

# 02. Assets & UI

Assets, Animation, UI, Level Management, TD2

---

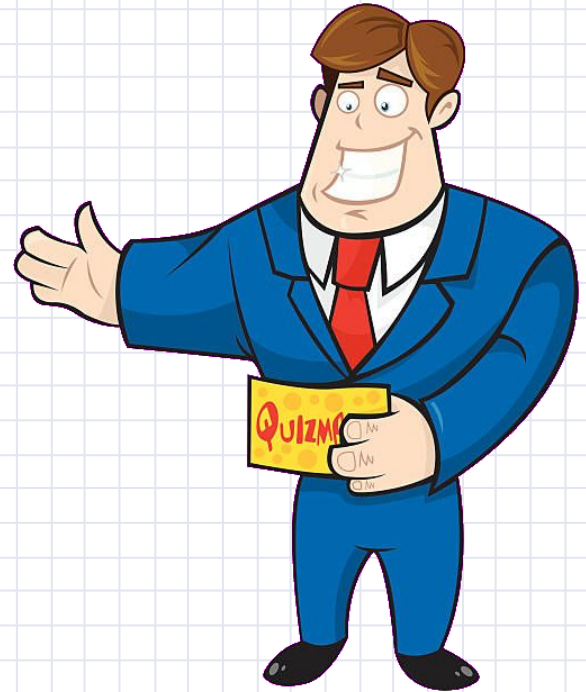
# Homework – PONG

Open your Pong Scene and press play

- Controller /1 point
  - Works properly on arrows and keys
  - The players don't go further than the screen
- Ball /1 point
  - Moves around
  - The ball doesn't go further than the screen
  - Has some kind of physics / bounces on walls and players

# Quizz

- For the next three courses (this one included) you will have a few questions at the beginning of the course
- Each good answer will grant you between 1 and 3 Quizz Point
- For each 5 Quizz Points you get you will have 0.5 bonus point on your Game Engine grade
- Don't shout the answers, raise your hand. If the answer is shout out, the question will be skipped and no points will be awarded.



## Question 0 – Example

What component is used to move, rotate and scale a GameObject ?

0 Quizz Point

## Question 1

**When creating a new C# Script in Unity, what class does the script inherits by default?**

**1 Quizz Point**

## Question 2

**What Component can be used to change the Physics properties of a GameObject? (Change its velocity for example)**

**1 Quizz Point**

## Question 3

What is the difference between the “Update” and the “FixedUpdate” methods?

1 Quizz Point

## Question 4

What Monobehaviour method is called before “Start”?

1 Quizz Point



## Question 5

**What is a coroutine and how do you start it?**

**2 Quizz Points**

## Question 6

**What property of a Transform can you use to change a GameObject rotation?**

**2 Quizz Points**

## Question 7

**What method can be used to access another Component on a GameObject?**

**2 Quizz Points**

## Question 8

**Explain what is object pooling and how to implement it.**

**3 Quizz Points**

# Unity Asset Store

- Go to the Asset Store and log in (you can use a google account)
- Search for Character Pack: Free Sample by Supercyan
- Add it to your assets
- Go back to Unity and open the Package Manager (Window > Package Manager)
- Log in to your Unity account
- Refresh your packages
- Download and Install the package Character Pack: Free Sample



**Character Pack: Free Sample**

by Supercyan

# Importing Assets

- Assets can be imported via your computer using the Assets folder or via the package manager
- Every type of asset has its own import settings
- Create a Folder and name it “Sprites” add the “Circle” image to this folder
- Set the import settings as such
  - Texture Type : Sprite (2D and UI)
  - Max Size : 512
- It's important to set a Max Size to your Sprites and Textures, this will improve performance and reduce the weight of your game.
- Apply your settings

Now let's slice this circle so we can change its shape in the UI

- Go to your package manager and search for “2D Sprite” in Unity Registries
- Download and Install the package
- Go back to your “Circle” **Sprite** import settings
- Click on “Sprite Editor” and set all your borders to 256
- Apply your borders



