Jamie Bovell

Programming Tutorial

First Tutorial: Collectables

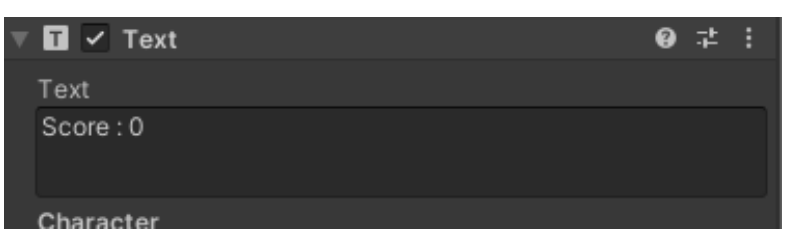
1) Begin by opening Unity and creating a new 3D Project.

2) Choose either a 3D shape or import a 3D model and then drag it into the scene. This will be your collectable.

3) Size your collectable correctly so it is a decent size, add a collider if it doesn't already have one, and tick the 'Trigger' checkbox.

4) Then right click in the hierarchy, go to UI then select Text. Name this text 'Score Text'.

5) In the textbox type "Score: 0' and anchor this to the top left-hand corner whilst holding the 'alt' key



6) Then right click in the hierarchy and create an empty gameObject and name it 'Scoring System'

7) Select it and in the inspector click 'Add Component' and add a script.

8) Name this script 'ScoringScript' and then double click it to open it.

9) At the top of the script make sure to write 'Using UnityEngine.UI;' This is so the script can be used to change UI elements

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

10) Then create two variables, One public 'gameObject' called 'Scoretxt' and another public static int called score;

```
public GameObject scoreTxt;
public static int score;
```

11) Then in the Update method use get component on the 'scoreTxt' and set the text to "Score " + score;.This will be used to update the score on the UI.

```
Unity Message | 0 references
void Update()
{

    scoreTxt.GetComponent<Text>().text = "Score: " + score;

}
```

12) Save this script and return to Unity

13) Select your collectable in the hierarchy then go to the inspector and click add component and then click script. Name this script ‘Collectable’ and then double click it to open it.

14) Delete the start and update method and create a new ‘onTriggerEnter’ method.

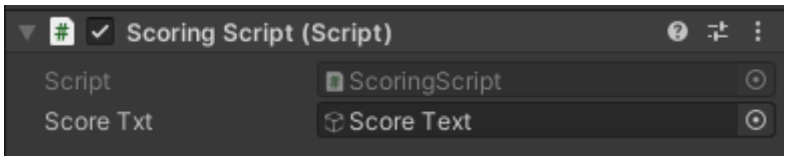
15) In this method use your ‘score’ variable from your ‘ScoringScipt’. And add 1 to its value. This value is the number of points you receive per collectable.

This will occur every time the player comes into contact with the collectable.

16) In the same method below it use Destroy(gameObject) to destroy the collectable. This is so the collectable is deleted once it is collected.

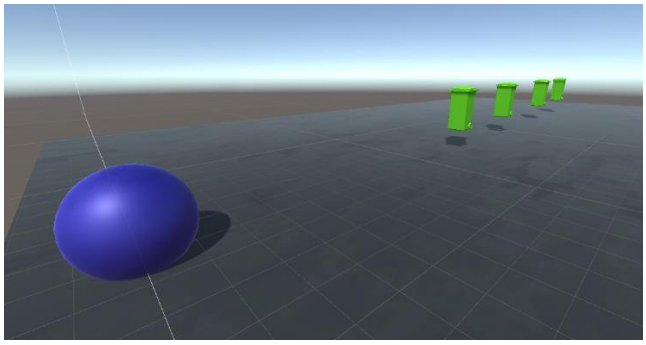
```
void OnTriggerEnter(Collider other)
{
    ScoringScript.score += 1;

    Destroy(gameObject);
}
```

17) Once again return to unity and then select ‘ScoringSystem’ and drag ‘Score Text into the ‘Score Txt’ variable in the inspector window.



18) Then select your collectable and drag it into the Assets or prefab folder. This will make your collectable a prefab.

19) Then drag as many collectables as you want into the scene.

