UNITY GAME DEVELOPMENT COURSES

Unit 4

Unity **Game Play and 2D game**

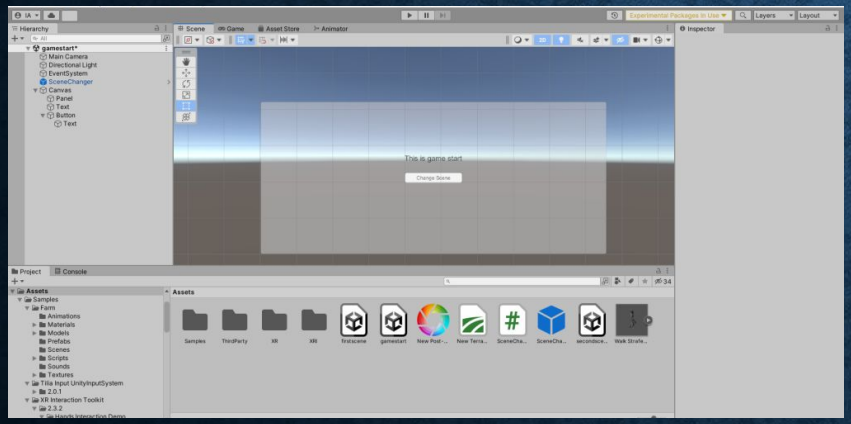
TOPICS

- Game Level Design using ProBuilder or other map editor
- Post Processing Unity : HDRP
- 3D Object Creation
  - Rendering
  - Texture
  - Lighting

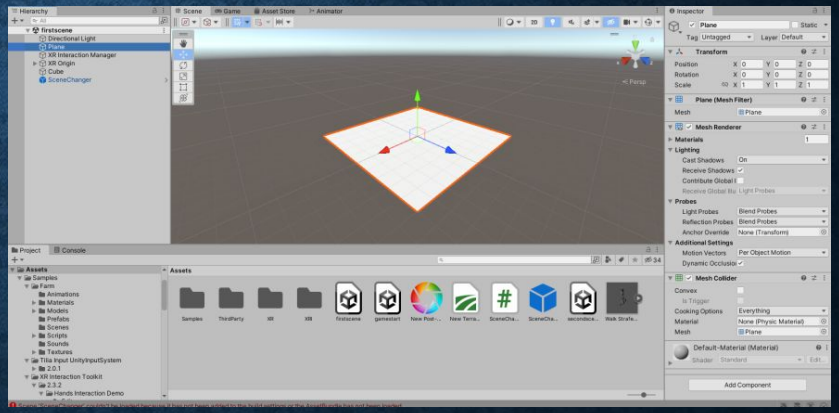
CHANGING SCENES IN UNITY 3D

- create 2 new scenes and name them **“gamestart”** and **“firstscene”**.

gamestart

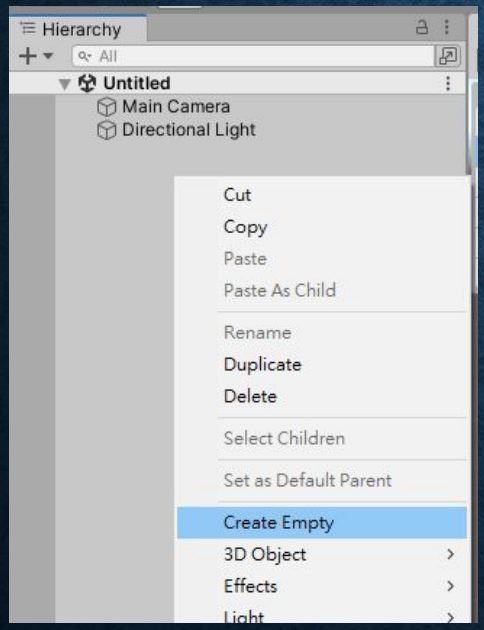


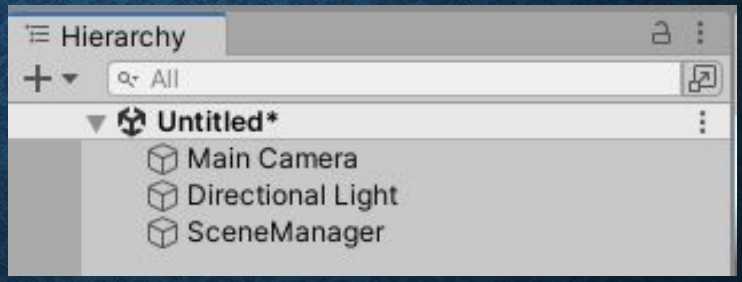
firstscene



CHANGING SCENES IN UNITY 3D

- In the **Hierarchy tab** of "**gamestart**", create an **Empty Game Object** and name it "**SceneChanger**“





CHANGING SCENES IN UNITY 3D

- Add a script named **"SceneChanger"** from github into the **Assets**

Inspector

Scene Changer (Mono Script) Import S

Open | Execution Order...

Imported Object

Scene Changer (Mono Script)

Assembly Information

Filename Assembly-CSharp.dll

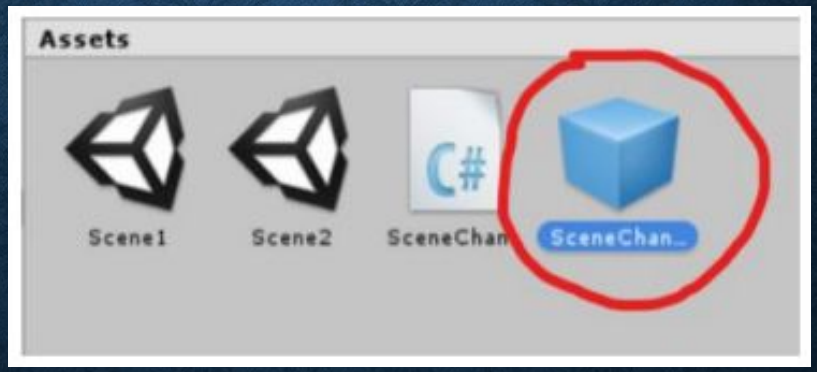
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneChanger : MonoBehaviour
{
        public void ChangeScene(string sceneName)
        {