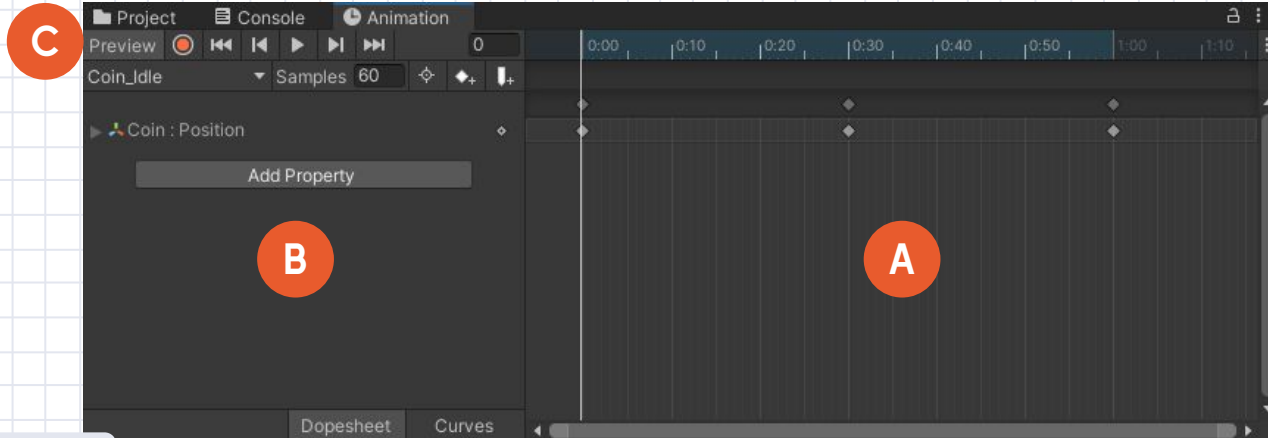
# Animation

**Animation** : A succession of keyframes (state of an object) are place on a timeline, frames in between are interpolated

**A - Timeline** : Animation keyframes are placed along a timeline

**B - Animated properties list** : Every property of every component can be changed using animation, here is the list of changed properties

**C - Toolbar** : The red dot is used to record an animation



# Animator

Let's animate a character!

- The Unity **Animator** is a state machine that handles animation transitions and parameters (trigger, integer, float, boolean)
- Create a new folder named "Animation" in your Assets
- Inside this folder create another folder named "Character"
- Inside the "Character" folder, right click and create a new **Animator Controller**, name it "Character"
- Create a new **Scene** with a plane, name it "Coin Runner"
- Add a **Texture** or a **Material** to the plane
- Create an empty **GameObject** at the origin of the scene and name it "Character"
- Go to Supercyan Character Free Sample > Prefabs > Base > High Quality
- Drag and drop the MaleFree1 **Prefab** as a child of your "Character" **GameObject** and reset its position
- In the MaleFree1 **GameObject**, go to the **Animator Component** and set its Controller to "Character"
- Open the **Animator** window (Window > Animation > Animator)

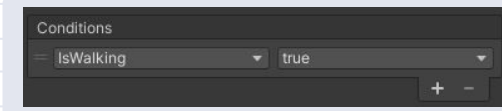


# Animator

- Right Click inside the Animator Window and create two empty states
- Rename the first one “Idle” and the second one “Walk”
- In the Idle state, set its motion to the “idle” animation
- In the Walk state, set its motion to the “walk” animation



- In the Animator Window, go to Parameters (top left) and add a boolean parameter named “IsWalking”
- Right click on your Idle state and create a transition to the Walk state
- In the transition condition set IsWalking : true
- Right click on your Walk state and create a transition to the Idle state
- In the transition condition set IsWalking : false



# Animator

- Place your camera **Transform** as such
  - Position (0, 0.5, 2)
  - Rotation (0, 180, 0)
- Press Play!



T-Pose



You can go in your **Animator** and change the state of animation by changing the value of the “IsWalking” boolean



# TD 2



## Coin Runner

Run and get as many coins as possible in a limited amount of time!

# Character Controller

Let's control the character by script!

- Place your camera **Transform** as such
  - Position (0, 10, 0)
  - Rotation (90, 0, 0)
- Create a new C# Script and name it "CharacterController"

## 1. Parameters

```
[SerializeField] private float speed = 3;

private float vertical;
private float horizontal;
private Vector3 currentPosition;
private Vector3 targetPosition;
```

## 2. Update

```
private void Update()
{
    vertical = Input.GetAxis("Vertical");
    horizontal = Input.GetAxis("Horizontal");
    currentPosition = transform.position;
    targetPosition = currentPosition + new Vector3(horizontal, 0, vertical);
    transform.LookAt(targetPosition);
    transform.position = Vector3.MoveTowards(currentPosition, targetPosition, Time.deltaTime * speed);
}
```

- Add the **Character Controller** Component to your Character and play

# Character Controller

Let's animate the character by script!

## 1. Parameters

```
[SerializeField] private Animator characterAnimator;  
  
private static readonly int IsWalking = Animator.StringToHash("IsWalking");
```

