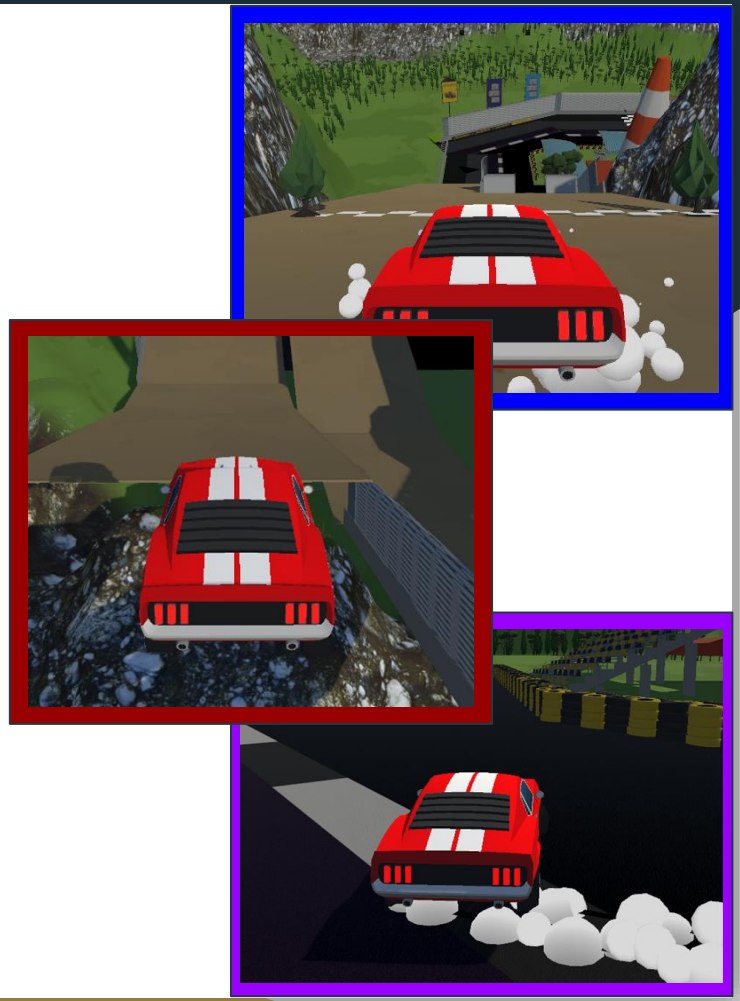
Extreme Barbie Car Racing

Marco Lo Dico, Joseph Picheca, Xavier Solano
SPH4U1
[Instructor's name removed for privacy]
April 3, 2023

1 - Physics Principles

Extreme Barbie Car Racing, inspired by the great sport local to Saint Jo, Texas - Barbie Jeep Racing - involves the use of several physics principles to achieve the desired result of a fun and interesting course. First, an **unpowered car must accelerate due to gravity**. Then the contestant is faced with either **completing a jump with the principles of projectile motion**, or having to avoid obstacles to complete the course. With this built-up speed, the car must complete a turn where it will experience **centripetal acceleration due** to the **circular motion** during the turn. Additionally, **friction is an important force** that constantly works to ensure the contestant maintains control of the car.



