Flappy Bird

Sebastian Grygorczuk - STEM Institute at CCNY

Assets

Sprite Sheet

2 Music Track

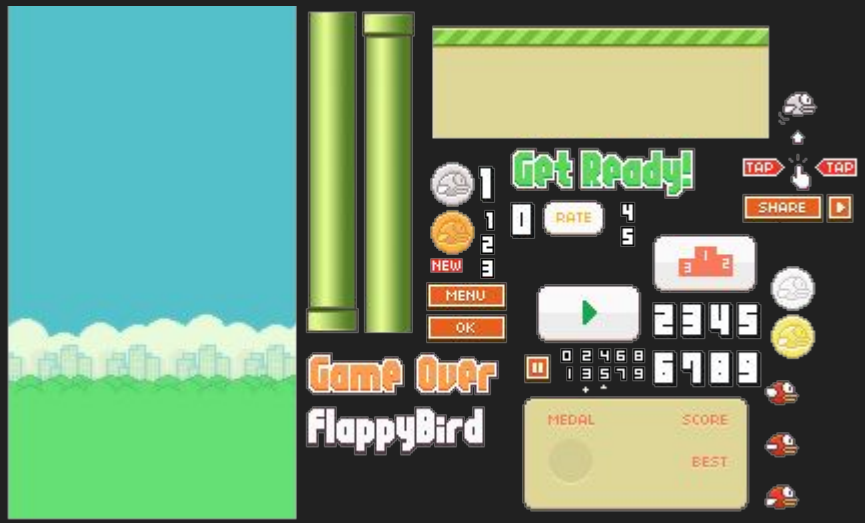
- Main Menu
- Level

3 SFX

- Button Click
- Point Gained
- Death

1 Font

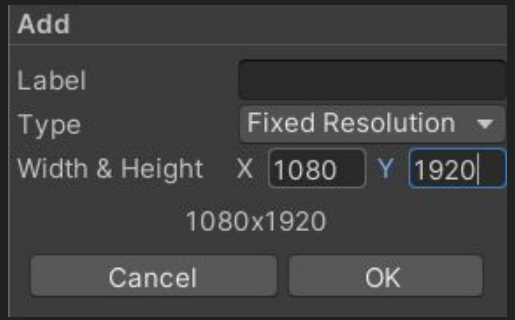
- Button & Score Font



Screen Resolution

Before we start making the game we have to make sure our screen fits the part.

In the Game View we can set the resolution to be 1080x1920 which is a standard phone aspect ratio.



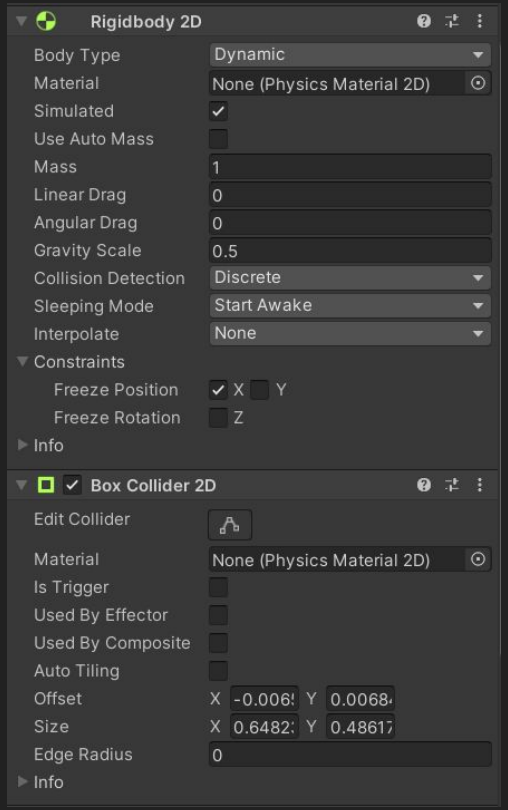
Starting Off With the Bird

Create a parent object that will hold another object that will have the sprite component.

We want the bird to fall down on its own and we want it to stay in place while the game is moving. We will add the Rigidbody2D component and freeze the X position only having them move up and down.

We also want to add a collider to the bird so it can interact with pipes and ground.

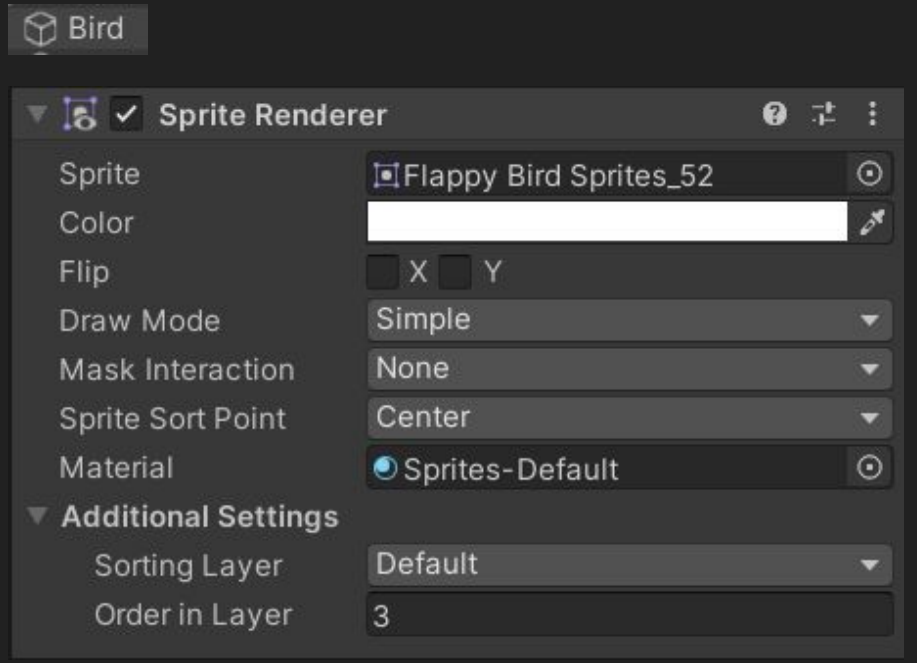
We'll set the gravity to be 0.5f so that it's easier for the bird to pass through the pipes.



Starting Off With the Bird

The child of the parent will have a sprite from the sprite sheet connected to it.

We also want to make the layer order higher so that it can be on top the Pipes, ground and background.



Building the World

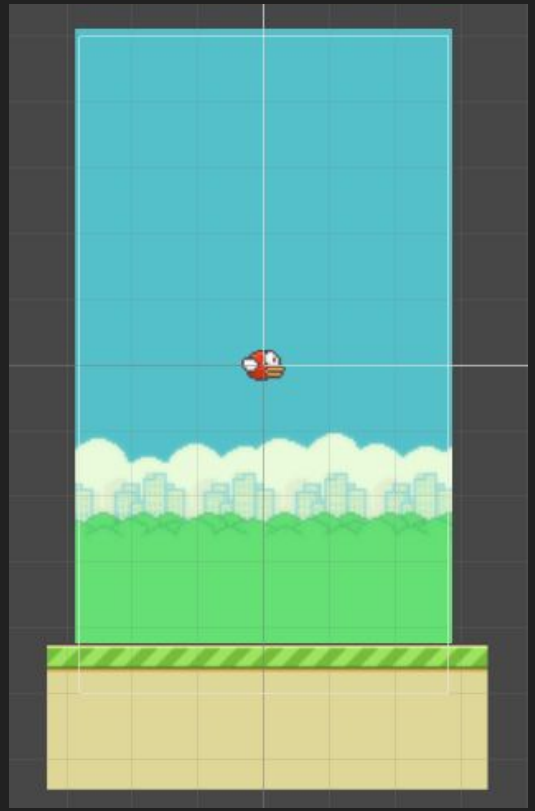
The bird now falls but we need something to catch it.