SceneManager.LoadScene(sceneName);
        }
        public void Exit()
        {
                Application.Quit ();
        }
}
```

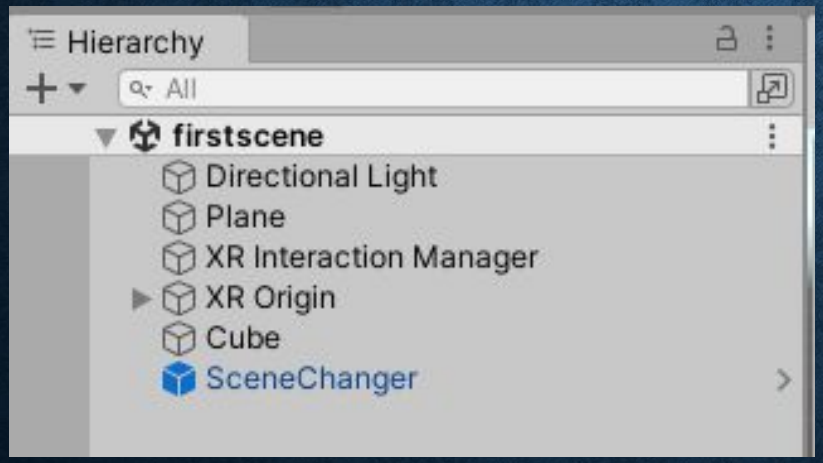
CHANGING SCENES IN UNITY 3D

- **Select the object in the Hierarchy** and **drag-drop it inside the Assets folder in the Project tab**. After doing this you will see a **blue cube** with the same name of the object. This is the object prefab.
- save the scene as "**gamestart**"



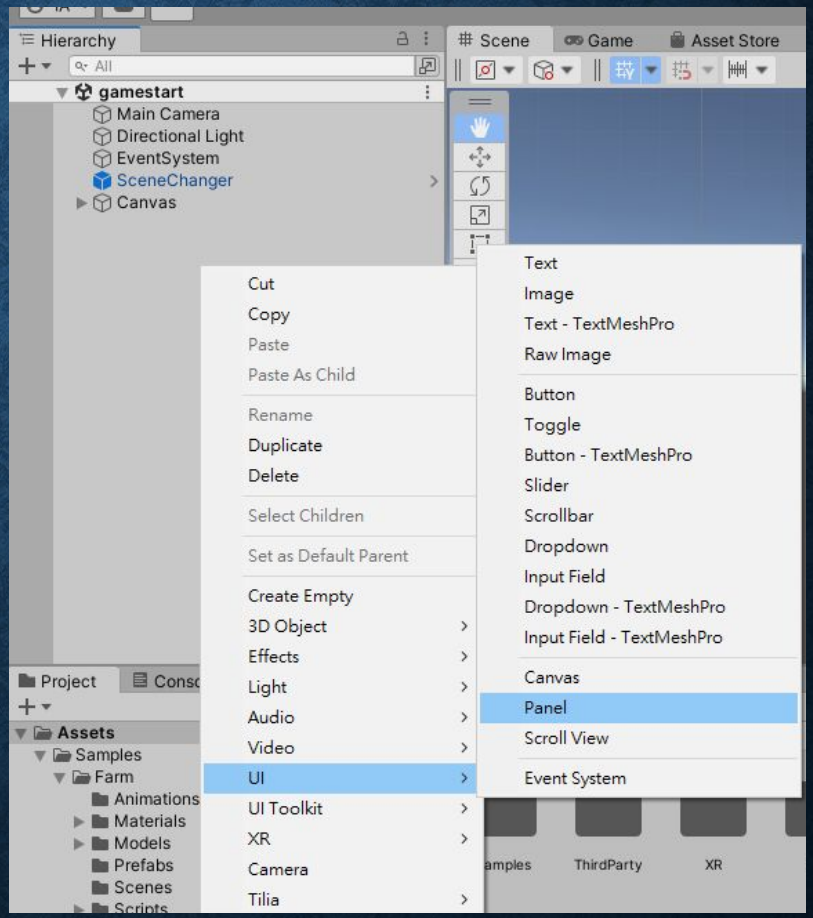
CHANGING SCENES IN UNITY 3D

- Double click on **“firstscene”.** In this scene, simply **drag and drop** the **SceneChanger prefab** into the **Viewport** or **Hierarchy tab.** An instance of the **SceneChanger** will be created.



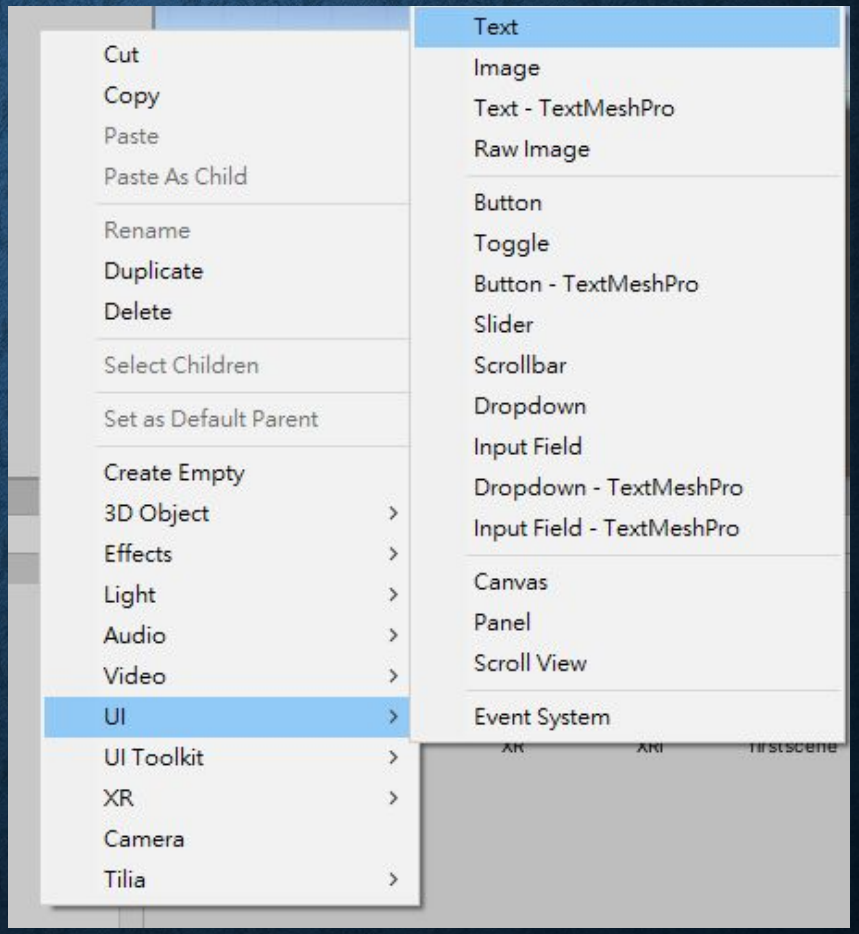
CREATING THE UI AND PREFAB

- **Right click** in the **Hierarchy tab** and **select UI > Panel**.
- Change the color of the panel to whatever you desire (try it yourself).



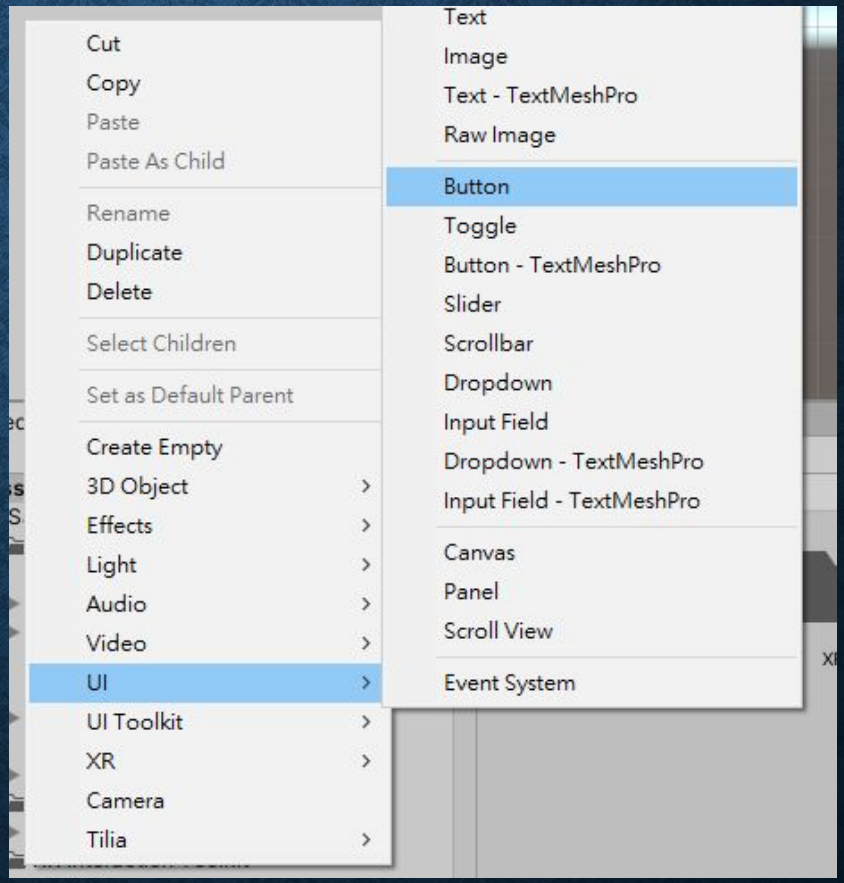
CREATING THE UI AND PREFAB

- Again **right click** on the **Hierarchy tab** and **select UI > Text.** Resize it and place it anywhere you desire (try yourself)



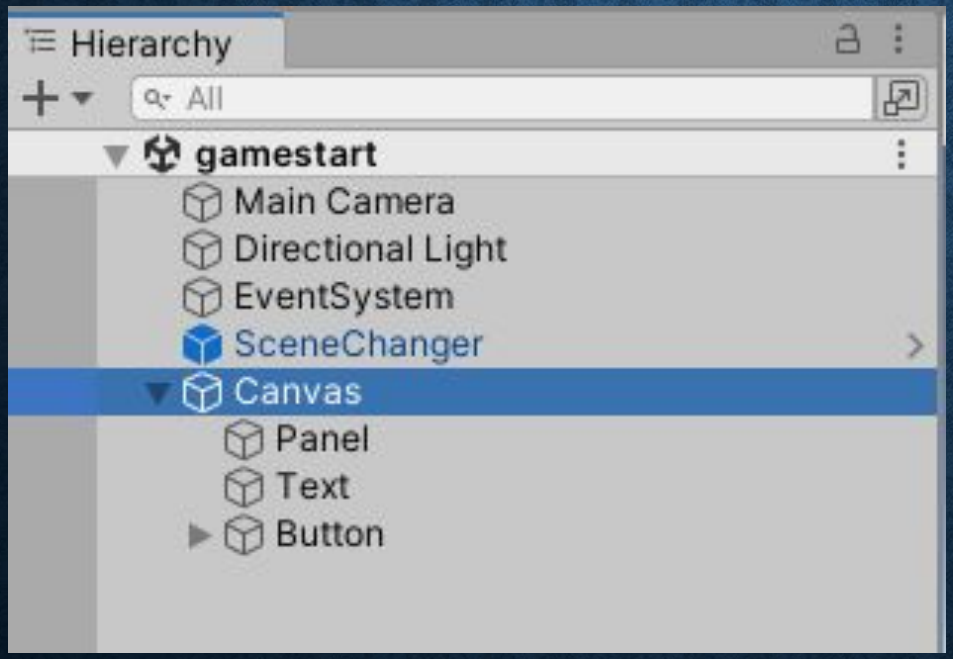
CREATING THE UI AND PREFAB

- Once more, **right click** in the **Hierarchy tab** and **select UI > Button.** Resize it and place it anywhere you desire (try yourself).



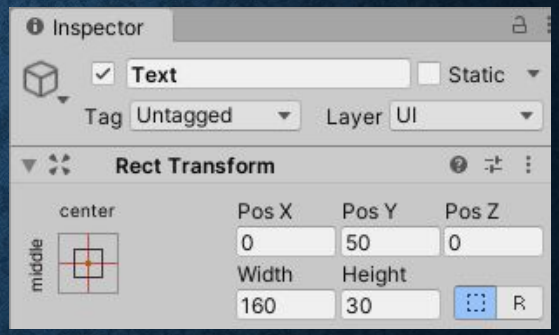
CREATING THE UI AND PREFAB

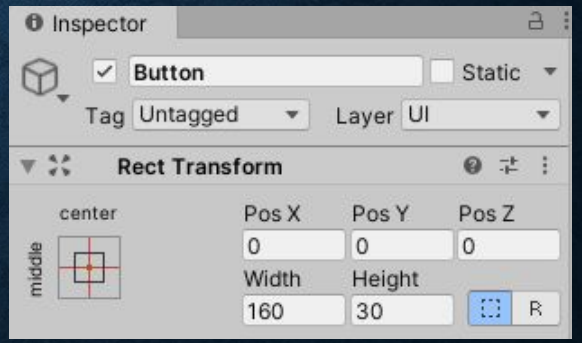
- The screen of all UI – Canvas is as follows:



CREATING THE UI AND PREFAB

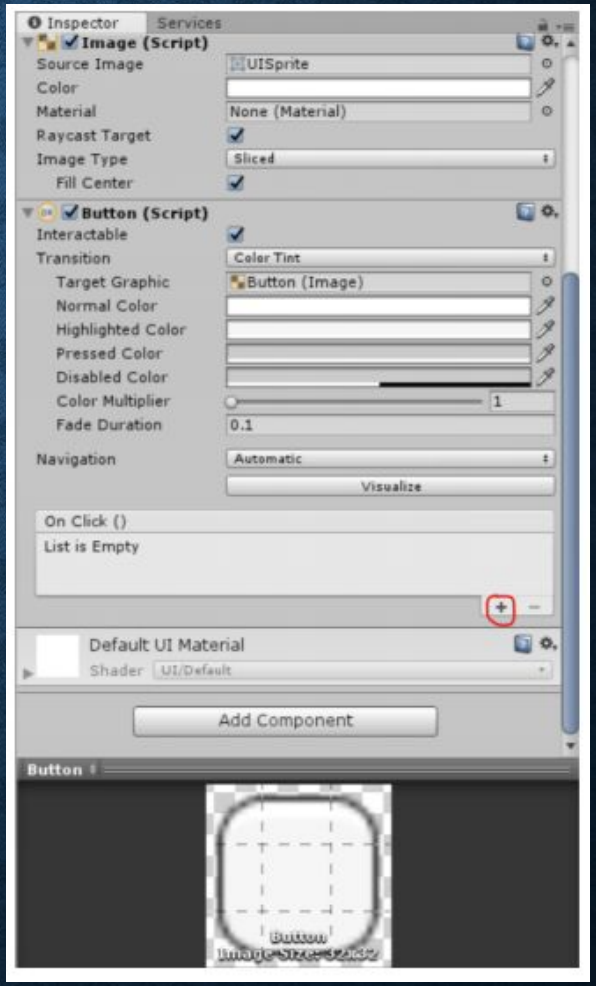
- change the text to **“This is game start”**. Also change the text in the button to **“Change Scene”**





CREATING THE UI AND PREFAB

- select the **Button** object in the **Hierarchy**. In the **Inspector** tab click on the plus icon in the **Button (Script)** component.

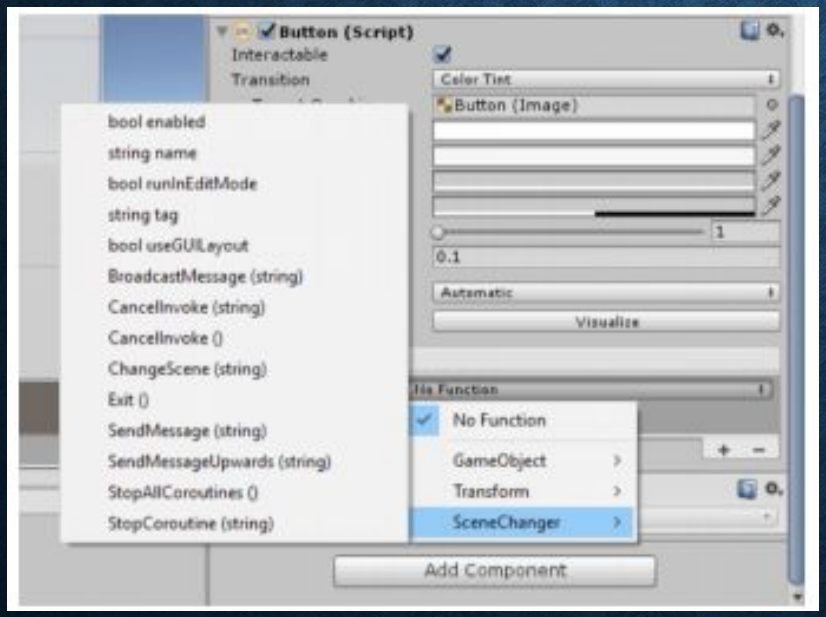


CREATING THE UI AND PREFAB

- Drag and drop the **SceneChanger** object from the **Hierarchy** tab into the **object reference space** (highlighted in red).

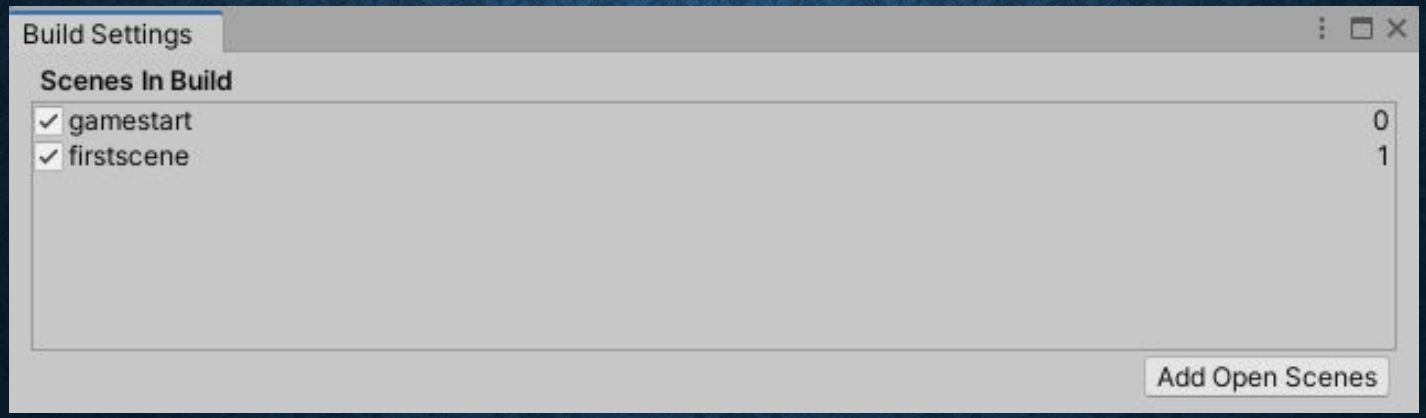
CREATING THE UI AND PREFAB

- Select the **Function Drop Down** (highlighted in red). From the drop down, select **SceneChanger > ChangeScene(string).**



DEMO

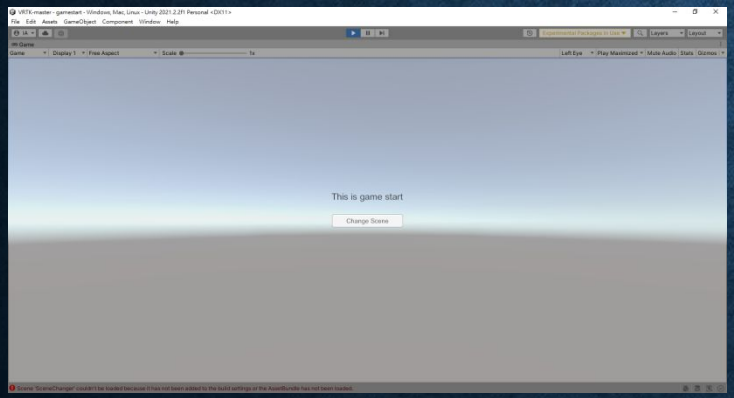
- Click **File -> Build Settings**
- Drag two scene into "**Scenes in Build**"



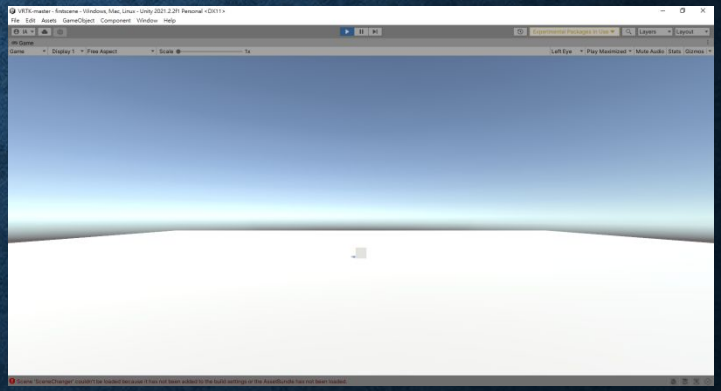
- Click Play

DEMO

gamestart



firstscene



QUICK PROTOTYPING TOOLS

- ProBuilder - https://unity3d.com/unity/features/worldbuilding/probuilder
- Realtime CSG - https://assetstore.unity.com/packages/tools/modeling/realtime-csg-69542
- **MAST - Modular Asset Staging Tool -** https://assetstore.unity.com/packages/tools/level-design/mast-modular-asset-staging-tool-154939