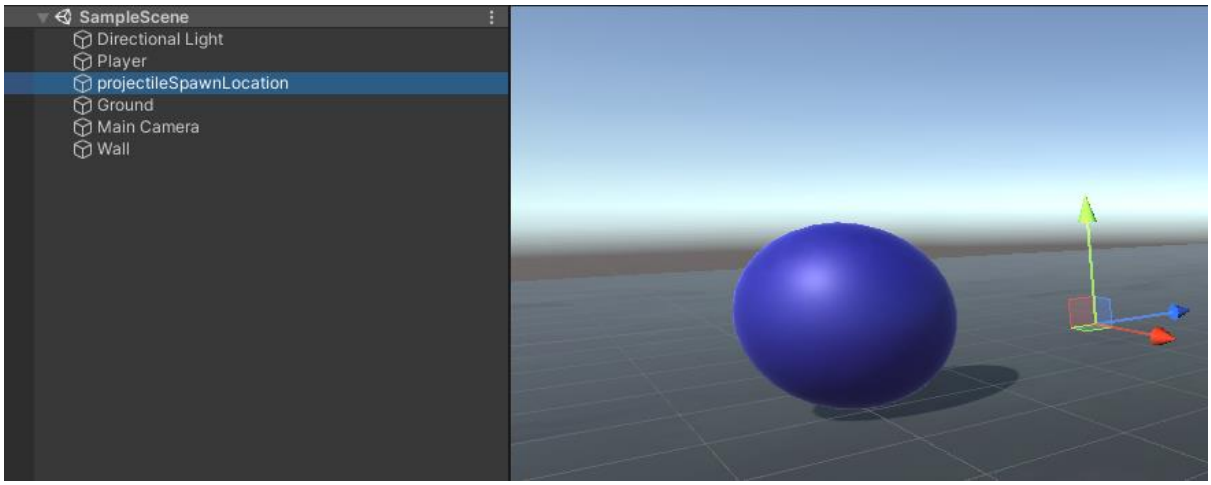


Second Tutorial: Projectile

1) Begin by opening Unity and creating a new 3D Project.

2) Choose either a 3D shape or import a 3D model and then drag it into the scene. This will be your Projectile. Add a rigid body to this object then drag it into the assets folder to make it a prefab

3) Click in the hierarchy and create an empty gameObject , name this projectileSpawnLocation and place it Infront of your player in the scene.



4) Click on your player in hierarchy and click add new component. Then click new script. Name this script ‘Shooting Projectile’. Then double click the script to open it.

5) then create two new variables. One public gameObject called Projectile and public Transform called projectileSpawnLocation.

```
public GameObject projectile;
public Transform projectileSpawnLocation;
```

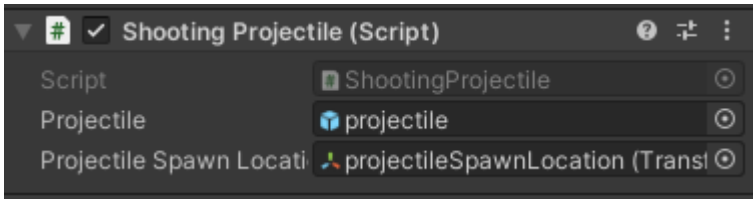
6) In the void Update method create an if statement where the condition is if the person presses the spacebar. This will instantiate the projectile every time the spacebar is pressed.

7) Inside the if statement Instantiate the projectile , with a position of the projectileSpawnLocation with a rotation of the projectileSpawnLocation.

```
if (Input.GetKeyDown(KeyCode.Space))
{
    Instantiate(projectile, projectileSpawnLocation.position, projectileSpawnLocation.rotation);
}
```

8) Return to Unity, click on your player and in the inspector on the ‘Shooting Projectile’ script, drag your projectile prefab from your assets folder into the public slot named projectile.

9) Then drag your’ projectileSpawnLocation’ gameObject into the ‘Projectile Spawn Location’ slot.



10) Then select your Projectile in the ‘Assets Folder’ click ‘New Component’ and add a new script. Name this script ‘Projectile’. And double click it to open it.

11) Create three variables. The first one is a public float named speed with a value of 1000f. This is the speed of our projectile. The second one is a public float called uptime; this is how long the projectile will exist for. The third one is a private Rigidbody called rb.

```
public float Speed = 1000f;
public float uptime = 3f;
private Rigidbody rb;
```

12) Then create a new void ‘Awake’ method. And inside of this method call the RigidBody component.

```
void Awake()
{
    rb = GetComponent<Rigidbody>();
}
```

13) Then in the start method add force to your projectile’s rigidbody in a forward’s direction and multiply this by speed.

14) Then on the next line destroy the game object but under the condition of your ‘uptime’.

```
void Start()
{
    rb.AddForce(rb.transform.forward * Speed);
    Destroy(gameObject, uptime);
}
```

15) Return to unity and select your projectileSpawnLocation gameObject and add new script called ‘LaunchLocation’ and double click it to open it.

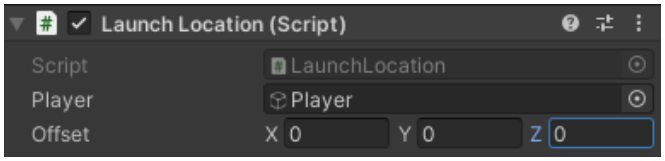
16) Once open create two variables. One public GameObject called player and a public Vector3 called offset.

```
public GameObject Player;
public Vector3 offset;
```

17) Then in the Update method set the transform position of the GameObject that this script is attached to equal to that off the player and then add offset. This sets the projectileSpawnLocation GameObject position to that of the player.

