Journal Entry

Week 1

Wednesday 18th August 2021:

First started off with the fps player and used standard assets to import a basic fps player, then I gave it a basic gun model of an ak-47 and started working on the gun mechanics. I started creating code for the raycast shooting and added ray cast to each bullet that was being fired from the ak-47 when your press "Fire1" or the left mouse button. I added raycasting to each bullet that was being fired so when the bullet hits something it either detects if it's an object which can lose health or if it's a stationary object like the terrain for example. Code at the bottom of week 1.

Thursday 19th August 2021:

Created a shootpoint where the raycasted bullets fire from the ak-47.

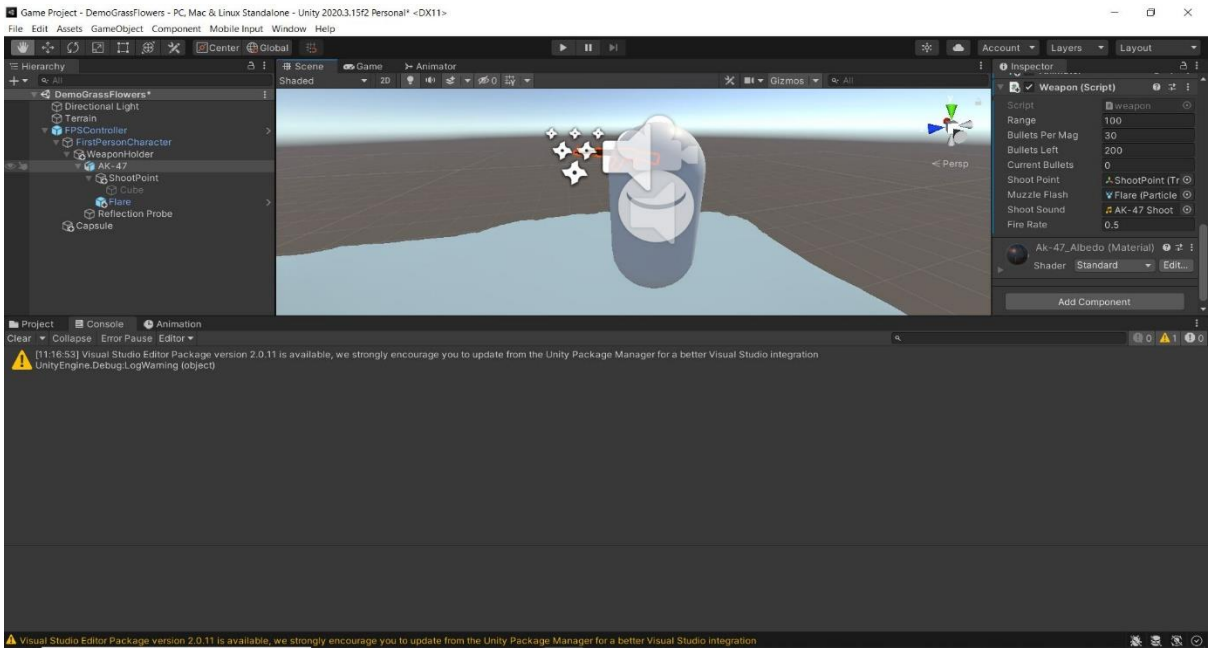
Friday 20th August 2021:

Created some simple gun animation for the idle and fire state of the gun on the unity animator. I created it so that when "Fire1" is pressed or the left mouse button is pressed, it switches from the idle animation to the fire animation. I also added in a muzzle flash each time the gun fired and also added an ak-47 sound effect each time the gun fired.

Sunday 22nd of August 2021:

I added a reloading mechanism where I created 3 variables, bullets per magazine, bullets left and current bullets. Where each time you fire it minuses one bullet from the variable current bullets. When the variable current bullets hit zero it takes 30 bullets from the variable bullets left because the variable bullets per magazine is equal to 30 and it adds it to the variable current bullets. If the variable bullets left is lower than the variable bullets per magazine then it just minuses the variable bullet left by itself and adds it to the variable current bullets. If there are no more bullets left in both the variables current bullets and bullets left then the gun won't play the fire animation and it will just stay in the idle animation.

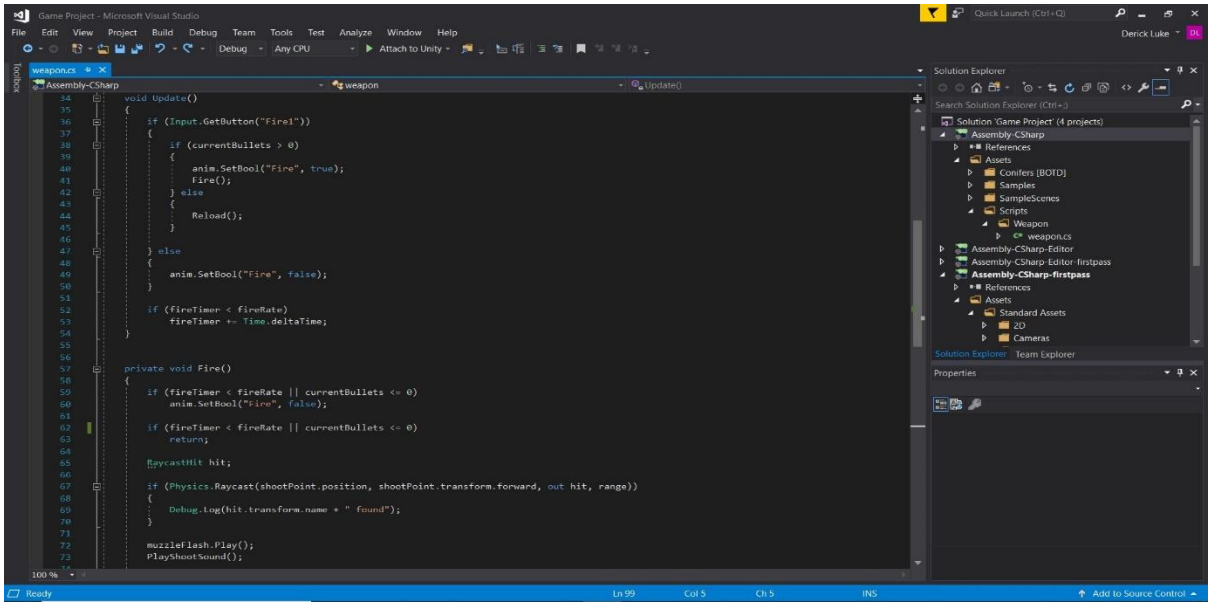
Proof of work picture:

