

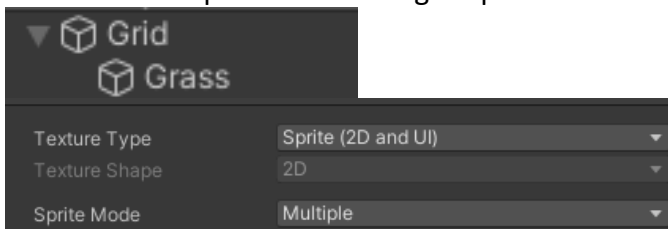
Microgame #6 Racing Game Progress Report

Name: Alan L. Perez

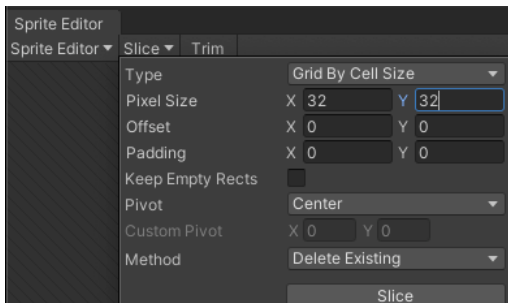
id: 004862867

Unity Play: <https://play.unity.com/mg/other/microgame-6-racing-game-cse-4410>

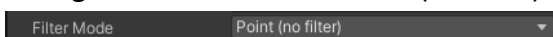
- 1.) Create a new project "Racing"
- 2.) Create folders under Assets: Prefabs, Scenes, Scripts, Sprites, Tiles
- 3.) Download RacingGame.zip from Canvas. Unzip, and drag sprites into Sprite folder. Put the Pride font under the Assets folder.
- 4.) Open the Sprites folder, select all the sprites and change "Pixel Per Unit" to 32.
- 5.) Then we go into GameObject -> 2D -> Tilemap. Now we have a "Grid", and "Tilemap". Rename "Tilemap" to "Grass". Then Go into the "Fantasy Tile Atlas Racing V1" sprite sheet in the inspector and change "Sprite Mode" to "Multiple".



- 6.) Then go into the Sprite Editor and click "Slice", then click "Grid By Cell Size". Then make "Pixel Size" X: 32 Y: 32. Then click "Slice" and "Apply".



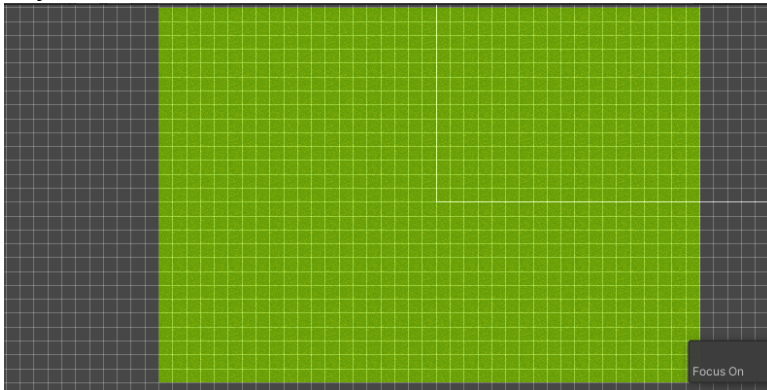
- 7.) Change "Filter Mode" to "Point (no filter)".



- 8.) Then we go into "Tile Palette" and "Create New Palette". We will name this "RacingBG". Then we save this under the "Tiles" folder.
- 9.) Then go into the "Tiles" folder and select everything under "Fantasy Tile Atlas Racing V1" and drag them into the "Tile Palette". We save this under "Tiles" folder.



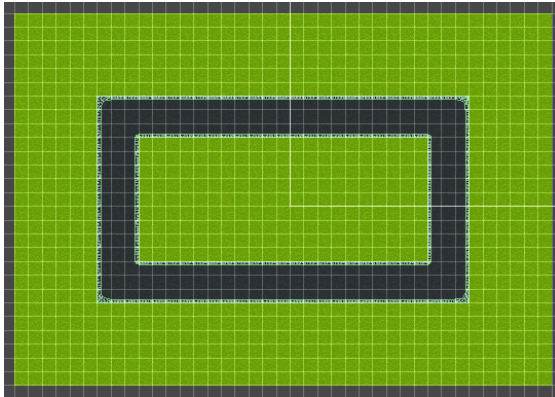
- 10.) Then we pick the green tile and add a grass background under the grass game object.



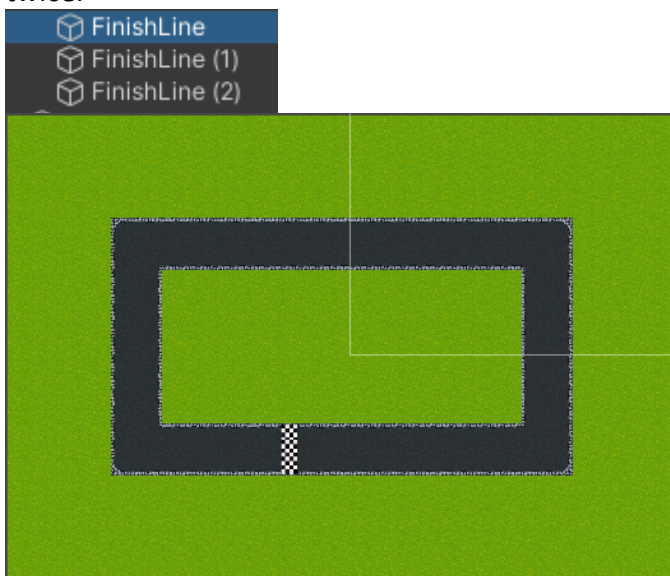
- 11.) Then we add another tile map and name it "Track".



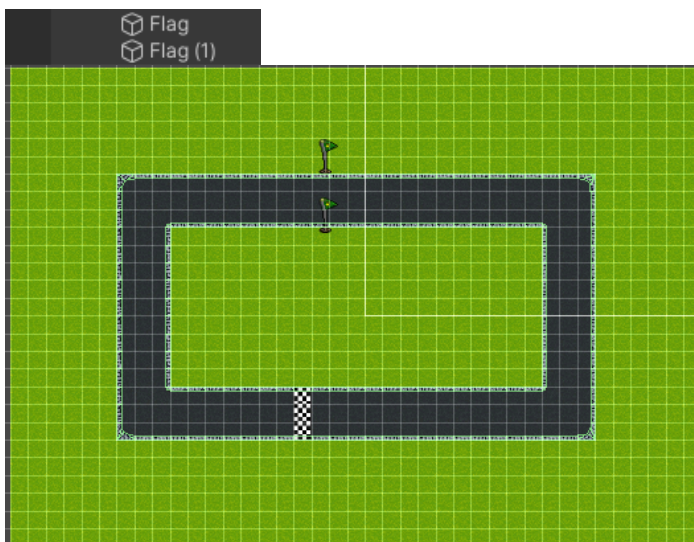
- 12.) Next we will create the track with the track pixels. We will adjust the tile's orientation if/when necessary.



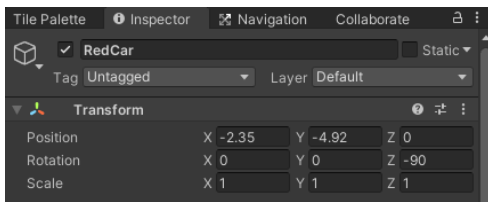
- 13.) Next we will add the finish line. We add the a finish line tile, and duplicated it twice.



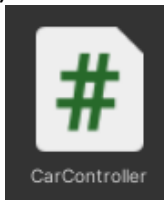
- 14.) Then we will add the check point flags under the Hierarchy. We will position them at the opposite side of the finish line.