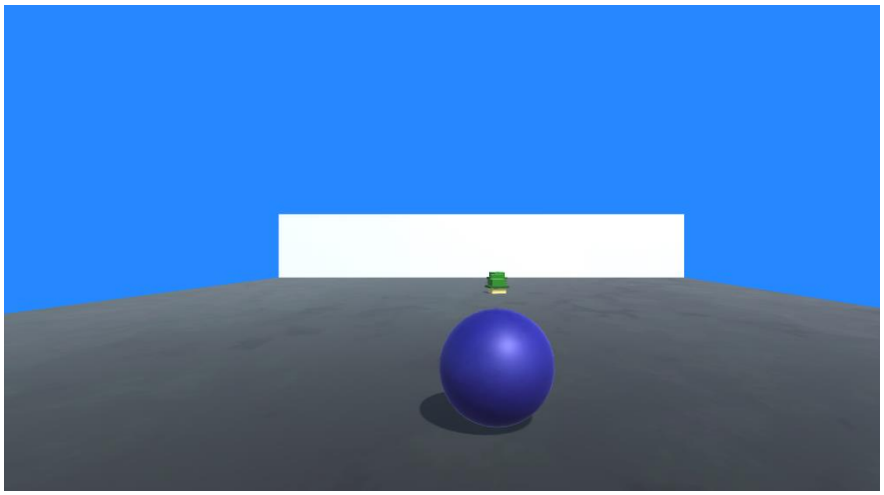
```
void Update()
{
    gameObject.transform.position = Player.transform.position + offset;
}
```

18) Return to unity and click and drag the player into the ‘player’ slot on the projectileSpawnLocation objects inspector



19) Return to unity and use the Vector3 Offset to position the projectileSpawnLocation GameObject Infront of the player.

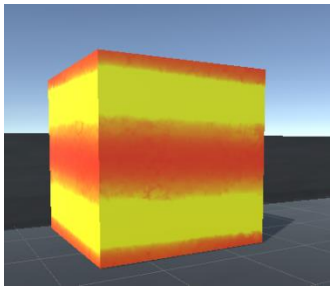
20) In play mode press the space bar key to shoot the projectiles



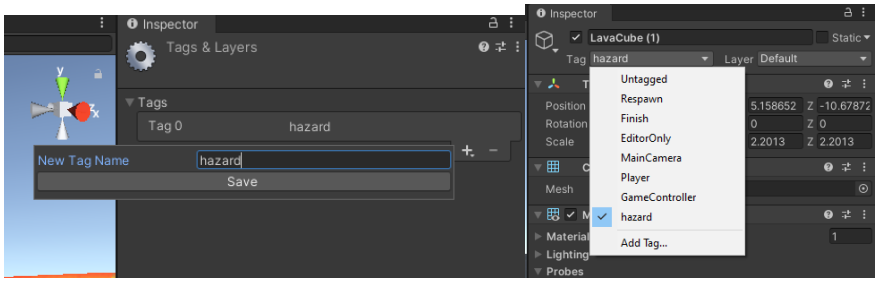
Third Tutorial: Health bar

1) Begin by opening Unity and creating a new 3D Project.

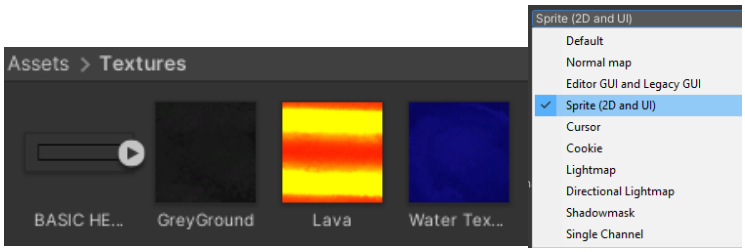
2) Choose either a 3D shape or import a 3D model and then drag it into the scene. This will be your Hazard or obstacle.



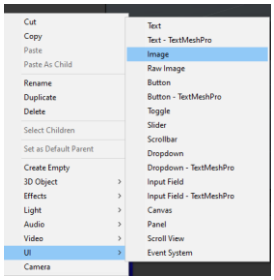
3) Click on it in the hierarchy go to the inspector window and create a new tag called ‘hazard’ and apply it to your hazard GameObject.