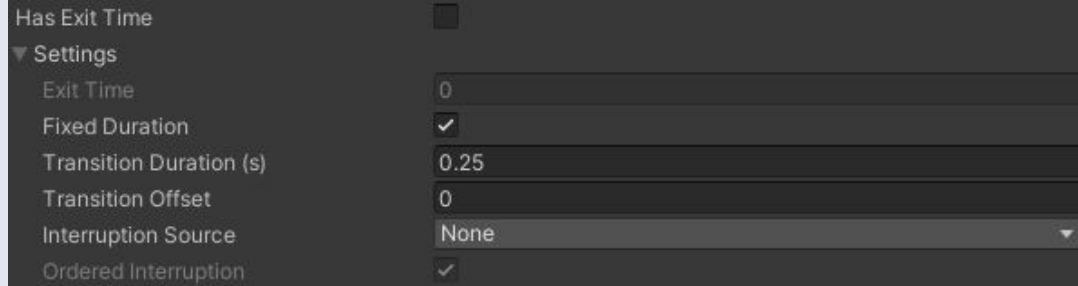
## 2. Update

```
private void Update()  
{  
    vertical = Input.GetAxis("Vertical");  
    horizontal = Input.GetAxis("Horizontal");  
    characterAnimator.SetBool(IsWalking, vertical != 0 || horizontal != 0);  
    currentPosition = transform.position;  
    targetPosition = currentPosition + new Vector3(horizontal, 0, vertical);  
    transform.LookAt(targetPosition);  
    transform.position = Vector3.MoveTowards(currentPosition, targetPosition, Time.deltaTime * speed);  
}
```

- Fill the “Character Animator” field in your **Character Controller** in the **Scene**
- Play!

# Animator Transitions

- Go to your Animator and set both transitions as such



The screenshot shows the settings for an Animator transition. The 'Has Exit Time' checkbox is unchecked. Under the 'Settings' section, 'Exit Time' is set to 0, 'Fixed Duration' is checked, 'Transition Duration (s)' is set to 0.25, 'Transition Offset' is set to 0, 'Interruption Source' is set to 'None' (via a dropdown menu), and 'Ordered Interruption' is checked.

Has Exit Time	<input type="checkbox"/>
▼ Settings	
Exit Time	0
Fixed Duration	<input checked="" type="checkbox"/>
Transition Duration (s)	0.25
Transition Offset	0
Interruption Source	None ▼
Ordered Interruption	<input checked="" type="checkbox"/>

**Has Exit Time** : If set to true, will wait until the animation is over to go to the next one

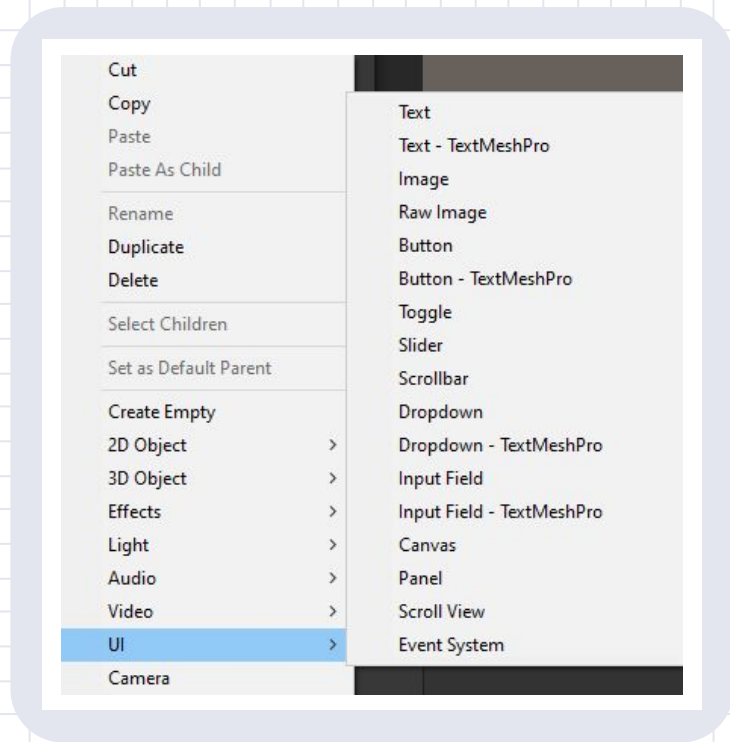
**Transition Duration** : Time (in seconds) to play the transition, set it to zero to play no transition

- Press Play!

# Inputs

- Your ZQSD keys might not work properly, that's because Unity is set to QWERTY keyboards by default
- To change these settings got to Edit > Project Settings > Input Manager
  - In "Horizontal" change the Alt Negative button to "q"
  - In "Vertical" change the Alt Positive button to "z"
- You can also create your own types of Inputs by changing the size of the list
- You can have duplicates in Input names if you want compatibility with other types of controllers

# User Interface



- You can access most UI Elements by Right clicking in the Hierarchy
- UI Elements must be placed inside a Canvas
- The EventSystem allows you to interact with the interface, it's created by default when you create a new Canvas



# Canvas

- A default canvas has 3 components
  - Canvas
    - Overlay : Linked to the screen
    - Camera : Linked to the camera
    - World Space : Placed as a 3D Object in the world
  - Canvas Scaler
    - Used to scale the canvas depending on the screen
    - Recommended settings :
      - Scale With Screen Size
      - Reference Resolution X : 1600 Y : 900
      - Match Width or Height
      - Match : 0
  - Graphic Raycaster
    - Used to get inputs from the user
- Create a canvas with these settings