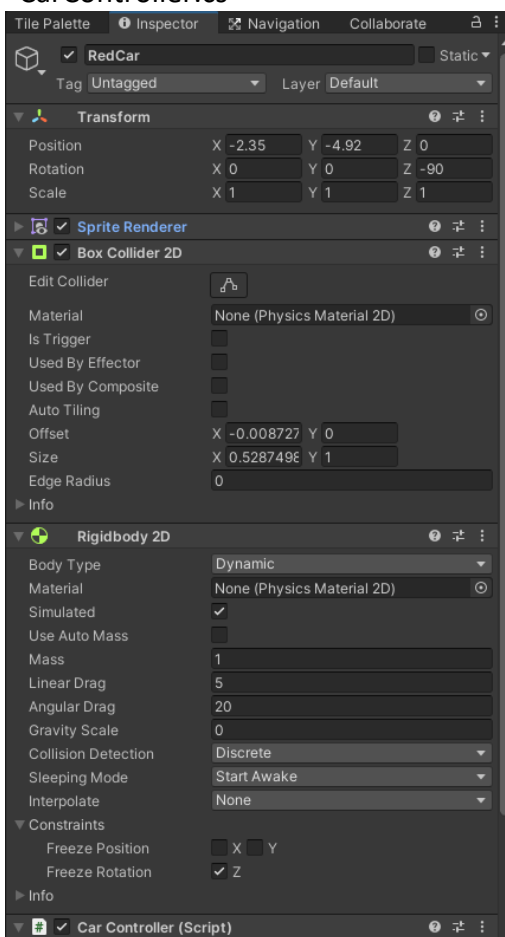
- 15.) Next we drag the “FantasyTileAtlasRacingV1_159” sprite to the Hierarchy. We will rename this “RedCar”. We will change its position, and orientation.



- 16.) Next we create the script file “CarController.cs”.



- 17.) We then add a “Box Collider 2D” to the car, and adjust the size. Next we will add a “Rigidbody2D” to the car. We set the “Linear Drag” to 5, “Angular Drag” to 10, and “Gravity” to zero. For constraints we “Freeze Rotation” of Z. Lastly, we attach “CarController.cs”



- 18.) Then we open “CarContoller.cs” we then add variables. Then we create the “Awake()” function and get the reference to the “Rigidbody2D”. Then in the “Update()” function we will take the input of the user. We will make a conditional statement saying

if the “input.x” is not zero the car will rotate. We will similarly do the same thing for “input.y” to make the car move forward and backward.

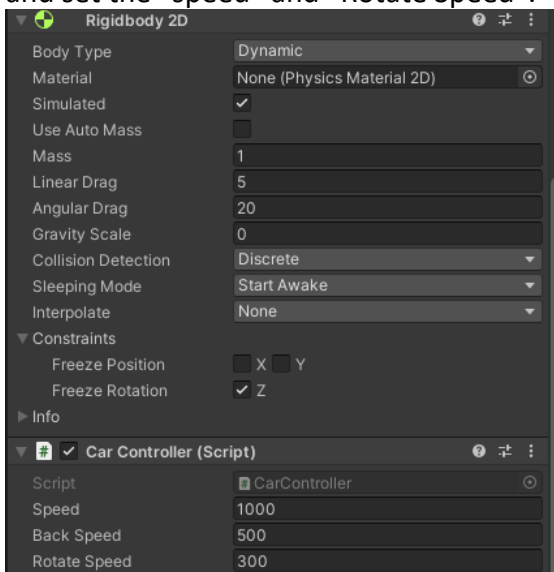
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CarController : MonoBehaviour
{
    Rigidbody2D carRigidbody;
    public float speed;
    public float backSpeed;
    public float rotateSpeed;
    Vector2 input;

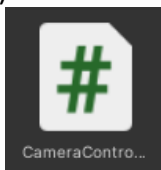
    private void Awake()
    {
        carRigidbody = GetComponent<Rigidbody2D>();
    }

    // Update is called once per frame
    void Update()
    {
        if (input.x != 0)
            transform.Rotate(0, 0, -rotateSpeed * Time.deltaTime * input.x);
        if (input.y > 0)
            carRigidbody.AddForce(transform.up * input.y * speed * Time.deltaTime);
    }
}
```

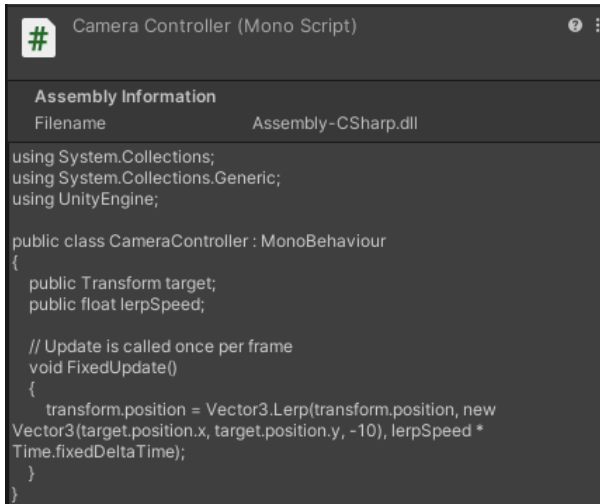
- 19.) We will then go back to Unity, and change the “Angular Drag” of the car to 20, and set the “speed” and “Rotate Speed”.



- 20.) Next we will create “CameraController.cs”



- 21.) Next we go into “CameraController.cs” and create variable. We then create the “FixedUpdate()” function. We will use this to transform the position of the camera.



```
# Camera Controller (Mono Script)

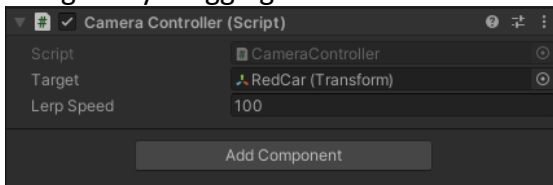
Assembly Information
Filename Assembly-CSharp.dll

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

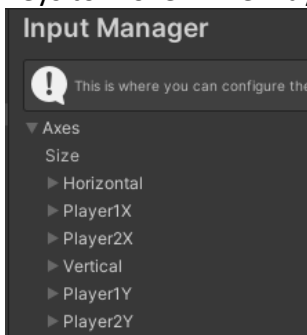
public class CameraController : MonoBehaviour
{
    public Transform target;
    public float lerpSpeed;