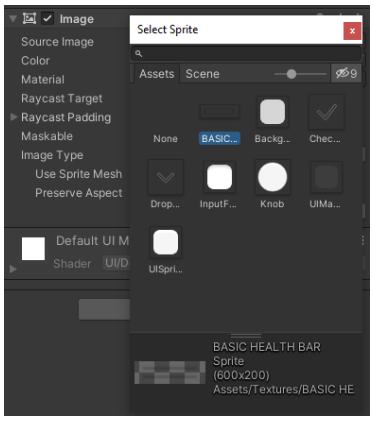
4) Import your health bar outline by dragging it into the assets folder and in the inspector window change the texture type from default to sprite



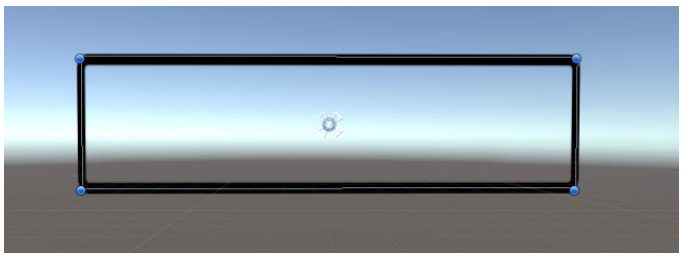
5)Right click in the hierarchy then click UI > Image. Name this image ‘Healthbar Outline’



6)Then in the inspector window click ‘Source Image’ and select your health bar image and scale it to your liking.

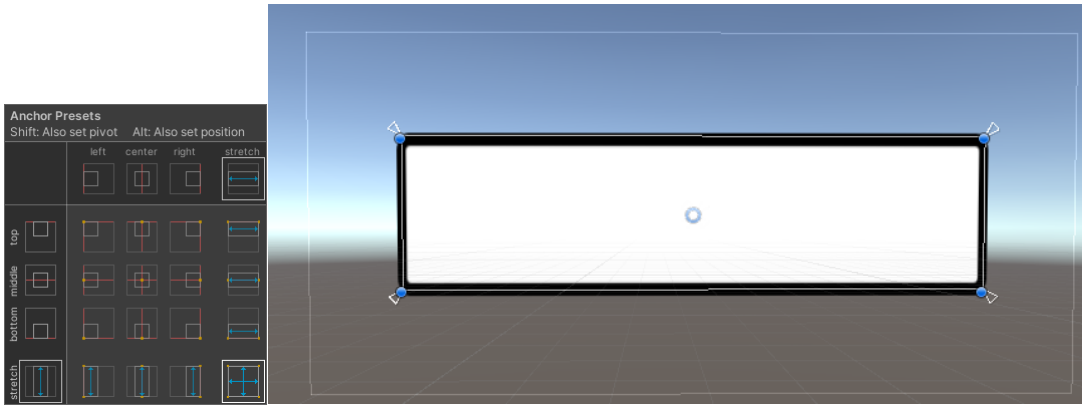


7)Then create a new GameObject that is a child of the canvas and call it Healthbar. Scale this gameObject to the outline of the Healthbar Outline.



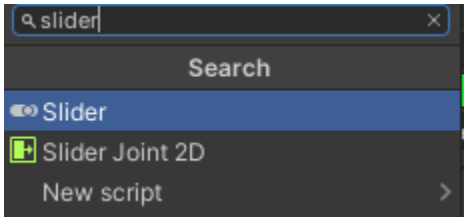
8) Next right click in the hierarchy UI > Image. Name this ‘Healthbar Health’ and make it a child of the ‘Healthbar’ gameObject.

9) Drag ‘HealthBar Health’ above ‘HealthBar Outline’ in the hierarchy then in the inspect window click the rect transform and click the bottom right anchor whilst holding the ‘alt’ key.

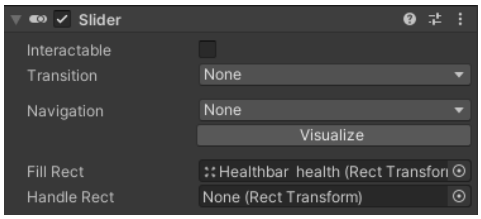


10) Then appropriately position the health bar on the UI. Typically, the somewhere on the top of the screen.

