2D Top-Down Car Racing Game

Submitted By

Student Name	Student ID
DIP SARKER	221-15-5274
Md. Sakib Khan	221-15-5728
Mahadi Hasan	221-15-5305

MINI LAB PROJECT REPORT

This Report Presented in Partial Fulfillment of the course **CSE422: Computer Graphics Lab in the Computer Science and Engineering Department**

DAFFODIL INTERNATIONAL UNIVERSITY
Dhaka, Bangladesh

December 14, 2025

DECLARATION

We hereby declare that this lab project has been done by us under the supervision of **Md Mahabul Alom Santo**, **Lecturer**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere as lab projects.

Submitted To:

Md Mahabul Alom Santo

Lecturer

Department of Computer Science and Engineering Daffodil International University

Submitted by

DIP SARKER ID: 221-15-5274 Dept. of CSE, DIU	
Md. Sakib Khan ID: 221-15-5728 Dept. of CSE, DIU	Mahadi Hasan ID: 221-15-5305 Dept. of CSE, DIU

COURSE & PROGRAM OUTCOME

The following course have course outcomes as following:

Table 1: Course Outcome Statements

CO's	Statements
CO1	Apply data mining and machine learning concepts to preprocess, normalize, and analyze data from various domains.
CO2	Implement appropriate data mining and/or machine learning techniques such as classification, regression, prediction, clustering, association rule mining, and artificial neural network to solve complex problems.
CO3	Evaluate the social implications of data mining and machine learning applications and solutions.
CO4	Assess the sustainability and impact of data mining and machine learning solutions on the environment.
CO5	Collaborate effectively in diverse and multidisciplinary teams for data mining and machine learning projects.
CO6	Apply engineering management principles and economic decision-making to plan, execute, and manage data mining and machine learning projects.
CO7	Communicate the results and insights of data mining and machine learning projects using effective reports, design documentation, presentations, and instructions to various stakeholders.

Table 2: Mapping of CO, PO, Blooms, KP and CEP

CO	PO	Blooms	KP	CEP
CO1	PO4	C2, A2, P2	K2 ,K3 ,K4, K8	EP1 ,EP4
CO2	PO5	C3, A3, P3	K2 ,K3 ,K4 ,K6 K8	EP1 ,EP2 ,EP7
CO3	PO6	C5 ,A4 ,P4	KP7	EP1 ,EP4
CO4	PO7	C4 ,A4 ,P4	KP7	EP1 ,EP2
CO5	PO9	C3 ,A4 ,P4	K5, K6	
CO6	PO11	C3 ,A3 ,P3	K5, K7	
CO7	PO10	C4 ,A4, P4	K5, K6 ,K7	

The mapping justification of this table is provided in section **4.3.1**, **4.3.2** and **4.3.3**.

Table of Contents

Chapter 1

Introduction

This chapter introduces the background, motivation, objectives, feasibility, and expected outcomes of the 2D top-down car racing project. It outlines the problem statement and establishes the relevance of computer graphics concepts applied in the project.

1.1 Introduction

Computer Graphics plays a vital role in the development of interactive visual applications such as games, simulations, and visualization systems. With the increasing demand for real-time graphical applications, understanding the fundamentals of 2D rendering, animation, and user interaction has become essential for computer science students.

This project focuses on the design and development of a 2D top-down car racing game using OpenGL with GLUT in C++. The game simulates a moving road environment where the player controls a car, avoids enemy vehicles, and scores points by surviving for a longer duration. The project demonstrates how basic computer graphics concepts such as coordinate systems, geometric primitives, transformations, animation, and collision detection can be combined to create an interactive game application.

The problem addressed by this project is to design a real-time 2D graphical game using low-level graphics primitives, without relying on external game engines, while maintaining smooth animation, player interaction, and logical game flow.

1.2 Motivation

The motivation behind this project is to gain hands-on experience in computer graphics programming by developing a complete interactive application from scratch. Instead of using high-level game engines, this project emphasizes direct use of OpenGL primitives and algorithms, which strengthens understanding of the graphics pipeline.

Developing a car racing game provides a practical and engaging way to apply theoretical concepts such as orthographic projection, animation using timer functions, line drawing algorithms, and collision detection. Additionally, implementing game logic such as scoring, enemy spawning, and restart mechanisms enhances problem-solving and programming skills.