2 + 4 - Testing Procedure and Safety

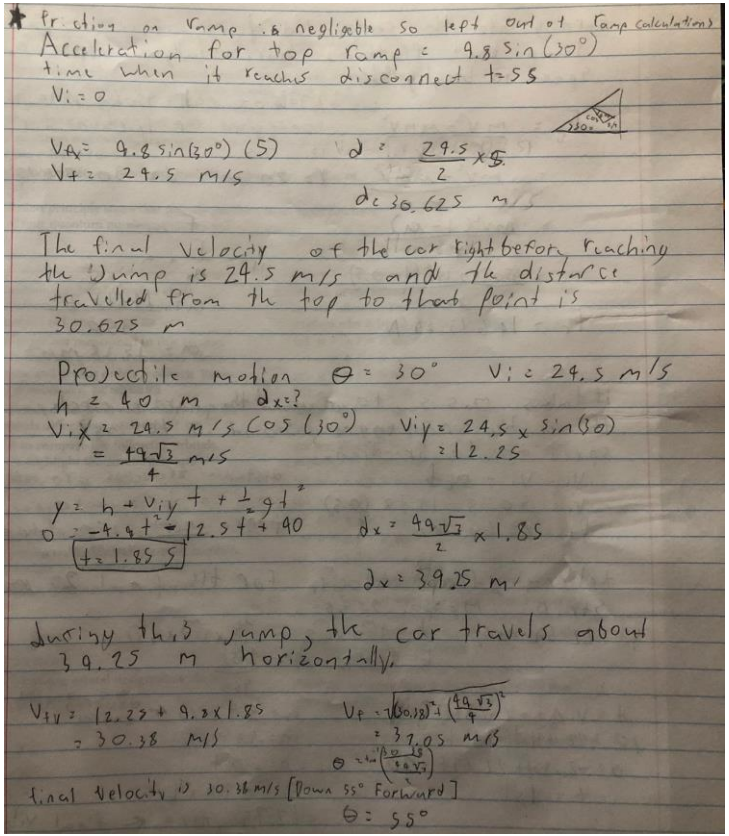
Testing is vital to ensure **safety and fun are both achieved**. This testing process was able to be rapidly experienced due to the modifiable nature of software. Testing consisted of:

- Making any necessary changes to **the code or environment.**
- Uploading those changes for group members to see.
- **Repeatedly running the game** to test out those changes in various situations.
- Making any corrections.
- Additional testing.

Calculations ensured that **obstacles like the jump and turn can be completed without crashing**. Furthermore, live values of distance and velocity could be obtained while testing, which allowed for adjustments to be made at sections where the car may have been moving too fast or too slow, which could have hindered the pace or made it hard to control. **Numerous changes were made to ensure the course is completed successfully**, such as changing the angles on the jump and turns. Since **the Unity game engine uses a physics engine, a realistic representation of physics is achieved, which allows for numerical parity with reality.** It is accepted that there is a certain safety risk, as is the nature of extreme sports, but significant physics knowledge allows for unforeseen risks to be eliminated.

```
yer.cs | PrometeoCarController.cs | PrometeoEditor.cs | physicsValues.cs
Sharp                                  CamFollowsPlayer
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 0 references
public class CamFollowsPlayer : MonoBehaviour
{

    public Transform cameraTarget;
    public float sSpeed = 10.0f;
    public Vector3 dist;
    public Transform lookTarget;

    Unity Message | 0 references
    void FixedUpdate()
    {
        Vector3 dPos = cameraTarget.position + dist;
        Vector3 sPos = Vector3.Lerp(transform.position, dPos, sSp
        transform.position = sPos;
        transform.LookAt(lookTarget.position);
    }
}
```

3 - Calculations (Additional Copy Submitted Physically)



Friction on ramp is negligible so left out of ramp calculations

Acceleration for top ramp = $9.8\sin(30°)$

time when it reaches disconnect t=5 s

$V_i = 0$

$V_{fx} = 9.8\sin(30°)(5)$

$V_f = 24.5$ m/s

$d = \frac{29.5}{2} \times 5$

$d = 30.625$ m

The final velocity of the car right before reaching the jump is 24.5 m/s and the distance travelled from the top to that point is 30.625 m

Projectile motion $\theta = 30°$ $V_i = 24.5$ m/s

$h = 40$ m $d_x = ?$

$V_{ix} = 24.5 \cos(30°) = \frac{49\sqrt{3}}{4}$ m/s

$V_{iy} = 24.5 \times \sin(30) = 12.25$

$y = h + v_{iy}t + \frac{1}{2}gt^2$

$0 = -4.9t^2 - 12.5t + 40$

$t = 1.85$ s

$d_x = \frac{49\sqrt{3}}{2} \times 1.85$

$d_x = 39.25$ m

during this jump, the car travels about 39.25 m horizontally.