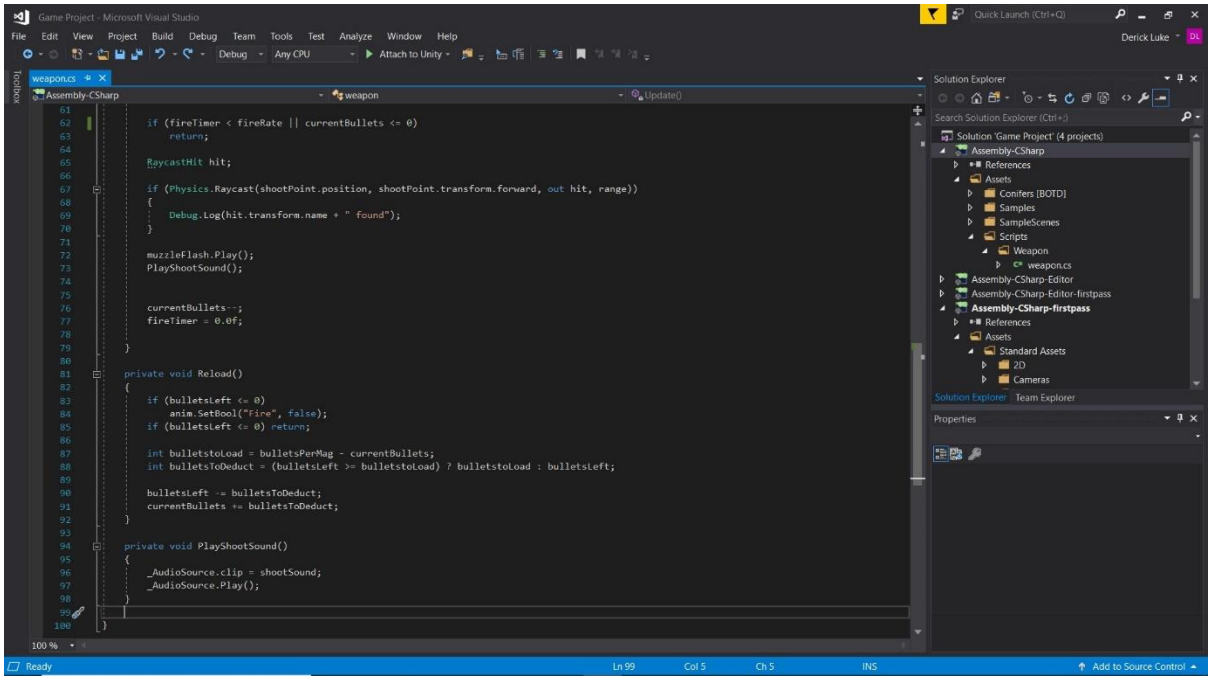


Journal Entry

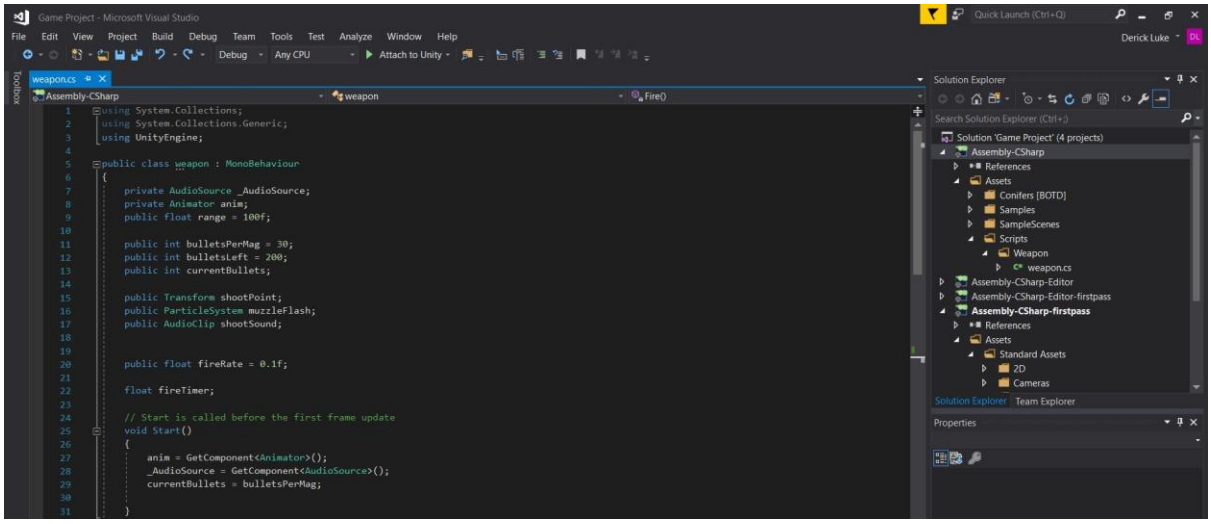
Proof of work code:

Journal Entry





Journal Entry



Questions:

1. **Did you do an effective job of communicating your learning to others?**

Yep, I did a pretty good job of that, every time I completed something I always informed my group member and we managed who does what and give each other roles.

2. **What were you most proud of?**

That I was able to learn C# pretty quickly even though I'm pretty new at it and was able to handle myself in a new subject I haven't learned, which is game dev.

3. **How did you agree or disagree with yourself / others?**

We argued about different ideas like raycasting and how we should use like colliders and tags instead of raycasting and we argued for like 4hrs and I was trying to say raycasting is fairly simple and we should just go with and my teammate said why it was bad and try to give me hints on how to improve raycasting but the problem is I've already done all the things my teammate suggested but since I was new to game dev I didn't understand half of what my teammate said until I showed my code and he was like alright you've already done everything I suggested, cool. So, we kind of wasted like 4hrs but it goes to show we had excellent communication and we were helping each other improve our ideas on each section of the game.

Journal Entry

Week 2

Wednesday 25th of August 2021:

I made a new C# script for the weapon object, which in this case is the ak-47 and called the new script "WeaponSway" all this script does is when you turn, like for example you make the character turn left, instead of the weapon instantly moving to your current position, it delays the weapons movement from where you were last, to your new position to add more of a realistic feel to the game.