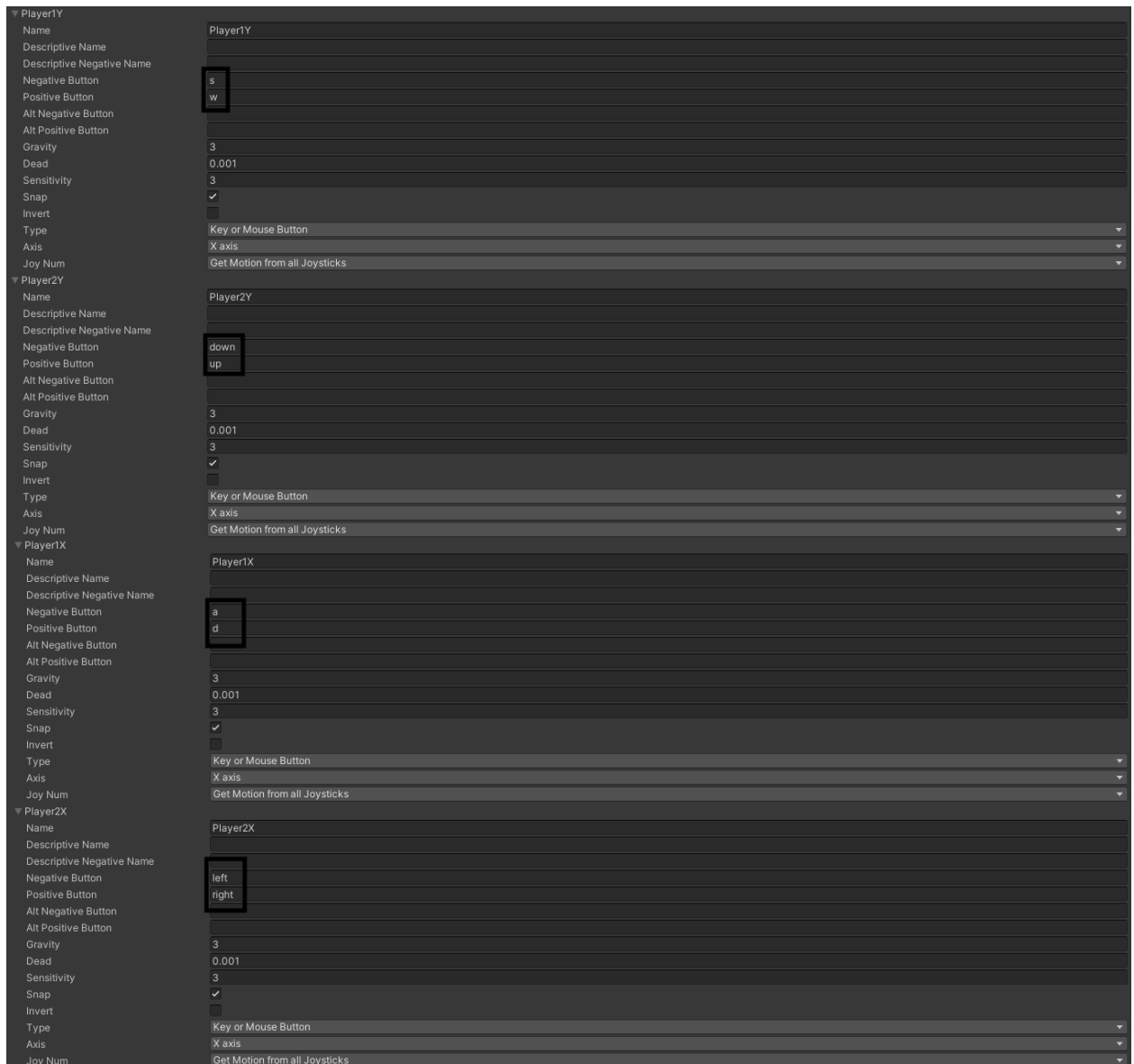
    // Update is called once per frame
    void FixedUpdate()
    {
        transform.position = Vector3.Lerp(transform.position, new
        Vector3(target.position.x, target.position.y, -10), lerpSpeed *
        Time.fixedDeltaTime);
    }
}
```

- 22.) Next we attach “CameraController.cs” to the “Main Camera” and we set the “Target” by dragging “RedCar” to it from the Hierarchy, and we set the “Lerp Speed”.



- 23.) Since we have two players, we must set the controls in “Project Settings” under “Input Manager”. We duplicate “Horizontal” as “Player1X” and “Player2X”. Then we duplicate vertical as “Player1Y” and “Player2Y”. Player one will be using the “WASD” keys to move while Player 2 will be using the “up down right left” keys to move.





24.) Next we go back into “CarController.cs”. We create variables, and we use them in the update function for “Input.GetAxis”.

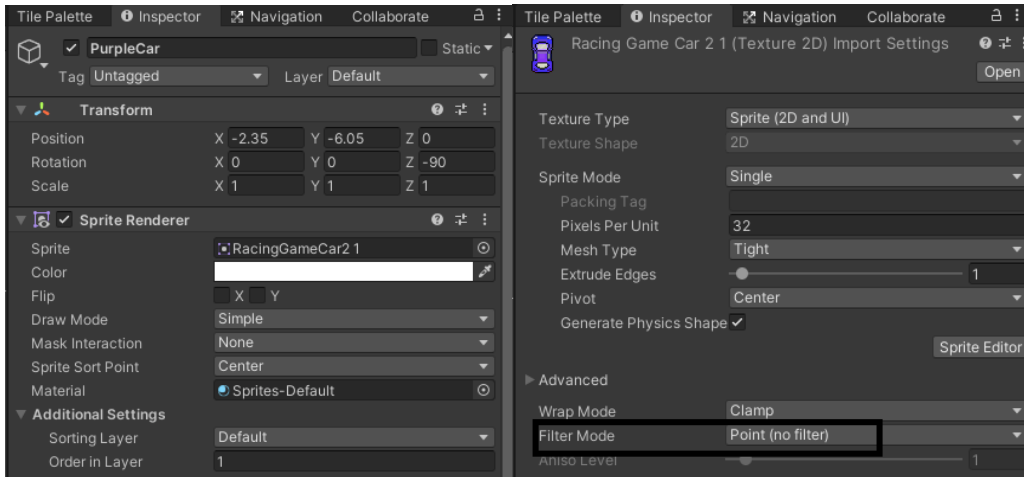
```
public string inputXName;
public string inputYName;