UNITY SCENE ASSETS

- Free Package
- https://assetstore.unity.com/packages/3d/characters/easyroads3d-free-v3-987
- https://assetstore.unity.com/packages/3d/environments/roadways/windridge-city-132222
- https://assetstore.unity.com/packages/3d/environments/3d-scifi-kit-starter-kit-92152

UNITY SCENE ASSETS

- Download https://assetstore.unity.com/packages/3d/environments/industrial/rpg-fps-game-assets-for-pc-mobile-industrial-set-v2-0-86679

UNITY SCENE ASSETS - FOLDER

- RPG_FPS_game_assets_industrial
  - Assets_showcase_scene.unity
  - Barrels
  - Boxes
  - Buildings
  - Containers
  - Dumpsters
  - Fences
  - Map_v1.unity
  - Map_v1
  - Oil_tanks
  - Other_props
  - Particles
  - Roads
  - Textures

UNITY SCENE ASSETS - FOLDER

- Buildings
  - Industrial
    - Hangars
      - Hangar_v1
        - Hangar_v1_full.prefab
        - Hangar_v1_half.prefab
        - Source
          - Hangar_v1.FBX
          - Hangar_v1.mat
          - Hangar_v1.tga
      - Hangar_v2
      - Hangar_v3
      - Hangar_v4

INV4015 Unity Game Development Courses

TOPICS

- Game Play Implementation
  - Mini-Map, Life Bar, Score
  - Voice and Music Production
  - Building Enemies, Navigation

TOPICS

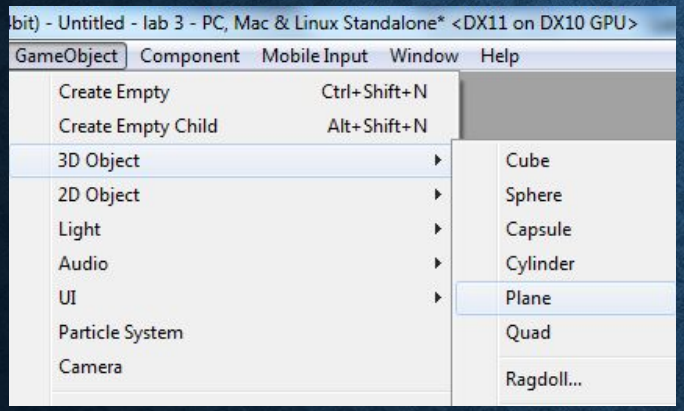
- Spawning and Tidying
- Timer
- Scene Interchange
- Game Data Manipulation

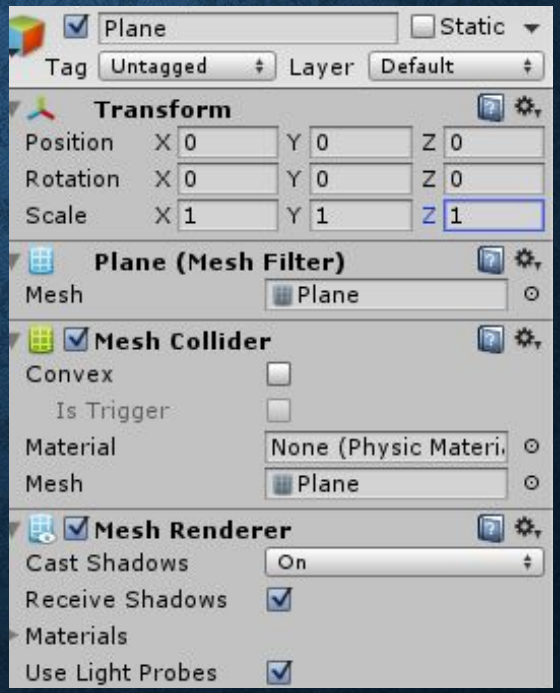
SPAWNING - INSTANTIATE GAME OBJECT

- During game play, we have to duplicate a number of game object such as bullet, explosion particles …
- Let's think of the generation of the coins "Coin Dozer"
- Before we can duplicate the game object, we have to convert it to **prefab**

INSTANTIATE GAME OBJECT

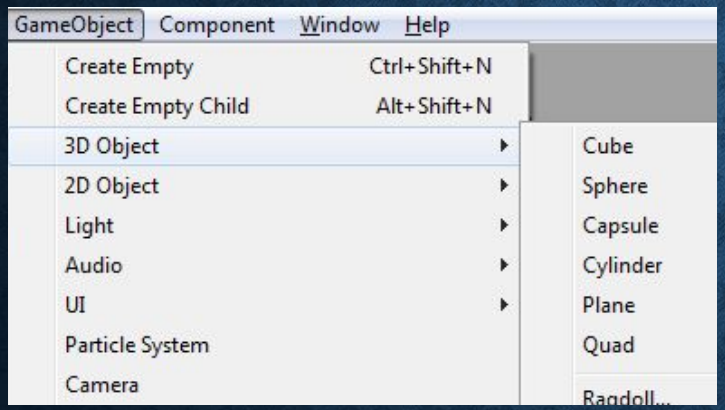
- Create a plane as follows:
  - GameObject >> Create 3D Object>> Plane

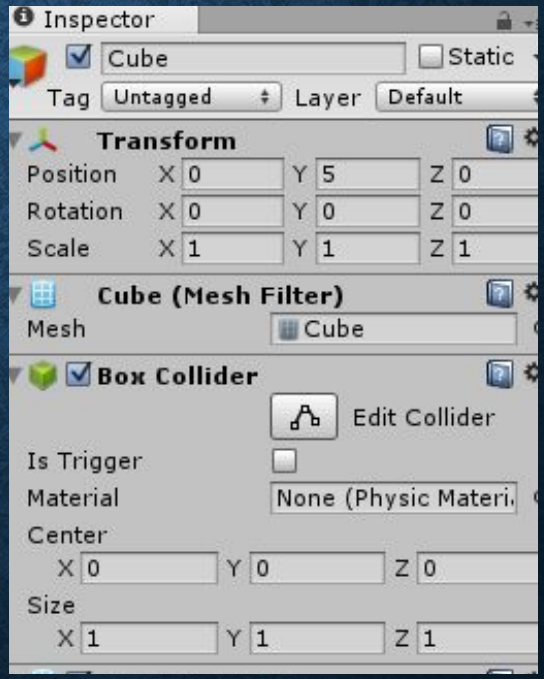




INSTANTIATE GAME OBJECT

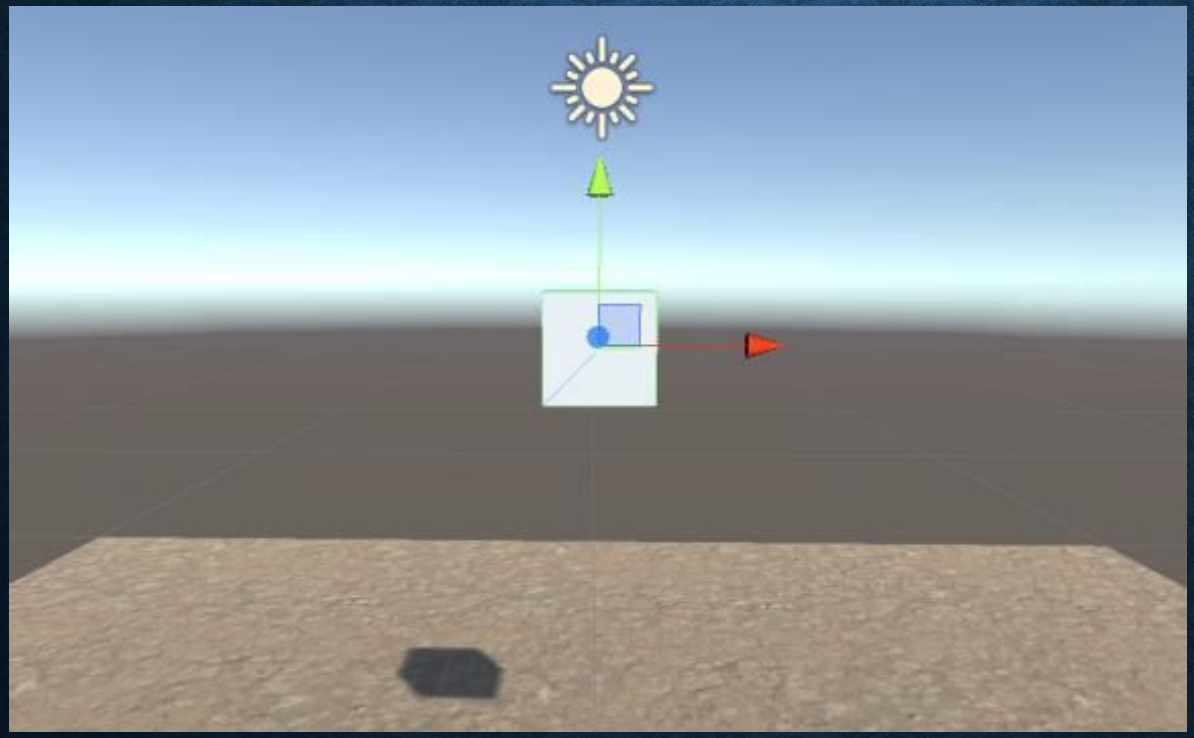
- Create an box and name it as "source"
  - GameObject -> Create 3D Object -> Cube





INSTANTIATE GAME OBJECT

- Place the box at the top



INSTANTIATE GAME OBJECT

- Import the model (e.g. Coin) to the scene for duplication
  - Add the rigid body component to it
  - Component -> Physics -> Rigidbody
- Drag it to the project panel so that it will become a prefab
  - The name will be changed to blue in color in hierarchy panel
- Delete the model from the scene

INSTANTIATE GAME OBJECT

- Create a C# script with the name "Duplicate"
- Put the following script inside

Duplicate.cs

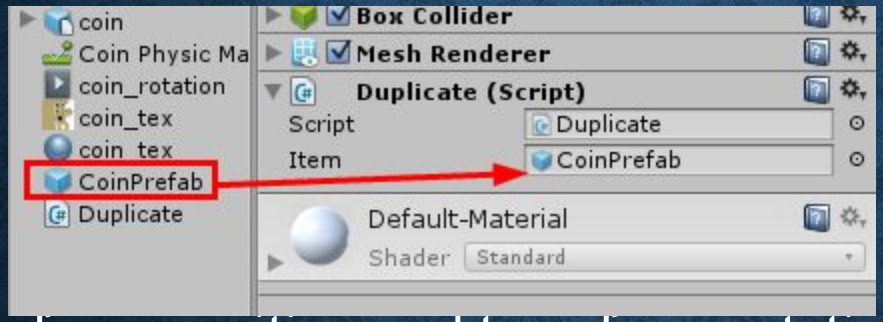
selection

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class Duplicate : MonoBehaviour {
5
6      public GameObject item;
7
8      void OnMouseDown () {
9          Instantiate(item, transform.position, transform.rotation);
10     }
11 }
```

