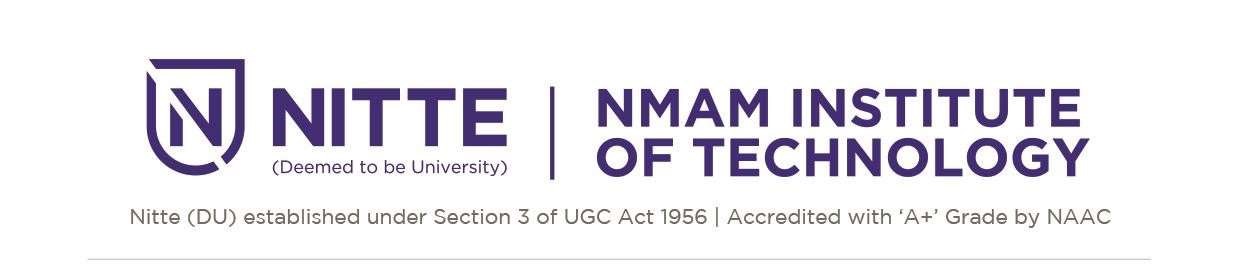
Nuclear Power Reactor



Department of Computer Science and Engineering

Report on Mini Project

3D – CAR RACE GAME

Course Code: 20CS603

Course Name: Computer Graphics

Semester: VI SEM Section: C

Submitted To:

Dr. Sannidhan M S

Associate Professor

Department of Computer Science and Engineering

Submitted By:

Rajesh C Shettigar – 4NM20CS142

Shaina Jyothica Crasta – 4NM20CS160

Date of submission: 08 -May -2023

Signature of Course Instructor

# **ABSTRACT**

The 3D Car Race Game is an immersive and dynamic project developed using OpenGL and GLUT libraries, featuring realistic gameplay elements, including audio backgrounds, dynamic speed variations, and a day-night mode. The objective of the game is to navigate a car through a challenging race track, avoiding collisions with obstacles while aiming for high-speed excitement. Leveraging the power of OpenGL and GLUT, the project implements visually appealing 3D graphics, smooth car movement, and accurate collision detection. The methodology involved designing the car model, creating a diverse race track environment, generating obstacles, and incorporating user input for car control. In addition to the visual experience, audio backgrounds complement the gameplay, enhancing the immersion factor. The car speed progressively increases over time, creating a thrilling sensation, while manual speed control is also available. The inclusion of a day-night mode, triggered by the 'n' and 'd' keys, adds an aesthetic dimension, influencing the lighting and atmosphere of the game. Furthermore, horn sound effects can be activated using the 'a' and 's' keys. The results showcase an engaging gaming experience, combining visual and audio elements seamlessly. Challenges during the project included optimizing audio integration, balancing speed progression, and achieving a smooth transition between day and night modes. The project demonstrates the potential of OpenGL and GLUT in creating dynamic and captivating 3D games, highlighting the importance of immersive audio and dynamic gameplay elements. Future enhancements could involve additional audio effects, customizable game settings, and further visual enhancements to elevate the overall gaming experience.

# TABLE OF CONTENTS

Title Page ................................................................................................................................ i

Abstract .................................................................................................................................. ii

Table of Contents ................................................................................................................. iii

Introduction ........................................................................................................................... 1

Problem statement ................................................................................................................. 2

Objectives .............................................................................................................................. 2

Hardware/Software Requirement .......................................................................................... 3

Methodology .......................................................................................................................... 4-7 Implementation Details ...........................................................................................................8-9

Results ....................................................................................................................................10-11

Conclusion and Future Scope .................................................................................................12

References .............................................................................................................................. 13

# **INTRODUCTION**

Computer Graphics has revolutionized the world of gaming by enabling the creation of immersive and visually stunning virtual environments. In line with this, our project focuses on developing a 3D car race game using the powerful OpenGL and GLUT libraries. The objective of the game is to provide players with an exciting and challenging experience as they navigate a virtual race track, avoiding collisions with obstacles while aiming for high speeds. By leveraging the capabilities of OpenGL and GLUT, we aim to create a visually appealing game with realistic graphics, smooth car movement, and accurate collision detection.