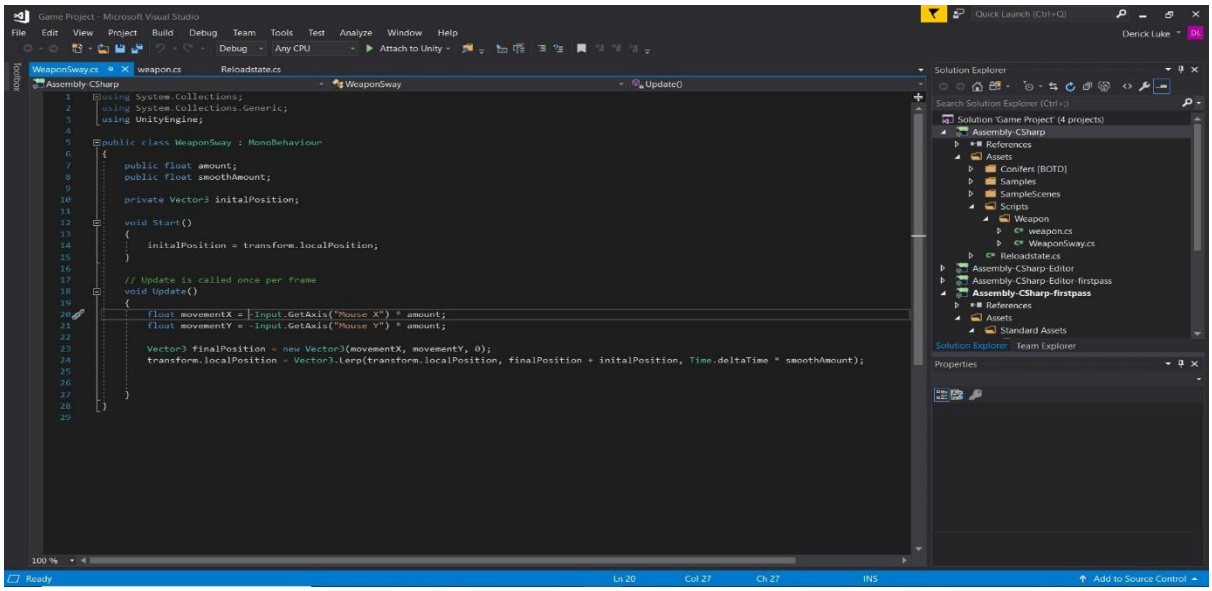
Thursday 26th of August 2021:

In the beginning, I couldn't create a good looking map because unity overloads my ram and deletes any progress I make, but I added more ram and was able to get started on one of the maps that me and my group planned for our game.

Proof of Work Picture:

Journal Entry

Proof of work code:



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

public class WeaponSway : MonoBehaviour
{
    public float amount;
    public float smoothAmount;

    private Vector3 initalPosition;

    void Start()
    {
        initalPosition = transform.localPosition;
    }

    // Update is called once per frame
    void Update()
    {
        float movementX = -Input.GetAxis("Mouse X") * amount;
        float movementY = -Input.GetAxis("Mouse Y") * amount;

        Vector3 finalPosition = new Vector3(movementX, movementY, 0);
        transform.localPosition = Vector3.Lerp(transform.localPosition, finalPosition + initalPosition, Time.deltaTime * smoothAmount);