We will create a general Background Game Object that will house three things in it.

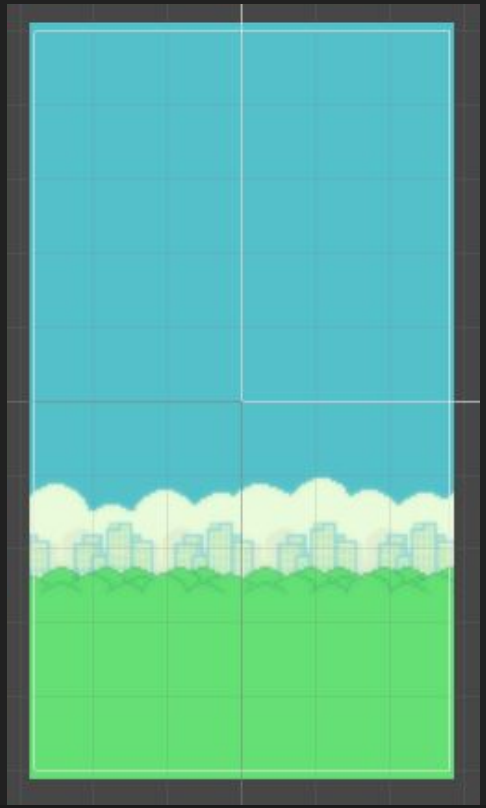
Background image, that will only have a Sprite Render,

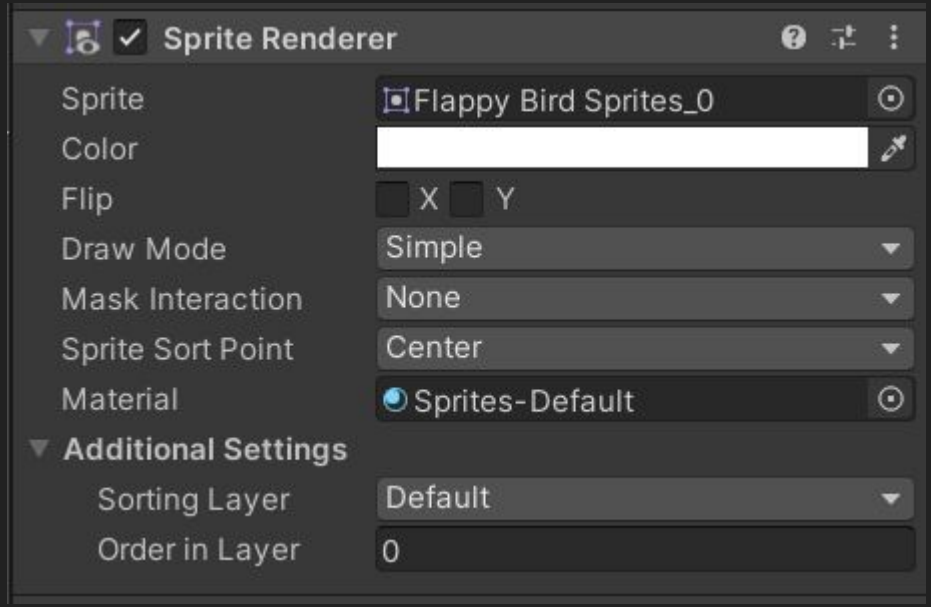
The Ground which will have a Sprite Renderer and a BoxCollider2D so that the Bird can fall on it.

The Sky Collision which will not have a sprite but will have a BoxCollider2D so that the player can’t fly off the screen.

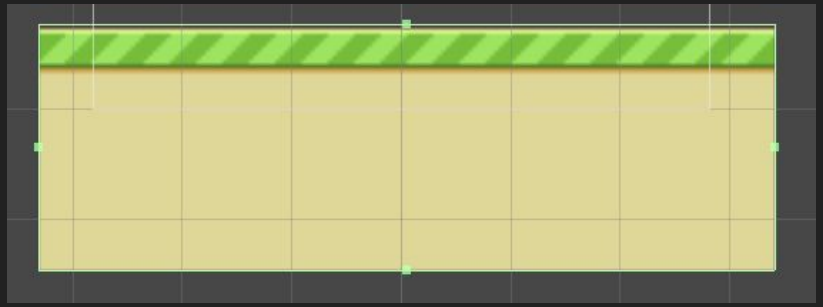


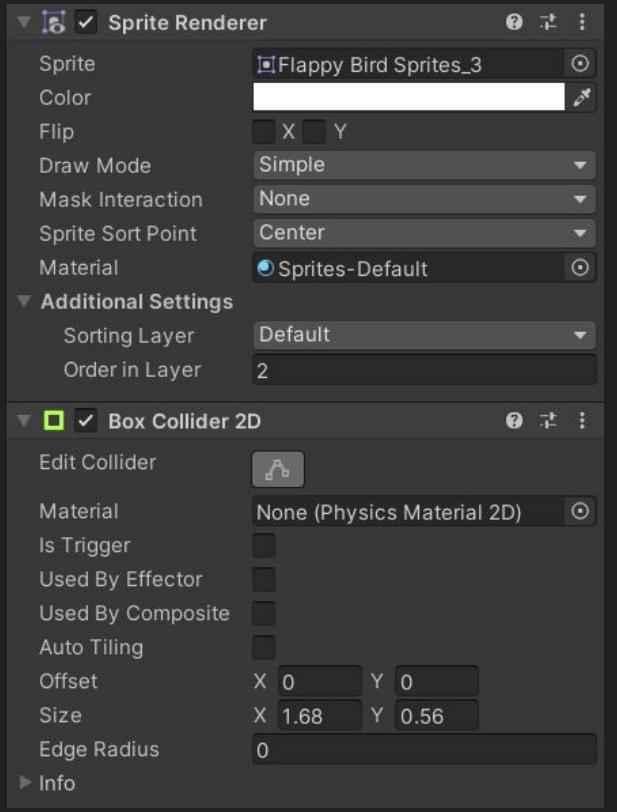
Background



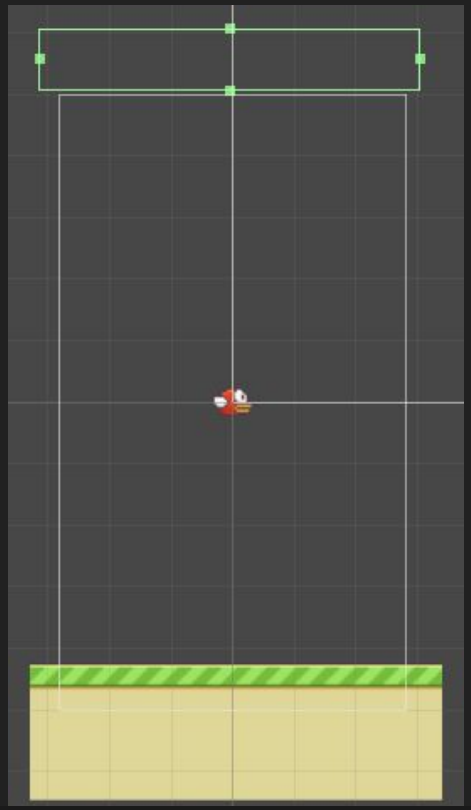

Sprite Renderer
Sprite
Flappy Bird Sprites_0
Color
Flip
X
Y
Draw Mode
Simple
Mask Interaction
None
Sprite Sort Point
Center
Material
Sprites-Default
Additional Settings
Sorting Layer
Default
Order in Layer
0