// Update is called once per frame
void Update()
{
    input = new Vector2(Input.GetAxis(inputXName),
        Input.GetAxis(inputYName));
}
```

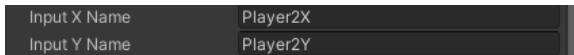
25.) Then we go into the “RedCar” game object under “CarController.cs” and we set the “Input X/Y Name” with “Player1X/Y”.

Input X Name	Player1X
Input Y Name	Player1Y

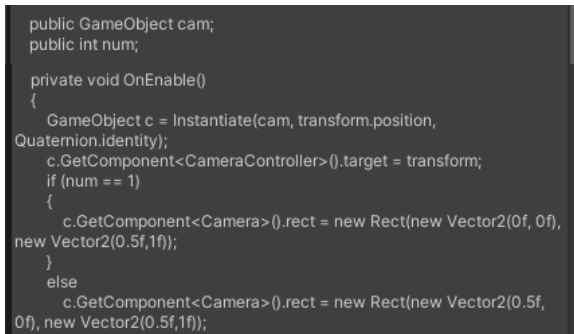
26.) Next we duplicate the “RedCar” game object and name it “PurpleCar”. We then go into our sprites and find “Racing Car 2 1” sprite and we drag it into the “Sprite” variable under “Sprite Renderer”. Then we select “Point (no filter)” under “Filter Mode”.



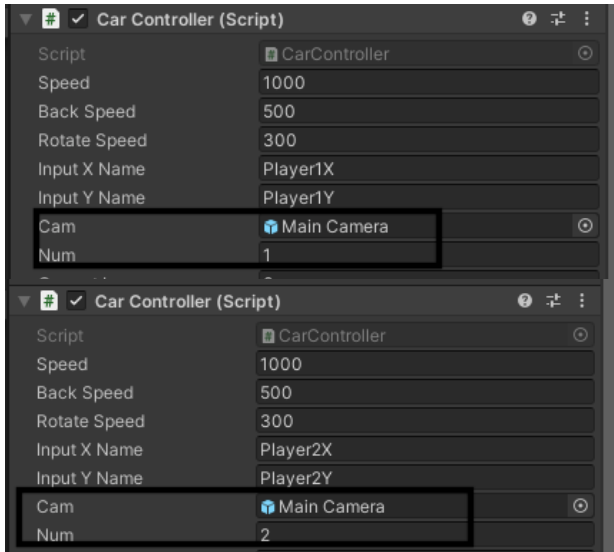
27.) Next under “CarController.cs” we will set “Input X/Y Name” with “Player2X/Y”.



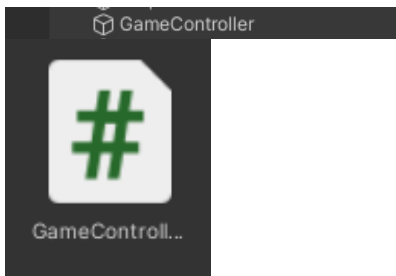
28.) Since there are two players in the game we want to separate cameras for each player. We go into “CarController.cs” and we declare variables. We then create an “Enable()” function to spawn the camera. Then we get the reference to the “CameraController.cs”.



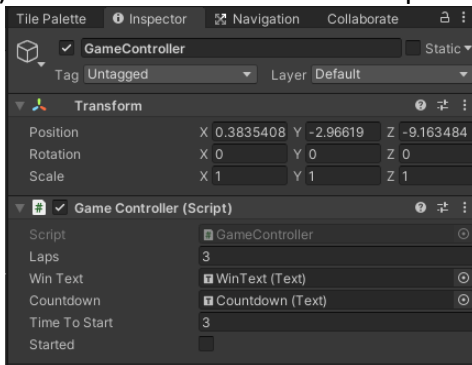
29.) Now we take the “Main Camera” and we save it as a prefab. Then we go to the “RedCar” game object and under the cam variable in “Car Controller” we drag the “Main Camera” Prefab to it. We do the exact same thing for the “PurpleCar”.



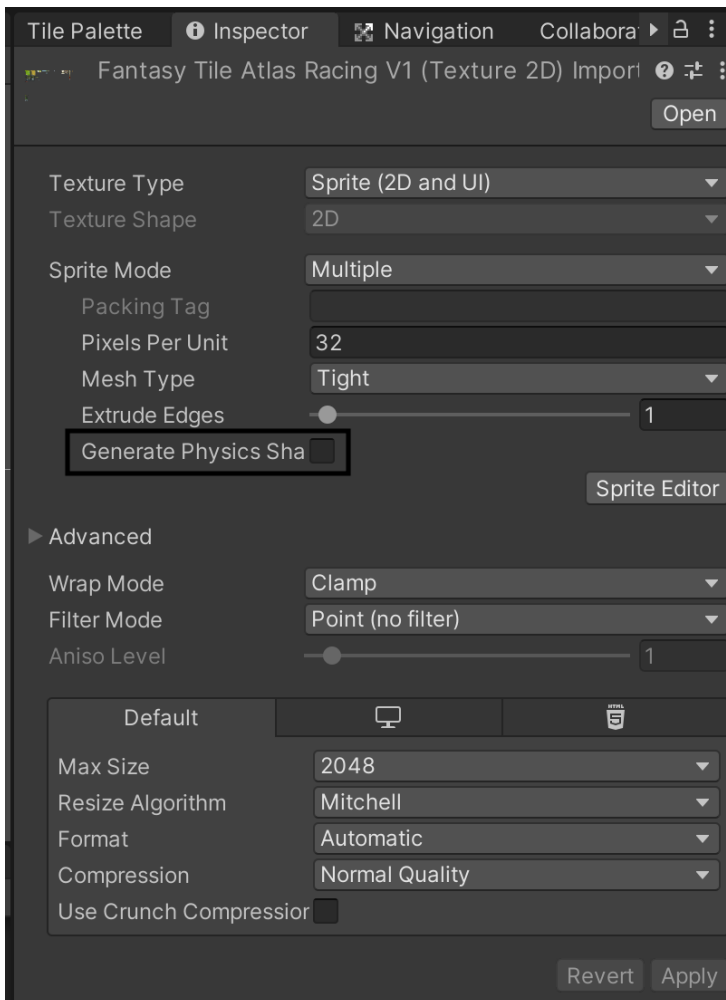
30.) Next we create the “GameController” game object and script file.



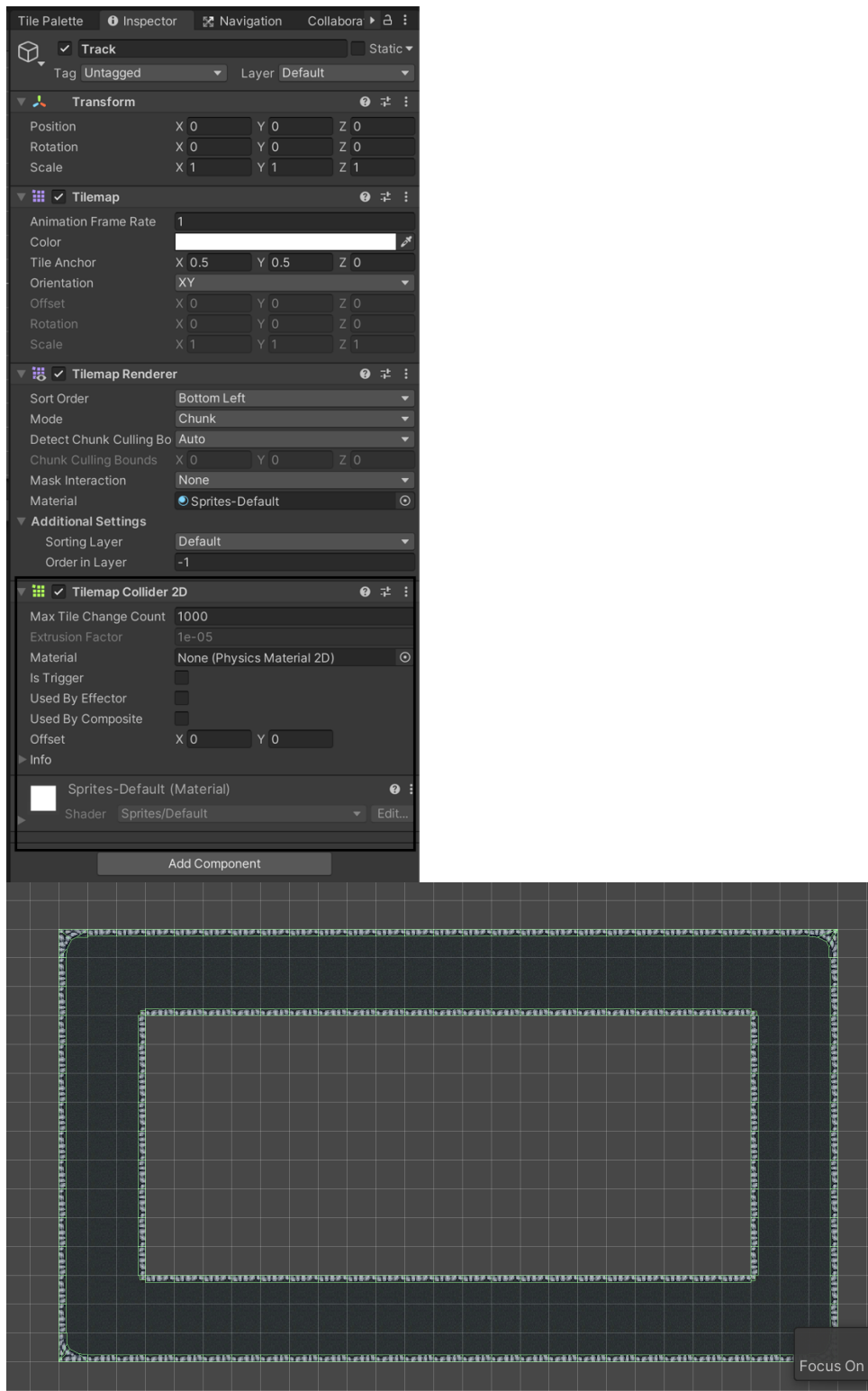
31.) Then we attach the script file to the game object.



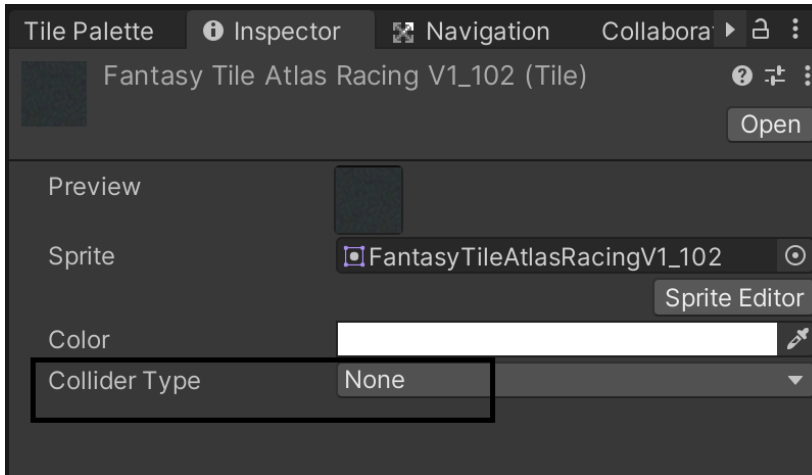
32.) Next we will add colliders to the edges of the track. We then click on “Fantasy Tile Atlas Racing V1” in the “Sprites” folder and uncheck “Generate Physics Shape”. Then click “Apply”.



- 33.) Then we go into the sprite editor and find the tiles we used for the track. Wherever there are boulders or the wall at the edge of the track we will outline a collider in that tile. We do this by clicking on "Sprite Editor" -> "Custom Physics Shape". Then we outline the tile. We then attach a "Tilemap Collider 2D" to the "Track" game object.



34.) Then we go into the “Tiles” folder and we find the tile we used for the middle of the track. We select it, go into the inspector under “Collider Type” we select “None”.



- 35.) We then go into “CarController.cs” and we make a variable to keep track of the laps completed and a “GameController” variable. Then in the “OnEnable()” function we initialized the first variable. Then we create the function “OnTriggerEnter2D()”. We want the finish line to be a “Trigger”. In the function we check if the tag is “Goal” which will be used for the finish line. If it hits the “Goal” it will increment the current lap.