    }
}
```

Questions:

1. **Were the strategies, skills and procedures you used effective?**

I used the basics I learned from C# tutorials and was able to use them to problem solve situations that came up while I was coding my game like for example, I couldn't get my reload animation to work when the bullets hit zero, so I found a smart work around for it.

2. **What lessons were learned from failure?**

Since I'm new to C# there were many failed attempts but after a bit of trial and error, I was able to fix the issue and learned different techniques on how to solve a specific problem along the way.

3. **What are some of the complexities we should consider?**

Since our group doesn't have an artist, it will take a lot longer to make 3d models like characters and objects but we do have a lot of time so we could be able to finish the 3d models without an artist, maybe one of the maps have to be scratched but at least we can finish around about 2 maps.

Journal Entry

Week 3

Wednesday 8th of September 2021:

I made spawn points for both teams to spawn in the beginning of a match.

Thursday 9th of September 2021:

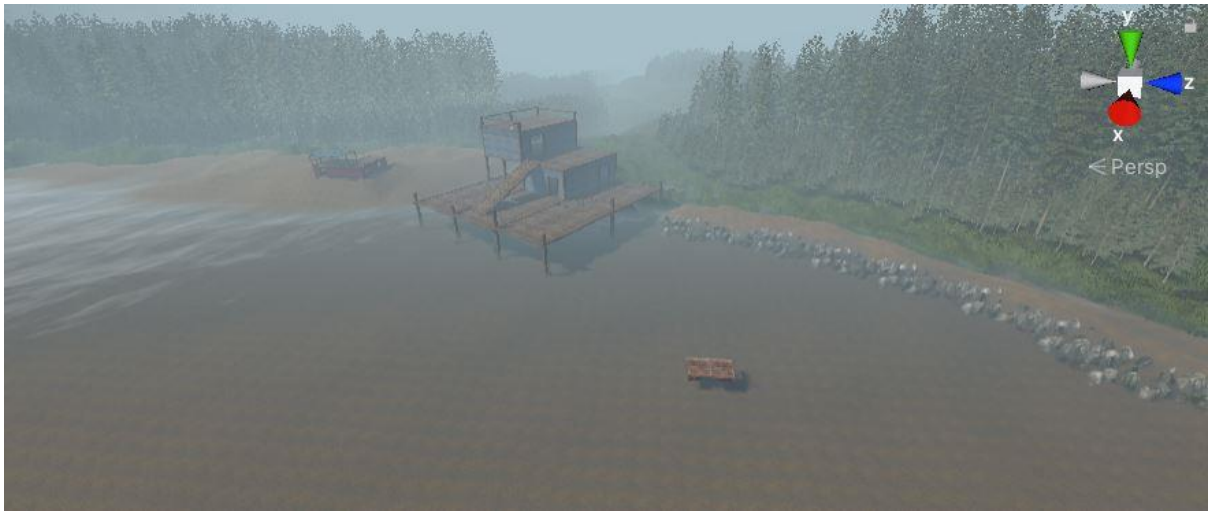
I added water to the scenery and gave it realistic water physics to look natural.

Friday 10th of September 2021:

I did a bit of scenery editing and added more detail to it to make a realistic looking environment by adding random building and adding little details in the environment like adding riverside rocks and a waterfall using the in-built particle systems in unity.

Proof of Work Picture:

Journal Entry



Questions:

1. **What did you learn this week?**
I learned that just making a simple environment doesn't make the environment look realistic like have a terrain with grass on it, but it's the small details that make the environment stand out like the riverside rocks I added to the side of the lake.

2. **What made you curious?**
It made me curious on how game designers make the environment look very realistic in games and I realized it's the little details they add to it to make it look very real like the lighting, or particles like fog or the props they add to the game like rocks and twigs.

3. **How can I verify or test this?**
I asked a couple of friends that game on a daily basis to rate the scenery and if it looks realistic or if it needs improvement

Journal Entry

Week 4

Tuesday 14th of September 2021:

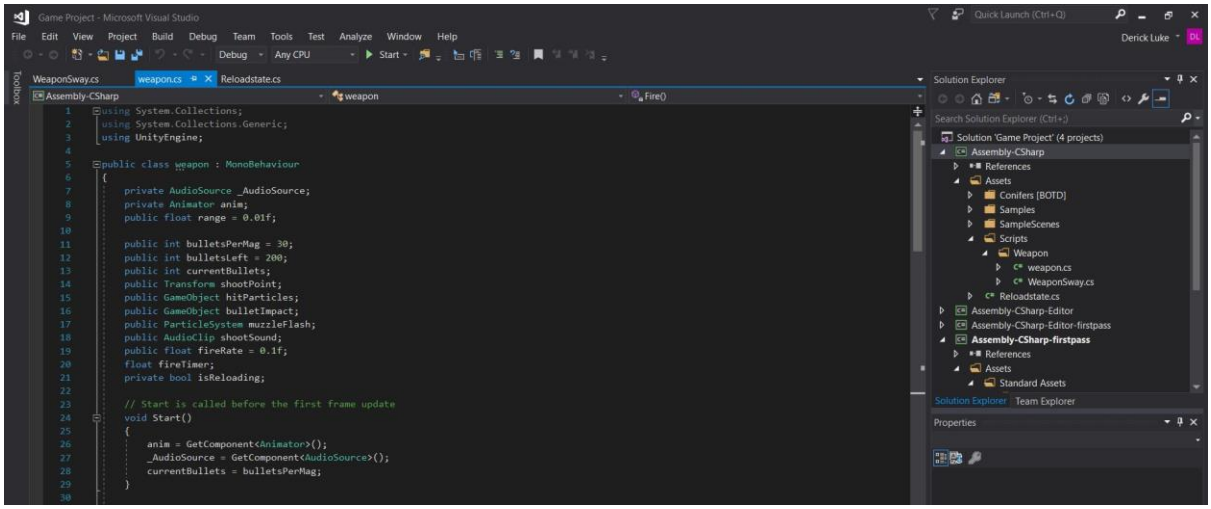
I made a new variable in the weapon script called hitparticles and I programmed it so that wherever the raycasted bullet lands or hits, it plays a short flare animation on that exact spot the bullet hit, recreating what happens in real life when a bullet hits a solid object.

Thursday 16th of September 2021:

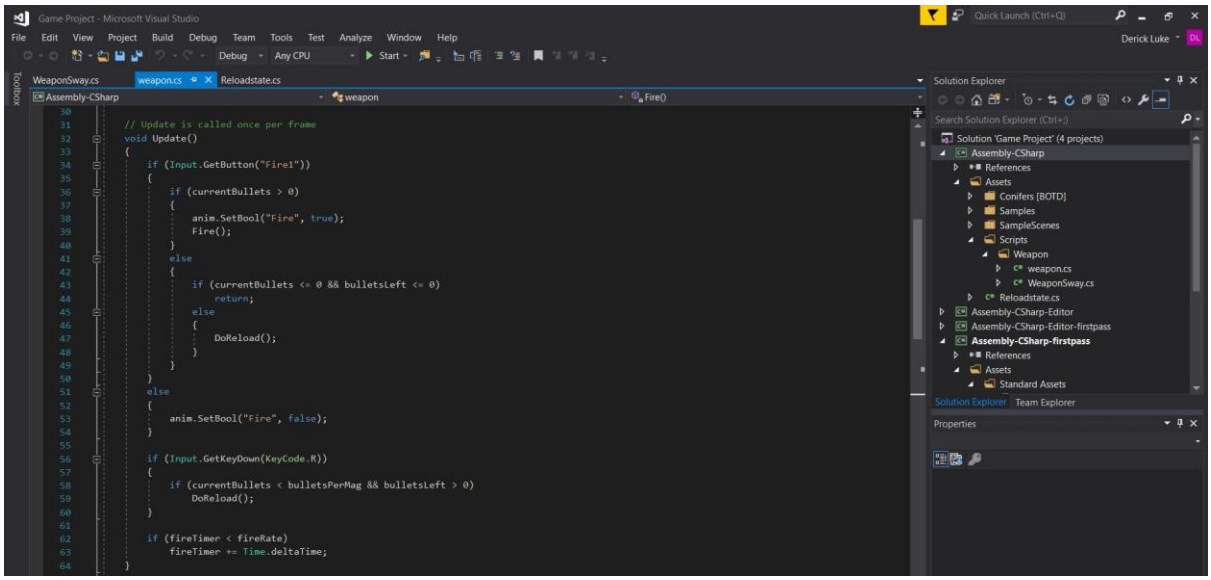
In the weapon script, I made a new variable called bulletimpact and I programmed it so that wherever the raycasted bullet lands or hits, it displays a bullet hole image on the exact spot the bullet hit, recreating what happens in real life when a bullet hits a solid object, e.g a bullet hits a block of wood and it creates a hole in the wood.

Proof of work code:

Journal Entry



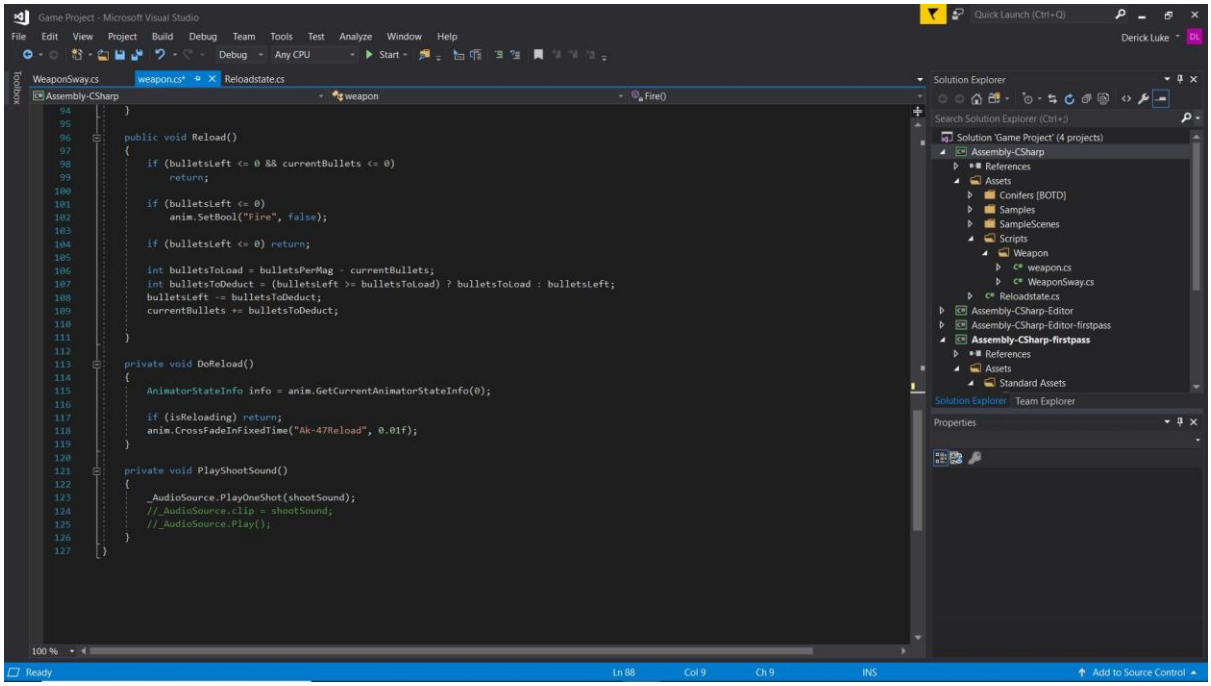
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class weapon : MonoBehaviour
{
    private AudioSource _AudioSource;
    private Animator anim;
    public float range = 0.01f;

    public int bulletsPerMag = 30;
    public int bulletsLeft = 200;
    public int currentBullets;
    public Transform shootPoint;
    public GameObject hitParticles;
    public GameObject bulletImpact;
    public ParticleSystem muzzleFlash;
    public AudioClip shootSound;
    public float fireRate = 0.1f;
    float fireTimer;
    private bool isReloading;

    // Start is called before the first frame update
    void Start()
    {
        anim = GetComponent<Animator>();
        _AudioSource = GetComponent<AudioSource>();
        currentBullets = bulletsPerMag;
    }
```



```
    // Update is called once per frame
    void Update()
    {
        if (Input.GetButton("Fire1"))
        {
            if (currentBullets > 0)
            {
                anim.SetBool("Fire", true);
                Fire();
            }
            else
            {
                if (currentBullets <= 0 && bulletsLeft <= 0)
                    return;
                else
                {
                    DoReload();
                }
            }
        }
        else
        {
            anim.SetBool("Fire", false);
        }

        if (Input.GetKeyDown(KeyCode.R))
        {
            if (currentBullets < bulletsPerMag && bulletsLeft > 0)
                DoReload();
        }

        if (fireTimer < fireRate)
            fireTimer += Time.deltaTime;
    }
```

```
    void FixedUpdate()
    {
        AnimatorStateInfo info = anim.GetCurrentAnimatorStateInfo(0);
        isReloading = info.IsName("Ak-47Reload");
    }

    private void Fire()
    {
        if (fireTimer < fireRate || currentBullets <= 0 || isReloading)
            anim.SetBool("Fire", false);

        if (fireTimer < fireRate || currentBullets <= 0 || isReloading)
            return;

        RaycastHit hit;

        if (Physics.Raycast(shootPoint.position, shootPoint.transform.forward, out hit, range))
        {
            Debug.Log(hit.transform.name + " found");
            GameObject hitParticleEffect = Instantiate(hitParticles, hit.point, Quaternion.FromToRotation(Vector3.up, hit.normal));
            GameObject bulletHole = Instantiate(bulletImpact, hit.point, Quaternion.FromToRotation(Vector3.forward, hit.normal));
        }

        muzzleFlash.Play();
        PlayShootSound();

        currentBullets--;
        fireTimer = 0.0f;
    }
```

