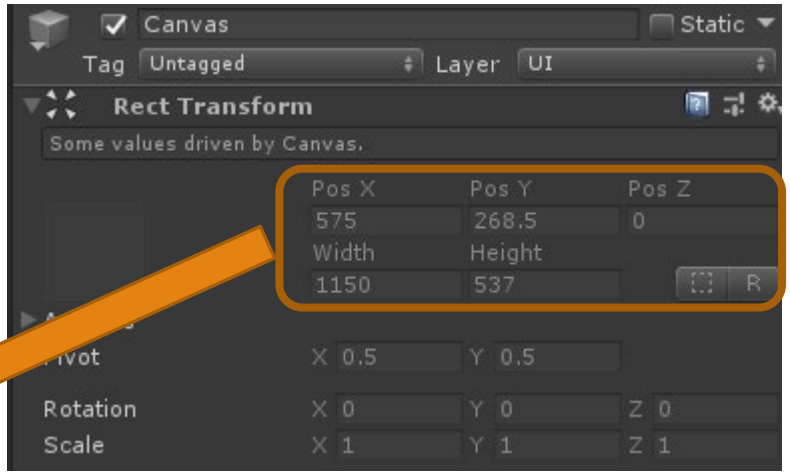




UI & Scale

- The canvas is huge, much bigger than typical game space
- For a Screen Space Canvas Unity scales it down to fit at runtime to fit the camera
- UI is more like pixels coordinates than what you see with GO

