CONTENTS

GAME OVERVIEW

INTRODUCTION

Glacier Rush is an ice-themed racing game. The player is driving through an icy terrain, enjoying the view, racing his opponents by avoiding the obstacles placed throughout the road and also achieving the best possible lap times.

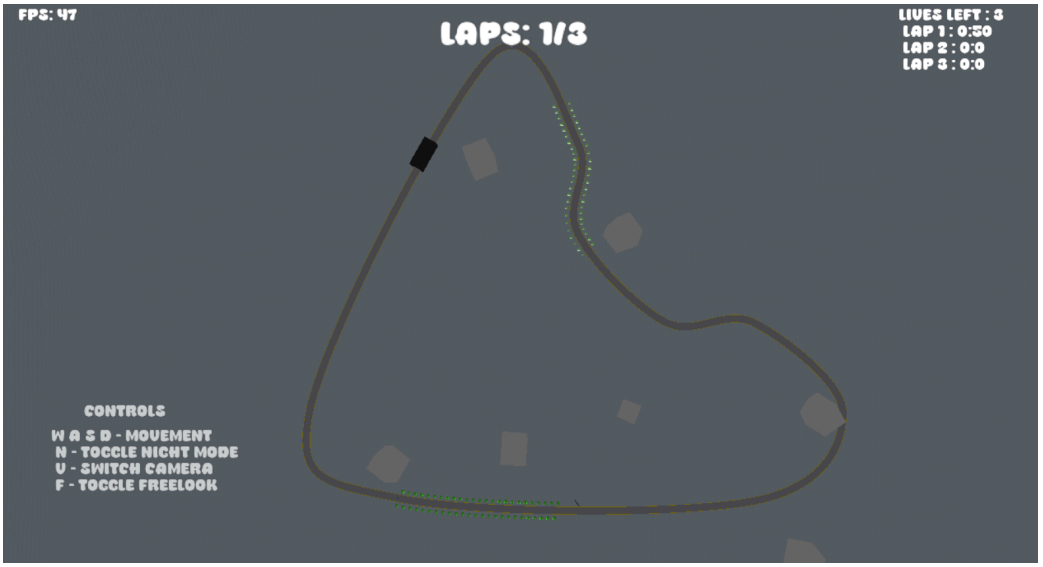
This is a prototype and is a solid foundation to build upon. Most of the bugs in the game have been fixed and all the parameters are editable. You can change the type of lighting using a boolean that you can pass through the shader to render it using normals for each face or average the vertices to get a more realistic feel. You can edit the track by inserting a point and all the objects on the track will be changed according to the track position or elevation. The objects on the track and the trees can be added just by inserting a new position into the object's vector respectively.

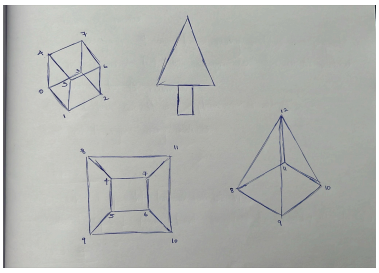
Different rendering techniques have been used to make the game more immersive. Explosion effect using geometry shaders, a TV screen that you can use to look at the race in freelook mode when the cars are racing through the fog.

Collision is based on distance. Any object you would like to add as a pickup or the player to collide into, its position can be added to the m_collidables variable and the collision is complete. As simple as that. There are collisions with barricades and other cars as well. So don’t forget to keep your car moving, we don’t want any accidents on the road.

This prototype is built with the mindset of serving as a template code for any future racing simulators.

PROTOTYPE





CONTROLS

W	Increase Speed
S	Decrease Speed
A	Move Left
D	Move Right
F	Toggle Freelook
V	Switch Camera View
N	Toggle Night Mode
R	Restart or Respawn

IMPLEMENTED REQUIREMENTS

ROUTE

- The track is generated programmatically using Catmull-Rom splines, ensuring smooth non-linear curves. A reference of the F1 Monzo track was used.
- Primitives have been created based on the centreline to visualise the track and appropriate textures have been applied.
- The edge of the route is again generated programmatically using the same line but with a certain width from both the sides of the track. A different texture has been applied to this edge to make it stand out.