$V_{fy} = 12.25 + 9.8 \times 1.85$

$= 30.38$ m/s

$V_f = \sqrt{(30.38)^2 + \left(\frac{49\sqrt{3}}{4}\right)^2}$

$= 37.05$ m/s

$\theta = \tan^{-1}\left(\frac{30.38}{\frac{49\sqrt{3}}{4}}\right)$

final velocity is 30.38 m/s [Down 55° Forward]

$\theta = 55°$

turning into the centripetal part, it is a curved ramp that flattens horizontally so loss of speed is negligible.

mass of car: 950kg $m = 950$

coefficient of friction for asphalt is 0.5

$r = 20$ m $V = 30.38$ m/s

$F_c = \frac{mV^2}{r} - \frac{\mu mV^2}{r}$

$= \frac{mV^2 - \mu mV^2}{r}$

$= \frac{mV^2(1-\mu)}{r}$

$= \frac{950(30.38)^2(1-0.5)}{30}$

$F_c = 14613.29$ N

$F = ma$

$\frac{F}{m} = a$

$a = \frac{14613.29}{950}$

$a = 15.38$ m/s^2

it takes 0.5 s to go around the quarter circle, the car has no engine, so it is losing speed due to friction.

$V_f = V_i - at$

$V_f = 30.38 - 15.38(0.5)$

$V_f = 22.69$ m/s

distance travelled around circle = $\frac{1}{4}$ circumference

$\frac{1}{4}2\pi r = \frac{2\pi(20)}{4} = 10\pi$ Meters travelled during turn

determining final velocity for the final 20 m strip

$m = 950$ $g = 9.8$

$V_i = 22.69$ m/s

$F = 950 \times 9.8 - 0.5 \times 950 \times 9.81$

$F = 4655$ N $a = \frac{4655}{950}$

$a = 4.9$ m/s ← decelerating b/c no engine

$a = -4.9$ m/s

$d = V_i\Delta t + \frac{1}{2}a(\Delta t)^2$

$10 = 22.69t + \frac{1}{2}(-4.9)t^2$

$0 = -2.45t^2 + 22.69t - 20$