# List of UI Elements

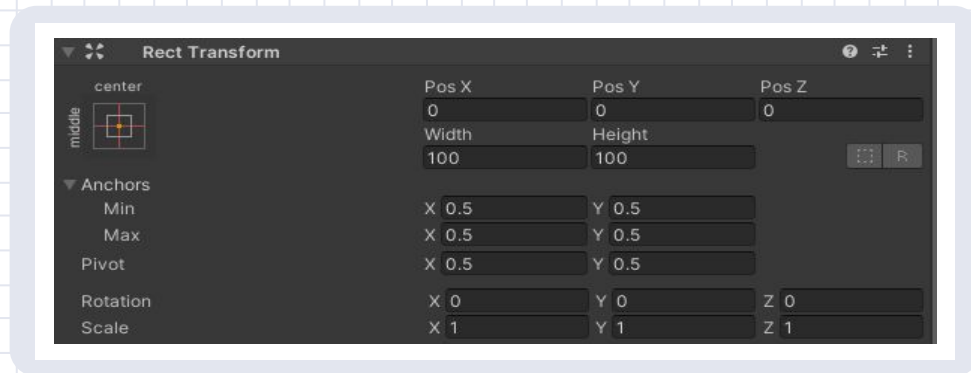
- **Text** : Display texts in the UI, I highly recommend using the **TextMeshPro** variant
- **Image** : Display images, can hold sprites, different modes (Simple, Sliced, Filled, Tiled)
- **RawImage** : Display images, can hold textures
- **Button** : Allows you to trigger methods from an input (**TextMeshPro** recommended)
- **Toggle** : ON / OFF buttons, can trigger methods on interaction
- **Slider** : Display gauges, can trigger methods on interaction
- **DropDown** : Displays a list of options, can trigger methods on interaction
- **InputField** : Allows the player to enter text or numbers, can trigger methods on interaction
- Many more that you will have to discover yourself...

Before going further, go in your package manager and Install the latest version of **TextMeshPro** from Unity Registry (**TextMeshPro** 3.0.6) Import only **TMP Essentials**.

**TextMeshPro** is basically a better version of some UI elements Unity already provides.

# RectTransform

- All UI Elements have a **RectTransform Component**, this class inherits from the **Transform** class
- Pos X / Pos Y / Pos Z : Position of the **GameObject** on the **Canvas**
- Width / Height : Size of the **RectTransform**
- Anchors : The **GameObject** is attached to its parent by the Anchors, Unity provides Anchors presets
- Pivot : From which point the **GameObject** is supposed to rotate (center by default)
- Rotation : same as **Transform Component**
- Scale : same as **Transform Component**