```
[SerializeField]
int currentLap = 0;
GameController cont;

private void OnTriggerEnter2D(Collider2D collision)
{
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if(collision.gameObject.CompareTag("Goal"))
        {
            currentLap++;
        }
    }
}
```

- 36.) Then we go into “GameController.cs and we create the variable for laps. Which means the amount of laps that need to be completed.

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

public class GameController : MonoBehaviour
{
    public int laps;
}
```

- 37.) Under the “Awake()” function under “CarController.cs” we get the reference to the game controller.

```
private void Awake()
{
    carRigidbody = GetComponent<Rigidbody2D>();
    cont = FindObjectOfType<GameController>();
}
```

- 38.) We go back into “OnTriggerEnter2D()”. We add a conditional statement saying that if the amount of laps is greater than or equal to the amount of laps that need to be finished the game will be ended.

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if(collision.gameObject.CompareTag("Goal"))
    {
        currentLap++;
        if (currentLap >= cont.laps )
        {
            cont.EndGame(num);
        }
    }
}
```

- 39.) Then we go into the “GameController.cs” to define the “EndGame()” function. We also add the variables needed in the function. In the “Update()” function if the player presses any key once the game is over the game will start again.

```

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

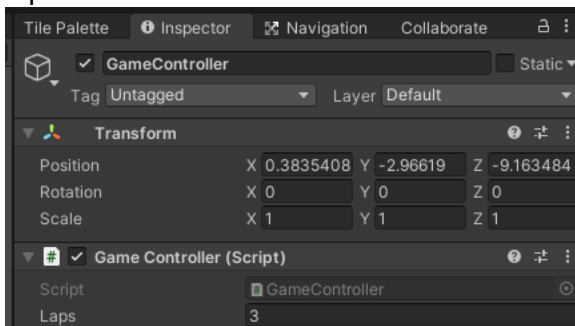
public class GameController : MonoBehaviour
{
    public int laps;
    public Text winText;
    bool endGame = false;

    // Update is called once per frame
    void Update()
    {
        if (endGame && Input.anyKeyDown)
            SceneManager.LoadScene("SampleScene");
    }

    public void EndGame(int num)
    {
        endGame = true;
        winText.gameObject.SetActive(true);
        winText.text = "Player " + num + " wins! Restart!";
    }
}

```

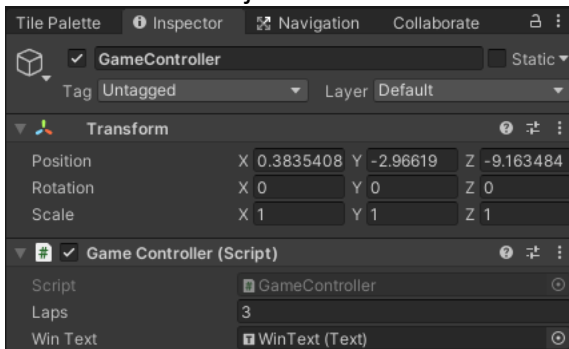
- 40.) Then we go into the “GameController” game object and we set the variable for laps.



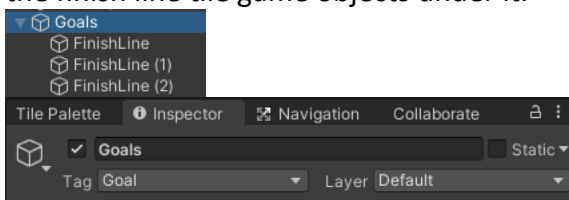
- 41.) Next we create we go into “UI”-> “Text”, and we create the “WinText”. We add the text for when the player wins. We change the position of the text. We use the “Pride” font, and we change font size. We set “Horizontal/Vertical Overflow” to “Overflow”. Then we make the color red. We center the text. We disable the “WinText”.



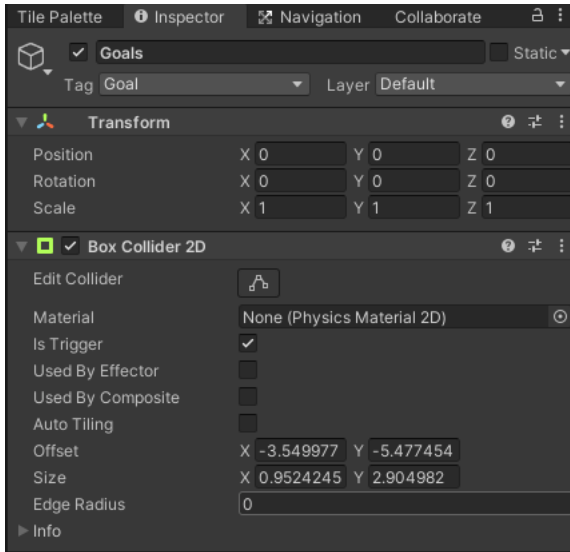
- 42.) We go back into the “GameController” game object and we set the “Win Text” to the “WinText” we just created.



- 43.) For the finish line tiles we create the tag “Goal” and tag them all as “Goal”. We do this by Creating a game object “Goals” and adding the tag “Goal”. We then place all the finish line tile game objects under it.



- 44.) In the “Goals” game object we attach a “Box Collider 2D” and make it “Is Trigger”.



- 45.) Next we will add a countdown for the game to begin in the “GameController.cs”. We add variables, and in the “Update()” function we will add a conditional statement to when the countdown can start, and when the game can begin.

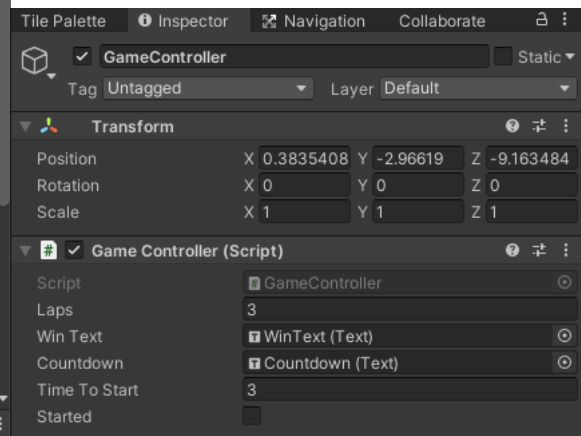
```
public Text countdown;
public float timeToStart = 3f;
public bool started = false;