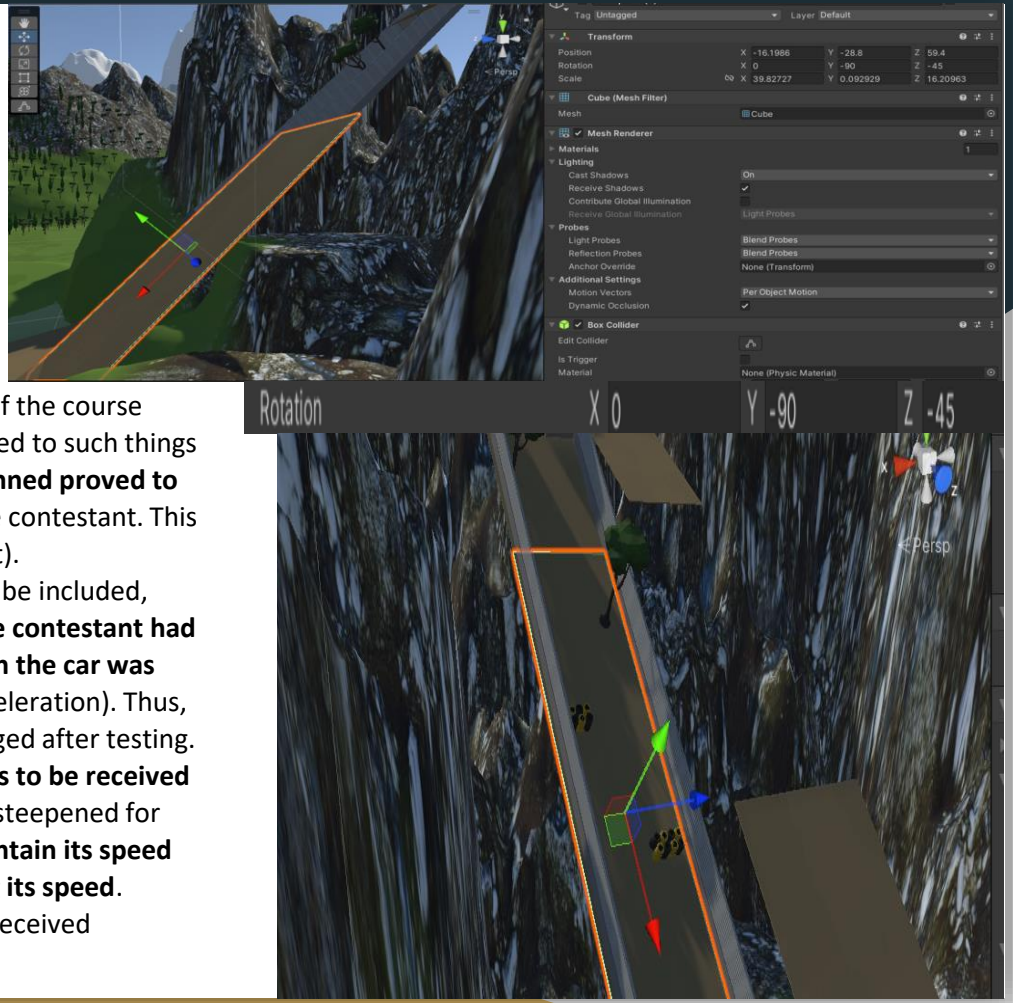
$t = 1$ s

$V_f = V_i + at$

$= 22.69 - 4.9 \times 1$

$= 17.79$ m/s ← final velocity

5 - Changes Made After Testing and which Parts were Functioning



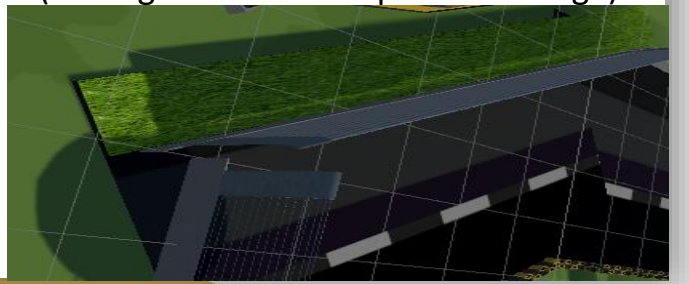
- Due to extensive planning, minimal changes to the layout of the course were required. However, some slight variations were needed to such things as **the angles of inclination, as the original 55 degrees planned proved to accelerate the car too greatly** to be safe or possible for the contestant. This value was instead **changed to 45 degrees** (see bottom right).
- Furthermore, significantly more obstacles were planned to be included, except we had **failed to account for the limited control the contestant had over the direction of motion** when the **only force acting on the car was due to gravity** (and it being the only effect on the car's acceleration). Thus, the layout of the course and its obstacles was slightly changed after testing.
- In addition to this, **the angle of incline at which the car was to be received from its projectile launch was too flat**, and was therefore steepened for safety, speed, and fluidity. This **allowed the vehicle to maintain its speed after landing, instead of the impact force greatly reducing its speed**. (Changed to allow the car from a 30 degree jump to to be received smoothly by a 45 degree plane, as shown above).