Journal Entry



Proof of Work Picture:

Journal Entry

Questions:

1. **Did you come to class prepared to learn, in both your attitude and with all your supplies?**

Yes, we came to class prepared, in both attitude and supplies. In each lessons we would have short meetings discussing what we did this week and what we contributed to the games and also brainstormed ideas on how to make it better.

2. **Should I change my perspective?**

Yes, I should change my perspective because if it was just me thinking on how to make the game perfect, the game would be more suited to me than the general public.

3. **Where did you meet success, and who might benefit most from what you've learned along the way?**

I met success from hard work and research, I would look at tutorials on how to make specific game mechanics and problem solve whenever there was a situation with the game with the limited my limited knowledge on game development.

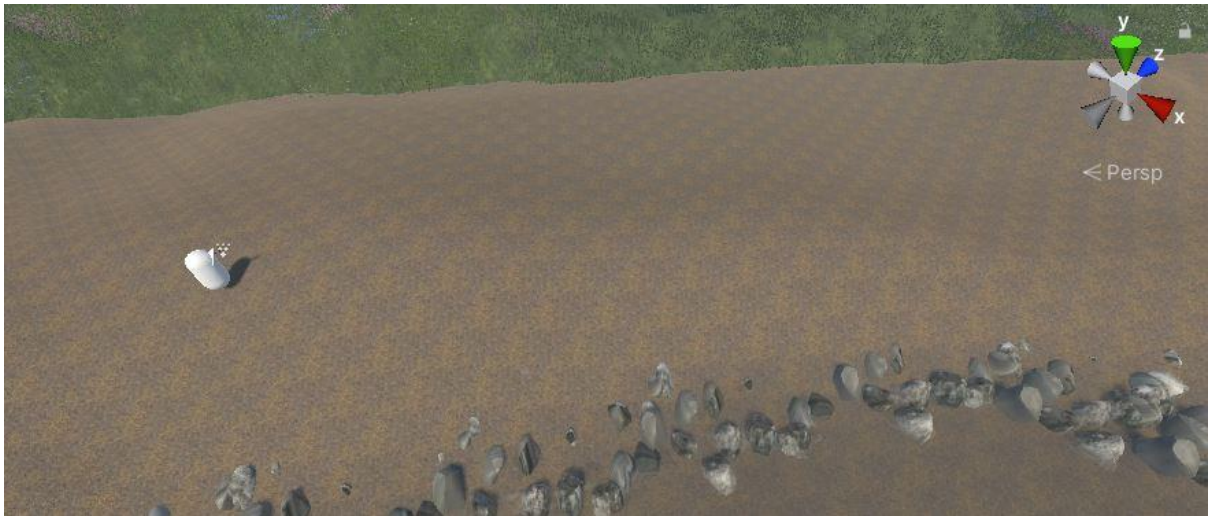
Journal Entry

Week 5

Monday 20th of September 2021:

I did some more scenery designs like finishing off the riverbed and increase the beach side width by 8 metres.

Proof of Work Picture:



Questions:

1. **Were the strategies, skills and procedures you used effective?**
For the scenery design, I got inspiration from real world sceneries like the Grand Canyon and Havasu water falls in Arizona.

2. **What made you curious?**
How small details made a huge difference in the scenery, like extending the beach side from a width of 3 metres to 8 metres, it really made a strong impact on the scenery.

3. **What is most relevant or important?**
Our games doesn't have the best graphics since we are using store bought assets, so we have to use scenery techniques to make it look realistic.

