CT3536: Games Programming

Project Report

Andrew Hayes
Student ID: 21321503

Contents

1 Introduction & Instructions **1**

2 Development Approach **3**

3 Source Code **5**

4 Third Party Assets Used **12**

1 Introduction & Instructions

This game is a single-player racing game in which the player races against a computer-controlled car on a race track. Each race consists of 3 laps, with the first "player" (i.e. the actual player or the computer-controlled car) to cross the finish line being declared the winner. All in all, the game is quite self-explanatory: it's a player vs computer car racing game, with the usual controls and behaviours one would expect. All of the art and sound assets in this game were originally sourced from a third party and were not created by me, although adjustments and edits were made to suit my use case.

Figure 1: Starting Point of Game

The main menu consists of the options "Start Game" or "Quit". When a race is started, the track and cars appear, but are immobile until the race has started. Three air horns are sounded at one second intervals to count down to the race start, followed by an air horn to indicate "Go". From there, the `WASD` keys can be used to control the player car to drive it around the track, by turning its wheel colliders to push it forward.

Figure 2: Start Menu

The computer-controlled car travels between a set of waypoints on the track. To give the player an edge over the computer, the NPC car will stop if it is on a collision course with the player's car, i.e. if the player's car is both close to and directly in from of the computer-controlled car.

The game ends when one "player" has completed three laps. When the game ends, the player is sent back to the main menu, and a message is displayed indicating whether they won or lost.

Figure 3: "Win" Screen

2 Development Approach

The first thing I did when I decided to start developing a racing game was to look for a free racing car prefab that I could use, and the first one that appeared on the Unity asset store seemed ideal:

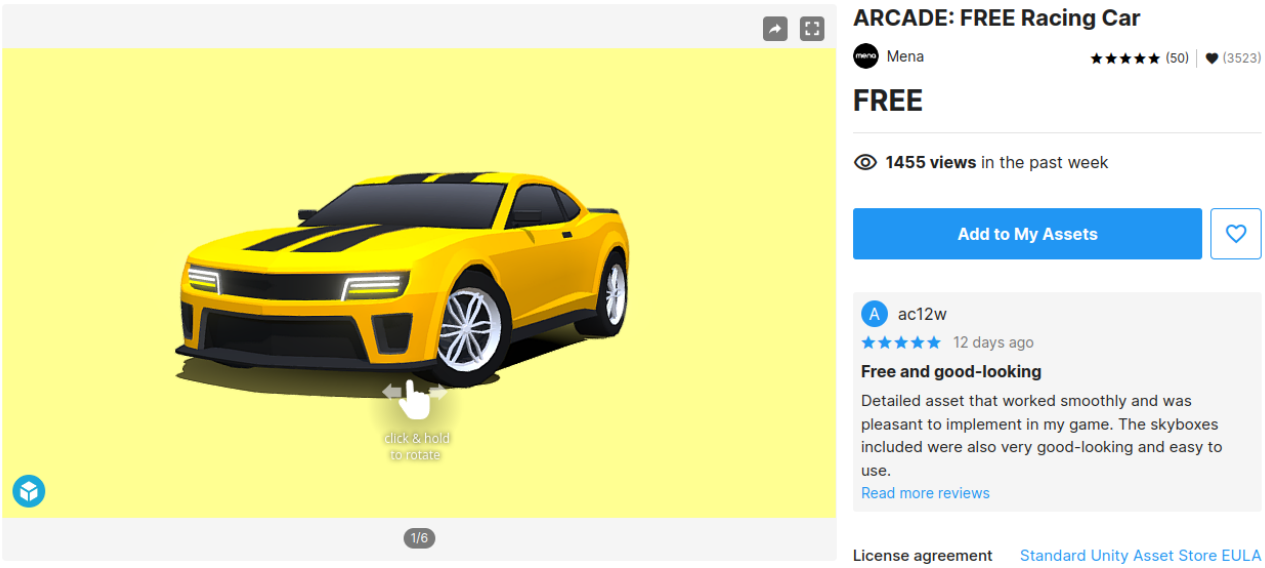

Figure 4: Racecar Prefab

Originally, I was going to just apply a physics force to the car object to make it go forwards, but I noticed that this car asset came with individual colliders and transforms for each wheel, allowing them to be manipulated independently. I later discovered that this was standard practice for cars in Unity, and the rotation of the wheel was typically used to propel the car. As this was definitely better than just applying a physics force to the back of the car, I opted to use the wheel rotation to propel the car. A mistake that I made here was to make the car front-wheel drive: as I don't drive myself, I just picked one at random when I was developing, ignorant of how uncommon it was for race cars to be front-wheel drive. The plus side of this was that it made the car lend itself quite easily to "drifting"-like motions.

When it came to making the main camera follow the car, I had a lot of trouble getting the camera to realise that the car's position had changed. For some reason, no matter what changes I made, the camera seemed to think that the player's car was always at (0,0,0). To get around this problem easily, I instead decided to make the car summon the camera to it, which ensured that the camera was always getting updated as the car's position changed. The drawback of this was that it led to less clean code architecture: the camera was being controlled by a script that was not attached to it, meaning that if I wanted to expand the game to allow the camera to move around to view other things than the car, development would be difficult. However, since I didn't plan on doing that for this project, I was prepared to sacrifice some code structure for quickly working code.

To get the car to make motor noises as one would expect, I initially considered getting several different car engine noise samples at different RPMs and mixing between them as appropriate. However this struck me as needlessly complicated, and it was prohibitively difficult to find a set of free-to-use engine noise samples that also looped cleanly: it proved difficult enough to find one decent engine noise sample. Instead, I decided to use one engine noise sample that would loop endlessly, and to just pitch it up or down to correspond the to the hypothetical "revs" of the engine. I still found it difficult to find a clean, looping sample of a car engine, but I eventually found one that was "close enough". It didn't loop perfectly though, so I edited it using Audacity to make it shorter, get rid of the silence, etc.