6 - Design Problems

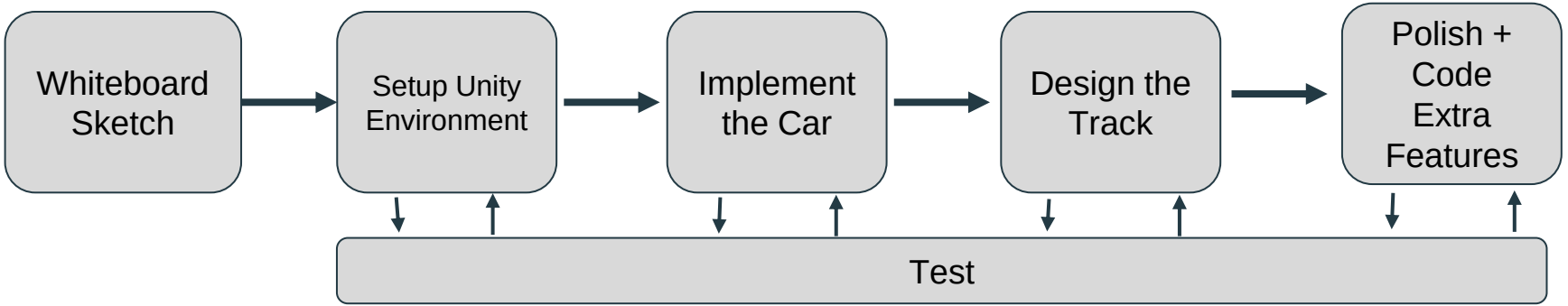
- Being developed on a **virtual workspace**, an issue arrived when attempting to work collaboratively, as it **required the used of extra cloud based software to sync and upload each group members' progress**. This software also contained a plethora of issues, such as the notorious "Terrain 1" (the terrain would not sync with the project), or various coding issues, such as calculating the car's acceleration in addition to its velocity (see right).
- On a physics level, there were very little problems, as a digital platform is not limited by needing physical materials. Therefore, **the variables** (such as the cars speeds, weight, and coefficient of friction) **became the issues**, as **adjusting these variables to match the authentic experience was difficult, but necessary to ground the sport in realism.**
- Furthermore, due to **the magnitude of the project** and the eventual accuracy of the variables, the scale of the track became a difficult problem. This is because building the track to that ratio required accurate angles of inclination to ensure safe launches and safe components of gravity to accelerate the car.
- Lastly, **the distance had to be scaled** as the ramp was adjusted, since the ramp affects the car's speed. If the car gained more speed, everything had to be made longer to ensure the obstacles were avoidable.

```
// This does not work
1 reference
void fixedUpdate()
{
    // Fetches the velocity value of the Y component
    velocityY = rb.velocity.y;

    veloYText.text = "Vy = " + velocityY.ToString();

    velocityX = rb.velocity.x;

    veloXText.text = "Vx = " + velocityX.ToString();

    velocityZ = rb.velocity.z;

    veloZText.text = "Vz = " + velocityZ.ToString();

    // Calculates the accelerations
    /*accelerationY = (velocityY - velocityYPrevious) / 0.02f;
    accelerationX = (velocityX - velocityXPrevious) / 0.02f;
    accelerationZ = (velocityZ - velocityZPrevious) / 0.02f;*/
```

(DIY'd grass to cover up asset leakage)