// Update is called once per frame
void Update()
{
    if(timeToStart > 0)
    {
        timeToStart -= Time.deltaTime;
        countdown.text = Mathf.RoundToInt(timeToStart).ToString();
    }
    else
    {
        started = true;
        countdown.gameObject.SetActive(false);
    }
}
```

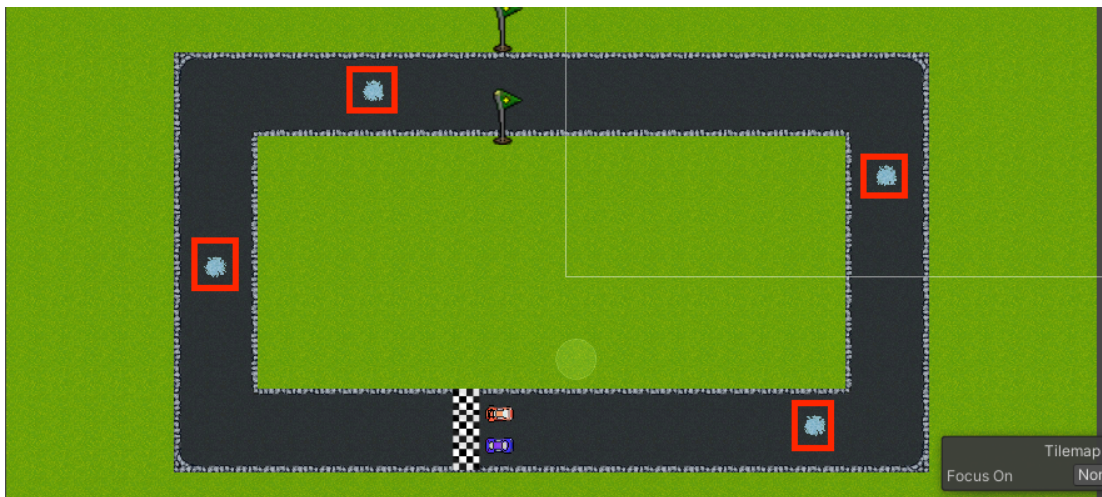
- 46.) Next we go into “CarController.cs” and we go into the “Update()” function and we add a conditional statement that if the controller has started the game can move.

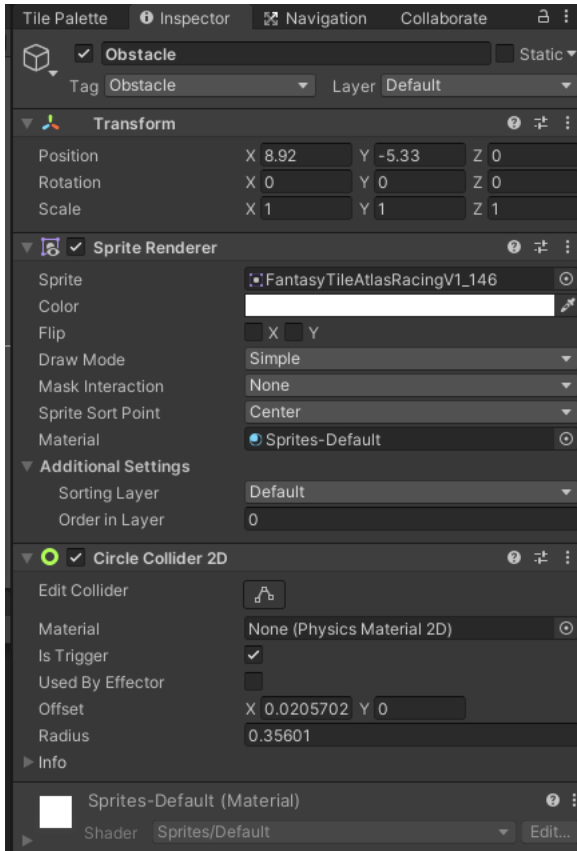
```
// Update is called once per frame
void Update()
{
    if (cont.started && !Islicked)
    {
        input = new Vector2(Input.GetAxis(inputXName),
        Input.GetAxis(inputYName));
        if (input.x != 0)
            transform.Rotate(0, 0, -rotateSpeed * Time.deltaTime * input.x);
        if (input.y > 0)
            carRigidbody.AddForce(transform.up * input.y * speed *
            Time.deltaTime);
    }
}
```

- 47.) Then we duplicate the “WinText” game object, and rename it “Countdown”. Then we change the position of it. We then change the text to 3. To start the countdown. Afterwards we go into “GameController” game object and set “Start To Time” as 3 and the “Countdown” variable as “Countdown”.



- 48.) Then we add obstacles to the game. We take the water sprite and we duplicate it until we get four of them throughout the course. We name the "Obsactle". We create a tag "Obstacle". We then attach a "Circle Collider 2D". We adjust the collider size. We then set "Is trigger".





49.) Next we go into the “CarController.cs” and we add variables. Then we go into “OnTriggerEnter2D()” function and we create a conditional statement for if the car hits the water.

```
float cools;
public float slickTimer;
public bool slicked = false;
public float slickRotation;
private void OnTriggerEnter2D(Collider2D collision)
{
    if(collision.gameObject.CompareTag("Goal") && hitCheckPoints)
    {
        currentLap++;
        if (currentLap >= cont.laps)
            cont.EndGame(num);
        hitCheckPoints = false;
    }

    if (collision.gameObject.CompareTag("Obstacle"))
    {
        slickedDirection = transform.up;
        cools = slickTimer;
        slicked = true;
    }
}
```

50.) Then we go to the update function and say if the car is not “slicked” then we can move the car.