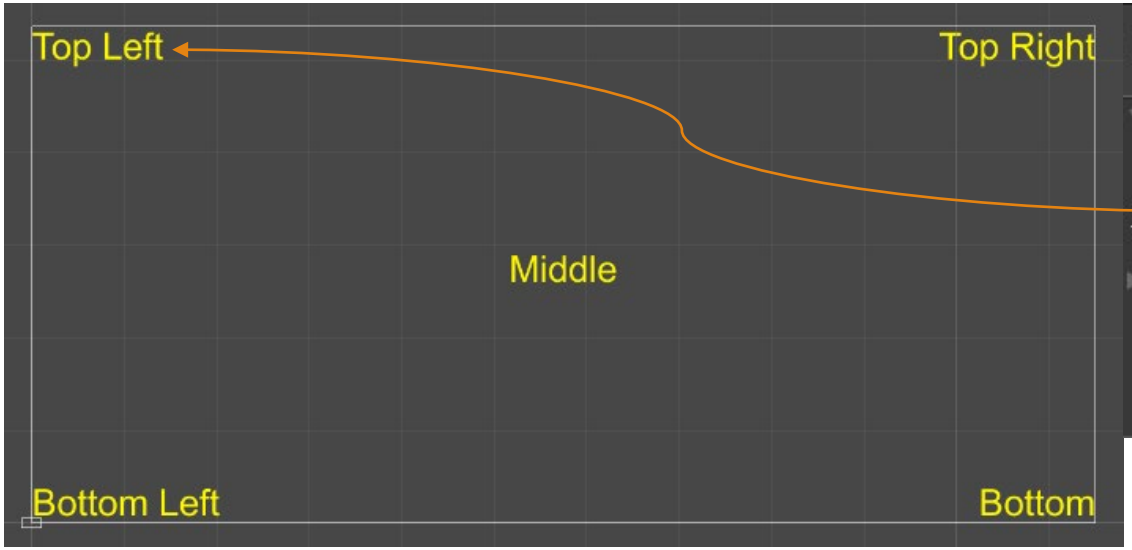


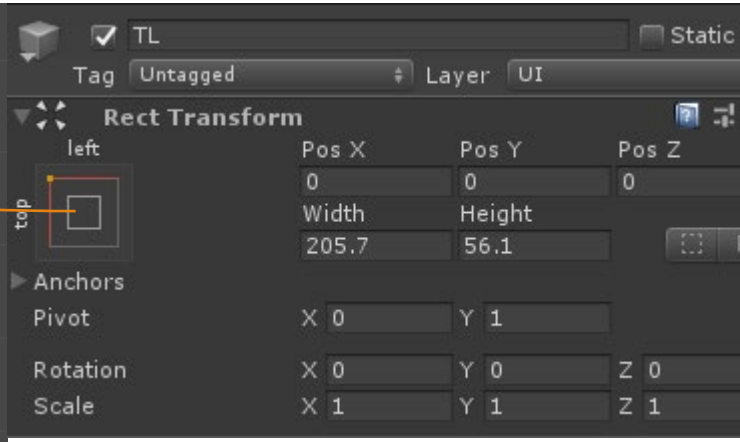


- UI uses a special transform called a Rect Transform

Using Anchors

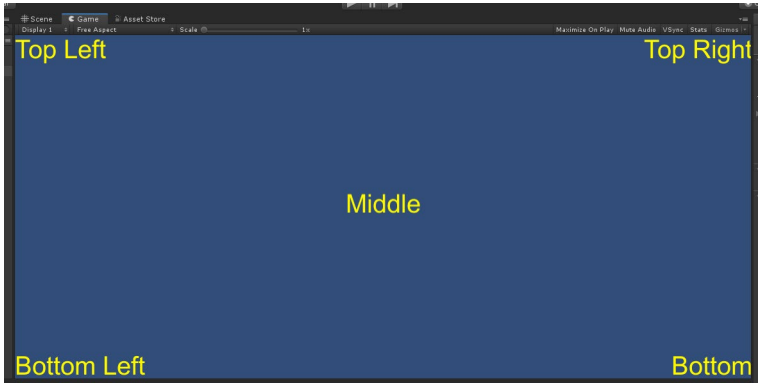
- As what the camera can see will vary with changes in screen size and aspect ratio it helps to have some invariants (things which don't change)

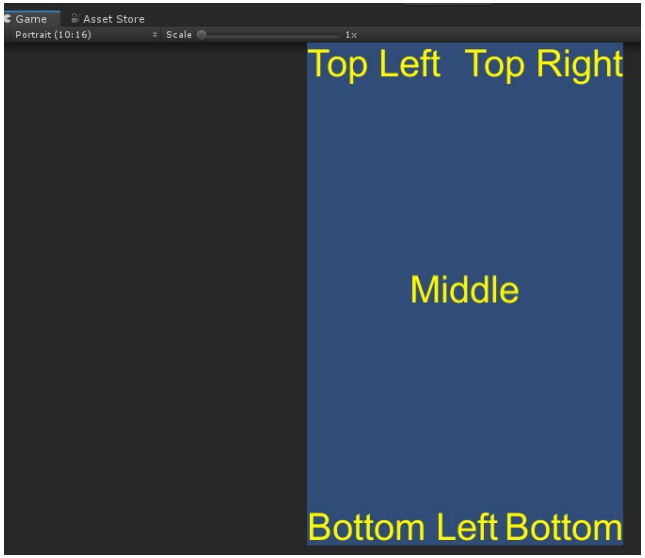




Device independence

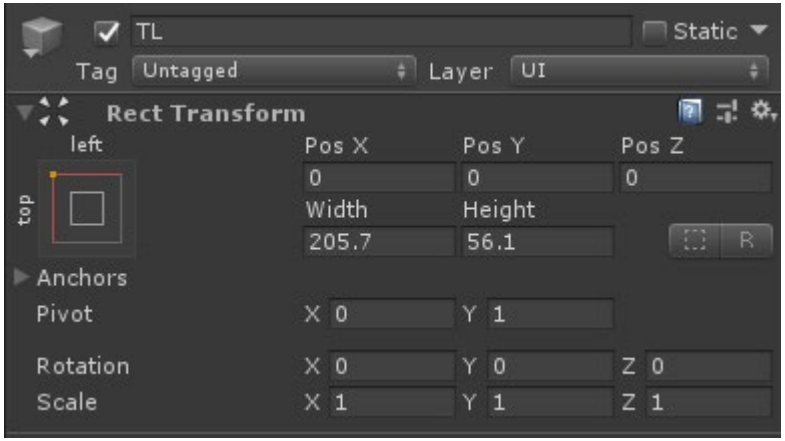
- Anchors offer some device independence





