Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Mini-Project Report On:

"Traffic Loop Game"

[Code No: COMP 342]

Submitted by

Chandan Kumar Mahato - 31

Submitted to

Mr. Dhiraj Shrestha

Department of Computer Science and Engineering

Submission Date: 05/18/2023

Contents

List of Figures:

1. Introduction

Traffic Loop is a racing game built using HTML, CSS, JavaScript and Three.js, which utilizes WebGL technology. The game is designed to give players an exciting experience of driving a vehicle on a race track that is in the shape of the digit 8. Player controls the vehicle on the left side of the track while the computer generates vehicles automatically on the right side. As the game progresses, more and more vehicles will appear on the computer’s side. Players must avoid colliding with these vehicles and complete each lap to increase their score. The development of game helped in understanding of computer graphics, WebGL, related libraries, and other tools and technologies used in game development.

2. Library and Language Used

To develop Traffic Loop, I have used a combination of tools and technologies, including HTML, CSS, JavaScript, Three.js, WebGL.

HTML and CSS:

HTML (Hypertext Markup Language) is the standard markup language used to create web pages. It provides the structure of a web page, defining elements such as headings, paragraphs, and links. CSS (Cascading Style Sheets) is used to define the visuals appearance of a web page, such as the layout, colors, and fonts.

In Traffic Loop, HTML and CSS was used to create the user interface and layout of the game. This includes elements such as buttons, text and images, as well as the styling and positioning of these elements on the screen.

JavaScript:

JavaScript is a programming language used to create interactive and dynamic web pages. It is widely used in web development and provides a wide range of features for manipulating web pages, such as event handling animations, and data storage.

In Traffic Loop, JavaScript was used to create the gameplay mechanics, such as controlling the user vehicle, generating computer-controlled vehicles, and detecting collisions. JavaScript was also used to communicate with Three.js and WebGL to create the 3D graphics and animations.

Three.js:

Three.js is a JavaScript library used for creating and displaying 3D graphics in a web browser it provides a range of features for working with 3D models, such as loading and manipulating 3D objects, creating lighting and shadows, and rendering scenes. Three.js uses WebGL under the hood.

In Traffic Loop, Three.js was used to create the 3D models of the racetrack and vehicles, as well as to render the scenes and apply lighting and shadows. Three.js was also used to create animation.

WebGL:

WebGL (Web Graphics Library) is a JavaScript API for rendering 3D graphics in a web browser. It provides a low-level interface for accessing graphics hardware, allowing for high-performance rendering of complex 3D scenes.

In Traffic Loop, WebGL was used to render the 3D graphics created with Three.js providing a high-performance and visually stunning gaming experience. WebGL was used to apply shaders and special effects to the 3D models, such as reflections and textures.

3. Design and Implementation

In developing Traffic Loop, several mathematical functions and geometry was used to achieve and desired effects. For instance, the sin and cosine functions were used to create the motion of the vehicles and generate the turning angles. Pythagoras' theorem was used to calculate the distance between the user's vehicle and the computer-generated vehicles to detect collisions. Additionally, geometry was used to create the 3D shapes and vehicles and the race track.

Source Code: https://github.com/chandankmahato/trafficloop

3.1 Camera Setup

Orthographic camera setup:

```
const aspectRatio = window.innerWidth / window.innerHeight;
const cameraWidth = 1500;
const cameraHeight = cameraWidth / aspectRatio;
const camera = new THREE.OrthographicCamera(
  cameraWidth / -2,
  cameraWidth / 2,
  cameraHeight / 2,
  cameraHeight / -2,
  50,
  700
);
camera.position.set(0, -210, 300);
camera.lookAt(0, 0, 0);
```