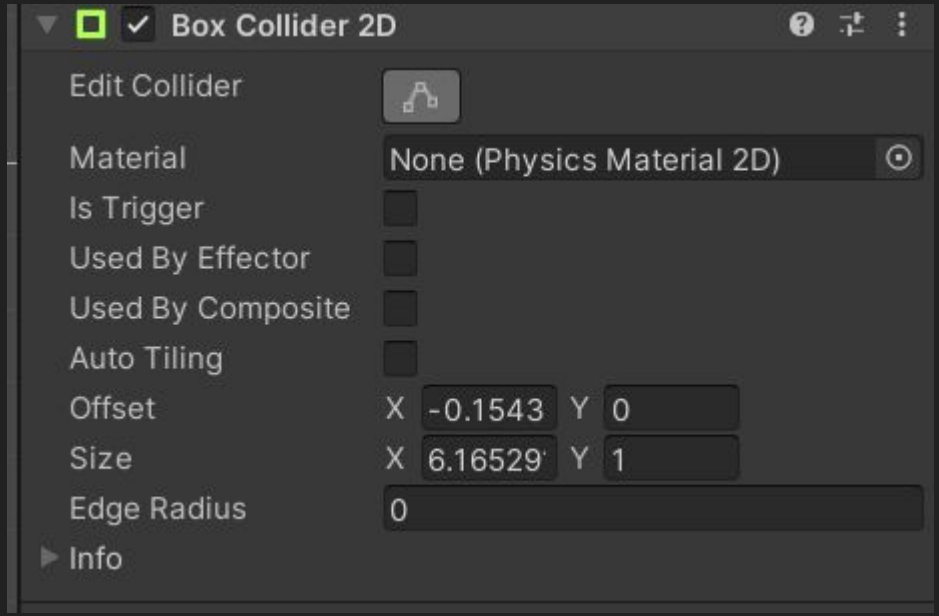
Ground




Sprite Renderer
Sprite
Flappy Bird Sprites_3
Color
Flip
X
Y
Draw Mode
Simple
Mask Interaction
None
Sprite Sort Point
Center
Material
Sprites-Default
Additional Settings
Sorting Layer
Default
Order in Layer
2
Box Collider 2D
Edit Collider
Material
None (Physics Material 2D)
Is Trigger
Used By Effector
Used By Composite
Auto Tiling
Offset
X 0
Y 0
Size
X 1.68
Y 0.56
Edge Radius
0
Info

Sky Collison





Scripting the Bird - Bird Script

We want to start off with the easiest bit having the bird move up.

We will grab the RigidBody2D that's connected to the parent and use GetComponent to connect it in the start function.

We will also create a public variable to we can adjust the strength of the jump up in the Inspector.

Lastly we will create a function that will listen to the click of the Left Mouse Button and send the Bird Up.

```
private Rigidbody2D rigidbody2D;
public float forceScale = 10f;

void Start()
rigidbody2D = GetComponent<Rigidbody2D>();
```

```
//Listens for the Player to click mouse to send Bird flying
0 references
public void Update()
{
    if (Input.GetKeyDown(KeyCode.Mouse0))
    {
        rigidbody2D.velocity = forceScale * Vector2.up;
    }
}
```

Scripting Pipe

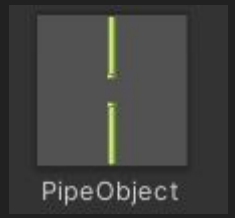
Now that we have a bird we need to provide it some obstacle.

We'll put together this Pipe Game object.

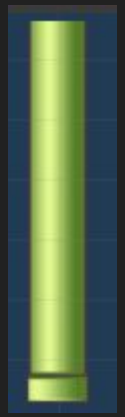
It will be made up of two Game Objects with Sprite Renderer which have BoxCollider2D on them and One Game object that only has BoxCollider2D on it. That way we can tell if the bird died or gained a point.

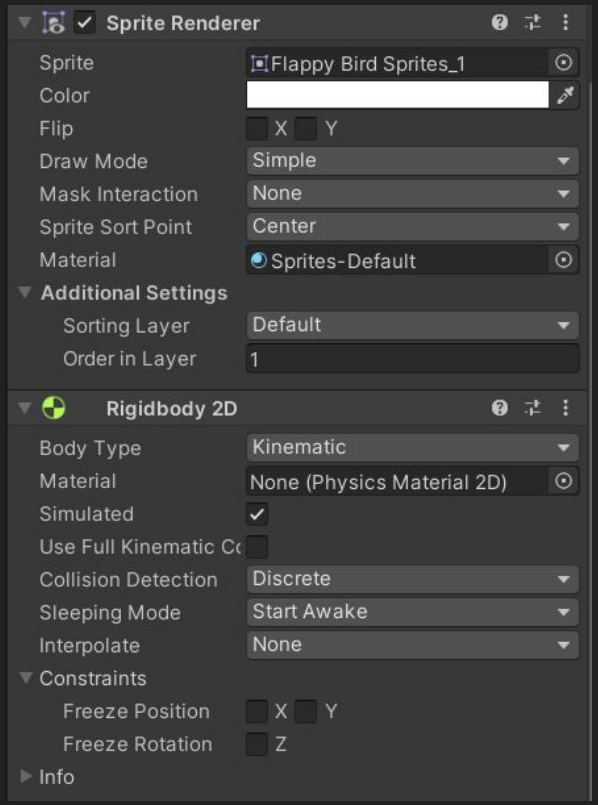
All of the colliders will be set to be Trigger so that the bird can use code with them.

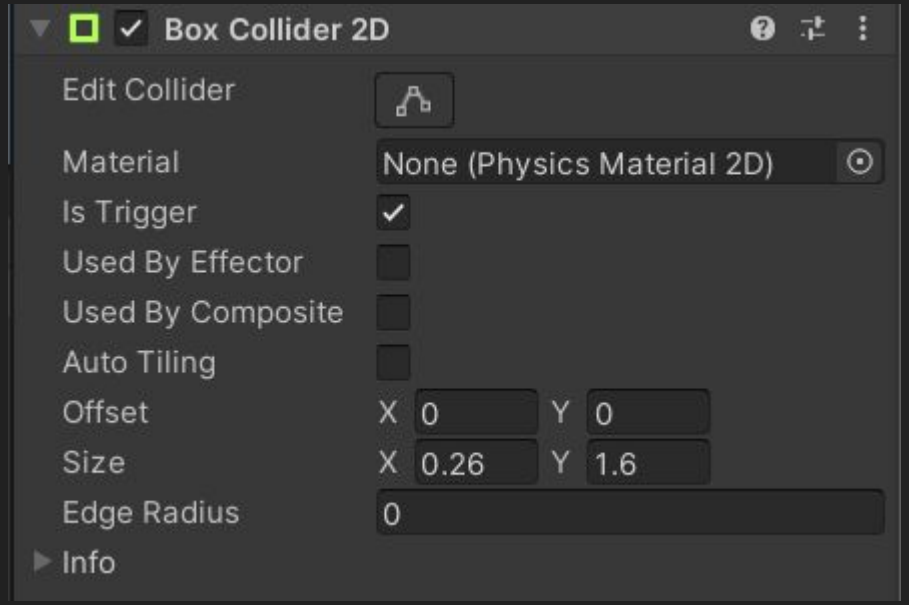
Once all of that is put together we will make the pipe into a PreFab.