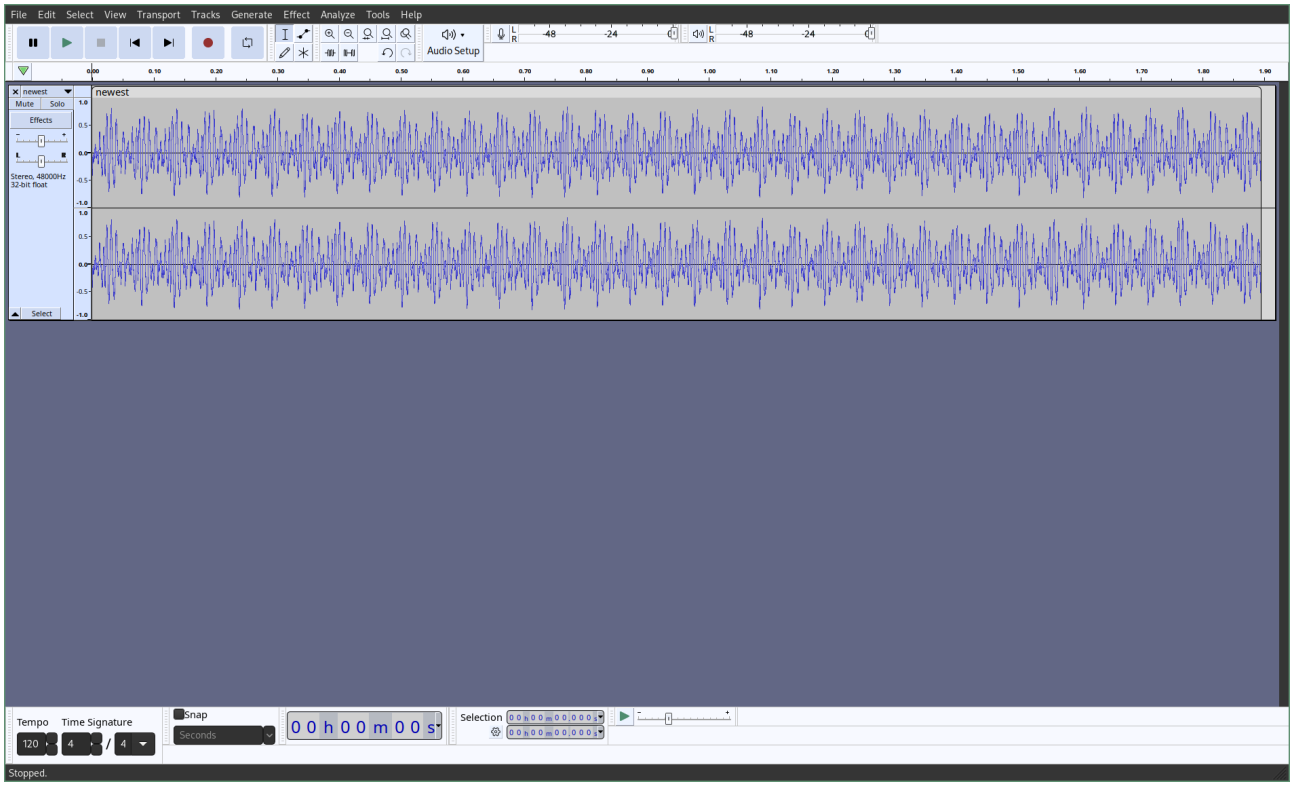

Figure 5: Editing the Engine Noise Sample in Audacity

To design the racetrack structure, I made use of a tool called RoadArchitect which provides a number of road structure prefabs and allows you to configure them into a road system consisting of several nodes. To avoid any confusion, I want to state clearly here that this tool does not execute code during the game loop nor is any of its code included in my game: a static road structure was created and designed using this tool, which is then used as a (code-free) GameObject in the game.

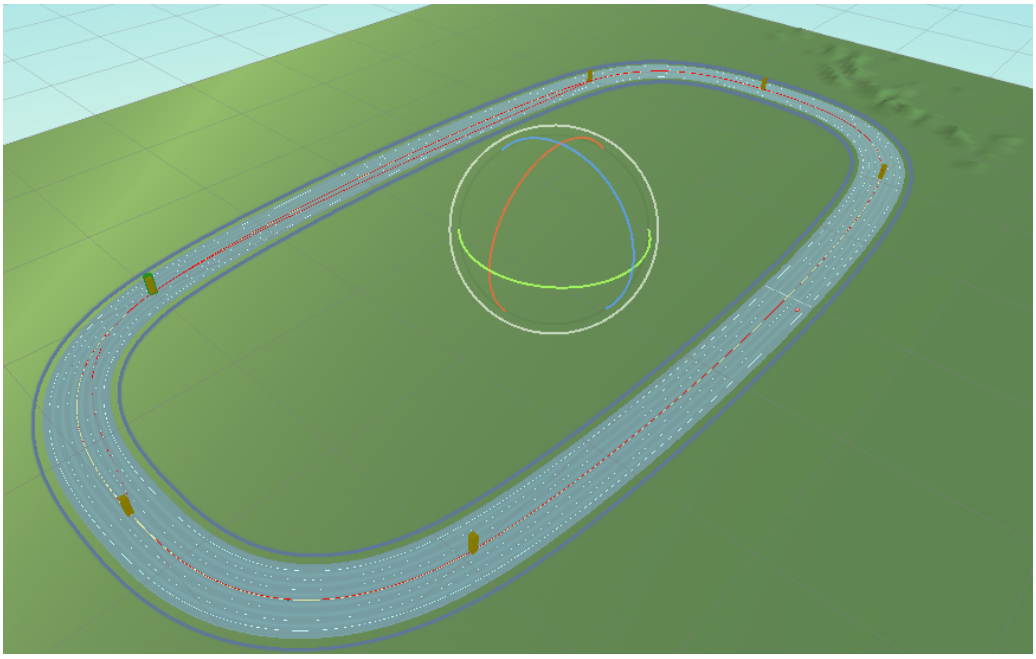

Figure 6: Racetrack Structure

The node-based structure of the road proved to be useful for creating the computer-controlled car; the car navigates the track by heading towards a series “waypoints”, many of which are the nodes of the road itself. To make the race

more interesting, the player is afforded an advantage: it can force the computer-controlled car to stop if the computer-controlled car is about to collide with it. To make this possible, the computer-controlled car is constantly casting rays out in its forward direction and checking to see if they hit any objects tagged "Player": if they do, it does not proceed until its raycasts are not hitting the player. These raycasts are limited to a short distance so as not to render the computer-controlled car entirely helpless.