Journal Entry

Week 6

Wednesday 29th of September:

I did some more scenery designs because, our whole teams decided on a story for our game and I had to change a few things about the scenery so it fits with our story and game mechanics. I started making a base for the enemy team like a fortress as it will match with our game story.

Proof of Work Picture:

Questions:

1. **Did you do an effective job of communicating your learning to others?**

   Yes, we all brainstormed together and implemented our own ideas and worked together to decide on how the scenery will look for our rescripted game story.

2. **How did I hinder others?**

   I might have hindered others by saying their ideas won't match the story line and won't make it compelling but this was quickly solved by voting which idea should go into the game.

3. **How did you agree or disagree with yourself / others?**

   I was able to make everyone agree on the new story I came up with, since we were struggling with story ideas and my story was not only compelling but it also matches with the game mechanics I incorporated. I also disagreed with others when one of my teammates wanted to keep the UI backdrop but I told him to make it again because it didn't match with the game storyline.

Journal Entry

Week 7

Friday 8th of October 2021:

I again did some more scenery adjustments by adding a second water fall which make it look like water is going into the riverbed into the lake.

Proof of Work Picture:

Questions:

1. **Which activities helped you learn the most?**
   Tutorials on how to make a realistic scenery really helped me in turning the scenery from average looking to somewhat realistic, despite the fact I had terrible graphics.

2. **How does this relate to current issues / problems / solutions?**
   Since, we had a whole rewrite on our game story, we had to change up the environment so it will match with the game story, but since the story was more suited to this kind of environment, it was simple, we only had to change a few variables for it to work.

3. **How can I verify or test this?**
   I verified if my scenery looked realistic by asking other teams if the scenery looked realistic and they said I was on the right path but had to tweak a few bits with the lighting and riverbed side as the riverbed side was just gras and that wouldn't be very realistic.

Journal Entry

Week 8

Tuesday 12th, Wednesday 13th,Thursday 14th, Friday 15th, Saturday 16th of October 2021:

Our game had a full rewrite of the story so now I had to code a few extra game mechanics like a car because the car will be apart of the game main objectives as the enforcer team has to get their car into the terrorist teams base or fort. So, I got a 3d car model of the unity asset store and made a carController script. In this script I coded the basic mechanics for a car by adding colliders, wheel functions, lights(reflective car body), particles(skid marks) and sounds(e.g revving sounds, skid mark sounds).

Proof of Work Code:

```
using System;
using UnityEngine;

#pragma warning disable 649
namespace UnityStandardAssets.Vehicles.Car
{
    internal enum CarDriveType
    {
        FrontWheelDrive,
        RearWheelDrive,
        FourWheelDrive
    }

    internal enum SpeedType
    {
        MPH,
        KPH
    }

    public class CarController : MonoBehaviour
    {
        [SerializeField] private CarDriveType m_CarDriveType = CarDriveType.FourWheelDrive;
        [SerializeField] private WheelCollider[] m_WheelColliders = new WheelCollider[4];
        [SerializeField] private GameObject[] m_WheelMeshes = new GameObject[4];
        [SerializeField] private WheelEffects[] m_WheelEffects = new WheelEffects[4];
        [SerializeField] private Vector3 m_CentreOfMassOffset;
        [SerializeField] private float m_MaximumSteerAngle;
        [Range(0, 1)] [SerializeField] private float m_SteerHelper;
        [Range(0, 1)] [SerializeField] private float m_TractionControl;
        [SerializeField] private float m_FullTorqueOverAllWheels;
        [SerializeField] private float m_ReverseTorque;
        [SerializeField] private float m_MaxHandbrakeTorque;
        [SerializeField] private float m_Downforce = 100f;
        [SerializeField] private SpeedType m_SpeedType;
        [SerializeField] private float m_Topspeed = 200;
        [SerializeField] private static int NoOfGears = 5;
        [SerializeField] private float m_RevRangeBoundary = 1f;
        [SerializeField] private float m_SlipLimit;
        [SerializeField] private float m_BrakeTorque;

        private Quaternion[] m_WheelMeshLocalRotations;
        private Vector3 m_Prevpos, m_Pos;
        private float m_SteerAngle;
        private int m_GearNum;
        private float m_GearFactor;
        private float m_OldRotation;
        private float m_CurrentTorque;
        private Rigidbody m_Rigidbody;
        private const float k_ReversingThreshold = 0.01f;

        public bool Skidding { get; private set; }
        public float BrakeInput { get; private set; }
        public float CurrentSteerAngle{ get { return m_SteerAngle; }}
        public float CurrentSpeed{ get { return m_Rigidbody.velocity.magnitude*2.23693629f; }}
        public float MaxSpeed{get { return m_Topspeed; }}
        public float Revs { get; private set; }
        public float AccelInput { get; private set; }

```

Journal Entry

```
// Use this for initialization
private void Start()
{
    m_WheelMeshLocalRotations = new Quaternion[4];
    for (int i = 0; i < 4; i++)
    {
        m_WheelMeshLocalRotations[i] = m_WheelMeshes[i].transform.localRotation;
    }
    m_WheelColliders[0].attachedRigidbody.centerOfMass = m_CentreOfMassOffset;

    m_MaxHandbrakeTorque = float.MaxValue;

    m_Rigidbody = GetComponent<Rigidbody>();
    m_CurrentTorque = m_FullTorqueOverAllWheels - (m_TractionControl*m_FullTorqueOverAllWheels);
}


private void GearChanging()
{
    float f = Mathf.Abs(CurrentSpeed/MaxSpeed);
    float upgearlimit = (1/(float) NoOfGears)*(m_GearNum + 1);
    float downgearlimit = (1/(float) NoOfGears)*m_GearNum;

    if (m_GearNum > 0 && f < downgearlimit)
    {
        m_GearNum--;
    }

    if (f > upgearlimit && (m_GearNum < (NoOfGears - 1)))
    {
        m_GearNum++;
    }
}


// simple function to add a curved bias towards 1 for a value in the 0-1 range
private static float CurveFactor(float factor)
{
    return 1 - (1 - factor)*(1 - factor);
}


// unclamped version of Lerp, to allow value to exceed the from-to range
private static float ULerp(float from, float to, float value)
{
    return (1.0f - value)*from + value*to;
}


private void CalculateGearFactor()
{
    float f = (1/(float) NoOfGears);
    // gear factor is a normalised representation of the current speed within the current gear's range of speeds.
    // We smooth towards the 'target' gear factor, so that revs don't instantly snap up or down when changing gear.
    var targetGearFactor = Mathf.InverseLerp(f*m_GearNum, f*(m_GearNum + 1), Mathf.Abs(CurrentSpeed/MaxSpeed));
    m_GearFactor = Mathf.Lerp(m_GearFactor, targetGearFactor, Time.deltaTime*5f);
}


private void CalculateRevs()
{
    // calculate engine revs (for display / sound)
    // (this is done in retrospect - revs are not used in force/power calculations)
    CalculateGearFactor();
    var gearNumFactor = m_GearNum/(float) NoOfGears;
    var revsRangeMin = ULerp(0f, m_RevRangeBoundary, CurveFactor(gearNumFactor));
    var revsRangeMax = ULerp(m_RevRangeBoundary, 1f, gearNumFactor);
    Revs = ULerp(revsRangeMin, revsRangeMax, m_GearFactor);
}
```