Completing this project improves confidence in graphics programming and provides a strong foundation for advanced topics such as 3D graphics, shader programming, and game development.

1.3 Objectives

The main objectives of this project are:

1. To design a 2D top-down car racing game using OpenGL and GLUT.
2. To implement orthographic projection for 2D rendering.
3. To apply basic geometric primitives (points, lines, triangles, and quadrilaterals) for drawing game objects.
4. To implement the DDA line drawing algorithm.
5. To simulate real-time animation using GLUT timer functions.
6. To implement keyboard-based user interaction for car movement.
7. To perform collision detection between player and enemy cars.
8. To design a scoring system and game-over logic.
9. To gain practical exposure to computer graphics concepts and algorithms.

1.4 Feasibility Study

Several existing 2D racing games and academic projects demonstrate the feasibility of this work. Many simple racing games are developed using game engines or higher-level libraries; however, similar educational projects have been implemented using OpenGL and GLUT to teach graphics fundamentals.

Previous academic studies and student projects have shown that 2D top-down racing games are effective for demonstrating real-time rendering, animation, and collision detection. Web-based and mobile racing games often rely on advanced engines, whereas this project focuses on methodological learning, implementing everything manually using OpenGL.

The required tools—C++, OpenGL, and GLUT—are freely available and lightweight, making the project technically and economically feasible. The hardware requirements are minimal, and the project can run efficiently on standard personal computers.

1.5 Gap Analysis

Although many 2D racing games exist, most rely heavily on pre-built engines and libraries, which hide the internal working of graphics algorithms. There is a gap in projects that clearly demonstrate low-level graphics implementation for educational purposes.

This project addresses that gap by:

- Implementing graphics using basic OpenGL primitives
- Using a custom DDA line drawing algorithm instead of built-in line functions
- Managing animation, object movement, and collision detection manually

Thus, the project bridges the gap between theoretical computer graphics concepts and practical implementation.

1.6 Project Outcome

The expected outcomes of this project are:

- A fully functional 2D top-down car racing game
- Clear understanding of OpenGL rendering and GLUT event handling
- Practical implementation of graphics algorithms and animation
- Improved knowledge of collision detection and game logic
- A reusable academic project suitable for demonstrations and evaluations

Additionally, this project serves as a strong base for future enhancements such as textured graphics, sound integration, multiple levels, and 3D extensions.

Chapter 2

Proposed Methodology/Architecture

This chapter describes the proposed methodology, system architecture, requirement analysis, and design specifications of the project. It explains the overall workflow, module interaction, and user interface design of the application.

2.1 Requirement Analysis & Design Specification

Requirement analysis and design specification define the functional and non-functional needs of the system before implementation. This phase ensures that the game design aligns with the project objectives and that all graphical and logical components are properly planned.

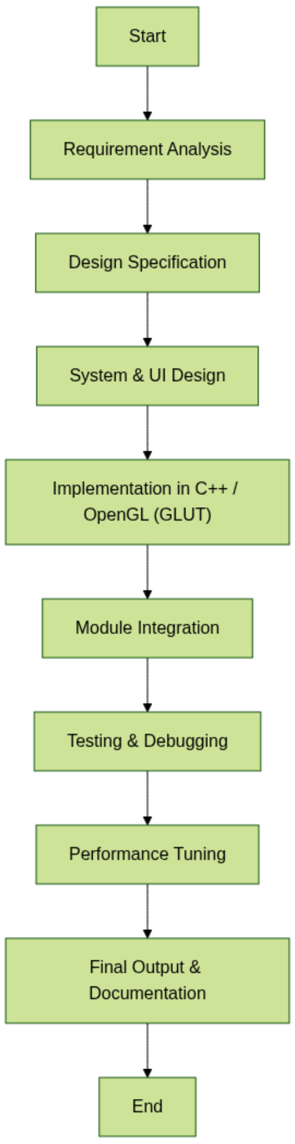

Figure: Project Development Phases

2.1.1 Overview

The proposed system is a 2D top-down car racing game developed using OpenGL and GLUT in C++. The game allows the player to control a car using keyboard inputs and avoid incoming enemy cars on a multi-lane road. The system continuously updates the scene to simulate motion and maintains a score based on player survival.

The project is designed to be simple, efficient, and suitable for academic purposes, focusing on graphics fundamentals rather than advanced game engines. The overall system operates in real-time and provides immediate visual feedback to user actions.