Finally, different touches were added to the game to make it more cohesive: trees were painted, terrain was moulded, and horn & buzzer sound effects were added to the countdown before the race begins. A main menu was added which allows the player to quit or race again, which also doubles as the announcement area of whether the player won or lost the race.

3 Source Code

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

// script to control the sound of the car's engine
public class EngineSound : MonoBehaviour
{
    public AudioClip engineClip;
    public float minPitch = 0.8f;
    public float maxPitch = 1.2f;
    public float minVolume = 0.5f;
    public float maxVolume = 1.0f;
    private AudioSource audioSource;

    void Start()
    {
        audioSource = gameObject.AddComponent<AudioSource>();
        audioSource.clip = engineClip;
        audioSource.loop = true;
        audioSource.Play();
    }

    void Update()
    {
        float speed = Mathf.Abs(Input.GetAxis("Vertical"));

        float pitch = Mathf.Lerp(minPitch, maxPitch, speed);
        float volume = Mathf.Lerp(minVolume, maxVolume, speed);

        audioSource.pitch = pitch;
        audioSource.volume = volume;
    }
}
```

Listing 1: `EngineSound.cs`

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

```
using UnityEngine.SceneManagement;
using TMPro;

public class GameManager : MonoBehaviour
{
    public TextMeshProUGUI textMesh;
    public GameObject playerCar;
    public Vector3 startingPosition;
    public static bool isRaceOn = false;
    public AudioClip buzzer;
    public AudioClip airHorn;
    private static AudioSource audioSource;

    void Start()
    {
        Instantiate(playerCar, startingPosition, Quaternion.identity);

        audioSource = gameObject.AddComponent<AudioSource>();
        audioSource.clip = buzzer;

        StartCoroutine(Countdown());
    }

    void Update() {
        // escape to main menu
        if (Input.GetKeyDown(KeyCode.Escape)) {
            isRaceOn = false;
            SceneManager.LoadScene(0);
        }
    }

    // countdown for race to start
    IEnumerator Countdown() {
        WaitForSeconds onesec = new WaitForSeconds(1);  // reuse waitforseconds for efficiency

        for (int i = 3; i > 0; i--) {
            // play horn noise for each pip
            audioSource.Play();
            yield return onesec;
        }

        // switch audio to airhorn and play to signify go
        audioSource.clip = airHorn;
        audioSource.Play();

        isRaceOn = true;
    }

    // static method for one of the cars to call when they have completed 3 laps
    public static void EndRace(string winner) {
        isRaceOn = false;

        // if winner is the player, load the win screen
```

```
        if (winner == "Player") {
            SceneManager.LoadScene(2);
        }
        // else load the lose screen
        else {
            SceneManager.LoadScene(3);
        }
    }
}
```

Listing 2: `GameManager.cs`

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

// script to handle updating the lap indicator in the UI
public class LapDisplay : MonoBehaviour
{
    public TextMeshProUGUI textMesh;
    private static int lapCount = 0;

    // static method to allow player car to update the lap count
    public static void IncrementCount() {
        lapCount++;
    }

    void Update() {
        textMesh.text = "Lap " + lapCount + "/3";
    }
}
```

Listing 3: `LapDisplay.cs`

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

// car controller for the NPC car
public class NPCCar : MonoBehaviour
{
    // wheel transforms and colliders for driving animation
    public Transform frontLeftWheelTransform;
    public Transform frontRightWheelTransform;
    public Transform rearLeftWheelTransform;
    public Transform rearRightWheelTransform;

    public WheelCollider frontLeftWheelCollider;
    public WheelCollider frontRightWheelCollider;
    public WheelCollider rearLeftWheelCollider;
```