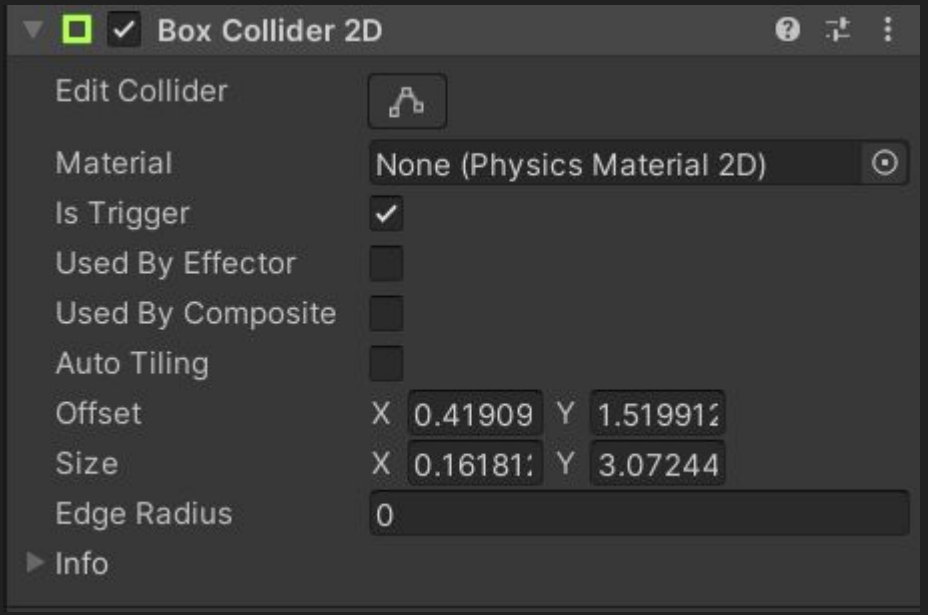
Pipe Game Object







Goal Game Object



Scripting the Pipe - Pipe Script

We want the pipe to be able to do two things: move left and die after X amount of time.

Let's start with moving left. We'll create a public float variable that we can adjust in Inspector to control how fast the Pipe is moving.

Then we'll add a Vector3.Left(-1,0,0) times the variable we made times Time.deltaTime (to normalize the speed per second).

That will have the pipe move left.

```
public float speedMag = 2;
```

```
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    transform.position +=
        speedMag * Vector3.left * Time.deltaTime;
}
```

Scripting the Pipe - Pipe Script

Now we need it do die after it leaves the screen.

Otherwise we'll have hundreds of pipes going left eating up memory and potentially crashing the game.

To do that we will create a function called Death that uses IEnumerator as it's return value. This will allow us to wait a for X amount of time before doing something. In this case Destroy(gameObject);

Lastly we have to start this function so we will call it from Start using StartCoroutine(Death());

```
public float endTime = 5;
```

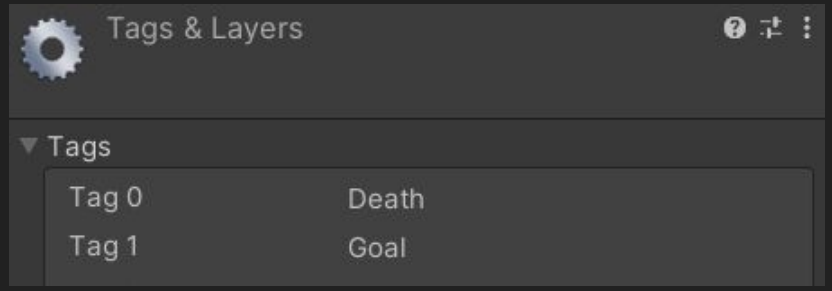
```
private IEnumerator Death()
{
    yield return new WaitForSeconds(endTime);
    Destroy(gameObject);
}
```

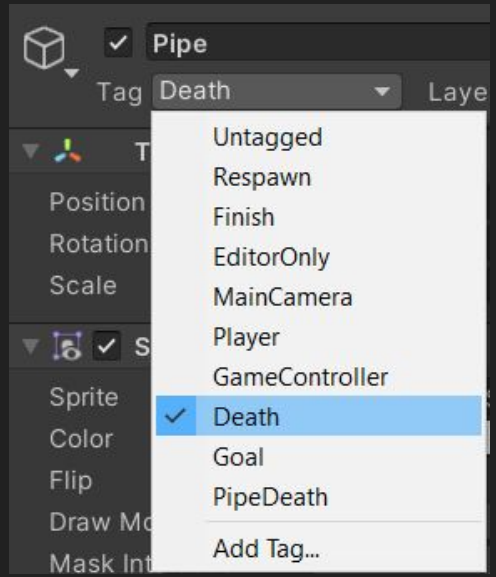
```
private void Start()
{
    StartCoroutine(Death());
}
```

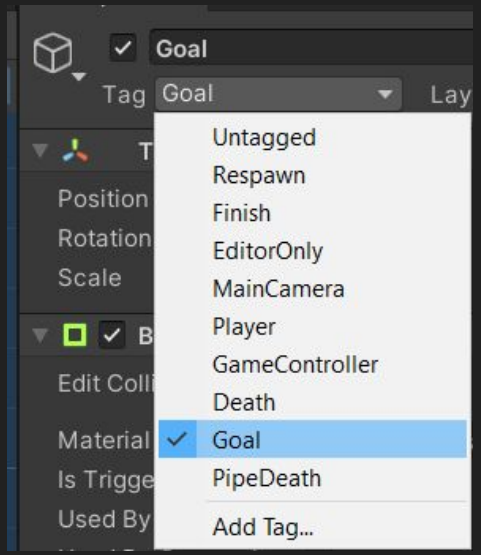
Bird Interacting With Pipes

We will create two function each listening for the Bird entering a Trigger Collider.

But to be able to tell what Trigger the bird has entered we will have to add Custom Tags that will allow us to difference between the game objects.







Bird Interacting With Pipes - Bird Script

Now that we can tell what trigger collider is what we will create two function that will check if the Bird has entered or exited the colliders.

We will check if the bird has hit the pipe and will play out it dying.

We will check if the bird has passed through the goal and increase the score.