- Drag the script to the "Cube" object
- Drag the coin prefab to the item slot

INSTANTIATE GAME OBJECT

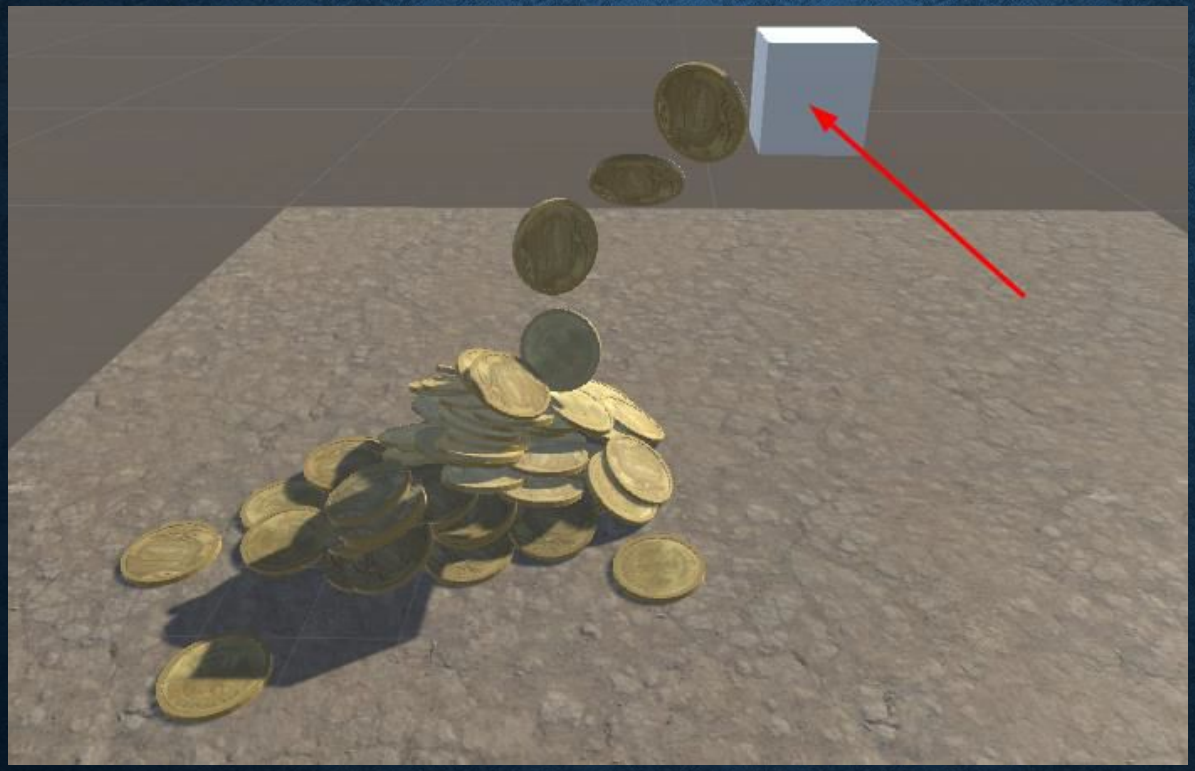
- Select the source object, drag the box prefab from the project panel to the item variable of the script



- The transform.position and transform.rotation will return the source position and rotation to the newly duplicated object

INSTANTIATE GAME OBJECT

- Try to click the source to duplicate the box



INSTANTIATE GAME OBJECT

- As the "Cube" also contains collider, it will collider with the duplicated items
- Use the Physics.IgnoreCollision to ignore the collision between the source and the duplicated items
  - E.g. prevent bullet collider with the gun

Duplicate.cs

o selection

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class Duplicate : MonoBehaviour {
5
6      public GameObject item;
7
8      void OnMouseDown () {
9          GameObject clone = Instantiate(item, transform.position, transform.rotation) as GameObject;
10         Physics.IgnoreCollision (clone.GetComponent<Collider>(), GetComponent<Collider>());
11     }
12 }
13
```

INSTANTIATE GAME OBJECT

- Now the coins drop vertically without colliding with the source

SPAWNER

- Use the InvokeRepeating function to repeat calling any function by a particular time interval
  - Usage: InvokeRepeating("function name", start delay, interval)

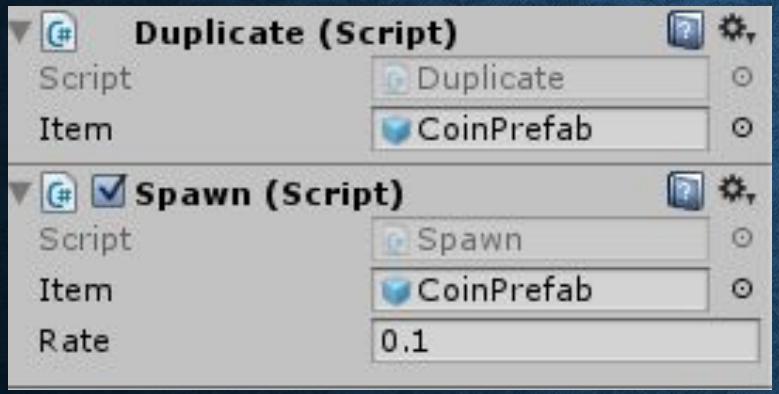
Spawn.cs | Duplicate.cs

selection

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class Spawn : MonoBehaviour {
5
6      public GameObject item;
7      public float rate=0.5f;
8
9      void Start () {
10         InvokeRepeating ("Duplicate", 0f, rate);
11     }
12
13     void Duplicate () {
14         GameObject clone = Instantiate(item, transform.position, transform.rotation) as GameObject;
15         Physics.IgnoreCollision (clone.GetComponent<Collider>(), GetComponent<Collider>());
16     }
17 }
```