```
18      public WheelCollider rearRightWheelCollider;
19
20      private int lapCount = 0;
21
22      // array of waypoints for car to navigate track
23      public Transform[] waypoints;
24
25      public float speed = 30f;
26      public float rotationSpeed = 5f;
27
28      private int waypoint = 0;   // next waypoint
29
30      // variables for handling stopping if about to collide with player
31      private Ray ray;            // ray to reuse for raycasting
32      public float maxDistanceToPlayer = 15f;
33
34      void Start() {
35          ray = new Ray();
36      }
37
38      void Update()
39      {
40          if (GameManager.isRaceOn) {
41              if (waypoint < waypoints.Length)
42              {
43                  ray.origin = transform.position;
44                  ray.direction = transform.TransformDirection(Vector3.forward);
45
46                  // only proceed if route is not blocked by player
47                  if (Physics.Raycast(ray, out RaycastHit hit, maxDistanceToPlayer)) {
48                      if (!hit.collider.CompareTag("Player")) {
49                          DriveToNextWaypoint();
50                      }
51                      else {
52                          Debug.Log("not proceeding because blocked by player");
53                      }
54                  }
55                  else {
56                      DriveToNextWaypoint();
57                  }
58              }
59              // if all waypoints have been visited, set next waypoint to the 0th one
60              // keep doing circuit until race ends
61              else
62              {
63                  waypoint = 0;
64              }
65          }
66      }
67
68      void DriveToNextWaypoint()
69      {
70          // direction to next waypoint
```

```
        Vector3 direction = waypoints[waypoint].position - transform.position;
        direction.y = 0f;

        // look at next waypoint
        Quaternion rotation = Quaternion.LookRotation(direction);
        transform.rotation = Quaternion.Slerp(transform.rotation, rotation, rotationSpeed *
        ↪  Time.deltaTime);

        transform.Translate(Vector3.forward * speed * Time.deltaTime);

        // if within 3m of goal waypoint, move on to next one
        if (Vector3.Distance(transform.position, waypoints[waypoint].position) < 3f)
        {
            waypoint++;
        }

        // rotate wheels for driving animation
        Vector3 pos;
        Quaternion rot;

        frontLeftWheelCollider.GetWorldPose(out pos, out rot);
        frontLeftWheelTransform.rotation = rot;
        frontLeftWheelTransform.position = pos;

        frontRightWheelCollider.GetWorldPose(out pos, out rot);
        frontRightWheelTransform.rotation = rot;
        frontRightWheelTransform.position = pos;

        rearLeftWheelCollider.GetWorldPose(out pos, out rot);
        rearLeftWheelTransform.rotation = rot;
        rearLeftWheelTransform.position = pos;

        rearRightWheelCollider.GetWorldPose(out pos, out rot);
        rearRightWheelTransform.rotation = rot;
        rearRightWheelTransform.position = pos;
    }

    private void OnTriggerEnter(Collider other) {
        // if collided with finish line
        if (other.gameObject.CompareTag("FinishLine")) {
            lapCount++;

            // if completed 3 laps and race is not finished, declare victory
            if (lapCount > 3 && GameManager.isRaceOn) {
                GameManager.EndRace("NPC");
            }
        }
    }
}
```

Listing 4: NPCCar.cs