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.CompareTag("Death"))
    {
```

```
//Checks if the player passed the goal post then up
Unity Message | 0 references
private void OnTriggerExit2D(Collider2D collision)
{
    if (collision.CompareTag("Goal"))
```

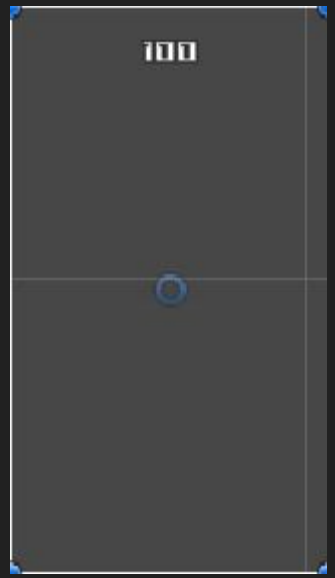
Score

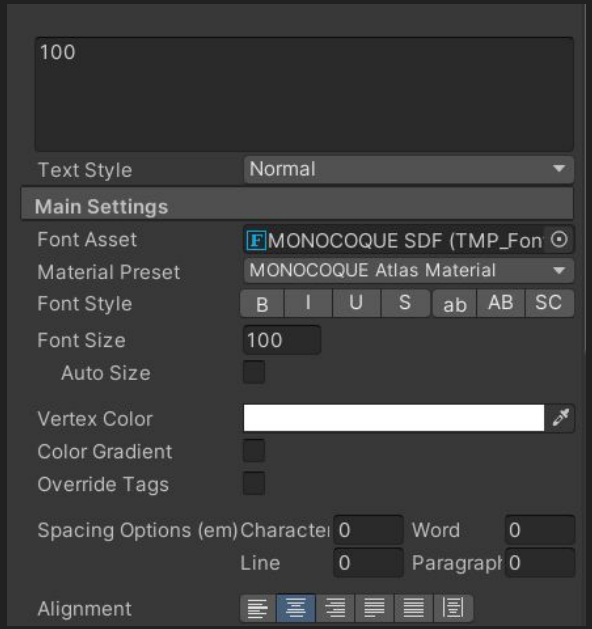
We'll start with the score as it will be easier to implement.

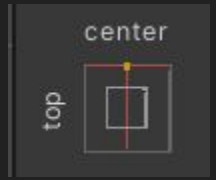
First we will create Canvas and inside that Cavans we will make a TextMesh Pro that will display the text.

We will place it near the top of the screen and anchor it to that.

We'll pass in our custom font make sure the text is always in the center and add thickness to the outline so it stands out.







Score Code - Bird Script

We will start with creating two new variables inside the Bird, one that will allow us to update the Text and one that will keep track of the current score.

Inside the Start Function we will connect the Text using GameObject.Find() and get the TextMesh Pro Component. And we will update it to say that the score is 0.

Inside our ONTriggerExit2D function we will increase the counter and update the text each time the bird passes through the goal trigger collider.

```
private TextMeshProUGUI scoreText;
private int scoreCounter = 0;
```

```
void Start()
scoreText = GameObject.Find("Canvas").transform.Find("ScoreText")
    .GetComponent<TextMeshProUGUI>();
scoreText.text = "" + scoreCounter;
```

```
if (collision.CompareTag("Goal"))
{
    //Increase score
    scoreCounter++;
    //Update UI
    scoreText.text = "" + scoreCounter;
```

Death

When the bird dies we want to make it drop down really fast outside of the bounds of the screen, and the sprite to change to a grayed out.

To do that we will get three variables, the BoxCollider2D so it won't interact with the Ground anymore, SpriteRenderer so we can change the sprite and the Sprite that we'll set it to upon death.

Inside the start function we connect to the BoxCollider from the gameObject itself and the SpirteRenderd from the child game object.

```
private BoxCollider2D boxCollider2D;

private SpriteRenderer spriteRenderer;
public Sprite deathSprite;
```

```
void Start()
boxCollider2D = GetComponent<BoxCollider2D>();
spriteRenderer = transform.GetChild(0)
    .GetComponent<SpriteRenderer>();
```

Death

We connect the corresponding Sprite in the Inspector View to the public variable.

When the bird hits the pipe we will increase the gravity from the 0.5f to 1 so it falls really fast, we turn off the BoxCollider2D so it can pass through the ground and we change the sprite that's connected to the SpriteRenderer to the gray out version.

Now the player will still have the ability to tap we will work on that in a moment.

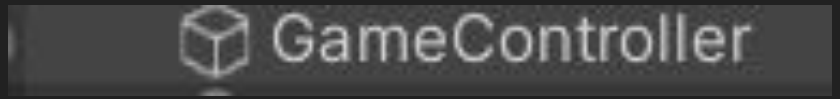
```
if (collision.CompareTag("Death"))
{
    //Let the bird fall faseter
    rigidbody2D.gravityScale = 1;
    //Let it fall off the screen
    boxCollider2D.enabled = false;
    //Change the sprite to be gray
    spriteRenderer.sprite = deathSprite;
```

Game Script

As we previously said after the player hits a pipe they will still have the ability to control the bird. We don’t want that to happen.

We will create a new script that will control how the game should behave at different States, this game will have three states.

1. Waiting For Player to Start Level
2. Playing Level
3. Lose State



Game Script

First thing we're going to do is create an enum, enum basically allows us to connect numbers to name so in this case Waiting Click repsernets 0, Playing represents 1 and LostMenu represents 2. It's easier to tell what's happening with the text rather than numbers.

We will also create a variable that will track what state we're currently in. The game will always start with Waiting For Player To Click.

```
//======================= Game States
//Defines if the game is performing automatic action or play
7 references
private enum GameState
{
    WaitingToClick,   //Player has to click to start game
    Playing,          //Play can fly up and pipes are spawn
    LostMenu,         //Lost Menu Buttons
}
private GameState _currentGame = GameState.WaitingToClick;
```

Game Script

We want all of the game object that update to be connected to this script so we will create a switch statement that check what state we're in and add function that will execute only when we are in that state.

Went we wait we will want the player to not move and the pipes to not spawn.

When we're playing player can move the bird and pipes can spawn and lastly when the player lost we want to stop the player from moving and any more pipes from spawning.

```
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    switch (_currentGame)
    {
        case GameState.WaitingToClick:
            WaitingToClick();
            break;
        case GameState.Playing:
            UpdatePlaying();
            break;
        case GameState.LostMenu:

            break;
    }
}
```

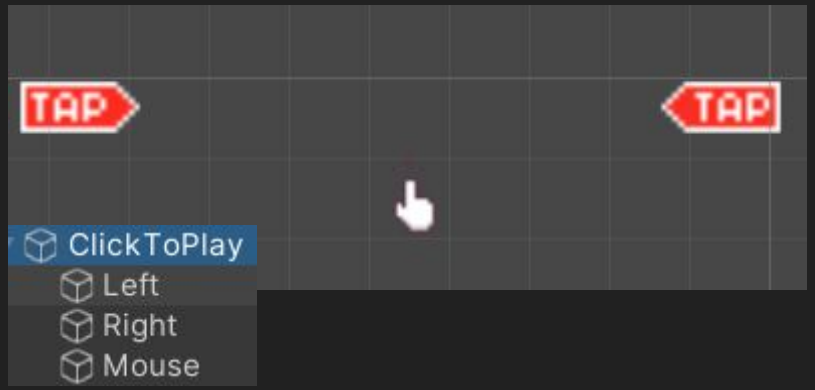
WaitForClick - Game Script

Waiting for click is pretty simple we want to give the player a moment to start the game.

The function only is set to listen if the player has clicked, once they did the bird should start falling and pipes should start showing up.

We should also have a UI that prompts the player to click and should disappear once the game has started.