Journal Entry

```
129
130     public void Move(float steering, float accel, float footbrake, float handbrake)
131     {
132         for (int i = 0; i < 4; i++)
133         {
134             Quaternion quat;
135             Vector3 position;
136             m_WheelColliders[i].GetWorldPose(out position, out quat);
137             m_WheelMeshes[i].transform.position = position;
138             m_WheelMeshes[i].transform.rotation = quat;
139         }
140
141         //clamp input values
142         steering = Mathf.Clamp(steering, -1, 1);
143         AccelInput = accel = Mathf.Clamp(accel, 0, 1);
144         BrakeInput = footbrake = -1*Mathf.Clamp(footbrake, -1, 0);
145         handbrake = Mathf.Clamp(handbrake, 0, 1);
146
147         //Set the steer on the front wheels.
148         //Assuming that wheels 0 and 1 are the front wheels.
149         m_SteerAngle = steering*m_MaximumSteerAngle;
150         m_WheelColliders[0].steerAngle = m_SteerAngle;
151         m_WheelColliders[1].steerAngle = m_SteerAngle;
152
153         SteerHelper();
154         ApplyDrive(accel, footbrake);
155         CapSpeed();
156
157         //Set the handbrake.
158         //Assuming that wheels 2 and 3 are the rear wheels.
159         if (handbrake > 0f)
160         {
161             var hbTorque = handbrake*m_MaxHandbrakeTorque;
162             m_WheelColliders[2].brakeTorque = hbTorque;
163             m_WheelColliders[3].brakeTorque = hbTorque;
164         }
165
166
167         CalculateRevs();
168         GearChanging();
169
170         AddDownForce();
171         CheckForWheelSpin();
172         TractionControl();
173     }
174
175
176     private void CapSpeed()
177     {
178         float speed = m_Rigidbody.velocity.magnitude;
179         switch (m_SpeedType)
180         {
181             case SpeedType.MPH:
182
183                 speed *= 2.23693629f;
184                 if (speed > m_Topspeed)
185                     m_Rigidbody.velocity = (m_Topspeed/2.23693629f) * m_Rigidbody.velocity.normalized;
186                 break;
187
188             case SpeedType.KPH:
189                 speed *= 3.6f;
190                 if (speed > m_Topspeed)
191                     m_Rigidbody.velocity = (m_Topspeed/3.6f) * m_Rigidbody.velocity.normalized;
192                 break;
193             }
194     }
195
196
197     private void ApplyDrive(float accel, float footbrake)
198     {
199
200         float thrustTorque;
201         switch (m_CarDriveType)
202         {
203             case CarDriveType.FourWheelDrive:
204                 thrustTorque = accel * (m_CurrentTorque / 4f);
205                 for (int i = 0; i < 4; i++)
206                 {
207                     m_WheelColliders[i].motorTorque = thrustTorque;
208                 }
209                 break;
210
211             case CarDriveType.FrontWheelDrive:
212                 thrustTorque = accel * (m_CurrentTorque / 2f);
213                 m_WheelColliders[0].motorTorque = m_WheelColliders[1].motorTorque = thrustTorque;
214                 break;
215
216             case CarDriveType.RearWheelDrive:
217                 thrustTorque = accel * (m_CurrentTorque / 2f);
218                 m_WheelColliders[2].motorTorque = m_WheelColliders[3].motorTorque = thrustTorque;
219                 break;
220
221             }
222
223             for (int i = 0; i < 4; i++)
224             {
225                 if (CurrentSpeed > 5 && Vector3.Angle(transform.forward, m_Rigidbody.velocity) < 50f)
226                 {
227                     m_WheelColliders[i].brakeTorque = m_BrakeTorque*footbrake;
228                 }
229                 else if (footbrake > 0)
230                 {
231                     m_WheelColliders[i].brakeTorque = 0f;
232                     m_WheelColliders[i].motorTorque = -m_ReverseTorque*footbrake;
233                 }
234             }
235     }
```

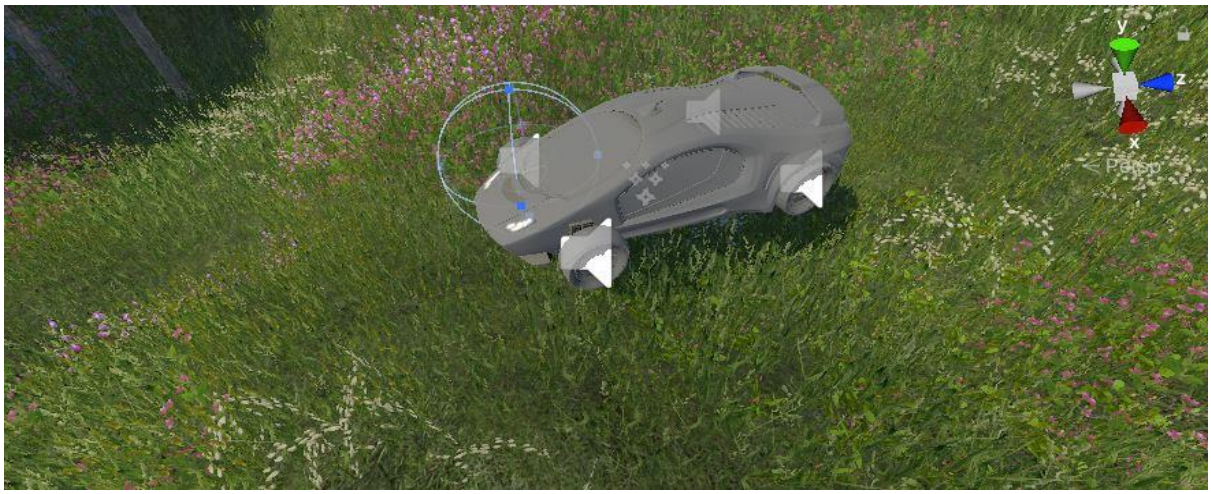
Journal Entry

```
267     // checks if the wheels are spinning and is so does three things
268     // 1) emits particles
269     // 2) plays tiure skidding sounds
270     // 3) leaves skidmarks on the ground
271
272     private void CheckForWheelSpin()
273     {
274         // loop through all wheels
275         for (int i = 0; i < 4; i++)
276         {
277             WheelHit wheelHit;
278             m_WheelColliders[i].GetGroundHit(out wheelHit);
279
280             // is the tire slipping above the given threshhold
281             if (Mathf.Abs(wheelHit.forwardSlip) >= m_SlipLimit || Mathf.Abs(wheelHit.sidewaysSlip) >= m_SlipLimit)
282             {
283                 m_WheelEffects[i].EmitTyreSmoke();
284
285                 // avoiding all four tires screeching at the same time
286
287                 if (!AnySkidSoundPlaying())
288                 {
289                     m_WheelEffects[i].PlayAudio();
290                 }
291                 continue;
292             }
293
294             // if it wasnt slipping stop all the audio
295             if (m_WheelEffects[i].PlayingAudio)
296             {
297                 m_WheelEffects[i].StopAudio();
298             }
299             // end the trail generation
300             m_WheelEffects[i].EndSkidTrail();
301         }
302     }
236
237
238     private void SteerHelper()
239     {
240         for (int i = 0; i < 4; i++)
241         {
242             WheelHit wheelhit;
243             m_WheelColliders[i].GetGroundHit(out wheelhit);
244             if (wheelhit.normal == Vector3.zero)
245                 return; // wheels arent on the ground so dont realign the rigidbody velocity
246         }
247
248         // this if is needed to avoid gimbal lock problems that will make the car suddenly shift direction
249         if (Mathf.Abs(m_OldRotation - transform.eulerAngles.y) < 10f)
250         {
251             var turnadjust = (transform.eulerAngles.y - m_OldRotation) * m_SteerHelper;
252             Quaternion velRotation = Quaternion.AngleAxis(turnadjust, Vector3.up);
253             m_Rigidbody.velocity = velRotation * m_Rigidbody.velocity;
254         }
255         m_OldRotation = transform.eulerAngles.y;
256     }
257
258
259     // this is used to add more grip in relation to speed
260     private void AddDownForce()
261     {
262         m_WheelColliders[0].attachedRigidbody.AddForce(-transform.up*m_Downforce*
263                                                 m_WheelColliders[0].attachedRigidbody.velocity.magnitude);
264     }
265
266
```