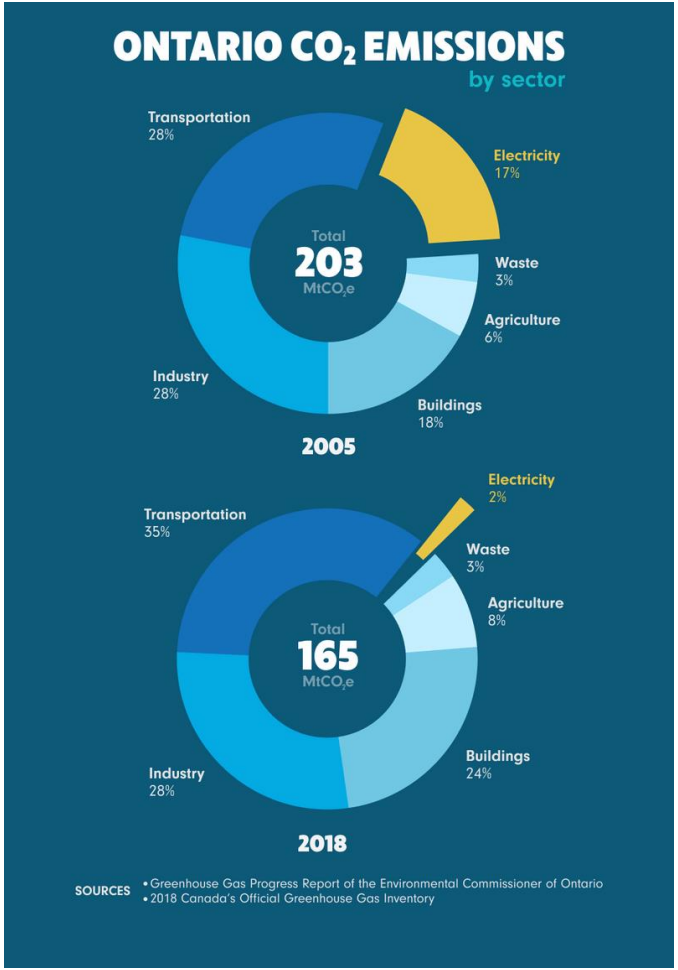
7 - Design Process



The final design was a **near-perfect match of the whiteboard design**. **The general layout, obstacles, and physics principles remained the same**. However, some **minor changes were made**, like the ramp into the jump not being angled upwards because that would result in a violent landing. Furthermore, it was during the track design process, within the Unity game engine, that **testing could be done to help decide the final specifications**. Overall, the process was perfect, as it allowed for constant collaboration and multiple tasks to work on at the same time. Some changes could be made to the design, like smoother transitions from the ramp to the flat turn. Additionally, visual improvements and additional gameplay features, like multiplayer, can be added. However, these tasks were low-priority as they have no impact on the physics principles and core gameplay.

8 - Environmental Impact

- The design of the sport was developed with sustainability in mind. For starters, the model for the project was made **electronically**. Therefore, when about **94% of Ontario's power is emissions free,** the development was immensely responsible and low in resource consumption and greenhouse gas emissions.
- In addition to this, the tangible design of the project incorporated an **engineless car on a dirt slope down a hill**. Thus, the sport requires **no fuel, nor a significant amount of asphalt production**. This means that the components of gravity accelerate the car, and track **has a small impact on the environment** since there is only a **small stretch of asphalt** on the turn of the track. This was incorporated for **driver safety,** as the turn demanded a reliable amount of friction to prevent unsafe slipping.
- Constructing the jump requires **slight terraforming by digging a hole** for the car to jump over.
- Therefore, the sport requires **some destruction of wildlife** at the exact location of the track.
- Therefore, **the development and concept are mostly eco-friendly**, with the only exceptions towards the end of the track for safety.

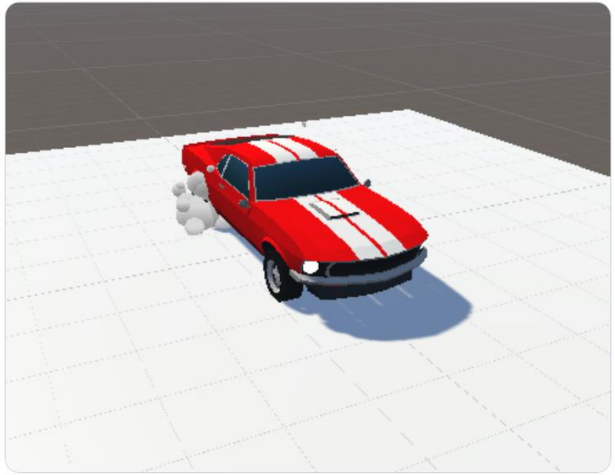


Ontario's Clean Energy Advantage: Home. Ontario's Clean Energy Advantage | Home. (n.d.). Retrieved April 3, 2023, from https://www.ontarioscleanenergyadvantage.ca/#:~:text=Today%2C%20clean%2C%20efficient%20and%20reliable,94%20per%20cent%20emissions%20free.

Social Media Posts

Extremer Barbie Car Racing
@HumberGames

The engine-less "prometheus", has been designed and spawned into the world. May his gravity be accurate, and non-existent exaust be eco-friendly.