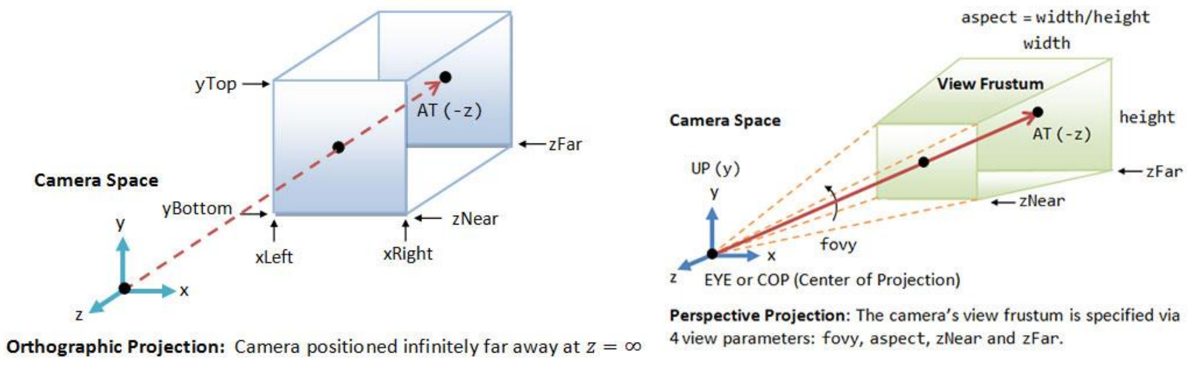

yTop
AT (-z)
zFar
Camera Space
yBottom
y
x
z
xLeft
xRight
zNear
Orthographic Projection: Camera positioned infinitely far away at z = ∞
aspect = width/height
width
View Frustum
Camera Space
height
AT (-z)
UP (y)
y
zFar
zNear
fovy
x
z
EYE or COP (Center of Projection)
Perspective Projection: The camera's view frustum is specified via 4 view parameters: fovy, aspect, zNear and zFar.

Figure 1 Orthographic and Perspective Projection

3.2 Racetrack

These images are showing the calculations for the race track:

```
trackRadius = 225;
trackWidth = 45;
innerTrackRadius = trackRadius - trackWidth;
outerTrackRadius = trackRadius + trackWidth;

arcAngle1 = (1 / 3) * Math.PI;
deltaY = Math.sin(arcAngle1) * innerTrackRadius;
arcAngle2 = Math.asin(deltaY / outerTrackRadius);

arcCenterX =
(Math.cos(arcAngle1) * innerTrackRadius +
 Math.cos(arcAngle2) * outerTrackRadius) /2;

arcAngle3 = Math.acos(arcCenterX / innerTrackRadius);
arcAngle4 = Math.acos(arcCenterX / outerTrackRadius);
```