SPAWNER

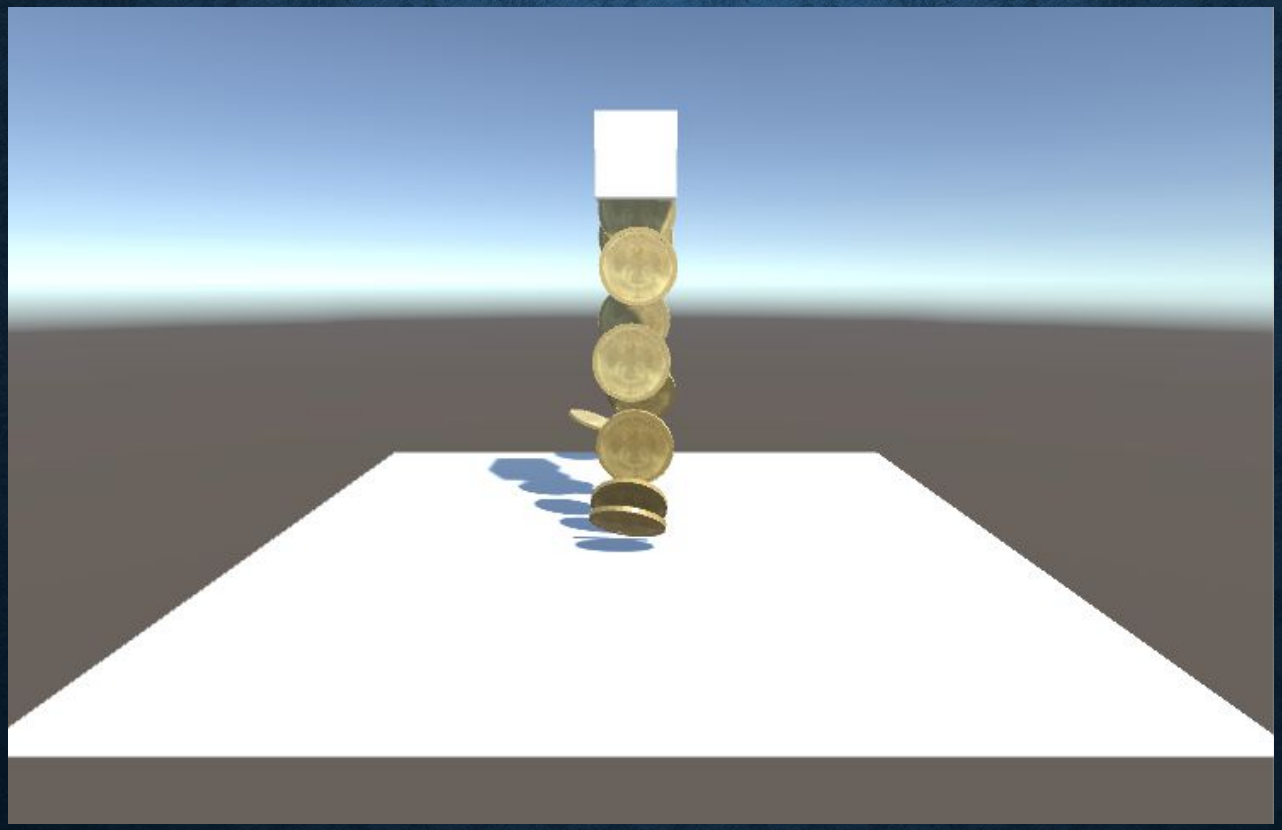
- Select the "Cube" object, drag the C# script from the project panel to it.



- You may change the rate to allow different speed on the "Coin" Object.

SPAWNER

- Now the coins automatically drop vertically without clicking the Cube.



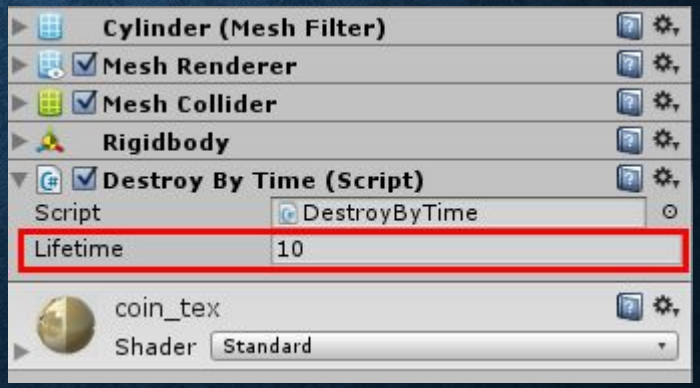
GUN FIRE

- Some items may not be used any more have to be destroyed to maintain performance
  - E.g. bullet shell, fire effect…
- Use Destroy (gameobject, delay) function
- Add the following script "DestroyByTime" to the coin prefab and set the life time

DestroyByTime.cs

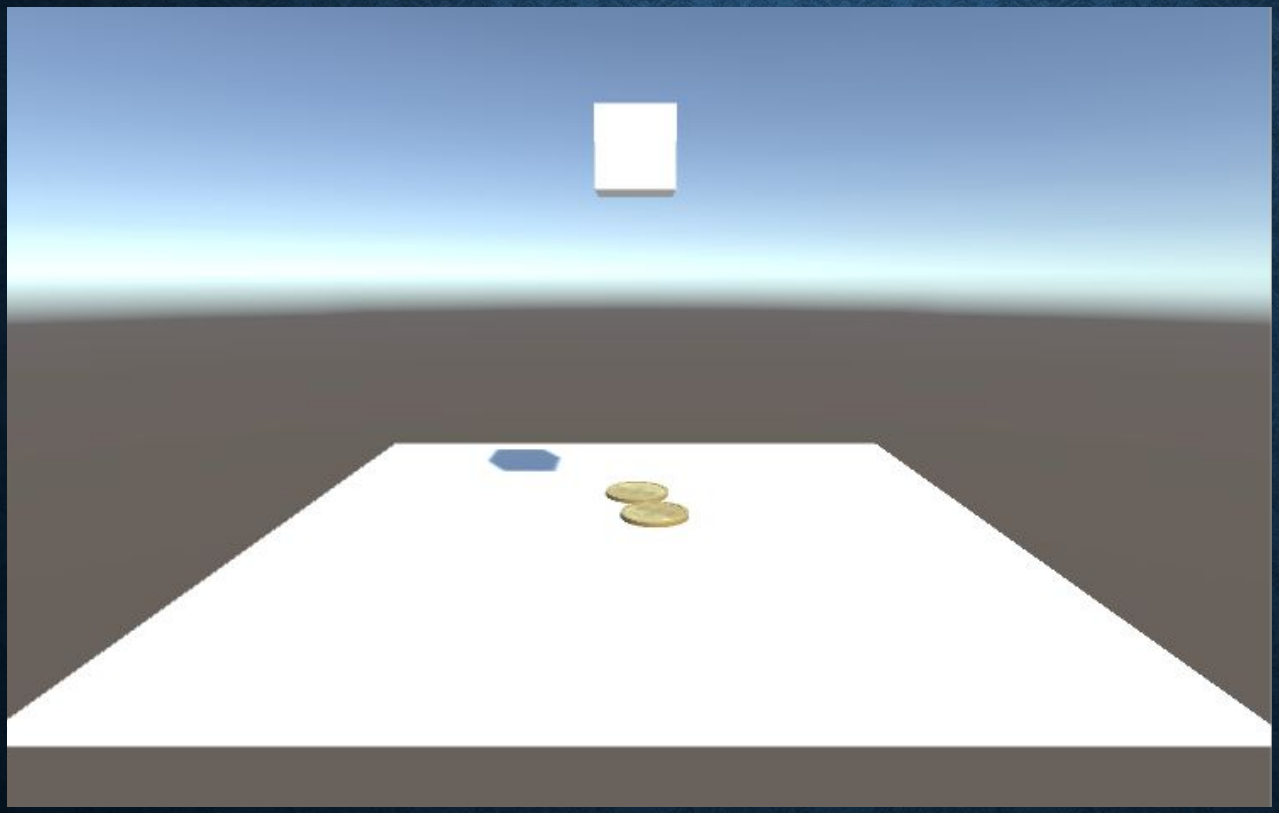
o selection

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class DestroyByTime : MonoBehaviour {
5
6      public float lifetime;
7
8      void Start () {
9          Destroy (gameObject, lifetime);
10     }
11 }
```



GUN FIRE

- Now the coins disappear after suitable period.



GUN FIRE

- Import package “Bullet”
  - Assets >> Import Package >>Custom Package

GUN FIRE

- The following script used raycast to fire
- When objects with Rigidbody is hit, force will be applied and the spark FX is instantiated

Fire.cs

o selection

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class Fire : MonoBehaviour {
5
6      public float range=20;
7      public float force= 20;
8      public GameObject hitFX;
9      public AudioClip hitSound;
10
11     AudioSource audio;
12
13     void Start () {
14         audio = GetComponent<AudioSource>();
15     }
```

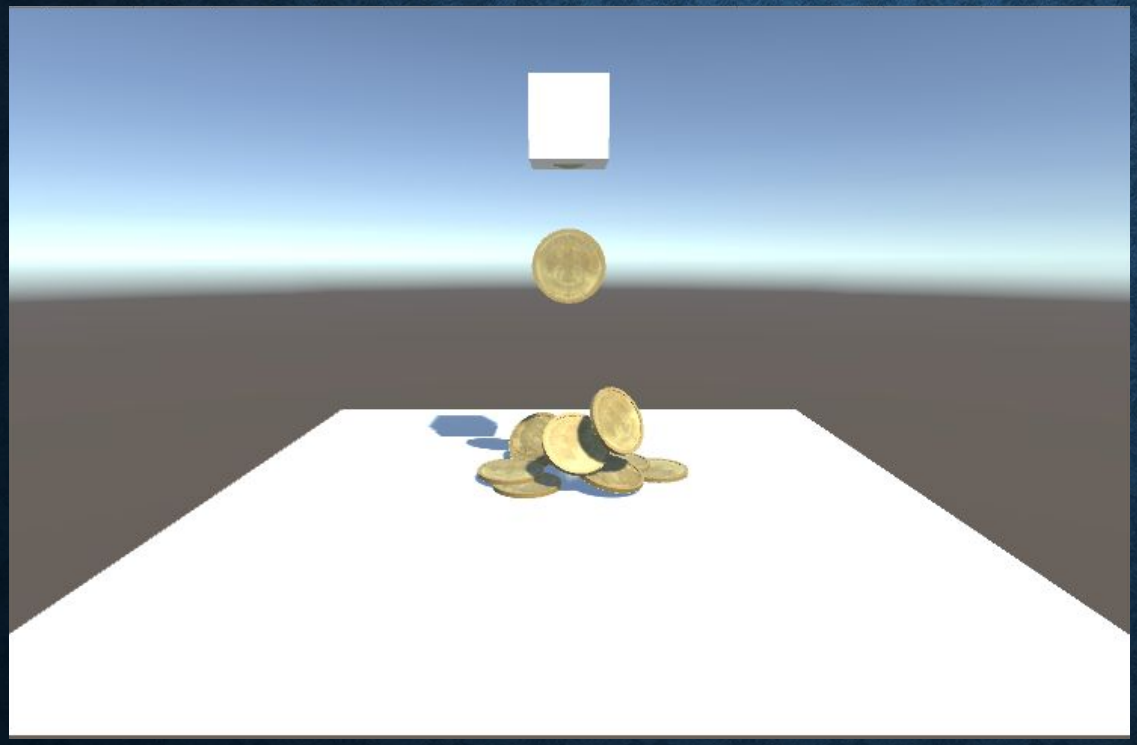
GUN FIRE

- Create an empty object with audio source and assign the script to "Main Camera".


```
16
17      void Update () {
18          if (Input.GetButtonDown("Fire1")) {
19              Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
20              Debug.DrawRay(ray.origin, ray.direction*range, Color.green);
21              RaycastHit hit;
22              audio.PlayOneShot(hitSound);
23              if (Physics.Raycast (ray, out hit, range)) {
24                  Instantiate (hitFX, hit.point, Quaternion.identity);
25                  if (hit.rigidbody) {
26                      hit.rigidbody.AddForceAtPosition(force*ray.direction,
27                                                       hit.point, ForceMode.Impulse);
28                  }
29              }
30          }
31      }
32  }
```

GUN FIRE

- Now you can shot the coins and destroy it.