2.1.2 System Design

The system design follows a modular structure, where each module is responsible for a specific task:

Input Module

- Handles user input using keyboard events.
- Left and right arrow keys are used to move the player's car.
- A restart key (R) is used to reset the game after a collision.

Game Logic Module

- Controls player movement within road boundaries.
- Manages enemy car generation with random speed and lane positions.
- Detects collisions between player and enemy cars.
- Updates score and game-over state.

Graphics Rendering Module

- Renders road, lane markings, player car, enemy cars, trees, and background.
- Uses OpenGL primitives such as quads, triangles, and points.
- Implements orthographic projection for 2D visualization.

Animation & Timing Module

- Uses GLUT timer functions to update object positions.
- Ensures smooth animation and consistent frame updates.
- Synchronizes enemy movement, background scrolling, and scoring.

This modular design improves readability, maintainability, and scalability of the code.

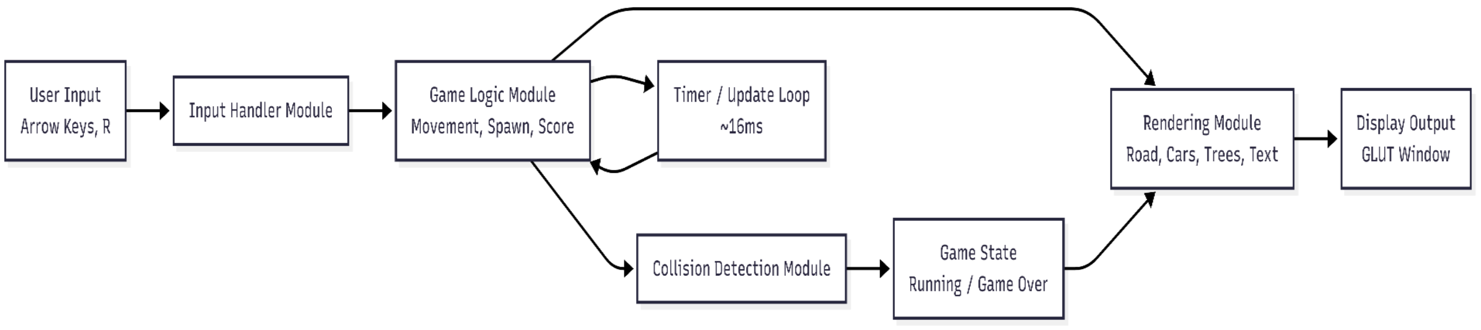

Figure: Block Diagram (Modules)

2.1.3 UI Design

The user interface (UI) is designed to be simple, intuitive, and minimalistic, focusing on gameplay clarity rather than visual complexity.

Main UI Elements

- Player Car: Displayed as a colored rectangle with a windshield.
- Enemy Cars: Differently colored rectangles to distinguish them from the player.
- Road & Lanes: Gray Road with dashed white lane markings created using the DDA algorithm.
- Background: Green grass and scrolling trees on both sides of the road.
- Score Display: Displayed at the top-left corner of the screen.
- Game Over Message: Displayed at the center of the screen with restart instructions.

The UI avoids clutter and ensures that the player can easily understand the game state at all times.

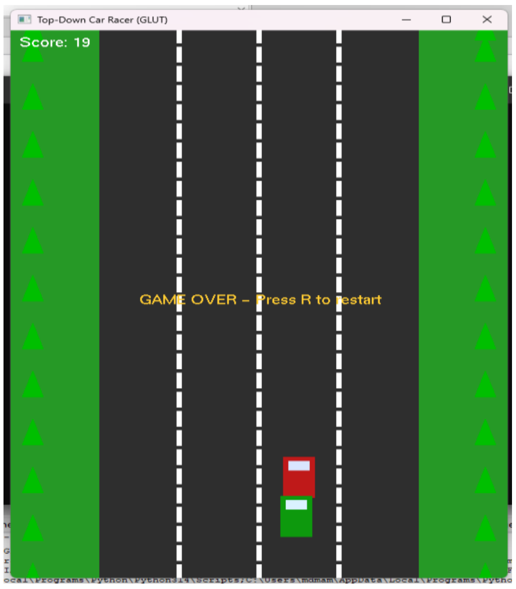

Figure: Game Interface

2.2 Overall Project Plan

The project was executed in a structured and phased manner to ensure systematic development:

Requirement Analysis

- Identified game features and functional requirements.
- Selected OpenGL and GLUT as development tools.