# Game Menu – Main Menu

Coin Runner

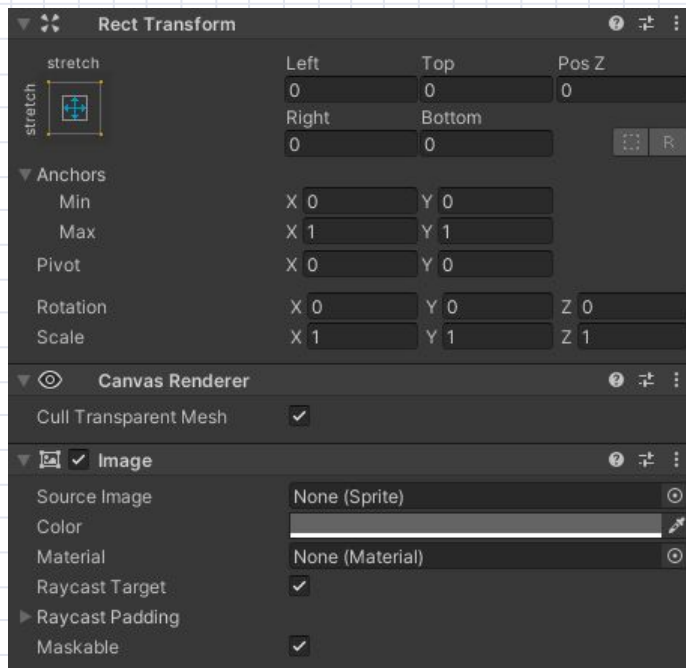
Level 1

Level 2

Level 3

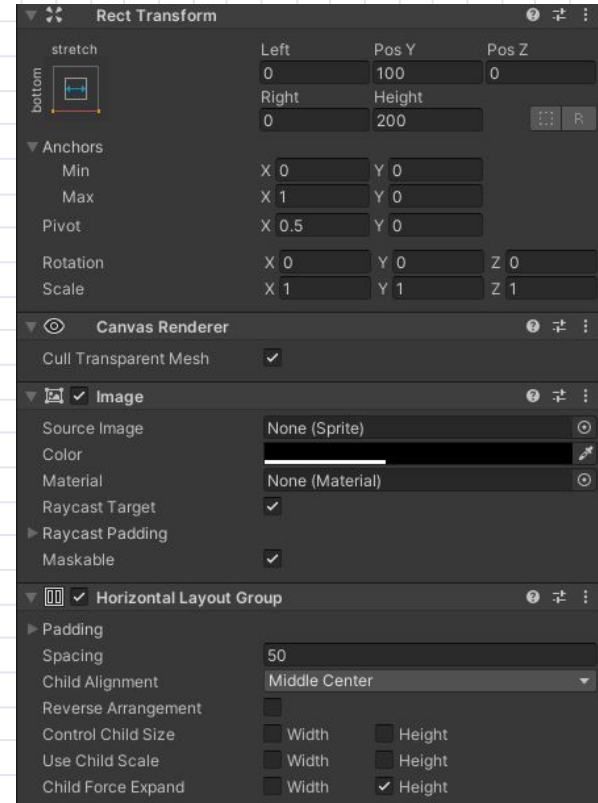
# Game Menu – Main Menu

- Inside your **Canvas**, create an **Image** and name it “Background”, set it like so :



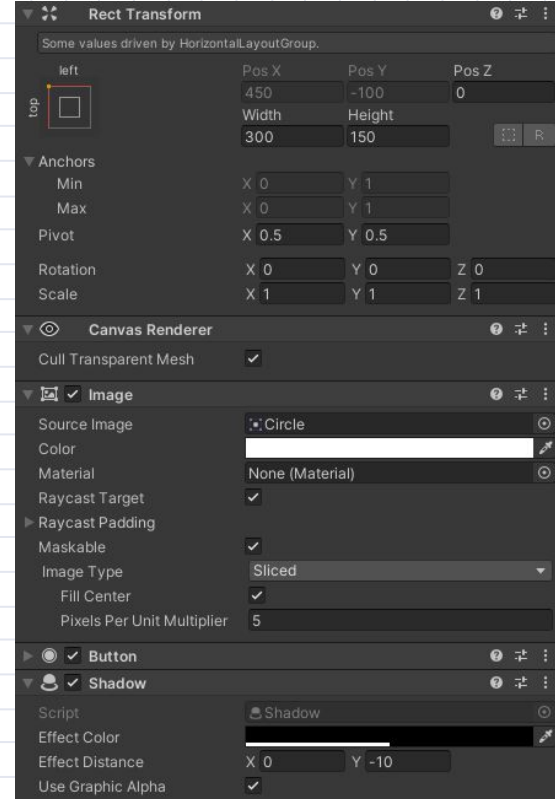
# Game Menu – Main Menu

- Inside your **Canvas**, create an other Image, name it “Buttons” and add the **Horizontal Layout Group Component** to it and set it like so :
- The **Horizontal Layout Group** allows you automatically place children GameObjects properly by given rules (Spacing, Padding, Child Alignment...)
- An **Image Component** without a **Sprite** (source image) will fill the rect with a **Color**
- An **Image Color** can be transparent



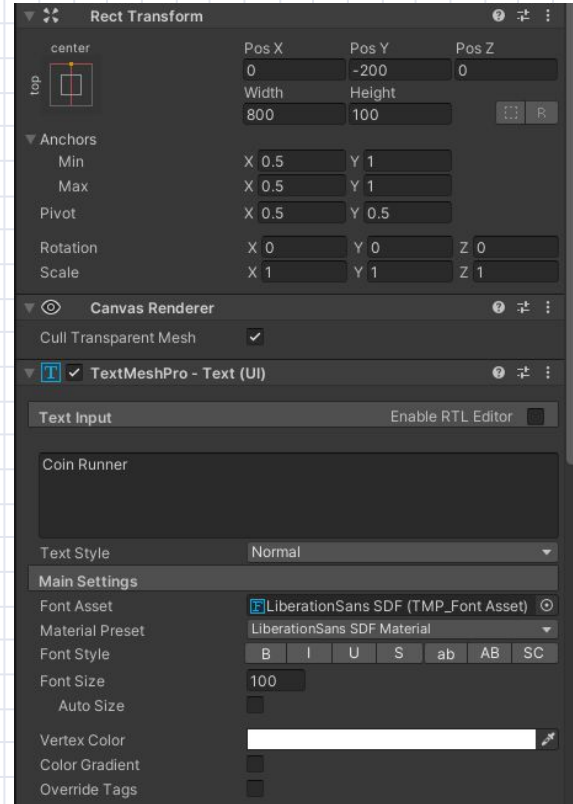
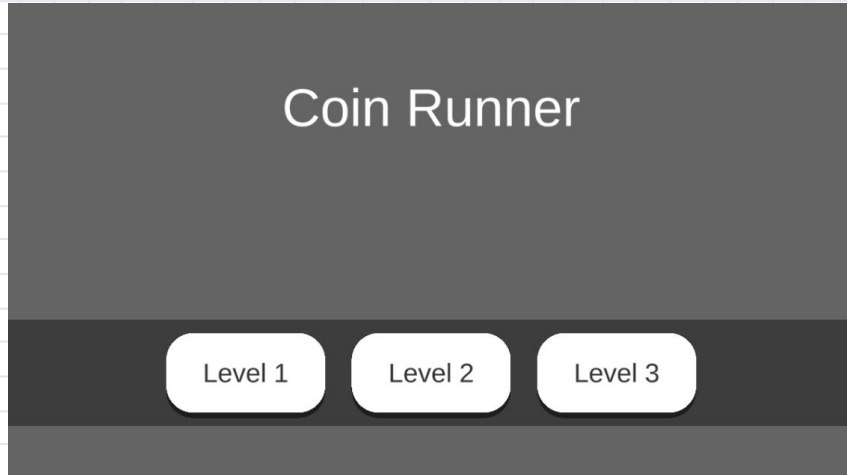
# Game Menu – Main Menu