COIN HIT

- Give life to each coins by the following script

Fire.cs | CoinBehaviour.cs

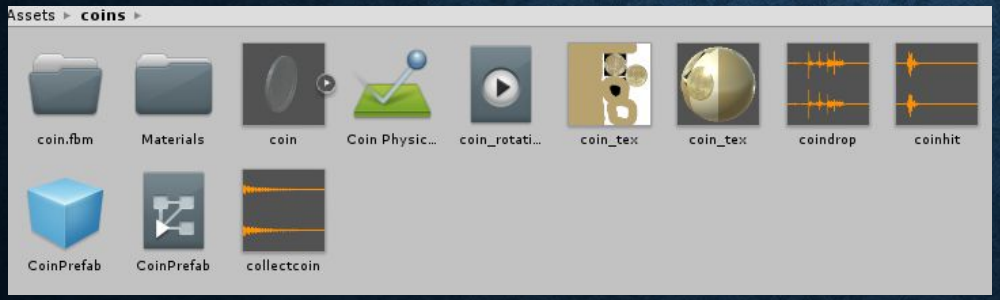
selection

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class CoinBehaviour : MonoBehaviour {
5
6      public float life=1;
7      public AudioClip hitSound;
8      public AudioClip collectSound;
9
10     AudioSource AS;
11
12     void Start () {
13         AS = GetComponent<AudioSource>();
14     }
15
16     void Hit (float damage) {
17         life -= damage;
18         if (life > 0)
19             AS.PlayOneShot(hitSound);
20         else
21             Kill ();
22     }
23
24     void Kill () {
25         AudioSource.PlayClipAtPoint(collectSound, transform.position);
26         Destroy(gameObject);
27     }
28 }
```

COIN HIT

- Drag the "CoinBehaviour" script to the "CoinPrefab" object.

- Drag "coinhit", "collectcoin" and "coindrop" into the three variable on Coin Behaviour Script



COIN HIT

- Modified the Fire script, use Sendmessage to call the Hit function of the coin

```
if (Physics.Raycast (ray, out hit, range)) {
    Instantiate (hitFX, hit.point, Quaternion.identity);
    if (hit.rigidbody) {
        hit.rigidbody.AddForceAtPosition(force*ray.direction,
                                         hit.point, ForceMode.Impulse);
        hit.transform.SendMessage("Hit", 1);
    }
}
```

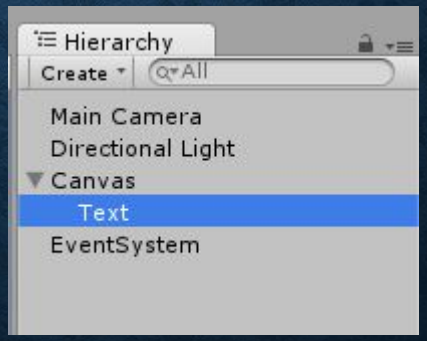
- Coin will be destroy by itself when its life is equal or lower than 0

TIMER

- Every game has their own goal, e.g.
  - Get highest score
  - Solve puzzle
  - Achieve something within limit time
  - Use the minimum time to finish the game…
- Let's introduce two kind of timer
  - Count down
  - Timer

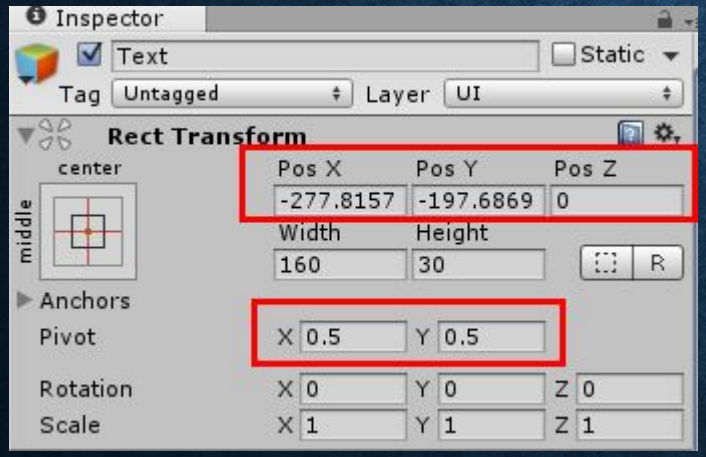
TEXT DISPLAY

- Before creating a timer, we should be able to display basic text on the screen so that we can know how much time is taken or passed
- Create a GUI Text to display the text in 2D level
  - GameObject > UI -> Text
- Canvas will be created automatically, all the UI element must be the child of canvas
- EventSystem will also be created to handle the UI input



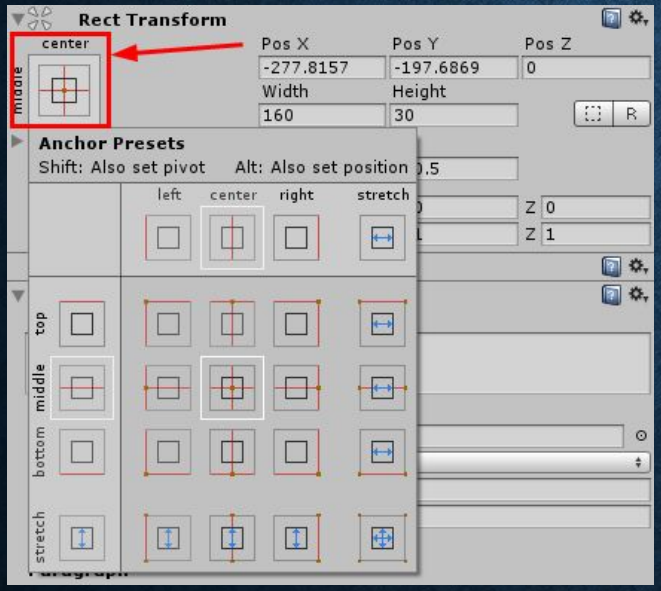
TEXT POSITIONING

- UI element will be placed according to the **Rect Transform position**
- The position is relative to the Pivot
  - Upper left corner is (1,1)
  - Lower right corner is (0,0)



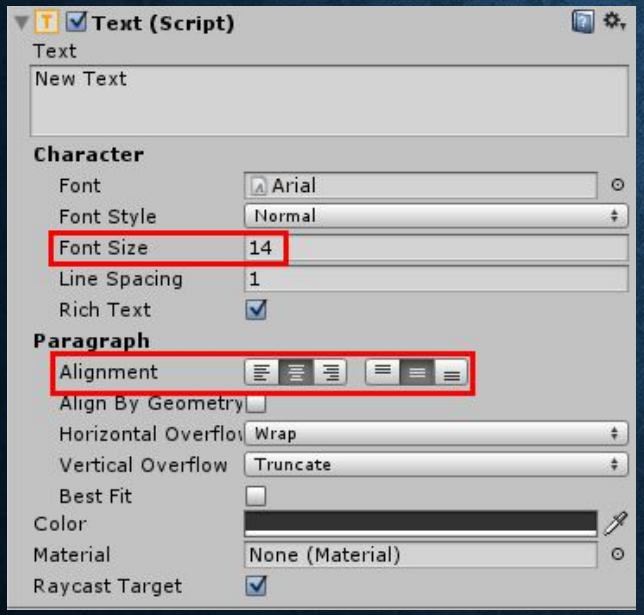
TEXT POSITIONING

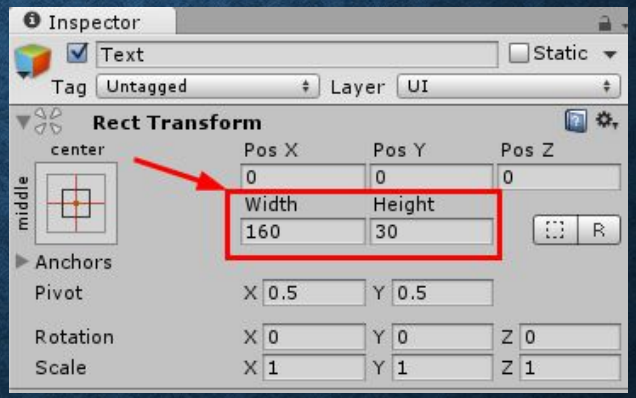
- Use the Anchor Preset to setup the position quickly
- Hold Shift for setting the pivot
- Hold Alt for setting the position but also move your text



TEXT ADJUSTMENT

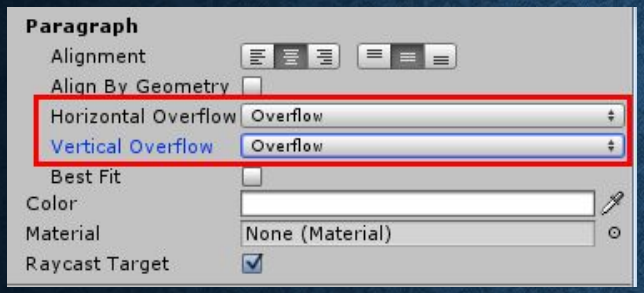
- Adjust the alignment and font size
- Text is bounded by the width and height
- If font size is too large, clipping will occur



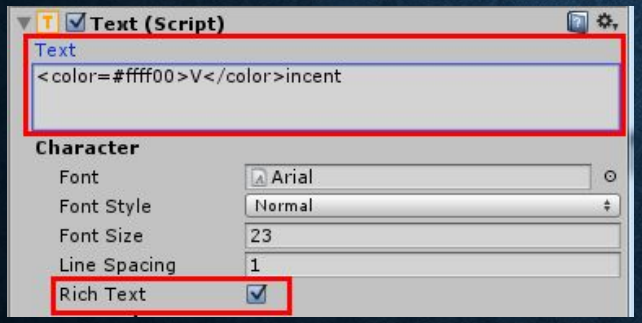


TEXT ADJUSTMENT

- Change the Horizontal / Vertical Overflow to Overflow can prevent text clipping



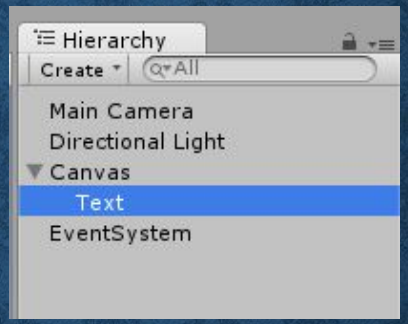
- Rich text allow to use makeup tags for the text field



<color=#ffff00>V</color>incent

TIMER

- Create a GUI Text to display the text in 2D level
  - GameObject > UI -> Text



- Change the name from "Text" to "Timer".
- Change the Text value, Alignment and Overflow configuration on the text

TIMER

- Now create a script with the name "Timer"
- Type in the following script and put it to the text

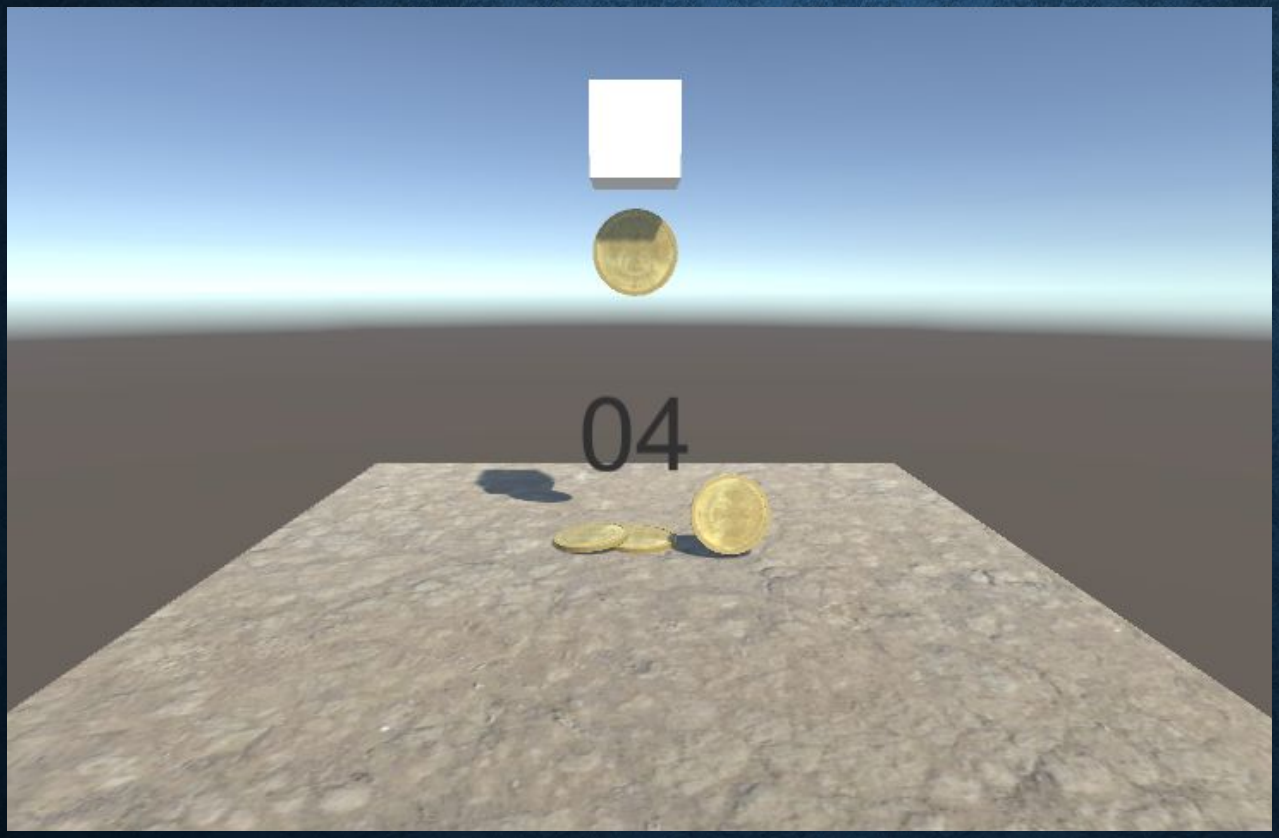
Timer.cs

selection