The motivation behind this project lies in the fascination with racing games and the desire to explore the potential of computer graphics in creating dynamic and engaging gameplay experiences. By utilizing OpenGL, we can harness its rendering capabilities to generate lifelike 3D visuals, incorporating lighting effects, transformations, and textures to create a captivating virtual world. GLUT, on the other hand, provides a user-friendly interface for window creation, input handling, and event management, facilitating the development of a responsive and interactive gaming environment.

Throughout this project, we will discuss the various steps and techniques employed to achieve our objectives. This includes designing the car model, creating the race track environment, generating obstacles, implementing smooth car movement, and integrating accurate collision detection algorithms. Moreover, we will explore the utilization of audio backgrounds to enhance the immersive experience and add depth to the gameplay. Additionally, the introduction of a day-night mode and interactive sound effects, such as the horn, further contribute to the realism and excitement of the game.

Through this project, we aim to showcase the potential of computer graphics in gaming and highlight the significance of OpenGL and GLUT in creating visually stunning and interactive 3D games. The implementation challenges we encountered and overcame, such as optimizing performance, achieving smooth transitions, and balancing gameplay elements, will also be discussed. Ultimately, this project serves as an exploration of the possibilities within computer graphics and provides a foundation for further enhancements and developments in the field.

# **PROBLEM STATEMENT**

The objective of this project is to develop a 3D car race game using OpenGL and GLUT that provides an immersive and challenging gaming experience. The project aims to address several key challenges in computer graphics, including creating realistic 3D graphics, implementing smooth car movement, accurate collision detection, and integrating dynamic elements such as a day-night mode and interactive sound effects. The main problem to be addressed is how to utilize the capabilities of OpenGL and GLUT to design and develop a visually appealing game environment where players can control a car, navigate a race track, and avoid collisions with obstacles while aiming for high speeds. Additionally, the project seeks to explore the efficient use of resources to ensure optimal performance and responsiveness of the game. By addressing these challenges, the project aims to showcase the potential of computer graphics in gaming and demonstrate the effectiveness of OpenGL and GLUT in creating immersive and visually engaging 3D games.

# **OBJECTIVES**

* Design and implement a visually appealing 3D race track environment using OpenGL and GLUT.
* Create a realistic car model with smooth movement and controls, utilizing transformations and animations.
* Develop accurate collision detection algorithms to ensure the car avoids obstacles and maintains its position on the race track.
* Implement a progressive speed variation system that increases the car's speed over time, providing an exciting gameplay experience.
* Incorporate a day-night mode that dynamically changes the lighting and atmosphere of the game environment.
* Integrate interactive sound effects, such as horn sounds, to enhance the immersive experience and add realism to the gameplay.
* Optimize performance by efficiently utilizing resources and implementing rendering techniques to ensure smooth and responsive gameplay.
* Provide an intuitive user interface that allows players to control the car using keyboard inputs and interact with various game features.
* Test and debug the game thoroughly, addressing any issues related to graphical rendering, collision detection, and gameplay mechanics.
* Demonstrate the completed project, showcasing the visual appeal, smooth gameplay, and immersive experience of the 3D car race game.

# **HARDWARE / SOFTWARE Requirements**

To run an OpenGL program, your computer must meet these minimum requirements:

1. Operating System: Windows, Linux, or macOS
2. Graphics card: Your graphics card must support OpenGL 3.3 or higher. You can check your graphics card specifications to ensure it supports the required OpenGL version.
3. Graphics drivers: You must have the latest graphics drivers installed for your graphics card. These can usually be downloaded from the graphics card manufacturer's website.
4. Development Environment: You'll need a development environment such as Code::Blocks, Visual Studio, or Eclipse that supports OpenGL.