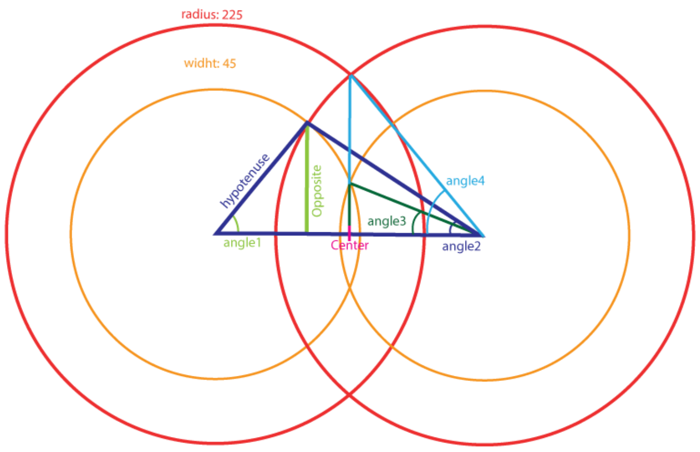


Figure 2 :Geometry for racetrack

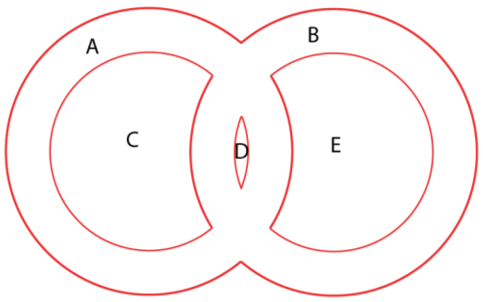
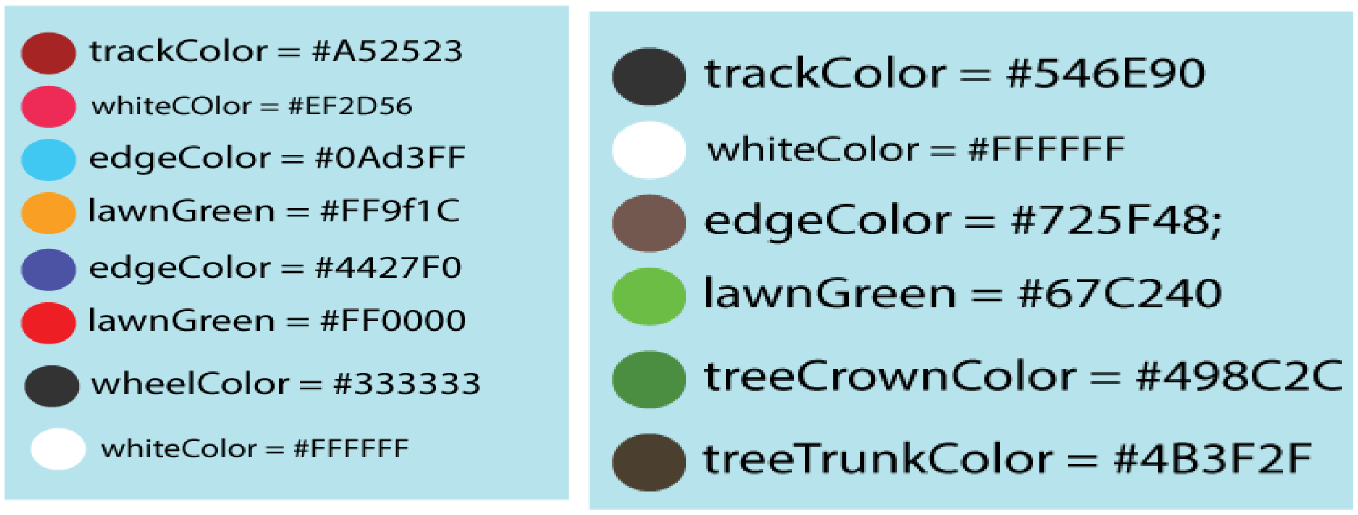


Figure 3 : Racetrack Outline

The colors used in the traffic loop game:



●	trackColor = #A52523
●	whiteCOlor = #EF2D56
●	edgeColor = #0Ad3FF
●	lawnGreen = #FF9f1C
●	edgeColor = #4427F0
●	lawnGreen = #FF0000
●	wheelColor = #333333
●	whiteColor = #FFFFFF

●	trackColor = #546E90
●	whiteColor = #FFFFFF
●	edgeColor = #725F48;
●	lawnGreen = #67C240
●	treeCrownColor = #498C2C
●	treeTrunkColor = #4B3F2F

Figure 4 : Colors Used

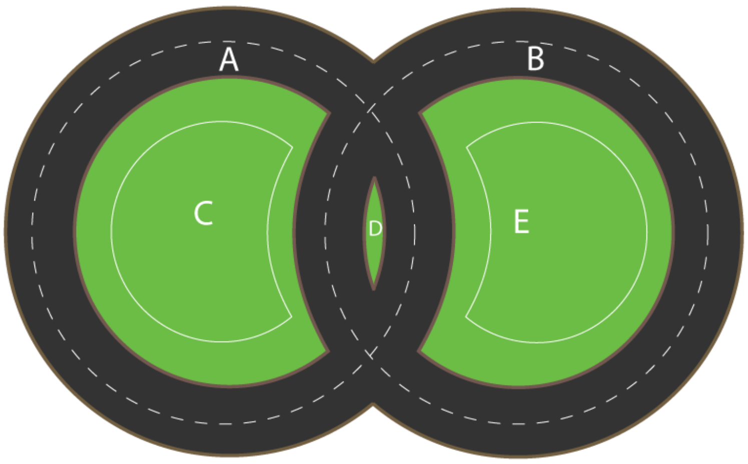


Figure 5 : Final Racetrack

3.3 Car

These figures are showing the making of car for the traffic loop game:

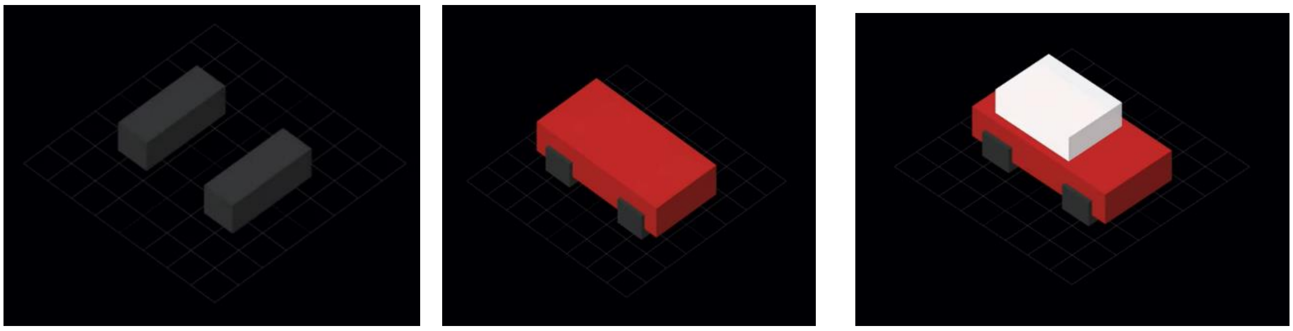


Figure 6 : Car

3.3.1 Car Position

These figures are showing the position and orientation of car on the racetrack:

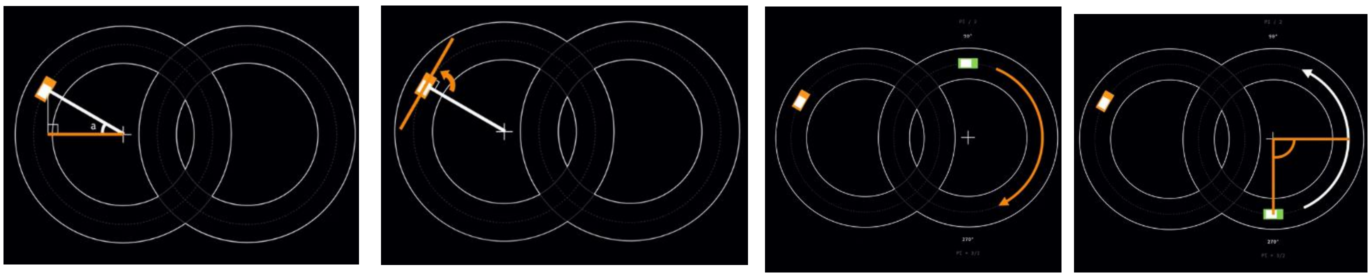


Figure 7 : Car Positioning

3.3.2 Car Collision

The figure showing the car collision on the racetrack:

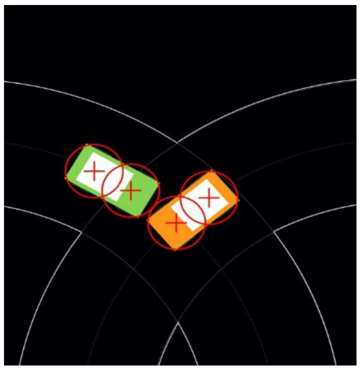


Figure 8 : Cars Collision

3.5 Canvas

This figure is showing the canvas rendered for the racetrack and background.