11) Next right click on the ‘HealthBar’ in the hierarchy and in the inspector window add a slider component

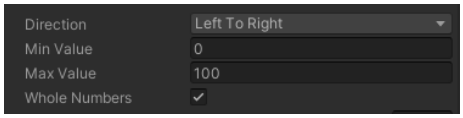


12) Click and drag ‘Healthbar Health’ into the ‘Fill Rect’ slot for the slider and then set transition and navigation to none



13) position and scale it so it fits the health bar’s outline

14) Next set the Max value of the slider to 100 and tick the Whole Numbers checkbox



15) Next click the Healthbar gameObject in the hierarchy then click add component then click script. Name this script ‘HealthBar’ and open it

16) At the top of the script write ‘uusing UnityEngine.UI;

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

17) Next crate three public variables. One Slider called healthBarSlider. Another called Gradient called Gradient and an Image called healthBarHealth.

```
public Slider healthbarSlider;
public Gradient Gradient;
public Image healthBarHealth;
```

18) First make a ‘public void’ Function called SetMaxHealth

19) Inside this function set Slider.MaxValue equal to health and slider.Value equal to health

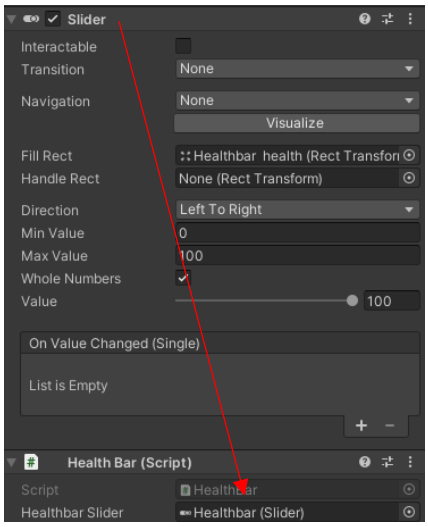
```
public void SetMaxHealth(int health)
{
    healthbarSlider.maxValue = health;
    healthbarSlider.value = health;
```

20)Next make a new ‘public void’ function called health with an int health.

21) Inside of this function set the value of the slider equal to that of the health

```
public void SetHealth(int health)
{
    healthbarSlider.value = health;
```

22) Return to Unity and click and drag the slider component from the inspector window into the slider public variable slot.

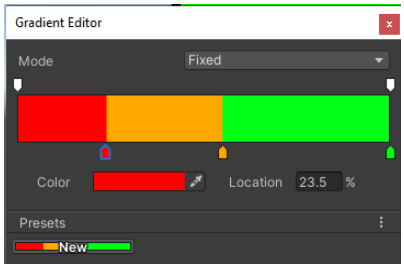


23) Then select gradient colour picker colour. This can be found on the HealthBar's 'Healthbar'Script.

24) Start by changing the gradients mode to Fixed by clicking the Mode drop down menu and clicking Fixed

25) Set the colour on the left side of the gradient to red and the colour on the right side of the gradient to green. Then click below the middle of the health bar to add a new colour in that spot and make it orange.

26) Evenly spread these colours across the gradient so they roughly consume the same amount of space along the bar.



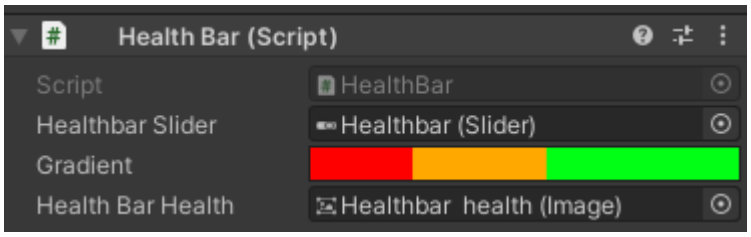
27)Return to HealthBar script and in the 'public void SetMaxHealth' function set healthBarHealth.color equal to Gradient.Evaluate with a value of 1 so that the health bar starts of green.

```
public void SetMaxHealth(int health)
{
    healthbarSlider.maxValue = health;
    healthbarSlider.value = health;
    healthBarHealth.color = Gradient.Evaluate(1f);
}
```

28) Next in the ‘SetHealth function set healthBarHealth.color to gradient.Evaluate with a value of the sliders normalized value

```
public void SetHealth(int health)
{
    healthbarslider.value = health;

    healthbarhealth.color = gradient.Evaluate(healthbarslider.normalizedValue);
}
```

29) Return to unity and click and drag the Healthbar Health into the Health Bar Health slot.



30) Next click on your player and then click add component and add a script called ‘PlayerHealth’ and open it

31)First create two public ints. One called maxHealth with a value of 100 and another called currentHealth with no given value.

```
public int maxHealth = 100;
public int currentHealth;
```

32) Then make a public ‘class’ HealthBar called healthBar.

```
public int maxHealth = 100;
public int currentHealth;
public HealthBar healthBar;
```

33) In the start method set currentHealth equal to maxHealth and then set the health on the health bar tomaxHealth.

```
void Start()
{
    currentHealth = maxHealth;
    healthBar.SetHealth(maxHealth);
}
```

34) Next create a new function and called it DamageHit with an int damage.