Pivot & Anchors

- The pivot is the origin of the UI object, it's a number from 0.0-1.0



- Its default it 0.5,0.5 i.e. the middle
- If its set to 0,1 as in this case the top left hand corner is now the origin and the object will take its position relative to this and the Anchor
- Pivots can be used to make it easier to deal with positions relative t anchors

Talking to UI from the game

- UI is relative new and was not part of Unity initially so it needs to be included manually

```
using UnityEngine.UI; //Allow us to talk to UI
```

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI; //Allow us to talk to UI
0 references
public class UpdateText : MonoBehaviour
{
    Text mText; //Variable to store (cache) Text Component
    0 references
    void Start()
    {
        mText = GetComponent<Text>(); //Get Text component from game Object
        Debug.Assert(mText != null, "You are missing the Text Component"); //Error Message
    }

    0 references
    void Update()
    {
        mText.text = string.Format("{0}", Random.Range(0, 100)); //Just print some random number 0-99 to test
    }
}
```

Debug.

- Debug.Log(“Error”);
  - Prints an error message
- Debug.LogFormat(“Error {0}”, error);
  - Prints formatted error message to console
- Debug.Assert(condition,”Error Message”);
  - Check a condition, if FALSE, prints error (get compiled out in production code)
- Debug.AssertFormat(condition,”Error {0}”,error);
  - Check a condition, if FALSE, prints formatted error message

Additional research

- UI
  - https://unity3d.com/learn/tutorials/topics/user-interface-ui/ui-canvas?playlist=17111
- String formatting
  - https://www.tutlane.com/tutorial/csharp/csharp-string-format-method

Workshop

ADDING SCORE AND

Adding score

1. Add a public Score variable (int) to GM
2. Make the singleton public, so we can access it from other classes

```
public static GM sSingleton; //Make static singleton, this will be shared by all

public int Score = 0;
```

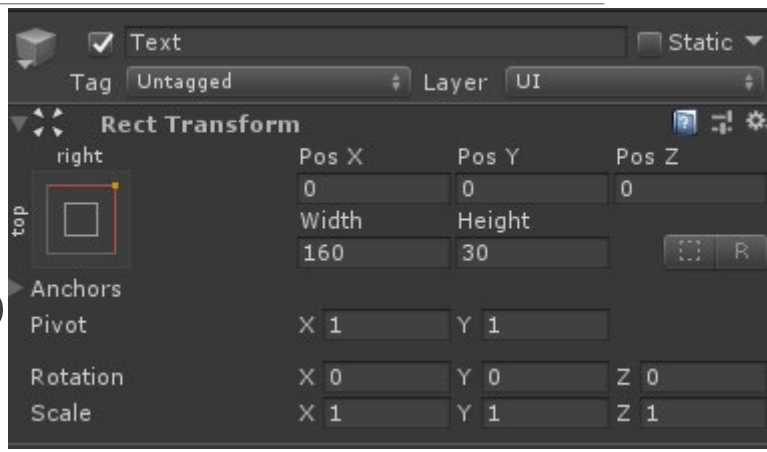
3. Add code to increase score when hitting rocks in BigRock.cs