```
private void WaitingToClick()
{
    if (Input.GetKeyDown(KeyCode.Mouse0))
    {
        FromClickToPlay();
    }
}
```




```
private GameObject clickToPlayText;
clickToPlayText = GameObject.Find("Canvas")
    .transform.Find("ClickToPlay").gameObject;
clickToPlayText.SetActive(true);
```

Sebastian Grygorczuk - STEM Institute at CCNY

Bird Waiting - Bird Script

Now that we have these states we're going to update the Bird Script so that it reflects the States.

First we don't want the Bird to start falling till player is ready, so we will set the gravity to 0 in the Start Function.

Next we want to give the Bird Gravity when the State is Playing so we will create Public function that the Game Script can call when switching states.

```
void Start()
rigidbody2D.gravityScale = 0;
```

```
public void SetUpBird()
{
    rigidbody2D.gravityScale = 0.4f;
    animator.enabled = true;
}
```

Switching to Play - Game Script

Now that the bird is ready we can connect it inside the Start Function of the Game Script.

When the player clicks the prompt will disappear, the bird will get gravity and start falling and the game will enter the Playing State.

```
private Bird bird;
```

```
bird = GameObject.Find("FlappyBird").
    GetComponent<Bird>();
```

```
private void FromClickToPlay()
{
    bird.SetUpBird();
    clickToPlayText.SetActive(false);
    _currentGame = GameState.Playing;
}
```

Playing State - Bird Script - Game Script

Now the bird is still able to be controlled by the player at any point beauce the clicking is connected to the update function.

We will go to the Bird Script and rename Update to Update Bird.

In the Game Script in Update Playing we will use the bird and call the UpdateBird function.

Now the bird will only fly up when the game is in Playing State.

Bird Script

```
public void UpdateBird()
{
    if (Input.GetKeyDown(KeyCode.Mouse0))
    {
        rigidbody2D.velocity = forceScale * Vector2.up;
    }
}
```

Game Script

```
private void UpdatePlaying()
{
    bird.UpdateBird();
```

Lose State

We lastly want to take away the player control when the player hits the pipe. So we will create this function called FromPlayToLose which will change the state to LostMenu.

But we only know if the lost if the bird hit the pipe so we will have to check it from inside the bird script.

We will bring in the game Script Variable, connect it in the Start Function and then add the FromPlayToLose function ithe OnTriggerEnter2D.

Now the player won't have control once they hit a pipe.

Game Script

```
public void FromPlayToLoes()
{
    _currentGame = GameState.LostMenu;
}
```

Bird Script

```
private GameScript gameScript;
```

```
Start(
gameScript = GameObject.
    Find("GameController").GetComponent<GameScript>();
```

```
OnTriggerEnter2D(
gameScript.FromPlayToLose(scoreCounter);
```

Lost State Menu

Now that the player lost we need a way to restart the level.

To do that we will create a menu with a button that will reload the scene. The menu consists of four Images

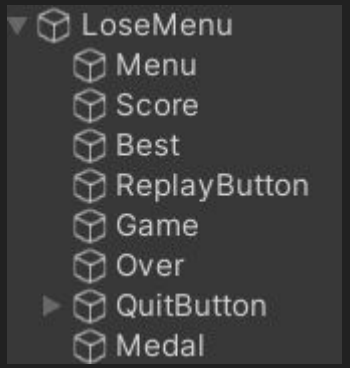
1. Game
2. Over
3. Medal
4. Board

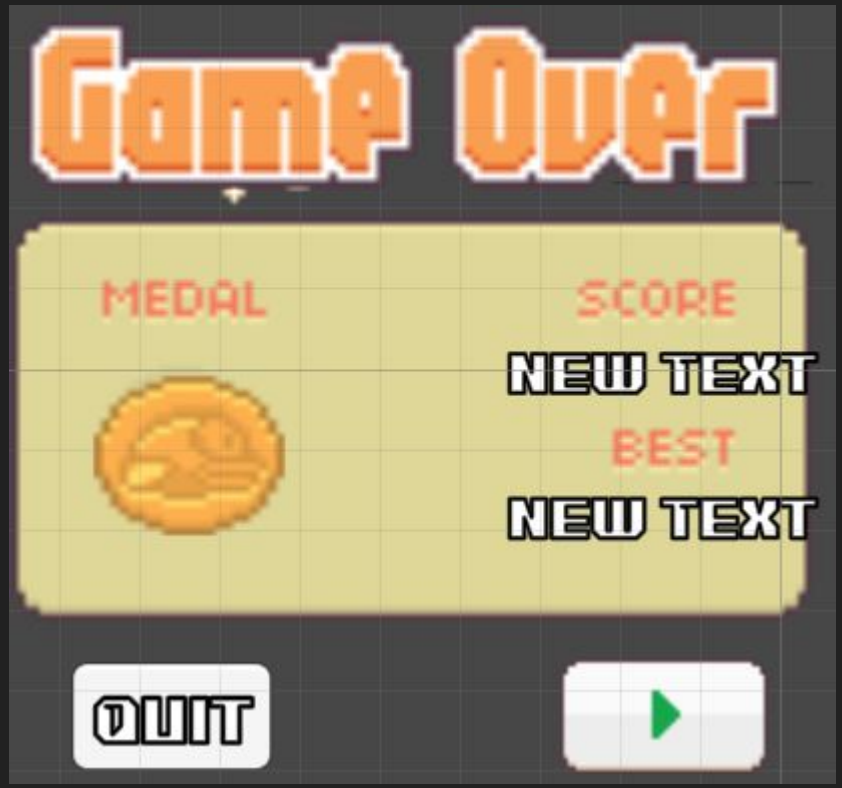
It has two buttons

1. Quit
2. Play

Two Text Mesh Pros

1. Score
2. Highscore





Lost Menu - Game Script

```
private GameObject loseMenu;
```

We will start off with connecting the Menu Game object to the Game Script, and setting it off in the Start Function.

```
loseMenu = GameObject.Find("Canvas").transform.Find("LoseMenu").gameObject;
loseMenu.SetActive(false);
```

Then we will add that it's turned on in the FromPlayToLose function allowing us to interact with the buttons.