```
// Update is called once per frame
void Update()
{
    if (cont.start && !slicked)
    {
        input = new Vector2(Input.GetAxis(inputXName),
        Input.GetAxis(inputYName));
        if (input.x != 0)
            transform.Rotate(0, 0, -rotateSpeed * Time.deltaTime * input.x);
        if (input.y > 0)
            carRigidbody.AddForce(transform.up * input.y * speed *
            Time.deltaTime);
    }
}
```

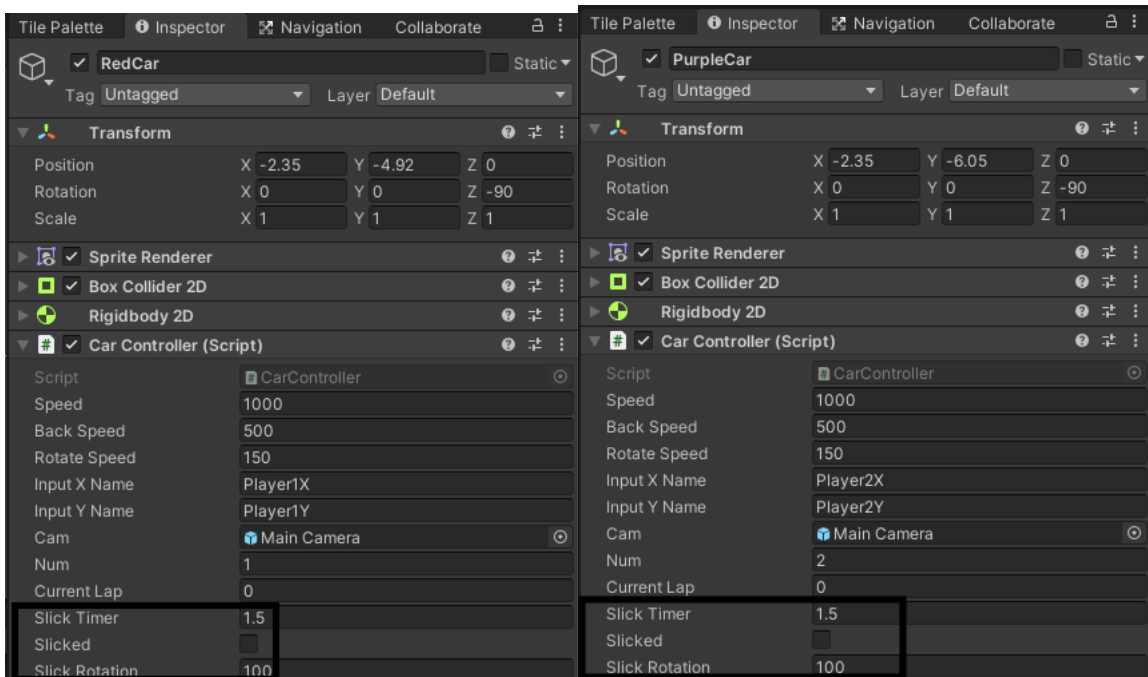
- 51.) Then we add a conditional statement for if the car is slicked, the car cannot move. The car will rotate. Then we reset "slicked"

```

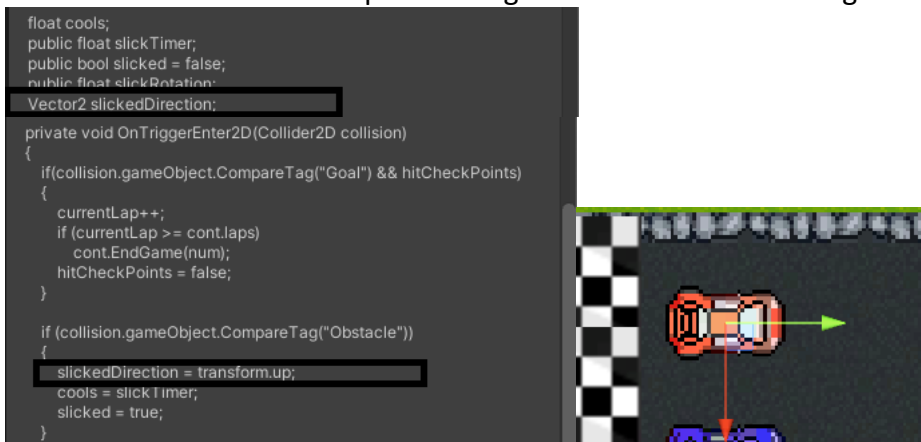
if (slicked)
{
    carRigidbody.AddForce(slickedDirection * backSpeed *
    Time.deltaTime);
    transform.Rotate(0, 0, slickRotation * Time.deltaTime);
    if (cools <= 0) slicked = false;
}

```

- 52.) Then for both cars in the inspector we set "Slicked Timer" and "Slick Rotation" under "CarController.cs".



- 53.) Then we want to check the slicked direction which is a Vector2 variable. We then go into "OnTriggerEnter2D()" Under the "Obstacle" conditional statement. We make it so the slicked direction is equal to the green arrow vector in the game object.



- 54.) Then in the "Update()" function we change "transform.up" in the "AddForce()" to "slickDirection".

```
// Update is called once per frame
void Update()
{
    if (cont.started && !slicked)
    {
        input = new Vector2(Input.GetAxis(inputXName),
        Input.GetAxis(inputYName));
        if (input.x != 0)
            transform.Rotate(0, 0, -rotateSpeed * Time.deltaTime * input.x);
        if (input.y > 0)
            carRigidbody.AddForce(transform.up * input.y * speed *
            Time.deltaTime);
        if (input.y < 0)
            carRigidbody.AddForce(transform.up * input.y * backSpeed *
            Time.deltaTime);
    }
    if (slicked)
    {
        carRigidbody.AddForce(slickedDirection * backSpeed *
        Time.deltaTime);
        transform.Rotate(0, 0, slickRotation * Time.deltaTime);
        if (cools <= 0) slicked = false;
    }
}
```

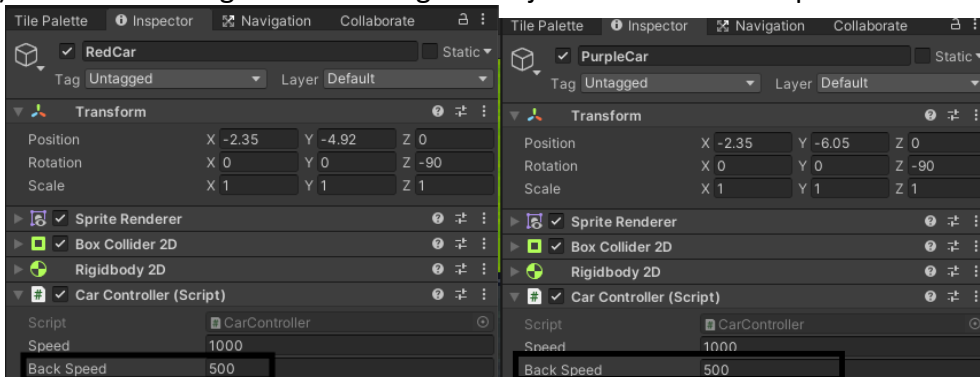
55.) Next we will add “backspeed” so that when the car goes backwards it will be slower than when it is going forward. Also, if the car is slicked, it will also use “backspeed” we implement this in “Update()”.

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

public class CarController : MonoBehaviour
{
    Rigidbody2D carRigidbody;
    public float speed;
    public float backSpeed;
    public float rotateSpeed;
    Vector2 input;

    // Update is called once per frame
    void Update()
    {
        if (cont.started && !slicked)
        {
            input = new Vector2(Input.GetAxis(inputXName),
            Input.GetAxis(inputYName));
            if (input.x != 0)
                transform.Rotate(0, 0, -rotateSpeed * Time.deltaTime * input.x);
            if (input.y > 0)
                carRigidbody.AddForce(transform.up * input.y * speed *
                Time.deltaTime);
            if (input.y < 0)
                carRigidbody.AddForce(transform.up * input.y * backSpeed *
                Time.deltaTime);
        }
        if (slicked)
        {
            carRigidbody.AddForce(slickedDirection * backSpeed *
            Time.deltaTime);
            transform.Rotate(0, 0, slickRotation * Time.deltaTime);
            if (cools <= 0) slicked = false;
        }
    }
}
```

56.) Then we go into the car game objects and set “Back Speed”.