```
protected override void CollidedWith(FakePhysics vOtherFF) {
    //We do not call parent as we will handle
    if (vOtherFF is BulletFP) {  //Were we hit by a bullet
        Destroy(gameObject);    //Destroy Asteroid
        Destroy(vOtherFF.gameObject);    //Destroy Bullet
        GM.SpawnPrefab(GM.SpawnIDs.Explosion, transform.position, Random.Range(-180, 180));// Exposion
        GM.SpawnPrefab(GM.SpawnIDs.AsteroidMedium, transform.position, Random.Range(-180, 180)); //Medium Rock
        GM.SpawnPrefab(GM.SpawnIDs.AsteroidMedium, transform.position, Random.Range(-180, 180)); //Medium Rock

        GM.sSingleton.Score += 100; //Give player score
    }
}
```

Display the score

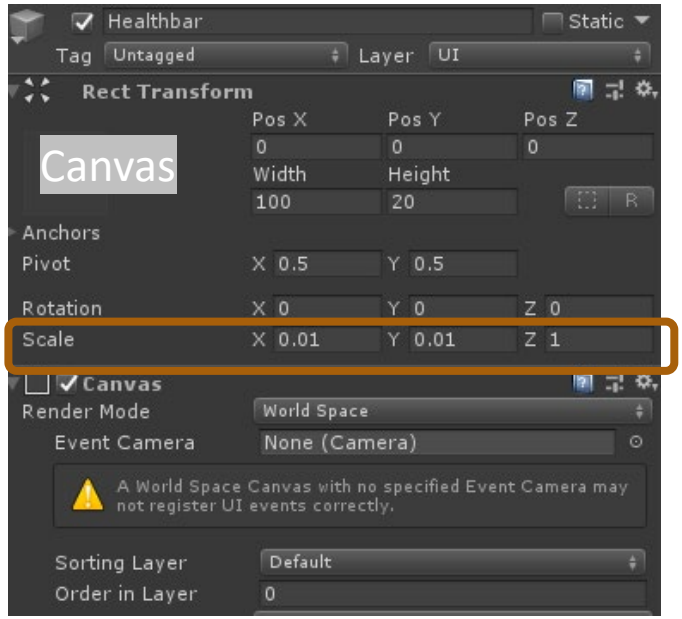
1. Add a Text UI component to game
2. Edit text to place it top right with anchor Note use of pivots
3. Add code to update score text to Text GO

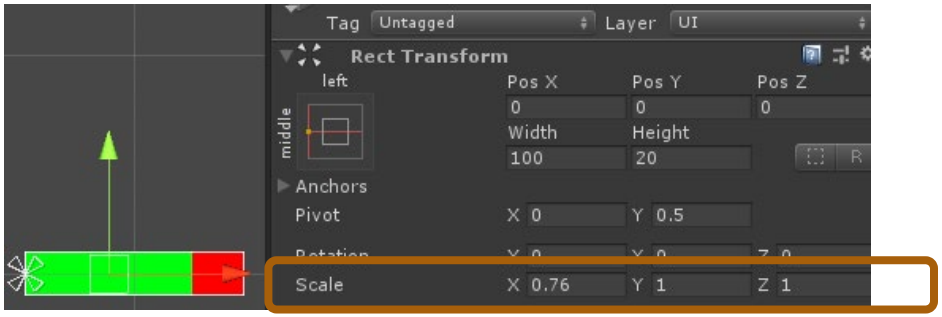



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
0 references
public class UpdateScore : MonoBehaviour
{
    Text mText; //Variable to store (cache) Text Component
    0 references
    void Start()
    {
        mText = GetComponent<Text>(); //Get Text component from game Object
        Debug.Assert(mText != null, "You are missing the Text Component"); //Error Message
    }
    0 references
    void Update()
    {
        mText.text = string.Format("Score:{0}", GM.sSingleton.Score); //Get Score from GM
    }
}
```

World space UI (Health bar)

- Lives in game world, manually scaled in IDE or code





Homework

- Make all the Rocks behave correctly
  1. Add score
  2. Reduce player Health
  3. Get destroyed
  4. And spawn new smaller rocks , except smallest one

Full catchup workshop download on student central