CAMERA / VIEWING

- There are three different views for the camera
  - ❖ First Person
  - ❖ Third Person
  - ❖ Top View

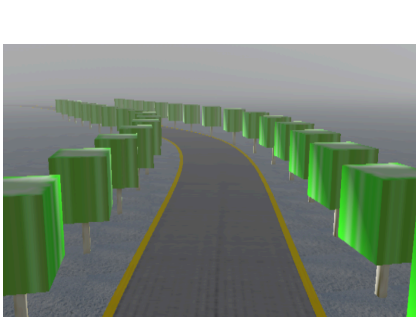
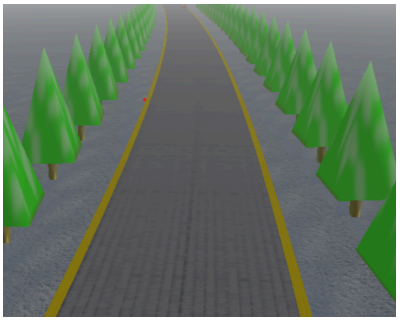
- The game also has a ‘Free Look’ camera that helps you debug or also watch the race in front of the TV in the game.
- The camera viewing geometry utilises the TNB frame in Glacier Rush.

```
m_playerT = glm::normalize(pNext - p);
m_playerN = glm::normalize(glm::cross(m_playerT, y));
m_playerB = glm::normalize(glm::cross(m_playerN, m_playerT));
```

- There is also a camera that constantly moves behind the car to render the texture to the TV.

BASIC OBJECTS

- There are two different basic objects (Trees) from primitives in the scenes. Appropriate textures and normals have been applied for these objects. One of the trees has a pyramid on top of the bark while the other one has a cube.

- These basic objects have been placed in two different parts of the map in a group on both the sides of the track using the offset points generated by the spline.

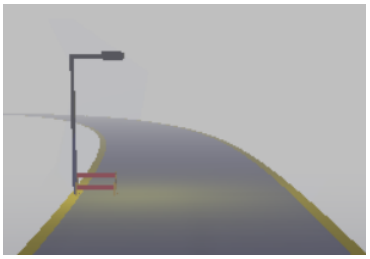
```
//Create the edge for the road
auto centreLinePoints = m_pCatmullRom->m_centrelinePoints;
auto rightOffsetPoints = m_pCatmullRom->m_rightOffsetPoints;
auto leftOffsetPoints = m_pCatmullRom->m_leftOffsetPoints;
for (int i = 0; i < rightOffsetPoints.size(); i++)
{
    rightOffsetPoints[i] += ((centreLinePoints[i] - rightOffsetPoints[i]) * 2.05f);
    leftOffsetPoints[i] += ((centreLinePoints[i] - leftOffsetPoints[i]) * 2.05f);

    //Set the tree positions
    if (i < 50 || (i > 250 && i < 300))
    {
        if (i % 2 == 0)
        {
            m_tree_positions.push_back(rightOffsetPoints[i] + (centreLinePoints[i] - rightOffsetPoints[i]) * 3.f);
        }
        else
        {
            m_tree_positions.push_back(leftOffsetPoints[i] + (centreLinePoints[i] - leftOffsetPoints[i]) * 3.f);
        }
    }
```

- The textures for these trees were made using Photoshop.