35) Inside of this function make it so currentHealth -= damage

36) Below this set the ‘healthBar’ health to current health

```
void DamageHit(int damage)
{
    currentHealth -= damage;
    healthBar.SetHealth(currentHealth);
}
```

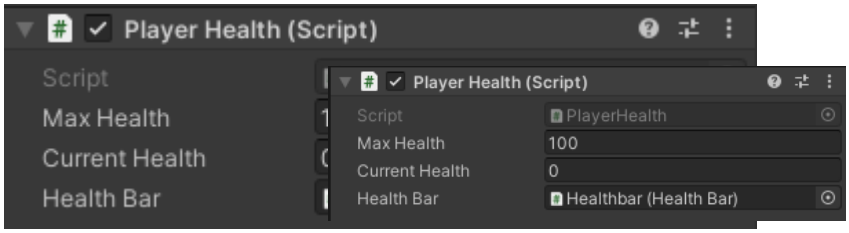
37) Next create a new ‘void OnCollisionEnter’ method.

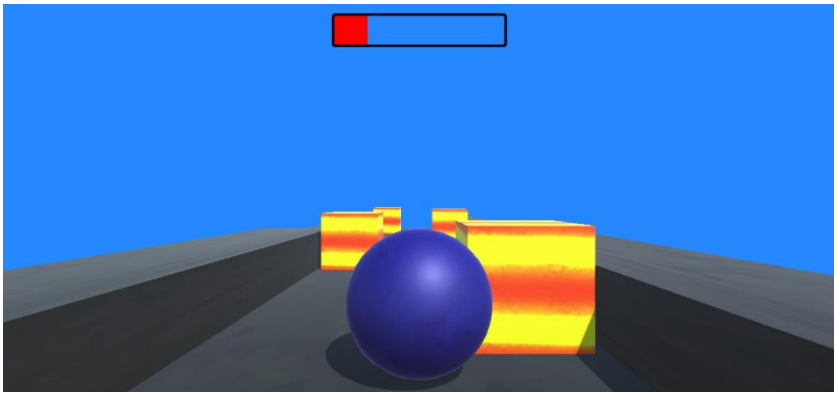
38)Inside of this method create an if statement that works of the condition if the player collides with another gameObject that has the ‘hazard’ tag the DamageHit function will be called with an int of 20;

```
void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.tag == "hazard")
    {
        DamageHit(20);
    }
}
```

39) Return to unity click on your player and then click drag your healthbar from the hierarchy into the healthBar slot on the Players PlayerHealth script.

40) On the Player Health script set the max health to 100



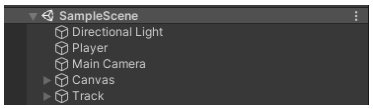


Jamie Bovell

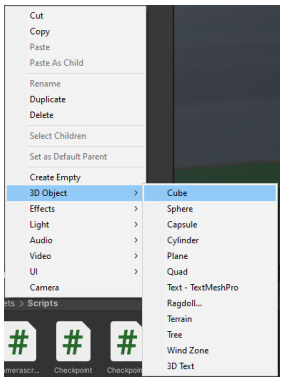
Fourth Tutorial: Racing Checkpoint

1) Begin by opening Unity and creating a new 3D Project.

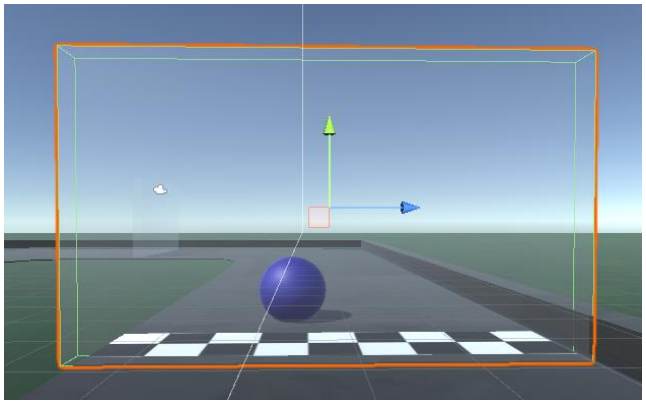
2) In the hierarchy click your track and add a new script component. Call this script ‘Checkpoint Tracker’. If you do not have a track, just make an empty game object and attach it to that.



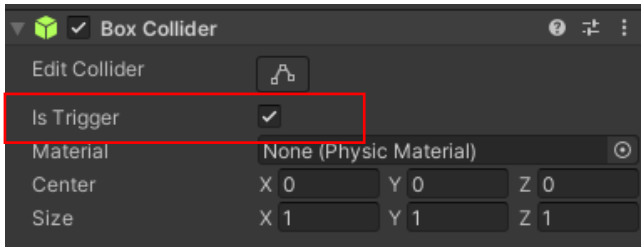
3) Right click in the hierarchy and click 3D Object > Cube



4) Name this cube Checkpointone and scale this to be a tall thin wall

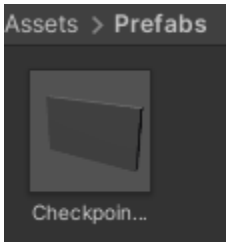


5) Next in the inspector window for Checkpointone go to the box Collider and tick the is Trigger box.