Design Phase

- Designed system architecture and modular structure.
- Planned UI layout and game mechanics.

Implementation Phase

- Implemented graphics rendering using OpenGL primitives.
- Added keyboard input handling and timer-based animation.
- Implemented collision detection and scoring system.

Testing Phase

- Tested player movement boundaries.
- Verified collision detection accuracy.
- Ensured smooth animation and stable frame updates.

Finalization

- Code optimization and cleanup.
- Documentation and report preparation.

This structured project plan ensured timely completion and a stable final application.

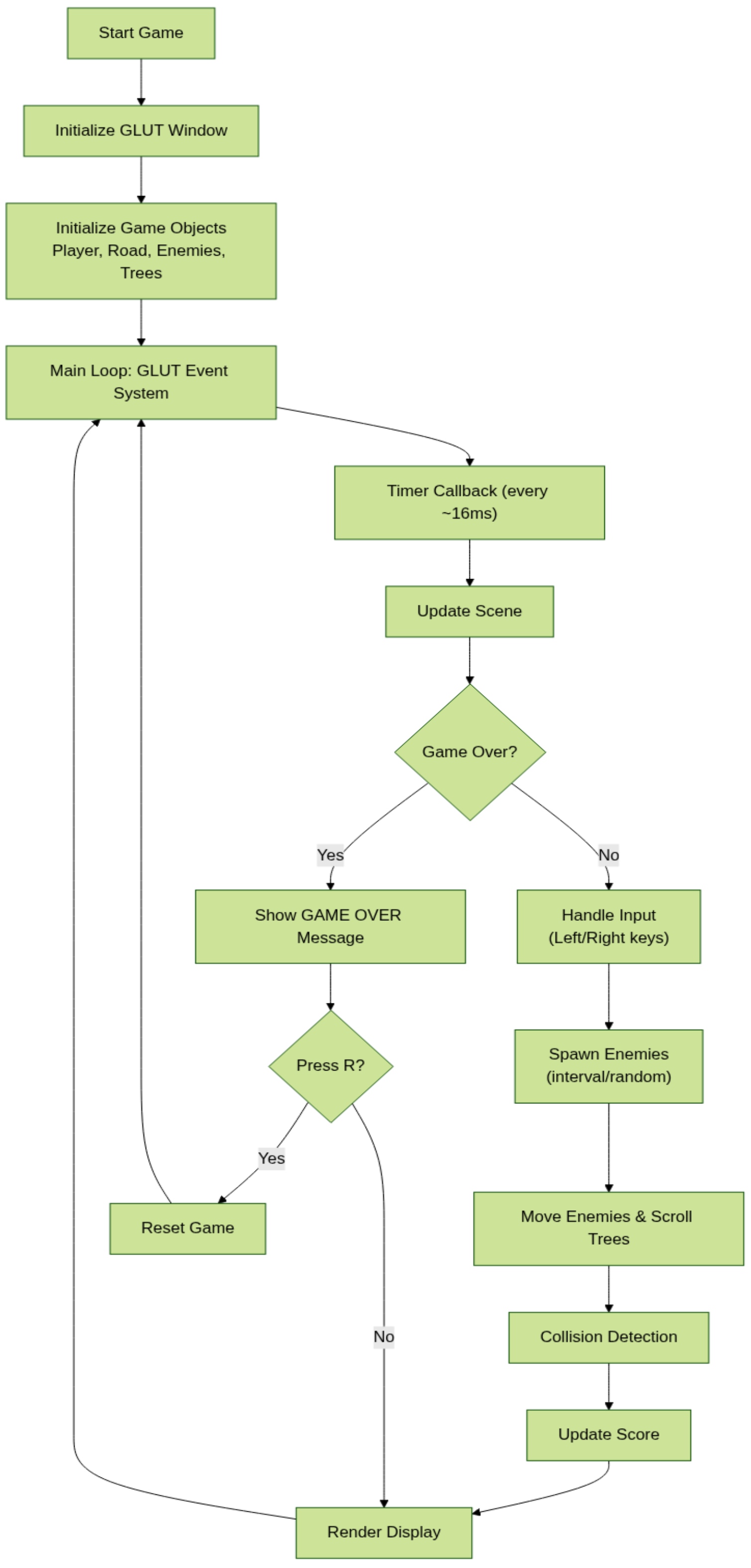
Start Game
Initialize GLUT Window
Initialize Game Objects Player, Road, Enemies, Trees
Main Loop: GLUT Event System
Timer Callback (every ~16ms)
Update Scene
Game Over?
Yes
No
Show GAME OVER Message
Handle Input (Left/Right keys)
Press R?
Spawn Enemies (interval/random)
Yes
Reset Game
Move Enemies & Scroll Trees
No
Collision Detection
Update Score
Render Display

Figure: Working Procedure.