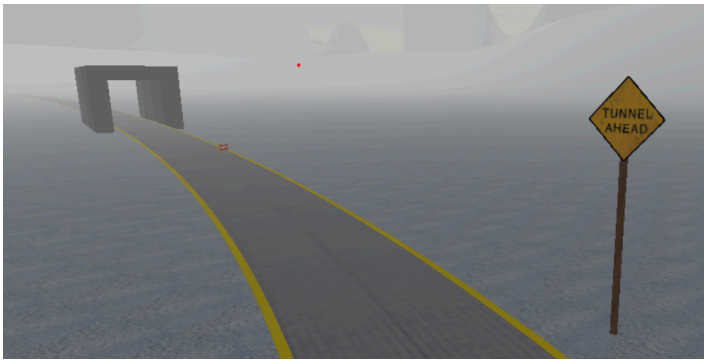
MESHES

- Seven different texture-mapped mesh models are added to the scene
    - ❖ Cars
    - ❖ Barricades
    - ❖ IceBergs
    - ❖ Snow man
    - ❖ Street Lamp
    - ❖ Tunnel Sign
    - ❖ Tunnel
- One of the Car models is the player which is used to record the laps. There are other cars which the player will be able to race with.
- The Barricades model are the obstacles for the player that has been placed repeatedly along the route and has been developed using Blender
- There are snowmen in the game to lighten the mood for the viewers. They vary in sizes from being giants to lilliputs. Keep an eye for any easter eggs.

- The Street Lamp models are used to light up the track in dark mode to make barricades mode easier to see.



- The Tunnel Sign model is a traffic sign saying “Tunnel Ahead” and the tunnel model is the tunnel ahead.



LiIGHTING

- General lighting has been used for most objects and there is also an option to switch between bright and dark modes and this is done by changing the ambient, specular, diffuse components of light

```
//Light properties for Light and Dark Modes
float la, ld, ls, ma;
if (m_gameMode == Light)
{
    la = 0.5f;
    ld = 0.9f;
    ls = 0.3f;
    ma = 1;
}
else
{
    la = 0.3f;
    ld = 0.3f;
    ls = 0.3f;
    ma = 0.7f;
}
```

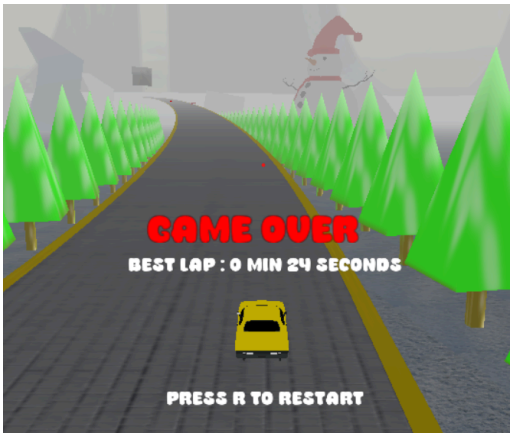
- There are six different spotlights that act as headlights to the cars. These lights will be turned on only in dark mode and all of these lights have a non-white colour. These lights are being rendered using the Blinn-Phong Spotlight model under the main shader. The main shader has a boolean and two different parameters that hold the light info for these spotlights. The boolean is used to turn these spotlights on and off.
- There are spotlights that are attached to street lamps as well that turn on in the dark mode. These spotlights are sent as an array to the main shader.

HUD

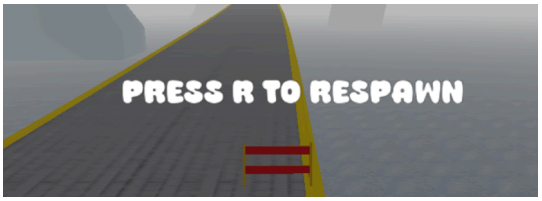
- A Head's up display has been implemented to show the number of lives remaining, number of laps and the lap times
- There is also a Controls HUD at the bottom left of the screen

- The player will have three laps and their best time will be

shown at the end with the Gameover screen.