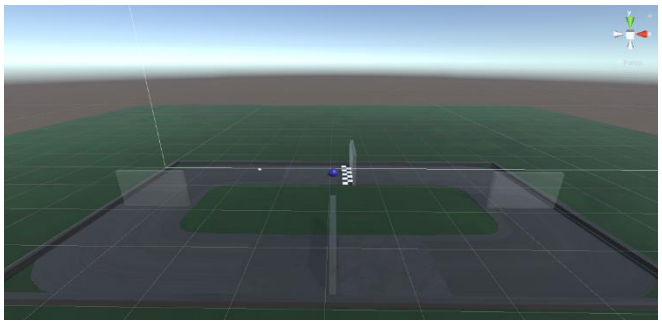


6) Next click add component and select script. Name the script ‘Checkpoint’.

7) Then make a prefab of this checkpoint by dragging it into your prefabs folder.



8) place the checkpoints around your track



9) Next make an empty game object called Checkpoints and make all your checkpoints a child of this game object, then make the Checkpoints game object a child of your track

10) Then Open the ‘Checkpoint’ script and create a new ‘private void’ OnTriggerEnter method

```
private void OnTriggerEnter(Collider other)
{
```

11) Inside this method create an if statement where if it another gameObject comes into contact with the checkpoint and it has a component called ‘Movement’ then the if statement will be true. The component called movement is the script for my player movement but if the script on your player has a different name use that script’s name in place of ‘Movement’.

```
private void OnTriggerEnter(Collider other)
{
    if (other.TryGetComponent<Movement>(out Movement movement))
    {
```

12) Now return to Unity and open the Checkpoint Tracker script on your Track gameObject.

13) inside the script create a new ‘private void’ Awake method.

```
private void Awake()
```

14) Inside this method begin by finding the checkpoints container by making a new Transform called CheckpointsTransform equal to transform.Find(“Checkpoints”)

```
private void Awake()
{
    Transform CheckpointsTransform = transform.Find("Checkpoints");
```

15) On the next line below make a for each statement that will cycle through the children of the ‘Checkpoints’ transform.

```
foreach (Transform checkpointOneTransform in CheckpointsTransform)
{
```

16)Next make a new public void called ‘PlayerPassedCheckpoint’ with an argument of ‘Checkpoint checkpointOne)

```
public void PlayerPassedCheckpoint(Checkpoint checkpointOne)
{
```

17)Checkpoint script reference PART

18) Return to the Checkpoint script and create a new private field CheckpointTracker called checkpointTracker.

```
private CheckpointTracker checkpointTracker;
```

19) Create a new ‘public void’ function called SetcheckpointTracker of (CheckpointTracker checkpointTracker)

20) Inside this function set this.checkpointTracker = checkpointTracker

```
public void SetcheckpointTracker(CheckpointTracker checkpointTracker)
{
    this.checkpointTracker = checkpointTracker;
}
```

21) Then inside the if statement located inside of OnTriggerEnter method write

checkpointTracker.PlayerPassedCheckpoint(this). This is used so that we know when the player goes through the checkpoint

```
private void OnTriggerEnter(Collider other)
{
    if (other.TryGetComponent<Movement>(out Movement movement)) // checks for a GO with a movement script
    {
        checkpointTracker.PlayerPassedCheckpoint(this);
    }
}
```

22) Next go back to the CheckpointTracker script and inside the for each statement get component of type CheckpointOneTransform.

```
foreach (Transform checkpointOneTransform in CheckpointsTransform)
{
    Checkpoint checkpointOne = checkpointOneTransform.GetComponent<Checkpoint>();
```

23) Next use CheckpointOne and call SetcheckpointTracker with (this)

```
foreach (Transform checkpointOneTransform in CheckpointsTransform)
{
    Checkpoint checkpointOne = checkpointOneTransform.GetComponent<Checkpoint>();
    checkpointOne.SetcheckpointTracker(this);
```

24) At the top of the CheckpointTracker script create a new ‘private’ List Checkpoint called checkpointList.

```
private List<Checkpoint> checkpointList;
```

25) Then in the awake method initialize the list and add checkpointOne to the list at the end of the foreach statement

```
private void Awake()
{
    Transform CheckpointsTransform = transform.Find("Checkpoints");

    checkpointList = new List<Checkpoint>();
    foreach (Transform checkpointOneTransform in CheckpointsTransform)
    {
        Checkpoint checkpointOne = checkpointOneTransform.GetComponent<Checkpoint>();
        checkpointOne.SetcheckpointTracker(this);
        checkpointList.Add(checkpointOne);

    }
    nextCheckpointOneIndex = 0;
}
```