Chapter 3

Implementation and Results

This chapter details the implementation of the system using C++ and OpenGL, followed by performance analysis and result discussion. It highlights how graphics algorithms, animation techniques, and game logic were implemented and evaluated.

3.1 Implementation

The implementation of the proposed system is carried out using **C++** with **OpenGL** and **GLUT**. The application follows a modular approach where rendering, input handling, game logic, and timing are handled independently to ensure clarity and maintainability.

The game window considers the orthographic 2D coordinate system using **gluOrtho2D**, which allows precise placement of game objects on the screen. All graphical components such as the road, player car, enemy cars, lane markings, trees, and background are created using basic OpenGL primitives including **quadrilaterals, triangles, and points.**

Player movement is controlled using special keyboard inputs (left and right arrow keys). The car is constrained within road boundaries to prevent it from moving outside the playable area. Enemy cars are generated at random intervals and lanes using a vector-based container structure. Each enemy vehicle is assigned a random speed to increase gameplay variability.

A timer-based animation mechanism using **glutTimerFunc** updates the game state approximately every 16 milliseconds. This mechanism ensures smooth movement of enemy vehicles and background elements such as scrolling trees. The **DDA line drawing algorithm** is implemented manually to render dashed lane markings, demonstrating the application of fundamental graphics algorithms.

Collision detection is performed using **axis-aligned bounding box** (AABB) logic. When a collision between the player and an enemy car is detected, the game state is set to game over. A scoring mechanism increments the score whenever an enemy successfully passes the player without collision. The game can be restarted using a keyboard input.

3.2 Performance Analysis

The performance of the system was evaluated based on frame consistency, responsiveness, and resource utilization. Since the application is based on 2D rendering and uses simple geometric primitives, the computational overhead is minimal.

The use of double buffering prevents flickering and ensures smooth rendering. Timer-based updates maintain a stable refresh rate, resulting in consistent animation even on systems with modest hardware capabilities. Enemy spawning and background scrolling are optimized to avoid unnecessary computations by removing off-screen objects.

Collision detection operations are lightweight, as they involve simple rectangle overlap checks. The application does not rely on textures or complex shaders, which further reduces GPU load. Overall, the system performs efficiently and maintains real-time responsiveness throughout gameplay.

3.3 Results and Discussion

The final application successfully demonstrates a functional 2D top-down car racing game. The player can smoothly control the car, avoid enemy vehicles, and observe real-time changes in score and game state. The dashed lane markings, scrolling background, and continuous enemy movement create an effective illusion of motion.

The project effectively applies key computer graphics concepts, including 2D projection, animation, geometric modeling, and collision detection. The implementation confirms that even with basic OpenGL primitives, engaging interactive applications can be developed.

While the visual design is intentionally simple, the game fulfills its academic purpose by focusing on algorithmic and graphical correctness. The results validate the feasibility of using OpenGL and GLUT for educational game development and provide a strong foundation for future enhancements.

Chapter 4

Engineering Standards and Mapping

This chapter discusses the impact of the project on society, environment, and sustainability, along with ethical considerations and project management practices. It evaluates the project in terms of engineering standards and responsible software development.

4.1 Impact on Society, Environment and Sustainability

Engineering solutions should not only address technical problems but also consider their societal, environmental, and ethical impact. Although this project is an academic 2D game application, it reflects responsible software engineering practices and contributes positively to learning and skill development.

4.1.1 Impact on Life

The project contributes to student learning by enhancing understanding of computer graphics fundamentals, problem-solving skills, and logical thinking. It promotes creativity and technical competence in software development. As an educational application, it helps learners gain hands-on experience with graphics programming, which is valuable for careers in gaming, simulation, visualization, and multimedia industries.