- Inside the **GameObject** “Buttons” create a “Button – TextMeshPro” **GameObject**, name it “Level (1)” add the “Shadow” **Component** to this **GameObject** and set it like so :
- The “Sliced” image type uses the slicing we did on the circle image to make round borders
- Go in the “Text (TMP)” child and change the text the “Level 1” and the Font Size to 50
- Copy / Paste the “Level (1)” **GameObject** two times
- Change the new objects texts to “Level 2” and “Level 3”



# Game Menu – Main Menu

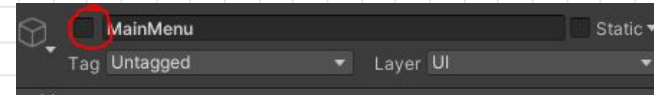
- Create a new **GameObject** from “Text – TextMeshPro” as a child of your **Canvas**, name it “Title” and set it like so :
- You now have a Level Selection UI!
- Rename your **Canvas** “MainMenu”





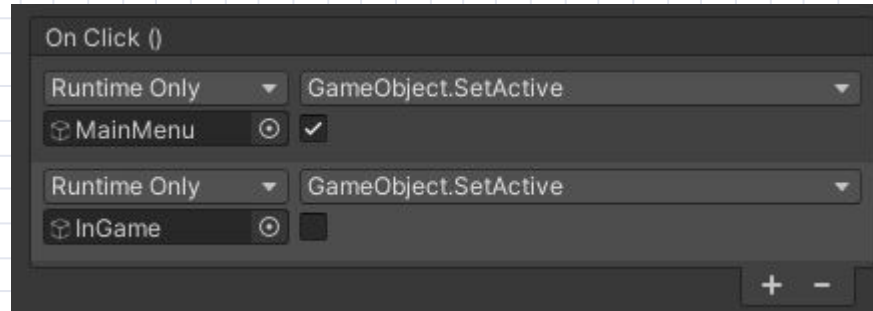
# Game Menu – In Game

- Create a new Canvas, name it “In Game” and create a “Back” button inside, place it anywhere you want.
- You can disable the “MainMenu” Canvas for the moment



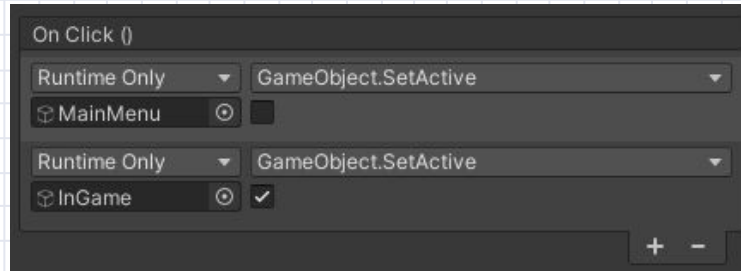
# Game Menu – In Game

- Go inside your **Button Component** and add an “On Click” listener
- Drag and drop your “MainMenu” **GameObject** and select the method “GameObject.SetActive”, tick the box
- Add another listener
- Drag and drop your “InGame” **GameObject** and select the method “GameObject.SetActive”, don't tick the box
- Now, when clicking this button, your main menu should be enabled and your InGame menu be disabled



# Game Menu – Main Menu

- Disable your “InGame” GameObject, enable your “MainMenu” GameObject
- Inside your levels buttons add these listeners