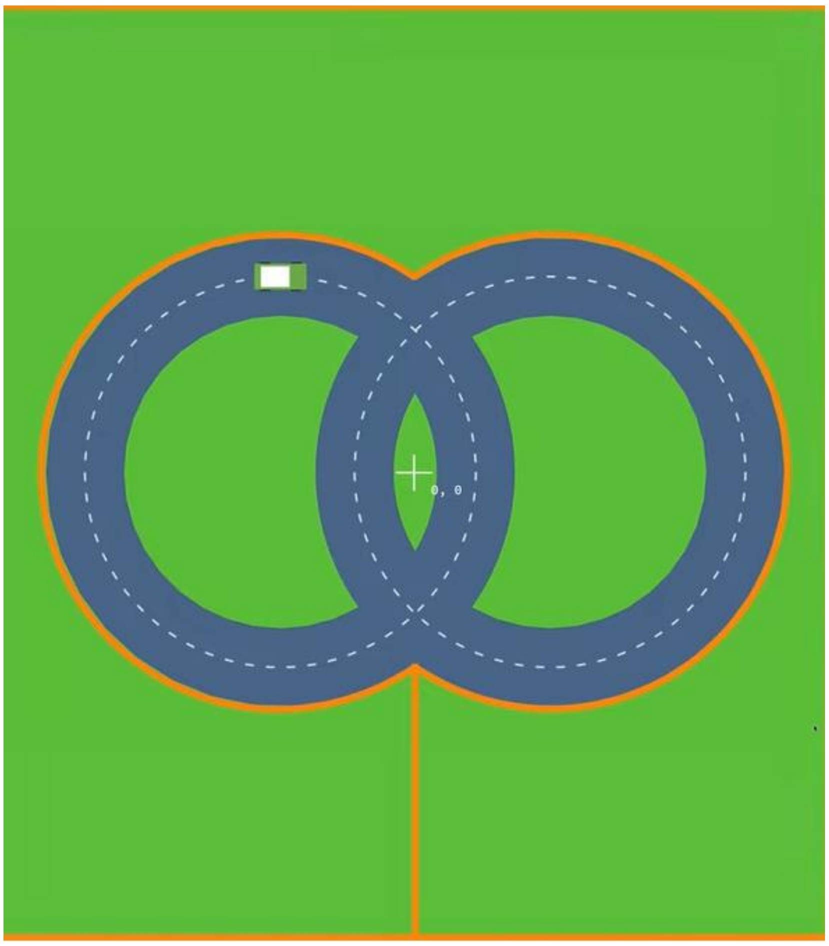
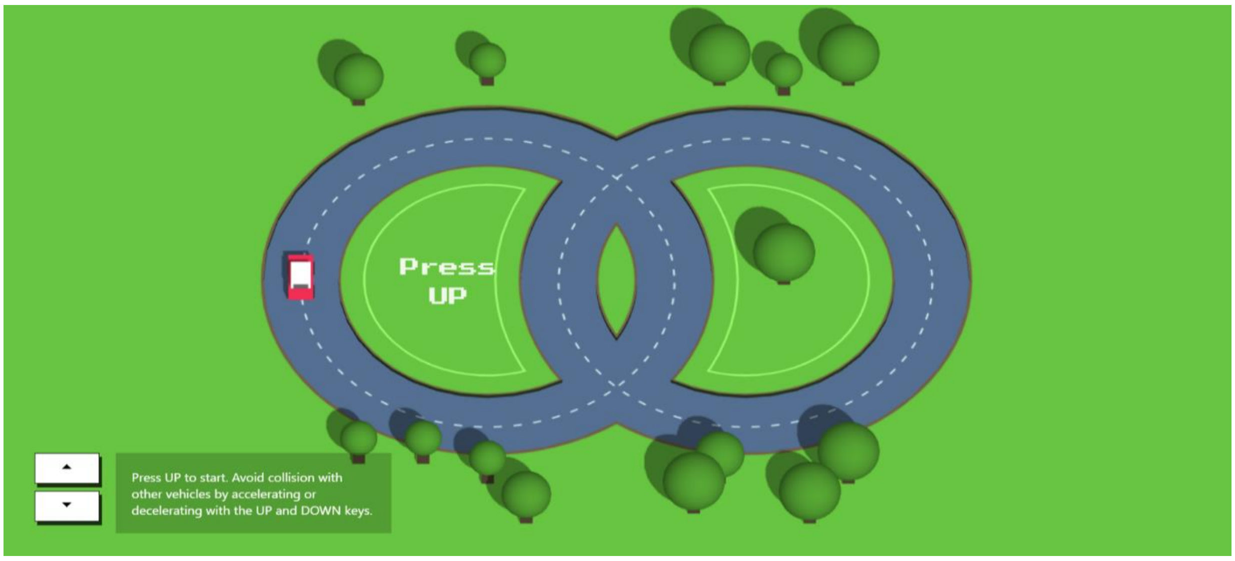


Figure 9 : Canvas with Racetrack & Car

4. Output



Press UP to start. Avoid collision with other vehicles by accelerating or decelerating with the UP and DOWN keys.

Figure 10 : Traffic Loop Game

Website: https://trafficloop.netlify.app

Demo Video: https://drive.google.com/file/d/1mCb5EkA4mLttmci4kYeNyPLRtwiwrbBn/view

5. Conclusion

Traffic Loop is a game that was developed using HTML, CSS, JavaScript, and Three.js, utilizing WebGL technology. This game provides players with thrilling racing experience, where they must avoid collisions with computer-generated vehicles and complete each lap to increase their score. The project aims to provide a comprehensive understanding of computer graphics, WebGL, related libraries, and tools and technologies used in game development. Mathematical functions and geometry were used to achieve the desired effects.

6. References

Three.js (2023). From dart.dev: https://threejs.org/

WebGL (2023). From dart.dev: https://www.khronos.org/webgl/wiki/Main_Page

JavaScript (2023). From dart.dev: https://devdocs.io/javascript/