Code::Blocks need these minimum system requirements to run:

* Operating System: Windows, Linux, or macOS
* Processor: 1 GHz Intel or AMD processor or higher
* RAM: 512 MB or higher
* Hard Disk Space: 100 MB of free space or higher
* Graphics Card: 1024x768 resolution or higher
* Optional: A C++ compiler, such as GCC, to build and run C++ programs

# **METHODOLOGY**

1. Requirements Analysis:

* Identify the project requirements and objectives, including the desired gameplay features, graphical elements, and interactive elements such as audio effects.
* Determine the scope of the project and define the target audience.

2. Design Phase:

* Plan the game's overall structure, including the game flow, user interface, and key components such as the car model, race track, and obstacles.
* Create concept art or sketches to visualize the game's graphical elements and aesthetics.
* Design the game mechanics, including the progressive speed variation system and collision detection algorithms.

3. Environment Setup:

* Install and configure the necessary development environment, including OpenGL and GLUT libraries.
* Set up the project structure, including source code organization and build configurations.

4. OpenGL and GLUT Integration:

* Initialize the OpenGL and GLUT environment.
* Set up the main game window, viewport, and projection settings.
* Implement rendering functions to display the game assets on the screen.
* Apply lighting and texturing effects to enhance the visual quality of the game.

5. Game Graphics creation and rendering:

* Create methods to render car, roads, houses, sky, trees etc.
* Methods for dynamic user interaction are created.

6. Car Movement and User Input:

* Set up keyboard callbacks to handle user input for car control (e.g., arrow keys for acceleration and steering).
* Implement car movement functions that update the car's position and orientation based on user inputs.
* Ensure smooth and responsive movement by utilizing appropriate transformation functions.

7. Collision Detection:

* Develop collision detection algorithms to detect collisions between the car and obstacles on the race track.
* Determine collision conditions based on bounding boxes, collision meshes, or other techniques.
* Handle collision events by updating the game state (e.g., reducing car speed or triggering game over).

8. Gameplay Dynamics:

* Implement the progressive speed variation system to increase the car's speed over time or allow manual speed control.
* Develop a scoring mechanism based on the distance covered, time taken, or other criteria.
* Implement a game timer and track lap completion for a comprehensive gameplay experience.

9. Audio Integration:

* Load and play background music or soundtracks to enhance the gaming experience.
* Implement sound effects for user interactions, such as horn sounds triggered by specific key inputs.

10. Testing and Debugging:

* Thoroughly test the game for graphical rendering issues, collision detection bugs, or gameplay glitches.
* Debug and fix any identified issues to ensure smooth and error-free gameplay.

The code contains the following function modules:

* **init()**

Used to initialize the state of the graphics system, such as setting up lighting, material properties, textures, and other parameters. It is called only once at the beginning of the program and is used to prepare the system for drawing graphics on the screen.

* **void drawScene()**

Used to render the game environment graphics. Contains methods sky(), roadside(), houses(), tree(), roadlines(), objectcube() used to render the whole scene of the game.

* **void keyboardown(int key, int x, int y)**

Used to control the movement and speed of the car using arrow keys. It defines the function of the following arrow keys. The Arrow key ‘left’ and ‘right’ is used for the movement of car. Arrow keys ‘up’ and ‘down’ are used to control speed

* **void keyboard(unsigned char key, int x, int y)**

This method allows users to dynamically interact with the game environment. The user can switch between day and night mode. Char key ‘d’ is used for day mode and char ‘n’ is used for night mode. User can also use the keys ‘a’ and ‘s’ to play the car horn. These values change interactively when the user presses the designated keys on the keyboard.

* **bool GameScore()**

This method calculates the score of the player. For every obstacle the player avoids the player’s score increase.

* **void sprint( float x, float y, string st)**

This method enables sprint mode. The sprint mode is enabled automatically after the player score passes a certain cut-off.

* **bool collision()**

This method checks for the collision of the car. The method returns true if the car collides with any one of the obstacles. Else it returns false.