```
public void FromPlayToLose(int counter)
//Turns UI on/off
scoreImage.SetActive(false);
loseMenu.SetActive(true);
```

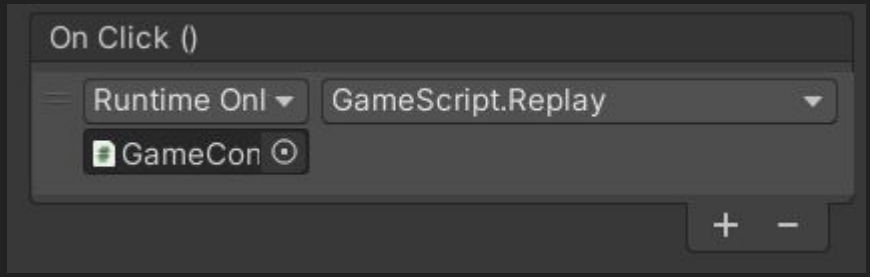
Lost Menu Buttons

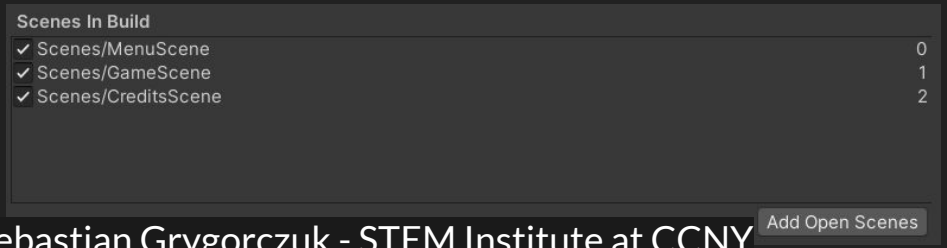
We will create two public functions each one connecting to a different that will send them to the appreciate scene.

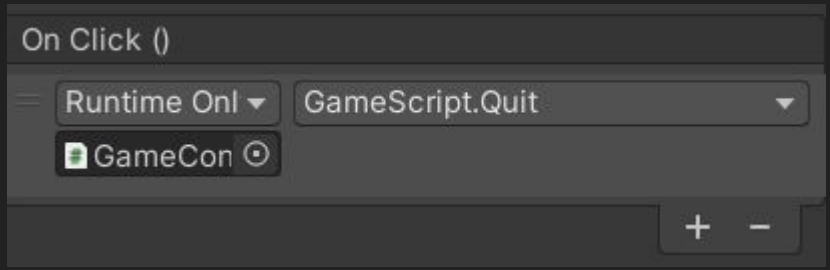
We will connect the through the OnClick() box in the button component.

We also make sure that the Scene is connected by going to File ->Build Setting->Add Open Scene.

```
public void Replay()
{
    SceneManager.LoadScene("GameScene");
}

public void Quit()
{
    SceneManager.LoadScene("MenuScene");
}
```







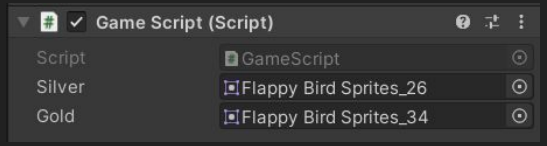
Updating the Score and Medal

We want to reward the player for doing a good job so we're going to display the score change the medal design based on how well they did.

We will connect the Image and the TextMeshProGUI in the Start function and we will have two sprites for the silver and gold, it will be bronze otherwise.

```
private TextMeshProUGUI score;
```

```
public Sprite silver;
public Sprite gold;
private Image medal;
```




```
//Medal
medal = GameObject.Find("Canvas")
    .transform.Find("LoseMenu")
    .transform.Find("Medal").GetComponent<Image>();

//Text
score = GameObject.Find("Canvas")
    .transform.Find("LoseMenu")
    .transform.Find("Score")
    .GetComponent<TextMeshProUGUI>();
```

Updating the Score and Medal

We will ask that the Bird passes the score when it calls to FromPlayToLose

Then we will use that score to set the sprite, if score is larger than 10 and less than 20 it'll be silver and if it's larger than 20 it'll be gold.

We also set the score to be the counter.

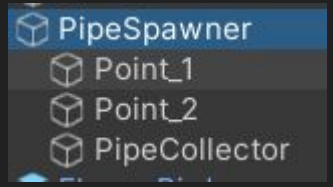
```
public void FromPlayToLose(int counter)
{
    //Checks if the new score is a high sco
    if (data.GetScore() < counter)
    {
        data.SetScore(counter);
    }

    //Changes medal sprite based on how hig
    if(counter >= 10 && counter < 20)
    {
        medal.sprite = silver;
    }
    else if(counter >= 20)
    {
        medal.sprite = gold;
    }

    //Updates the text
    score.SetText(counter.ToString());
```

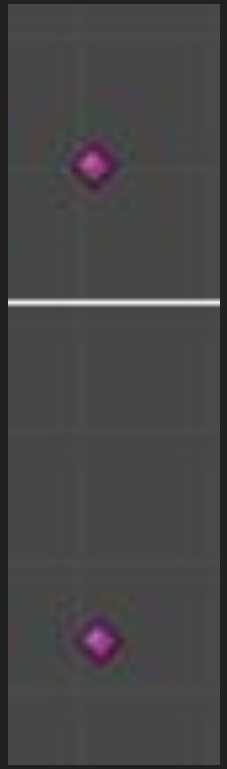
Pipe Spawner

Now we have a whole game loop set up. Game Start waiting for the player to click, the bird starts moving and if they hit the pipe they die and can restart the level, but we only got one pipe in the level.

We have to create a game object that will create more pipes.

It will consist of two points we will use to randomize the space and a empty game object where we'll spawn the extra pipes.





Pipe Spawner Script

Let’s start with the variables in the Pipe Spawner.

pipeSet is the Prefab that we will be Instatinging.

Pipe Collector is where we’re going to put the newly made pipes game objects.

spawn Points is the two point we’ll use to randomize the height.

TIMER is the time between each spawning of the pipes.

timeLeft is current time until a new pipe will be made.

```
//The game object that will be spawned
public GameObject pipeSet;
//The collector for newly created pipe
public GameObject pipeCollector;
//Used to tell where the bullet will spawned
public Transform[] spawnPoints;

//What the time resets to after we finish counting down to
public float TIMER = 10f;
//What the timer is at the moment
private float timeLeft = 10f;
```

Pipe Spawner Script

In the start function we will set the time Left to Timer and connect Pipe Collector game object so we can put the new pipes in it.

We will also create and Update Pipes Spawner that will count down till timeLeft is less than zero.

We'll create a random position based on the two points in our array.

We'll Instante the pipe, set it to be a child of Pipe Collector and reset the timer.

```
void Start()
{
    timeLeft = TIMER;
    pipeCollector = transform.Find("PipeCollector").gameObject;
}
```

```
public void UpdatePipeSpawner()
{

    //Count down the time
    timeLeft -= Time.deltaTime;
    if (timeLeft < 0)
    {
        Vector3 pos = new Vector3((float) spawnPoints[0].position.x,
            Random.Range((float) spawnPoints[0].position.y,
            (float) spawnPoints[1].position.y), 0);
        var var = Instantiate(pipeSet, pos, Quaternion.identity);
        var.transform.SetParent(pipeCollector.transform);
        //Reset the timer
        timeLeft = TIMER;
    }
}
```

Connecting to Game Script

We will lastly connect the pipe spawner to the Game Script so that it only spawn pipes when the game is in Playing State.

We connect the pipes spawner object to the game script through the start function.

We set it to be called whenever the UpdatePlaying() function is being executed.

```
private PipeSpawner pipeSpawner;
```

```
pipeSpawner = GameObject.
    Find("PipeSpawner").GetComponent<PipeSpawner>();
```

```
private void UpdatePlaying()
{
    bird.UpdateBird();
    pipeSpawner.UpdatePipeSpawner();
}
```

Data

We will want to store the high score and be able to move it between scene.

We will create game object and script called data.

The game object will have two variables that score int and the Data game script that we will use to save the information.

We will also have two functions one to store the data and one to give the data back.