Journal Entry

```
        // crude traction control that reduces the power to wheel if the car is wheel spinning too much
        private void TractionControl()
        {
            WheelHit wheelHit;
            switch (m_CarDriveType)
            {
                case CarDriveType.FourWheelDrive:
                    // loop through all wheels
                    for (int i = 0; i < 4; i++)
                    {
                        m_WheelColliders[i].GetGroundHit(out wheelHit);

                        AdjustTorque(wheelHit.forwardSlip);
                    }
                    break;

                case CarDriveType.RearWheelDrive:
                    m_WheelColliders[2].GetGroundHit(out wheelHit);
                    AdjustTorque(wheelHit.forwardSlip);

                    m_WheelColliders[3].GetGroundHit(out wheelHit);
                    AdjustTorque(wheelHit.forwardSlip);
                    break;

                case CarDriveType.FrontWheelDrive:
                    m_WheelColliders[0].GetGroundHit(out wheelHit);
                    AdjustTorque(wheelHit.forwardSlip);

                    m_WheelColliders[1].GetGroundHit(out wheelHit);
                    AdjustTorque(wheelHit.forwardSlip);
                    break;
            }
        }


        private void AdjustTorque(float forwardSlip)
        {
            if (forwardSlip >= m_SlipLimit && m_CurrentTorque >= 0)
            {
                m_CurrentTorque -= 10 * m_TractionControl;
            }
            else
            {
                m_CurrentTorque += 10 * m_TractionControl;
                if (m_CurrentTorque > m_FullTorqueOverAllWheels)
                {
                    m_CurrentTorque = m_FullTorqueOverAllWheels;
                }
            }
        }


        private bool AnySkidSoundPlaying()
        {
            for (int i = 0; i < 4; i++)
            {
                if (m_WheelEffects[i].PlayingAudio)
                {
                    return true;
                }
            }
            return false;
        }
    }
}
```

Proof of Work Picture:



Journal Entry

Questions:

1. **Which activities helped you learn the most?**
Tutorials on Udemy about game designs really helped me out when it came to the game mechanics of the car, as I am still an amateur game coder, so the tutorial really helped me out when it came to coding the car mechanics.

2. **What lessons were learned from failure?**
When I couldn't get the front and back wheels to work together and was stuck for 5hrs and I almost quit, but I stuck to it and eventually figured it out when debugging the entire code.

3. **What are some of the complexities we should consider?**
We should consider on how the car will function across the map, as there are some uneven terrain forming's, meaning the car can get stuck and we can't get it out meaning it will be a win by default for the terrorist team.

Journal Entry

Week 9

Thursday 21st of October 2021:

I did a bit more terrain fixing by making some bits a little less uneven and finished of some small detail in the environment like the lake side rocks.

Proof of Work Picture:

Questions:

1. **If you could do this week over, what would you do differently?**
   I would like to have done a bit more as I did not do much this week, because I was worn out from spending most of my time last week coding the car mechanics.

2. **Where did you encounter struggles and what did you do to deal with it?**
   I encountered struggles when I was trying to make the terrain even but I couldn't understand which terrain tool to use, so I kept getting uneven bumps in the terrain. I overcame this problem by actually learning what each tool does, instead of blindly using it.

3. **What is most relevant or important?**
   Having uneven terrain in the map because having uneven terrain could result in the car that I coded last week getting stuck in a hole meaning the terrorist team would win by default because the enforcer team couldn't get the car into their base as it was stuck in a hole.

Journal Entry

Week 10

Wednesday 27th of October 2021:
We mostly did game testing this week on how well the game is running so far. Currently the game is running on 10fps which is not great but that is mainly because we do not have a graphics card hence the low fps but the game can still be optimized in the future to run better and cause less lag during actual gameplay among multiple players.

Proof of Work Picture:

Questions:

1. **What did you learn this week?**
That making a game is very hard work without a team because there are so many variables. So far, I have only worked on the game mechanics and it took me a very long time, I could not imagine how long it would take me to do the UI or the story for the game.

2. **How did you convince others that your way/s were the best way?**
Since, I was working on the game mechanics there was a lot of disputes on how the game should work, like the matches and how they would operate, should it be a defuse the bomb sort of situation or protect the hostage match. We eventually decided on what type of match we wanted when I brought up the idea of a completely new gameplay experience and one that matches with our story and everyone agreed as it was unique and could be a fun experience.

3. **What is most relevant or important?**
Is that our game is fun and entertaining for the general public where gamers can relax and enjoy their free time by playing our fps game and also make our game playable as well so it won't frustrate gamers when their game lags every 10 seconds.