* **int winner(char a)**

If the collision() method returns true the winner() method is called .This displays that the player has lost and the game has come to an end.

* **display()**

contains all the code for displaying various components in the 3d space along with the animations used to sow the functionalities. It is called whenever a refresh of the screen is required, such as when the window is resized or moved, or when the content of the screen needs to be updated

* **main()**

this function contains the function calls for init and display functions and all other calls for creating event loops to respond to user input and update the display. The ‘main’ function is the starting point of a C++ program

# **IMPLEMENTATION**

To install Code::Blocks on your computer, you can follow these steps:

1. Go to the Code::Blocks website at [http://www.codeblocks.org/downloads](http://www.codeblocks.org/downloads%20) and click on the "Download" button for the version that matches your operating system.
2. Once the download is complete, double-click on the downloaded file to start the installation process.
3. Follow the on-screen instructions to complete the installation. You can choose the default settings or customize them according to your needs.
4. When the installation is complete, launch Code::Blocks by double-clicking on the desktop shortcut or by searching for it in the Start menu.
5. If this is your first time using Code::Blocks, you may need to configure it to work with your programming language of choice. To do this, go to the "Settings" menu and select "Compiler" and then "Toolchain executables." Here, you can select the compiler that you want to use with Code::Blocks.
6. Once you have configured Code::Blocks to work with your programming language, you can start coding. Create a new project by going to the "File" menu and selecting "New" and then "Project." From here, you can select the project type that matches your needs, such as a console application or a graphical user interface (GUI) application.
7. Write your code and save your project. When you're ready to compile and run your code, click on the "Build" button in the toolbar or go to the "Build" menu and select "Build and Run."

To install OpenGL in Code::Blocks, you can follow these steps:

1. First, ensure that you have installed a C++ compiler, such as MinGW or GCC, on your computer. Code::Blocks usually come with MinGW pre-installed, but you can also download and install it separately if needed.
2. Next, download the OpenGL libraries that match your operating system. You can download the libraries from the OpenGL website or other sources on the internet.
3. Once you have downloaded the OpenGL libraries, extract the contents to a location on your computer. Note the location of the library files, as you will need to reference them in your Code::Blocks project.
4. Open Code::Blocks and create a new project by going to the "File" menu and selecting "New" and then "Project." From here, you can select the project type that matches your needs, such as a console application or a GUI application.
5. In the project settings, go to the "Linker Settings" tab and add the OpenGL libraries that you downloaded earlier. You can do this by clicking on the "Add" button and browsing to the location of the library files. Add the following libraries: "opengl32", "glu32", and "gdi32".
6. Once you have added the libraries, save your project and build it. If there are no errors, you should be able to use OpenGL in your Code::Blocks project.

Setting up the project folder:

* Copy the project folder to the desired location
* Go to the folder where the .cbp file will be located
* Click on the file and open it with Codeblocks
* The project folder and the associated main file will be opened in Codeblocks IDE

To play the game:

* After opening the project click on run in the build option.
* Press the left and right arrow keys to avoid obstacles.
* Press the up and down arrow keys to control speed.
* Press ‘n’ key to change the settings to night mode .
* Press ‘d’ key to change the settings to day mode.