```
private static Data _instance;
public int score = 0;
```

```
private void Awake()
{
    //Checks if there already exits a
    if (_instance != null)
    {
        Destroy(gameObject);
        return;
    }

    _instance = this;
    DontDestroyOnLoad(gameObject);
}
```

```
public void SetScore(int s)
{
    score = s;
}
public int GetScore()
{
    return score;
}
```

Data in Game Script

We collect the data game object and the text mesh pro we're going to display it in inside the Start Function

Inside FromPlayToLose we check if the current score was larger than the previous best, if so we save it.Then we display it.

```
private Data data;
private TextMeshProUGUI best;
```

```
data = GameObject.Find("Data").GetComponent<Data>();
```

```
best = GameObject.Find("Canvas").
    transform.Find("LoseMenu").transform
    .Find("Best").GetComponent<TextMeshProUGUI>();
```

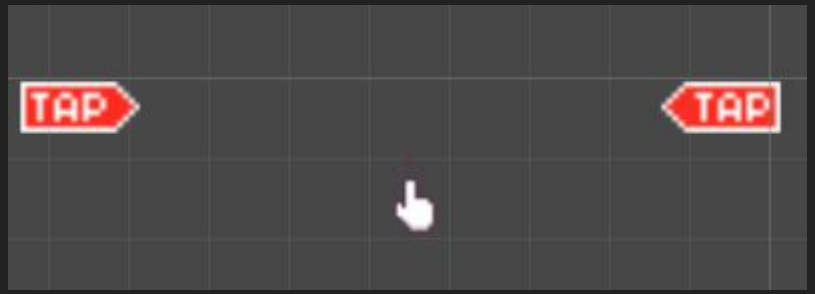
FromPlayToLose

```
//Checks if the new score is a high score
if (data.GetScore() < counter)
{
    data.SetScore(counter);
}
best.SetText(data.GetScore().ToString());
```

Animation

We will animate two things in the game screen.

The bird flapping its wings and the click prompt during the waiting state.



Waiting Animation

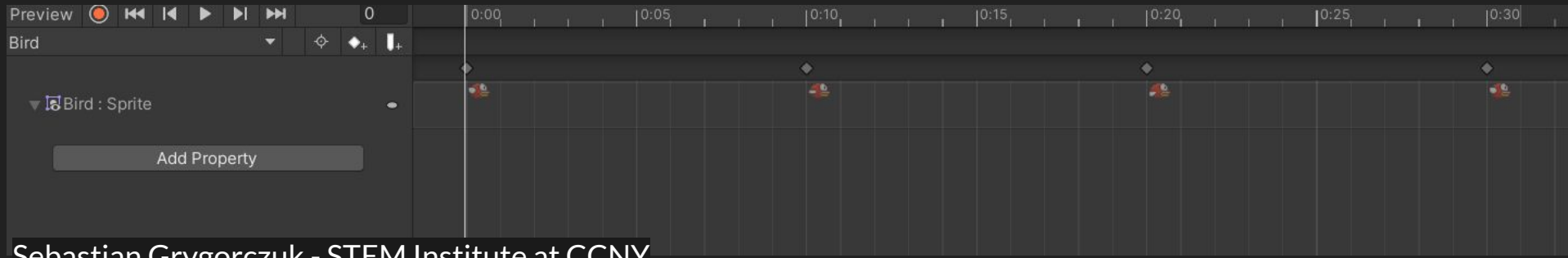
We will manipulate three things here, the Left Tap and the Right tap position, they’re gonna move closer and then back to their own position

Then Mouse image scale making it smaller then going back to its original size.

This will create a call to action to click on the screen.

Bird Animation

For the bird the animation is very simple just switching between the three frames of animation for the bird flapping its wings.



Bird Animation Script

We will want the bird to not be animated during the waiting state or the lost menu state.

We will connect the game object in the start function and turn it off at the start.

Inside the SetUpBird Function we will turn the animation on.

Inside the OnTriggerEnter2D we will turn it off again.

```
private Animator animator;
```

```
//Sets animation on and off
animator = GetComponent<Animator>();
animator.enabled = false;
```

```
public void SetUpBird()
{
    rigidbody2D.gravityScale = 0.4f;
    animator.enabled = true;
}
```

```
OnTriggerEnter2D
animator.enabled = false;
```

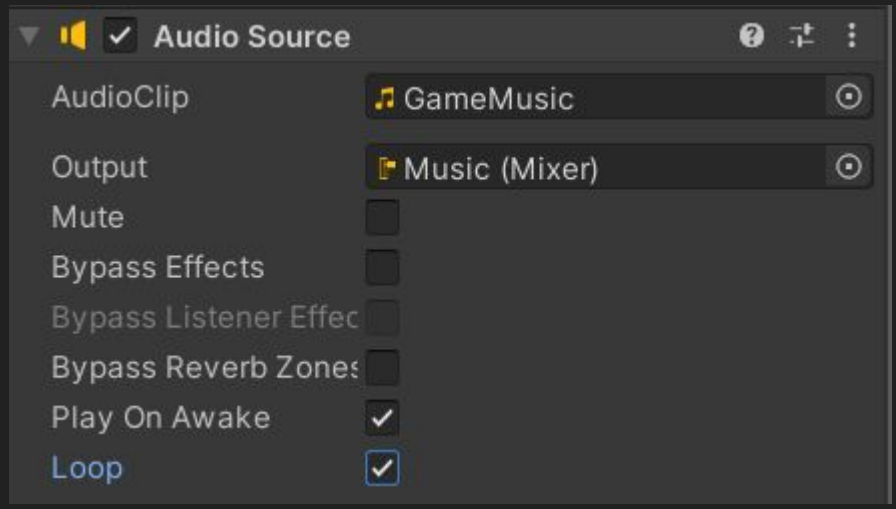
Music & SFX

We will create four game objects which will have audio sources.

Two of them will be under the player and two of them will be in the game world.

Three of them are SFX so make sure that the Play On Awake is turned off.

For the music make sure the loop is on.



SFX in Bird

We add in the two audio sources and connected them in the start function.

We add the dieSFX to play inside the TriggerEnter2D

We add the pointSFX to play inside TriggerExit2D

```
private AudioSource pointSFX;
private AudioSource dieSFX;
```

```
Start(
pointSFX = transform.GetChild(1).GetComponent<AudioSource>();
dieSFX = transform.GetChild(2).GetComponent<AudioSource>();
```

```
OnTriggerEnter2D(
dieSFX.Play();
```

```
OnTriggerExit2D
pointSFX.Play();
```

Button Script - Game Script

We add the button SFX through the Start function.

Then we have to redo our exit function as we want to wait for the button SFX to finish playing before we leave the scene.

We create an IEnumerator function that will play the sound, wait till it's over and then load the desired scene.

```
private AudioSource button;

Start(
//Auido
button = GameObject.Find("ButtonSFX").GetComponent<AudioSource>();

0 references
public void Replay()
{
    StartCoroutine(PlayAndLeave("GameScene"));
}

//Starts the process of moving to Menu Scene using button
0 references
public void Quit()
{
    StartCoroutine(PlayAndLeave("MenuScene"));
}

//Waits until the button SFX has stopped playing and then g
//requested level
2 references
private IEnumerator PlayAndLeave(string level)
{
    button.Play();
    yield return new WaitForSeconds(button.clip.length);
    SceneManager.LoadScene(level);
}
```