- There is a Restart and Revive screen that pops up when you are out of tries or you crash the car



- The font for all of these screens has been changed

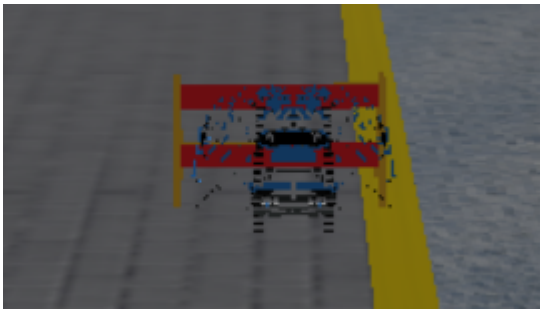
GAMEPLAY

- The car is the player and can be controlled using the keyboard. Whenever the player is moved to the left or right, the car turns a specific amount along its up vector showing the direction and that it is about to turn.
- The player has to complete three laps as soon as possible by overcoming the obstacles and avoiding other cars along the route and their best time will be visible at the game over screen
- The car has a property m_health which is set to the minimum for the prototype but when you set it a bit higher, every obstacle you hit will reduce its speed instead of destroying the car. This can be used later on if we plan on adding different types of cars with abilities.

- The player will be given 3 tries to complete these 3 laps and if failed, they could restart anytime
- Collision has been implemented by calculating the distance in between them. The cars can collide between themselves and the player can also collide with the barricades placed along the track.

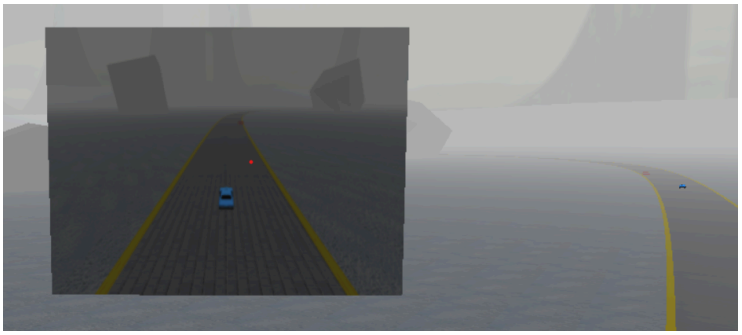
ADVANCED RENDERING

- Explosion effect : A separate shader has been implemented for the car. This shader has a uniform variable that is a boolean when set to true will explode the car and an explode factor that will keep increasing when the car is exploding. So basically the exploding factor is the speed at which the exploding should occur. A geometry shader has been utilised to implement this. So when the boolean is set to true, the basic primitives of the car will start moving in the direction of their normals and they will be scaled down uniformly until the scale hits zero.



- TV : You will be able to see the race on the TV when you are in your free look mode. A new camera has been attached to the car in the third person position and this camera renders the texture to the FBO. Later the main shader renders a plane and the texture from the FBO is applied to this plane. So even if we use the freelook camera and move around, we will still be able to see the

race perfectly from the TV.



- Fog effect : For this effect, we have two parameters, rho and fog colour. Fog colour basically indicates the colour of the fog and rho shows the density of the fog. The fog is calculated using e^(-rho * d) showing that the fog thickens as the distance is farther from the camera where d is the distance from the camera. It also has a boolean which will basically thicken the fog in dark mode.

DISCUSSION

- Engagement : Glacier Rush is a racing game in an icy terrain but it lacks the feel to keep the player engaged. Adding more elements and effects to keep the player engaged is required.
- Template : This code gives a solid foundation to build racing simulators upon. It is polished and most of the bugs have been fixed.
- Competitiveness : The game has a sense of competitiveness, but adding more elements, like power ups and pickups will get more engagement.
- Night mode and fog colour : The fog colour remains the same when the night mode is toggled. Changing the fog colour according to the global light setting will be more immersive.
- Car movement : The car movement is snappy right now as it is being changed with position, if the player position is interpolated between points, the movement will be smoother.
- Additional effects : Addition of effects like suspension for the car, smoke to leave trail behind the car etc. will add realism to the game.
- Multiplayer : The game although competitive, it would be better if PVP is added to the game. This will increase the player count and also engagement.
- Collision : The collision at the moment is based on distance and the position of the objects. Instead a separate class for any collidable objects will give more control over the code and will maintain a specific pattern.