```
1  using UnityEngine;
2  using System.Collections;
3  using UnityEngine.UI;
4
5  public class Timer : MonoBehaviour {
6
7      public float myTimer=0;
8
9      Text text;
10
11     void Start () {
12         text = GetComponent<Text>();
13     }
14
15     void Update () {
16         myTimer += Time.deltaTime;
17         text.text = Mathf.Floor(myTimer).ToString();
18     }
19 }
```

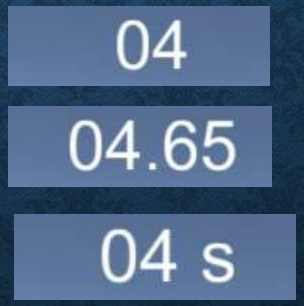
TIMER

- Drag the “Timer” script to the “Timer” and test it.



TIMER

- To handle the uGUI system by script, the UnityEngine.UI name space must be included
- Mathf.Floor is used to truncate the number into integer
- .ToString() is used to convert number into String, it can also used to format the number
  - E.g. .ToString ("00")
  - E.g. .ToString ("00.00")
  - E.g. .ToString ("00 's' ")



COUNT DOWN

- Similar to the basic timer, give a value of the starting time and subtract it by the time passed

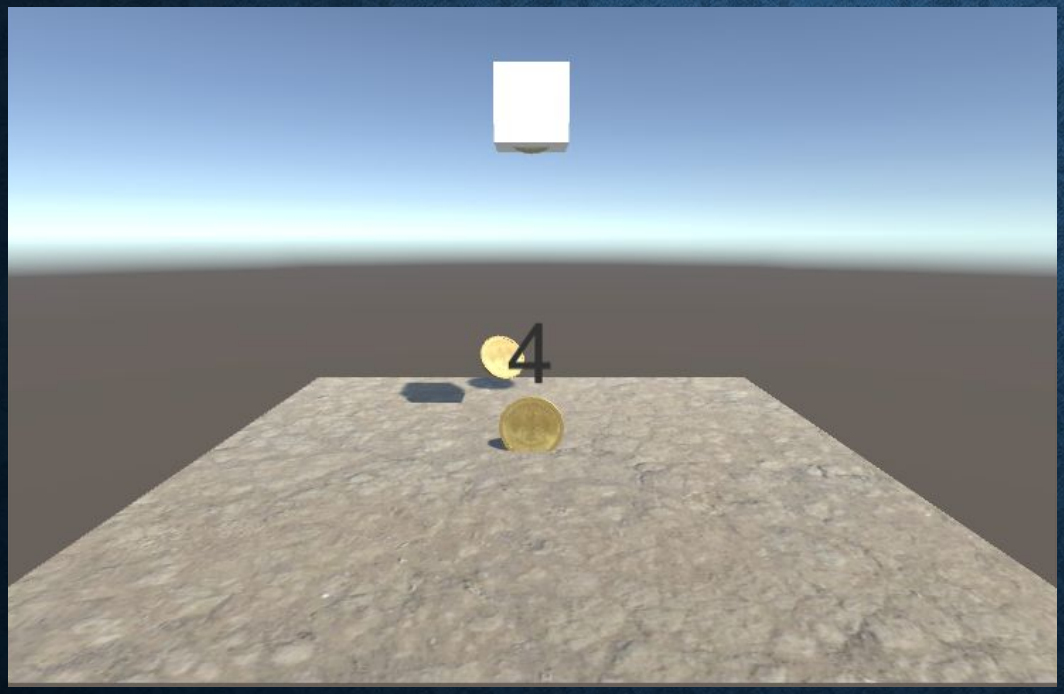
CountDown.cs

selection

```
1   using UnityEngine;
2   using System.Collections;
3   using UnityEngine.SceneManagement;
4   using UnityEngine.UI;
5
6   public class CountDown : MonoBehaviour {
7
8       public float myTimer=5;
9
10      Text text;
11
12      void Start () {
13          text = GetComponent<Text>();
14      }
15
16      void Update () {
17          if (myTimer > 0)
18              myTimer -= Time.deltaTime;
19          text.text = Mathf.Ceil(myTimer).ToString();
20      }
21  }
```

COUNT DOWN

- Drag the “CountDown” script to the “Timer”
- Disable the script “Timer” and test it.

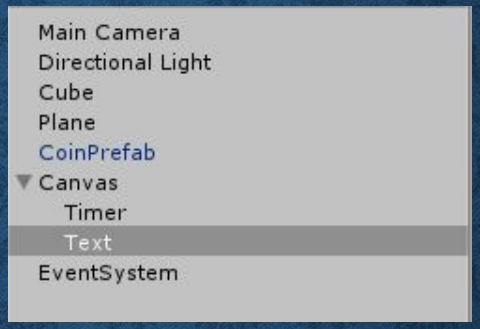


COUNT DOWN

- When the time is up, we can trigger different event, e.g.:
  - Display the game over text
  - Display the result…
- First create the game over text under canvas

COUNT DOWN

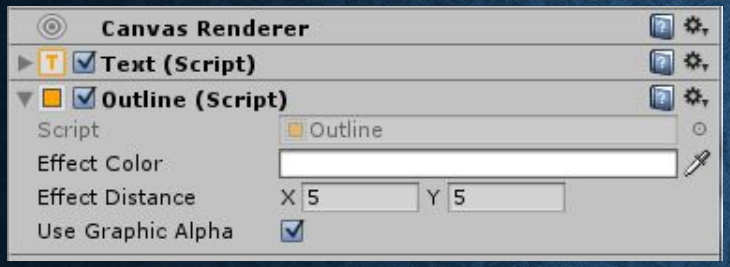
- Create a GUI Text to display the text in 2D level
  - GameObject > UI -> Text



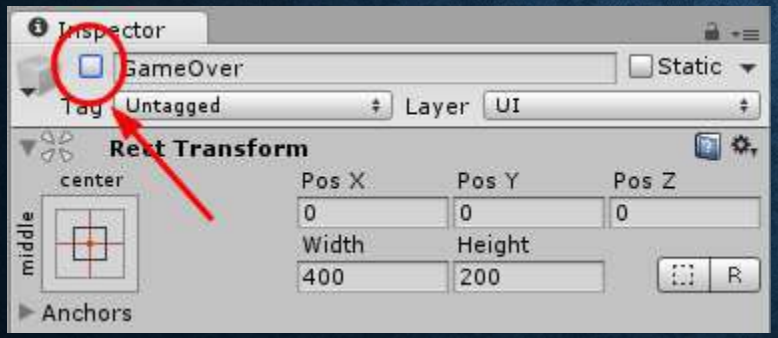
- Change the name from "Text" to "GameOver".
- Change the Text value, Alignment and Overflow configuration on the text.

COUNT DOWN

- Add the Outline text effect (Component -> UI -> Effects -> Outline)



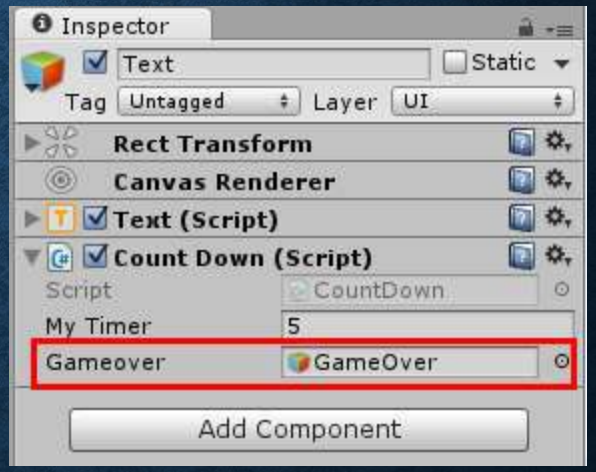
- Disable the game over text



COUNT DOWN

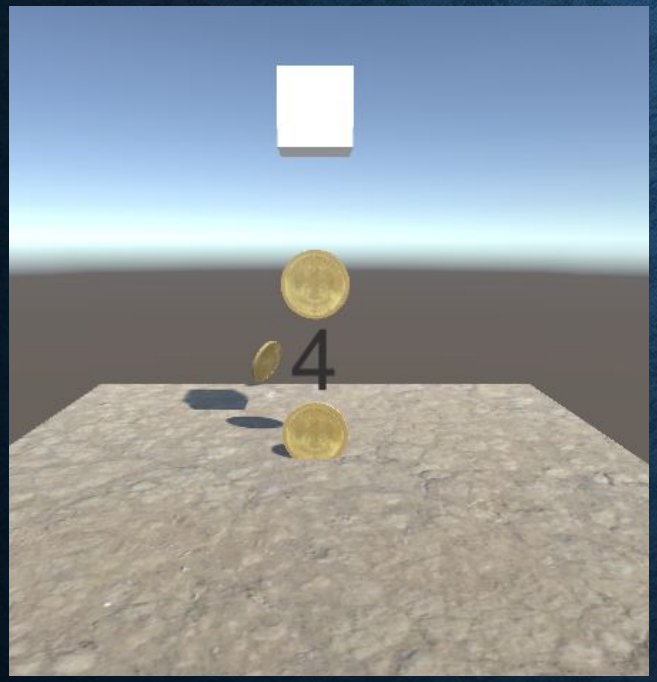
- Modifier the count down script
- Drag the gameover text to the gameover slot