26) next make an int for the ‘nextCheckpointOneIndex’ and make the value of this int ‘0’ on awake.

```
private int nextCheckpointOneIndex;
```

```
private void Awake()
{
    Transform CheckpointsTransform = transform.Find("Checkpoints");

    checkpointList = new List<Checkpoint>();
    foreach (Transform checkpointOneTransform in CheckpointsTransform)
    {
        Checkpoint checkpointOne = checkpointOneTransform.GetComponent<Checkpoint>();
        checkpointOne.SetcheckpointTracker(this);
        checkpointList.Add(checkpointOne);

    }
    nextCheckpointOneIndex = 0;
}
```

27) now inside of ‘PlayerPassedCheckpoint’ use an if statement that checks if the Index of the list we just created is equal to ‘nextCheckpointOneIndex;’

use the index of the list that was just created which passes through checkpointOne.

```
if (checkpointList.IndexOf(checkpointOne) == nextCheckpointOneIndex)
```

28) If this is true then nextCheckpointOneIndex is equal to nextCheckpointIndex +1) % checkpointList.Count; This is used to set the count to zero once a lap is complete

29) Inside this if statement use debug.log to indicate that this is the correct checkpoint

```
public void PlayerPassedCheckpoint(Checkpoint checkpointone)
{
    if (checkpointList.IndexOf(checkpointone) == nextCheckpointOneIndex)
    {
        nextCheckpointOneIndex = (nextCheckpointOneIndex + 1) % checkpointList.Count;
        Debug.Log("Correct");
    }
}
```

30) and ‘else’ to this if statement and then use debug.log to indicate that any other checkpoint is incorrect.

```
else
{
    Debug.Log("Incorrect");
}
```

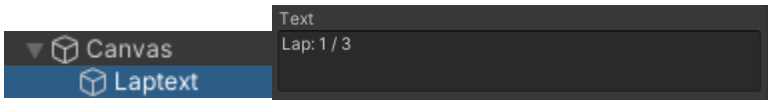
31) next create two ints, cpCount which is equal to zero and lpCount which is equal to one. CpCount is a checkpoint count and lpCount is a lapCount.

```
private int cpCount = 0;
private int lpCount = 1;
```

32) Next in the ‘public void’ Update method create an If statement that if cpCount is equal to checkpointList.Count + 1 then increase lpCount by one and sets cpCount equal to one.

```
if (cpCount == checkpointList.Count + 1)
{

    lpCount++;
    cpCount = 1;

}
```

33) Next return to Unity, right click the hierarchy and click UI> Text. Rename this text to ‘Laptext’ and write in the ‘Laptext’s’ textbox: ‘Lap: 1 / 3’ and position this in the top left hand corner.



34) return to the ‘CheckpointTracker’ Script and create a new ‘public’ GameObject called lapText. And at the top of the script write ‘using UnityEngine;’

```
public GameObject lapText;
```

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

35) Then below the first if statement in the update method create two new if statements. One if (lpCount == 2) and another if (lpCount == 3).

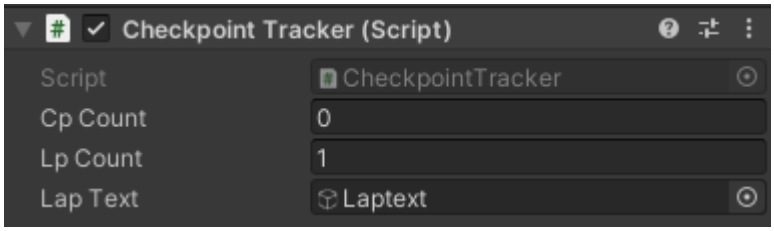
36) Inside both of these if statements use Getcomponent for the lapText and make it equal to "Lap: " + lpCount + " / 3". This is used to update the UI as a lap is completed.

```
if (lpCount == 2)
{

    lapText.GetComponent<Text>().text = "Lap: " + lpCount  + " / 3";
}
if (lpCount == 3)
{
    lapText.GetComponent<Text>().text = "Lap: " + lpCount + " / 3";
}
```

37) Then inside of the if statement located inside ‘PlayerPassedCheckpoint’ write ’cpCount++’

```
public void PlayerPassedCheckpoint(Checkpoint checkpointone)
{
    if (checkpointList.IndexOf(checkpointone) == nextCheckpointOneIndex)
    {
        nextCheckpointOneIndex = (nextCheckpointOneIndex + 1) % checkpointList.Count;
        Debug.Log("Correct");
        cpCount++;
```

38) Return to unity once again and then click and drag the Laptext into Lap Text slot located on the CheckpointTracker script on the Track GameObject.



Jamie Bovell