12:00 PM · Mar 27, 2023

1.2K Retweets **532** Quote Tweets **45.5K** Likes

Extremer Barbie Car Racing
@HumberGames

The final code has been prepared. The car now turns, accelerates (due to gravity), and hits the cleanest projectile motion launch the laws of physics, realism, (and even safety) demand!

```
//The following method turns the front car wheels to the right. The speed of this movement will depend on the steeringSpeed variable.
public void TurnRight(){
  steeringAxis = steeringAxis + (Time.deltaTime * 10f * steeringSpeed);
  if(steeringAxis > 1f){
    steeringAxis = 1f;
  }
  var steeringAngle = steeringAxis * maxSteeringAngle;
  frontLeftCollider.steerAngle = Mathf.Lerp(frontLeftCollider.steerAngle, steeringAngle, steeringSpeed);
  frontRightCollider.steerAngle = Mathf.Lerp(frontRightCollider.steerAngle, steeringAngle, steeringSpeed);
}

//The following method takes the front car wheels to their default position (rotation = 0). The speed of this movement will depend
// on the steeringSpeed variable.
public void ResetSteeringAngle(){
  if(steeringAxis < 0f){
    steeringAxis = steeringAxis + (Time.deltaTime * 10f * steeringSpeed);
  }else if(steeringAxis > 0f){
    steeringAxis = steeringAxis - (Time.deltaTime * 10f * steeringSpeed);
  }
  if(Mathf.Abs(frontLeftCollider.steerAngle) < 1f){
    steeringAxis = 0f;
  }
  var steeringAngle = steeringAxis * maxSteeringAngle;
  frontLeftCollider.steerAngle = Mathf.Lerp(frontLeftCollider.steerAngle, steeringAngle, steeringSpeed);
  frontRightCollider.steerAngle = Mathf.Lerp(frontRightCollider.steerAngle, steeringAngle, steeringSpeed);
}
```

12:04 PM · Mar 27, 2023

122.4K Retweets **534.3K** Quote Tweets **54.6M** Likes

Social Media Posts

Extremer Barbie Car Racing
@HumberGames

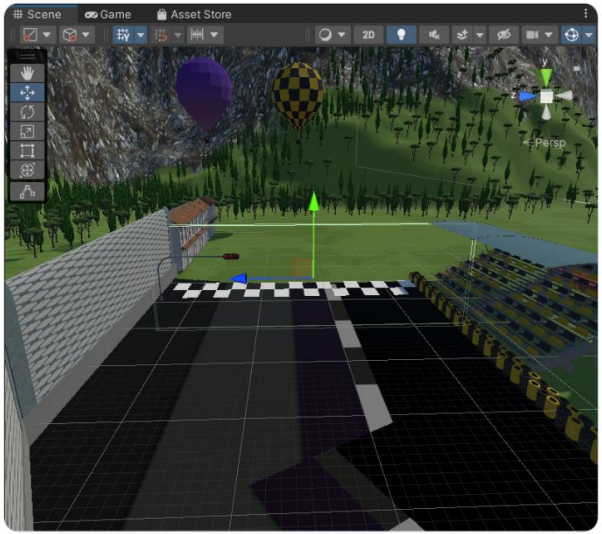
The track has been completed and mastered by yours truly! The angles are both safe and realistic, the track awe-inspiring, and the sky full of free asset balloons. Besides, whats an extreme sport without a giant pylon that doubles as a radius base for centripetal acceleration.

12:04 PM · Mar 28, 2023

7.1M Retweets **5.4M** Quote Tweets **556.5M** Likes

Extremer Barbie Car Racing
@HumberGames

After a gruelling 2 days, the magnum opus of gaming is complete. This final touch: figuring out how to make a checkpoint to reset prometheus. There is now unlimited, EBCR fun. I'd like to think ths made Texas proud. "Extremer Barbie Car Racing", out 2023/04/03 1:20AM Eastern.



1:21 AM · Apr 3, 2023

8.2M Retweets **7.6M** Quote Tweets **2.3B** Likes

References

DiGiuseppe, M., Stewart, C., Bruni, D., Dick, G., Speijer, J., & Stewart, C. (2012). *Physics 12*. Nelson Education.

Ontario's Clean Energy Advantage: Home. Ontario's Clean Energy Advantage | Home. (n.d.). Retrieved April 3, 2023, from https://www.ontarioscleanenergyadvantage.ca/#:~:text=Today%2C%20clean%2C%20efficient%20and%20reliable,94%20per%20cent%20emissions%20free.

YouTube. (2019). *Most Insane Barbie Jeep Racing Of All Time!! YouTube*. Retrieved April 4, 2023, from https://www.youtube.com/watch?v=BrFIsgRIIZ8.