```
6  public class CountDown : MonoBehaviour {
7
8      public float myTimer=5;
9      public GameObject gameover;
10
11     Text text;
12
13     void Start () {
14         text = GetComponent<Text>();
15     }
16
17     void Update () {
18         if (myTimer > 0) {
19             myTimer -= Time.deltaTime;
20             text.text = Mathf.Ceil(myTimer).ToString();
21         }
22         else {
23             gameover.SetActive(true);
24             text.enabled = false;
25         }
26     }
27 }
```



COUNT DOWN

- Test it and the game over text automically displayed after the count down.

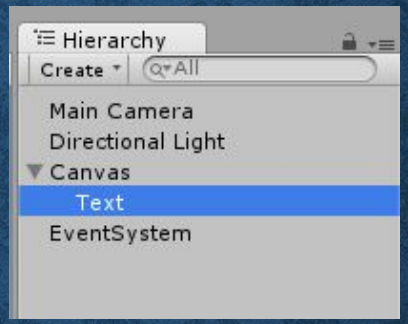


SCORE

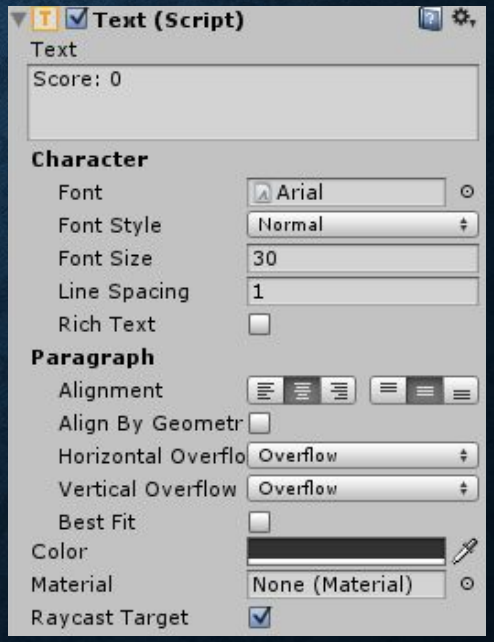
- Open the project of last lesson
- Create the score at the upper left corner

SCORE

- Create a GUI Text to display the text in 2D level
  - GameObject > UI -> Text



- Change the name from "Text" to "Score".
- Change the Text value, Alignment and Overflow configuration on the text



SCORE

- Add the ScoreCount script to the text
- Static is used as the global variable, also carry the value throughout the game

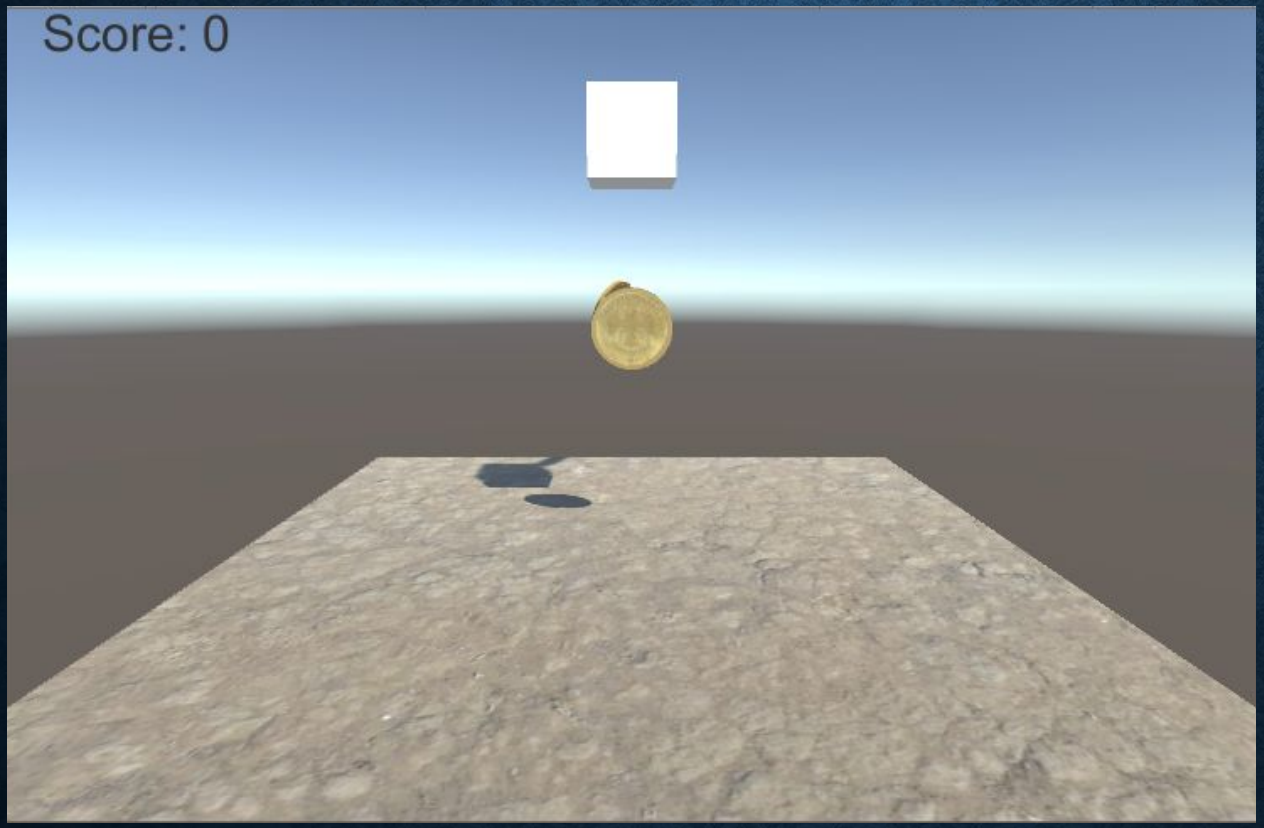
ScoreCount.cs

selection

```
1  using UnityEngine;
2  using System.Collections;
3  using UnityEngine.UI;
4
5  public class ScoreCount : MonoBehaviour {
6
7      public static int score;
8      public static Text scoreText;
9
10     void Start () {
11         scoreText = GetComponent<Text>();
12         UpdateScore (0);
13     }
14
15     public static void UpdateScore(int value) {
16         score += value;
17         scoreText.text= "Score: " + score.ToString();
18     }
19 }
```

SCORE

- Test it and the Score Text displayed.



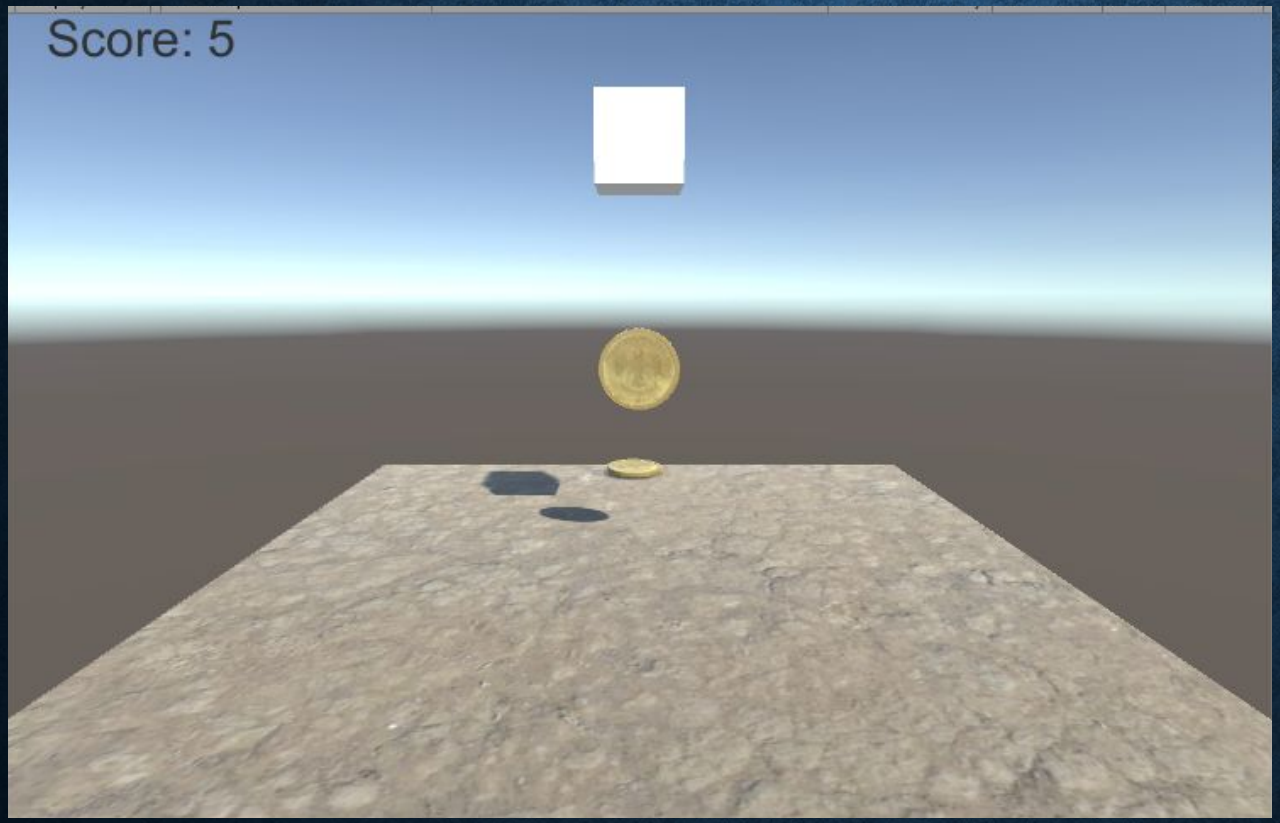
UPDATE SCORE

- Modify the CoinBehaviour script so that score will be gained when the coin is hit

```
17      void Hit (float damage) {
18          life -= damage;
19          ScoreCount.UpdateScore(1);
20          if (life > 0)
21              AS.PlayOneShot(hitSound);
22          else
23              Kill ();
24      }
25
26      void Kill () {
27          AudioSource.PlayClipAtPoint(collectSound, transform.position);
28          ScoreCount.UpdateScore(10);
29          Destroy(gameObject);
30      }
```

UPDATE SCORE

- The score will be gained when the coin is hit



PLAYERPREF

- The score will save to memory only, next time when you play, the value will return to zero
- Use the PlayerPref to save and read data
  - Support Int, float and string
  - Use Get to read from data
  - Use Set to write to data
  - Use DeleteAll or DeleteKey to remove data

PLAYERPREF

- Modified the ScoreCount script to store data

```
5  public class ScoreCount : MonoBehaviour {
6
7      public static int score;
8      public static Text scoreText;
9
10     void Start () {
11         scoreText = GetComponent<Text>();
12         score = PlayerPrefs.GetInt("scoreData",0);
13         UpdateScore (0);
14     }
15
16     public static void UpdateScore(int value) {
17         score += value;
18         PlayerPrefs.SetInt("scoreData", score);
19         PlayerPrefs.Save();
20         scoreText.text= "Score: " + score.ToString();
21     }
22 }
```

PLAYERPREF

- Test it.