- 57.) Then we go into “CarController.cs” and we add two different linear drags, and a current drag. We initialize the current drag in “OnEnable()”. Then in the “Update()” we have a conditional statement checking what the “currentDrag” will be. The “Lerp Drag” is just to make it look smooth.

```
public float regDrag;
public float slickedDrag;
float currentDrag;
public float dragLerp;

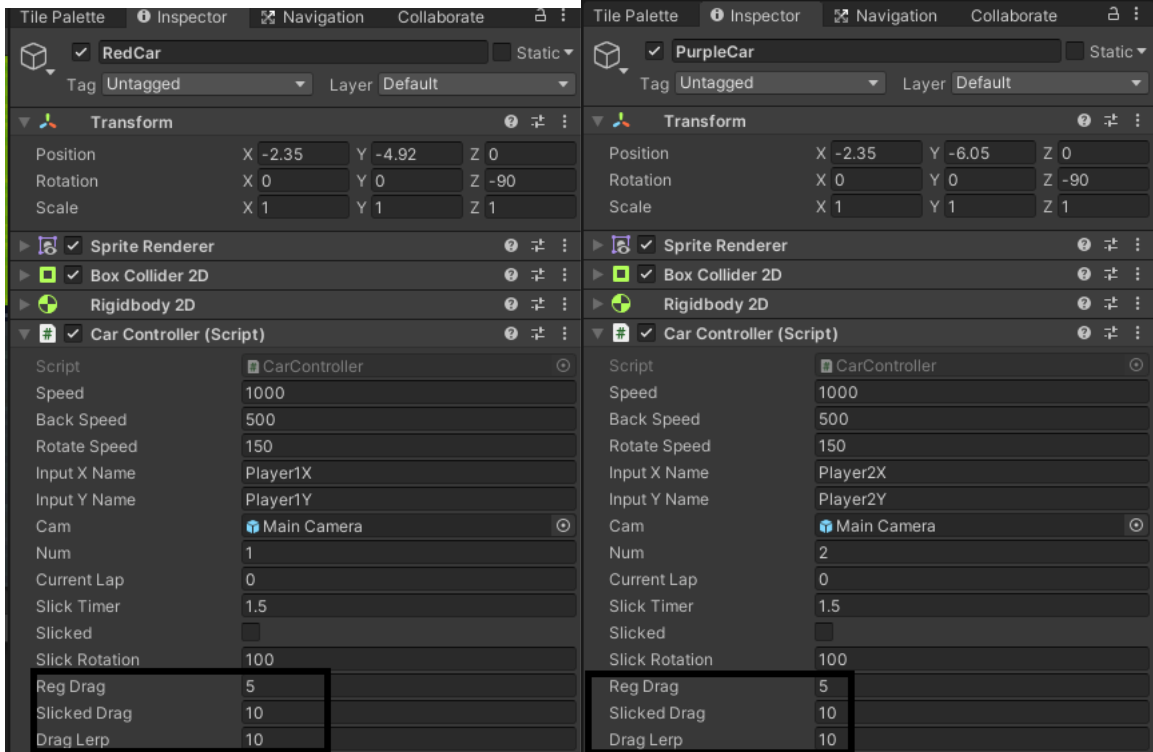
private void OnEnable()
{
    GameObject c = Instantiate(cam, transform.position,
Quaternion.identity);
    c.GetComponent<CameraController>().target = transform;
    if (num == 1)
    {
        c.GetComponent<Camera>().rect = new Rect(new Vector2(0f, 0f),
new Vector2(0.5f, 1f));
    }
    else
    {
        c.GetComponent<Camera>().rect = new Rect(new Vector2(0.5f,
0f), new Vector2(0.5f, 1f));
    }

    currentLap = 0;
    currentDrag = regDrag;
}

// Update is called once per frame
void Update()
{
    if (cont.started && !slicked)
    {
        input = new Vector2(Input.GetAxis(inputXName),
Input.GetAxis(inputYName));
        if (input.x != 0)
            transform.Rotate(0, 0, -rotateSpeed * Time.deltaTime * input.x);
        if (input.y > 0)
            carRigidbody.AddForce(transform.up * input.y * speed *
Time.deltaTime);
        if (input.y < 0)
            carRigidbody.AddForce(transform.up * input.y * backSpeed *
Time.deltaTime);
    }
    if (slicked)
    {
        carRigidbody.AddForce(slickedDirection * backSpeed *
Time.deltaTime);
        transform.Rotate(0, 0, slickRotation * Time.deltaTime);
        if (cools <= 0) slicked = false;
    }
    if (cools > 0) cools -= Time.deltaTime;

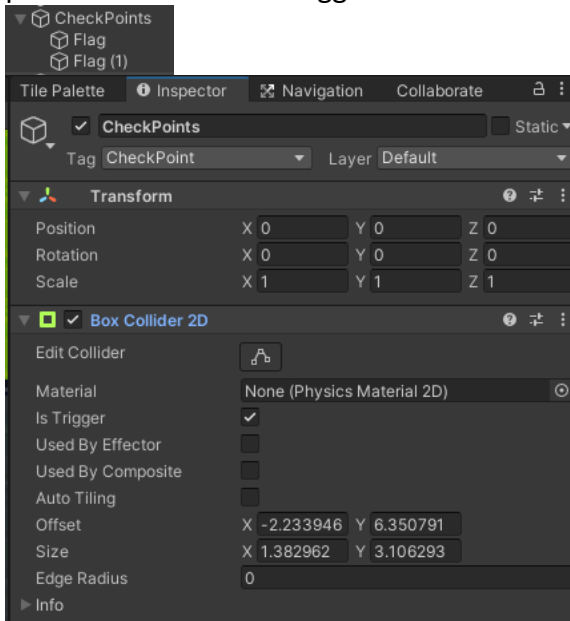
    currentDrag = slicked ? slickedDrag : regDrag;
    carRigidbody.drag = Mathf.Lerp(carRigidbody.drag, currentDrag,
dragLerp * Time.deltaTime);
}
```

- 58.) We set “Reg Drag”, “Slick Drag”, and “Drag Lerp” for both the “Red/PurpleCar” under the “Car Controller (Script)”.



59.) If the car goes backward, we don't want the lap to increment therefore, we will add a checkpoint so the lap can only increment after the fact. Also, once the lap increments we will reset the checkpoint.

60.) Similarly to the finish line we create an empty game object "CheckPoints" and place the flags that are already in the Hierarchy and make them a child of "Checkpoints". We set "CheckPoints" position as zeros for X,Y, and Z. We then make a tag "CheckPoint" and Tag "CheckPoints" with it. We also attach a "Box Collider 2D" and position it. We set "IsTrigger".



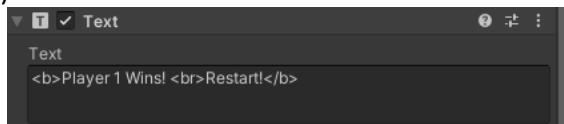
- 61.) Then we go into "CarController.cs" we add variables, and go into "TriggerEnter2D()". Here if the car hits the collider of "CheckPoints" then "hitCheckPoints" will be equal to true. We also reset "hitCheckPoints" in the "Goal" conditional statement.

```
public bool hitCheckPoints = false;
private void OnTriggerEnter2D(Collider2D collision)
{
    if(collision.gameObject.CompareTag("Goal") && hitCheckPoints)
    {
        currentLap++;
        if (currentLap >= cont.laps)
            cont.EndGame(num);
        hitCheckPoints = false;
    }

    if (collision.gameObject.CompareTag("Obstacle"))
    {
        slickedDirection = transform.up;
        cools = slickTimer;
        slicked = true;
    }

    if (collision.gameObject.CompareTag("CheckPoint"))
    {
        hitCheckPoints = true;
    }
}
```

- 62.) Then in the "WinText" we add some HTML to the text.



The screenshot shows the Unity Hierarchy window with a 'Text' object selected. The 'Text' component's 'Text' field contains the HTML code: `Player 1 Wins!
Restart!`. The text is displayed in a bold font with a line break between 'Wins!' and 'Restart!'.