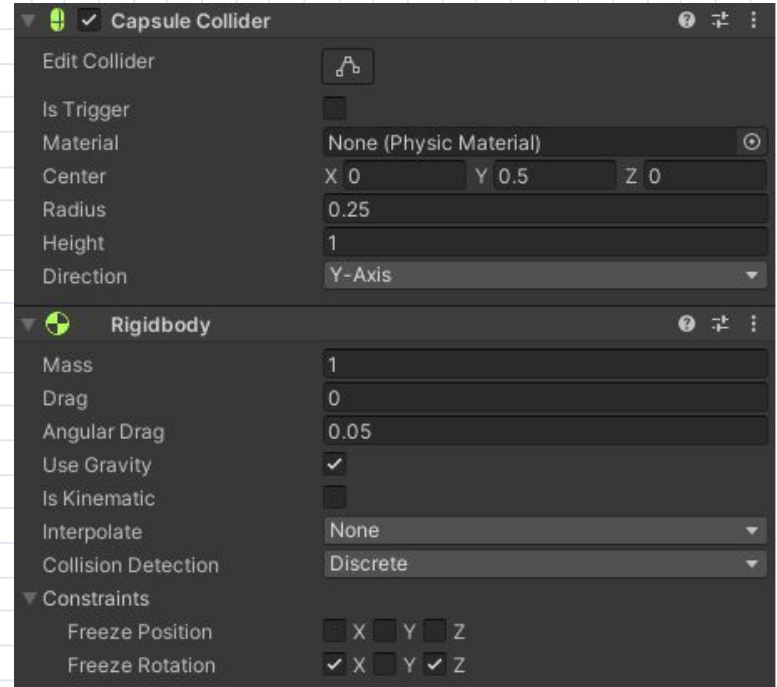


Press Play and test your buttons

# Character Controller

Let's make our character collision with physics objects

- Go to your Character GameObject and add a Capsule Collider and a Rigidbody with these settings :
- You can place a cube on the scene, your character will collide with it
- You might see that the character flickers on collision
- Collision are physics based, it's computed every FixedUpdate instead of every Update
- Go inside your Character Controller script and Replace "Update" with "FixedUpdate"



# Coin

Let's make a collectible Coin

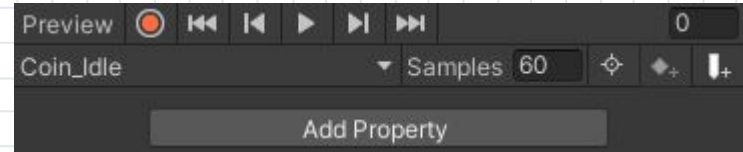
- In the **Scene** create an empty **GameObject** and name it "Coin"
- In this **GameObject** create a Sphere and set its Scale to (0.3, 0.3, 0.3), name it "Sphere"
- Remove the **Sphere Collider** of this object and place it on the "Coin" parent instead
- Create a new "Coin" **Material** in your Assets and assign it to the coin
- Drag and drop the coin into your "Prefabs" folder

Now let's Animate our Coin

- Go to your "Animation" folder and create a new "Coin" folder
- Create an **Animator Controller** and name it "Coin"
- Go into your "Coin" **Prefab** and add the **Component** "Animator" to its child ("Sphere")
- Set your "Coin" **Animator Controller** in the "Controller" field
- Open the **Animator** Window (Window > Animation > Animator)
- Open the **Animation** Window (Window > Animation > Animation)
- In the **Animation** Window click "Create" to create a new **Animation** and name it "Coin\_Idle"

# Coin

- Click on the red button in your **Animation** Window
- Set your Transform y position to 0.4
- Go to frame number 30 (0:30)
- Set your Transform y position to 0.6
- Got to frame number 60 (0:60)
- Set your Transform y position to 0.4
- Click again on the red button to finish recording your **Animation**
- If you set your **Animator Controller** correctly, the **Animator** should use your newly created **Animation** as the default one



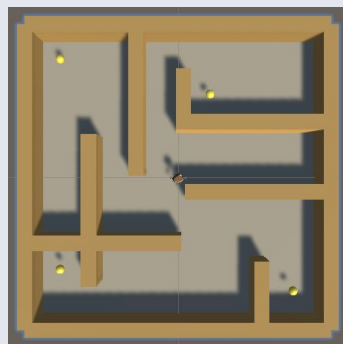
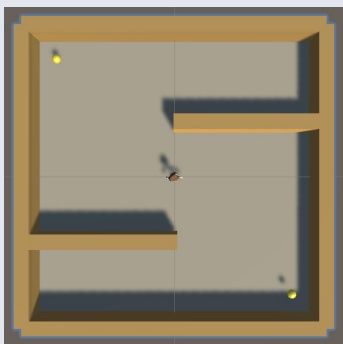
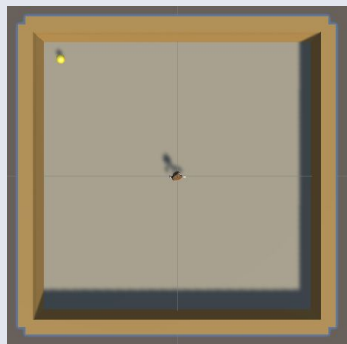
# Levels

Create 3 simple levels

- Each level has a plane and four walls
- “Level1” has no additional walls, and one coin in a corner
- “Level2” has some additional walls and two coins
- “Level3” is a tiny maze and has three to five coins