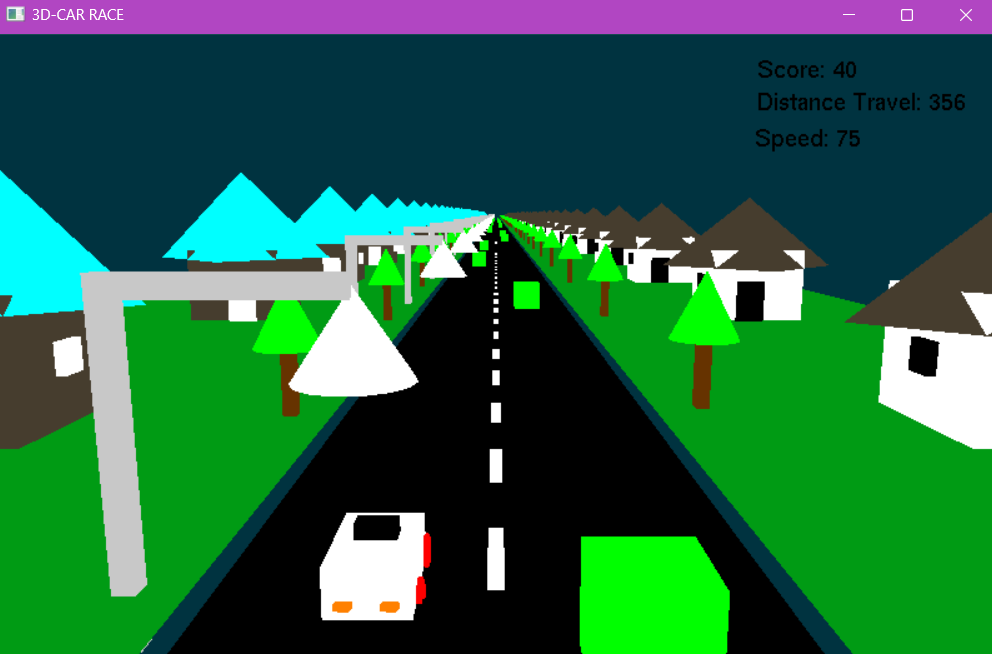
Press ‘a’ and ‘s’ keys for horn. **RESULTS AND DISCUSSIONS**

On starting the application:

On starting the application the initial welcome screen shows up . The game and the graphics get loaded up after this window gets closed.

Player Interaction

The game window is created and the game environment is rendered. The game starts immediately after. The objective of this game is to avoid the obstacles on the road. The player can use ‘left’ and ‘right’ arrow keys to move left and right and ‘top’ and ‘bottom’ keys to control speed.

Mode:

Player can switch between night and day modes. Player can turn on night mode by pressing “n” key and turn on daylight mode by pressing “d” key.

Speed And Score:

Player can use top and bottom arrow keys to control the speed. The game calculates the score of the player based on the number of blocks the player avoids.

Game End :



# **CONCLUSION AND FUTURE IMPROVEMENTS**

Conclusion:

The 3D car race game project using OpenGL and GLUT has been completed successfully. The project implemented a range of features including 3D graphics, smooth car movement, collision detection, dynamic day-night mode, and interactive sound effects. The game provides an immersive and engaging gaming experience and demonstrates the potential of computer graphics in creating visually stunning games.

Future Improvements:

Despite the success of the project, there are several areas where the game could be improved to provide an even better user experience:

1. More Realistic Physics: The current car movement is relatively simple. Enhancing the physics to make the car movement more realistic and true to life would make the game even more enjoyable.

2. Multiplayer Functionality: Introducing multiplayer capabilities would allow players to compete against each other in real-time races, adding an extra level of competition and excitement.

3. Additional Game Modes: Including more game modes, such as time trials or challenges, would add variety and replay value to the game.

4. Advanced Visual Effects: Incorporating advanced visual effects such as particle systems, dynamic lighting, and shadows would take the game's visual quality to the next level.

5. Customizable Car and Track: Allowing players to customize their car with different colors, textures, and performance attributes would give them a greater sense of ownership over their experience.

# **REFERENCES**

1. [https://www.opengl.org/resources/libraries/glut/glut-3.spec.pdf](https://www.opengl.org/resources/libraries/glut/glut-3.spec.pdf%20)
2. [https://www.khronos.org/opengl/wiki/OpenGL\_Reference](https://www.khronos.org/opengl/wiki/OpenGL_Reference%20)
3. (Red book) "OpenGL Programming Guide", 6th Eds, 2008, Addison-Wesley.
4. Edward Angel and David Shreiner "Interactive Computer Graphics - A top-down approach using OpenGL", 6th ed, Pearson Education, 2012.
5. https://learn.microsoft.com/en-us/windows/win32/opengl/opengl-reference