```
using System;
```

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

// player car controller
public class PlayerCar : MonoBehaviour
{
    private float hInput;
    private float vInput;

    public float engineStrength = 60000;
    public float maxSteerAngle = 90;

    public Transform frontLeftWheelTransform;
    public Transform frontRightWheelTransform;
    public Transform rearLeftWheelTransform;
    public Transform rearRightWheelTransform;

    public WheelCollider frontLeftWheelCollider;
    public WheelCollider frontRightWheelCollider;
    public WheelCollider rearLeftWheelCollider;
    public WheelCollider rearRightWheelCollider;

    private int lapCount = 0;

    private void FixedUpdate()
    {
        // only accept input if race has started
        if (GameManager.isRaceOn) {
            hInput = Input.GetAxis("Horizontal");
            vInput = Input.GetAxis("Vertical");

            // front wheel drive
            frontLeftWheelCollider.motorTorque = vInput * engineStrength;
            frontRightWheelCollider.motorTorque = vInput * engineStrength;

            // steering
            frontLeftWheelCollider.steerAngle = maxSteerAngle * hInput;
            frontRightWheelCollider.steerAngle = maxSteerAngle * hInput;

            // rotate wheels to propel car
            Vector3 pos;
            Quaternion rot;

            frontLeftWheelCollider.GetWorldPose(out pos, out rot);
            frontLeftWheelTransform.rotation = rot;
            frontLeftWheelTransform.position = pos;

            frontRightWheelCollider.GetWorldPose(out pos, out rot);
            frontRightWheelTransform.rotation = rot;
            frontRightWheelTransform.position = pos;

            rearLeftWheelCollider.GetWorldPose(out pos, out rot);
```

```
            rearLeftWheelTransform.rotation = rot;
            rearLeftWheelTransform.position = pos;

            rearRightWheelCollider.GetWorldPose(out pos, out rot);
            rearRightWheelTransform.rotation = rot;
            rearRightWheelTransform.position = pos;
        }
    }


    private void OnTriggerEnter(Collider other) {
        // if collided with finish line
        if (other.gameObject.CompareTag("FinishLine")) {
            lapCount++;

            // if finished laps and race not over, declare victory
            if (lapCount > 3 && GameManager.isRaceOn) {
                GameManager.EndRace("Player");
            }
            // else update lap counter display
            else {
                LapDisplay.IncrementCount();
            }
        }
    }
}
```

Listing 5: `PlayerCar.cs`

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

// script to enable quitting the game from the main menu
public class QuitGame : MonoBehaviour
{
    public void QuitGameApplication() {
        Application.Quit();
    }
}
```

Listing 6: `QuitGame.cs`

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

// script to enable starting the race from the main menu
public class StartRace : MonoBehaviour
{
    public void StartRaceScene() {
```

```
        SceneManager.LoadScene(1);
    }
}
```

Listing 7: `StartRace.cs`

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

// script to summon the camera to the player car
public class SummonCamera : MonoBehaviour
{
    public Vector3 offset = new Vector3(0f, 4f, -10f);
    public float speed = 10f; // speed of camera movement
    public Camera mc;

    void Start() {
        mc = Camera.main;
    }

    private void FixedUpdate()
    {
        mc.transform.position = Vector3.Lerp(mc.transform.position,
        ↪  transform.TransformPoint(offset), speed * Time.deltaTime);
        mc.transform.rotation = Quaternion.Lerp(mc.transform.rotation,
        ↪  Quaternion.LookRotation(transform.position - mc.transform.position, Vector3.up), speed
        ↪  * Time.deltaTime);
    }
}
```

Listing 8: `SummonCamera.cs`

4 Third Party Assets Used

Below is a list of all third party art assets that I used in this project:

- Grass textures: `https://assetstore.unity.com/packages/2d/textures-materials/floors/hand-painted-grass-texture-78552`
- Racecar prefabs: `https://assetstore.unity.com/packages/3d/vehicles/land/arcade-free-racing-car-161085`
- Asphalt textures: `https://assetstore.unity.com/packages/2d/textures-materials/roads/asphalt-materials-141036`
- Audio clip for engine noises: `https://pixabay.com/sound-effects/engine-6000/`
- Road prefabs: `https://github.com/FritzsHero/RoadArchitect`
- Image for finish line material: `https://www.deviantart.com/fantasystock/art/Beveled-Checker-Board-Seamless-15233976`
- Audio clip for buzzer sound: `https://soundbible.com/1501-Buzzer.html`

- Audio clip for airhorn sound: https://soundbible.com/1542-Air-Horn.html
- Tree prefabs: https://assetstore.unity.com/packages/3d/vegetation/trees/free-trees-103208
- Skybox prefabs: https://assetstore.unity.com/packages/2d/textures-materials/sky/free-stylized-skybox-212257