Save these levels as Prefabs and delete them from your scene

Examples :



# Scriptable Object

- **Scriptable Objects** allows you to store data in your assets
- They are useful to keep global references in the project (it can be shared between scenes)
- **Scriptable Objects** can be modified at runtime
- ⚠ In Unity Editor, modified **Scriptable Objects** during runtime will keep their changes after runtime, this will not occur in build
- Most of the time, **Scriptable Objects** are used to contain data but they can also contain logic



# Level

- Create a new C# script named "Level"

```
[CreateAssetMenu(fileName = "Level", menuName = "ScriptableObjects/Level", order = 1)]
public class Level : ScriptableObject
{
    [SerializeField] private int coinNumber;
    public int CoinNumber => coinNumber;

    [SerializeField] private Vector3 startPosition;
    public Vector3 StartPosition => startPosition;

    [SerializeField] private GameObject levelPrefab;
    public GameObject LevelPrefab => levelPrefab;
}
```

- Create one **Scriptable Object** per level inside a "Data" Folder and fill their fields

# Level Manager

```
[SerializeField] private List<Level> levels = new
List<Level>();
[SerializeField] private Transform levelParent;
[SerializeField] private Transform playerTransform;

private Level GetLevelByName(string levelName)
{
    Level result = levels.Find(level => level.name == levelName);
    if (result) return result;
    throw new ArgumentException("Couldn't find any level named " +
levelName);
}

private void DestroyLevels()
{
    int childCount = levelParent.childCount;
    for (int i = 0; i < childCount; i++)
    {
        Destroy(levelParent.GetChild(i).gameObject);
    }
}
```

# Level Manager

```
private void SetLevel(Level level)
{
    Instantiate(level.LevelPrefab, levelParent);
    playerTransform.position = level.StartPosition;
}

public void SetRandomLevel()
{
    if (levels.Count == 0) return;
    Level level = levels[Random.Range(0, levels.Count)];
    SetLevel(level);
}
```

- Create a new **GameObject** in the **Scene** and add the **LevelManager** component to it
- Fill the **LevelManager** fields
- In your level buttons add a reference to the **LevelManager** and select the method "SetLevel()"

# Coin Counter

- Let's make a counter for our coins

```
[SerializeField] private TextMeshProUGUI counterText;  
private int _currentValue = 0;  
private int _maxValue = 1;  
  
public void SetCounter(int max)  
{  
    _currentValue = 0;  
    _maxValue = max;  
    SetCounterText();  
}  
  
private void SetCounterText()  
{  
    counterText.text = _currentValue.ToString() + "/" + _maxValue.ToString();  
}  
  
public void AddValue(int valueToAdd)  
{  
    _currentValue += valueToAdd;  
    SetCounterText();  
}
```

# Coin

- Let's make our coin give us points

```
[SerializeField] private int coinValue = 1;

private Counter _counter;

private void Start()
{
    _counter = FindObjectOfType<Counter>();
}

private void OnCollisionEnter(Collision other)
{
    if (other.gameObject.CompareTag("Player"))
    {
        _counter.AddValue(coinValue);
        Destroy(gameObject);
    }
}
```

- Add this script to your coin Prefab
- Add a **Text** (Text Mesh Pro) in your UI to contain the counter
- Add the **Counter Component** to this **GameObject**

# Level Manager

- Set the player tag to “Player”
- Modifying the Level Manager to set our counter

```
[SerializeField] private Counter counter;  
  
...  
  
private void SetLevel(Level level)  
{  
    Instantiate(level.LevelPrefab, levelParent);  
    playerTransform.position = level.StartPosition;  
    counter.SetCounter(level.CoinNumber);  
}
```

- Set the reference of **Counter** to the **LevelManager**
- Play !

# TD 2 – Bonus Topics

If the time allows it we can go through some of these topics

Scripting C#

## Improve movement

Move in 3D instead of 2D, Run and Walk

## LD Improvement

Add doors and switches.



## Timer

Add a timer per level and save the best time



Unity

## Animation

UI Counter Animation, Door animation, Victory animation

## Procedural Generation

Generate levels procedurally

## Level Manager +

Save and display progression (Best time, completed or not)



# Homework – Coin Runner

- Coin Runner becomes an Infiltration game!
- Add enemies to the game
  - An enemy should be distinguishable from the player – 1 point
  - An enemy should be able to move (following a path for example) – 1 point
  - An enemy should be able to “spot” the player (when it happens, the player loses) – 1 point
- You should implement enemies in at least three levels
- The game should be playable at least in Unity Editor for next course
- Here is a list of bonus tasks if you want to dive deeper
  - Making a visual vision cone on the enemy to know where it's safe to go
  - Making different types of enemies (by changing their aspect, speed, vision range...)
  - Adding extra levels
  - Decorating levels using free Assets (You can find some in the asset store)
  - Changing the character controller to be first person instead or third person