Additionally, the project encourages interactive learning, which can improve engagement and motivation compared to purely theoretical study.

4.1.2 Impact on Society & Environment

From a societal perspective, the project supports digital education and skill development, which are essential in today's technology-driven world. It demonstrates how simple graphical applications can be built efficiently without excessive resource consumption.

Environmentally, the project has minimal impact, as it is a lightweight software application that does not require specialized hardware or high energy consumption. By focusing on efficient 2D graphics rendering, it aligns with environmentally responsible computing practices.

4.1.3 Ethical Aspects

The project adheres to ethical software development principles by:

- Avoiding harmful or misleading content
- Respecting user control and transparency
- Using open-source and legally available tools such as OpenGL and GLUT

There is no collection or misuse of personal data, and the application does not promote violence or unethical behavior. The game content is suitable for all age groups and serves educational purposes, ensuring compliance with ethical standards in software engineering.

4.1.4 Sustainability Plan

The sustainability of the project lies in its reusability, scalability, and educational value. The codebase is modular and can be easily extended or reused for future academic projects. The project can be maintained and enhanced over time by adding new features such as textures, sound, or 3D graphics.

From a resource standpoint, the project promotes sustainable computing by utilizing minimal system resources and avoiding unnecessary computational complexity. As an educational tool, it supports long-term knowledge development, which is a key aspect of sustainable engineering practice.

4.2 Project Management and Team Work

Effective project management and teamwork were essential to the successful completion of this project. The development of the 2D top-down car racing game was carried out by a three-member team, where collaboration, coordination, and shared responsibility were emphasized throughout the project lifecycle.

The project was executed using a phased and organized approach, beginning with requirement analysis and system design, followed by implementation, testing, and documentation. Tasks were planned and distributed among team members to ensure balanced workload and timely completion of each phase. Regular discussions and reviews were conducted to track progress, resolve technical challenges, and integrate different components of the system.

Strong communication among team members enabled effective problem-solving and smooth integration of graphics rendering, game logic, and user interaction modules. This collaborative approach improved productivity and ensured consistency across the project.

Overall, the teamwork and project management practices adopted during the development process contributed significantly to achieving the project objectives and enhanced the team's understanding of computer graphics concepts, software development practices, and cooperative engineering work.

4.3 Complex Engineering Problem

4.3.1 Mapping of Program Outcome

In this section, provide a mapping of the problem and provided solution with targeted Program Outcomes (PO's).

Table 4.1: Justification of Program Outcomes

PO's	**Justification**
PO1 (Engineering Knowledge)	Attained by applying core computer graphics concepts, algorithms, and C++ programming principles to design and implement the 2D racing game.
PO5 (Modern Tool Usage)	Attained through effective use of modern development tools such as OpenGL, GLUT, and C++ IDEs for real-time graphics rendering and interaction.
PO9 (Team Work)	Attained by collaboratively planning, developing, integrating, and testing the project as a three-member team with shared responsibilities.
PO10 (Communication)	Attained through clear documentation, code comments, report preparation, and effective discussion of design and implementation

	details among team members.

4.3.2 Complex Problem Solving

In this section, provide a mapping with problem solving categories. For each mapping add subsections to put rationale (Use Table 4.2). For P1, you need to put another mapping with

Knowledge profile and rational thereof.

Table 4.2: Mapping with complex problem solving.

EP1 Dept of Knowledge	**EP2** Range of Conflicting Requirements	**EP3** Depth of Analysis	**EP4** Familiarity of Issues	**EP5** Extent of Applicable Codes	**EP6** Extent Of Stakeholder Involvement	**EP7** Inter-dependence
✓	✓	✓	✓			✓

4.3.3 Engineering Activities

In this section, provide a mapping with engineering activities. For each mapping add subsections to put rationale (Use Table 4.3).

Table 4.3: Mapping with complex engineering activities.

EA1 Range of resources	**EA2** Level of Interaction	**EA3** Innovation	**EA4** Consequences for society and environment	**EA5** Familiarity
✓	✓		✓	✓

Chapter 5

Conclusion

This chapter summarizes the work carried out, discusses limitations of the current system, and presents possible future enhancements. It reflects on the learning outcomes and overall success of the project.

5.1 Summary

This project successfully demonstrates the development of a 2D top-down car racing game using OpenGL and GLUT in C++. The main objective was to apply fundamental computer graphics concepts to create an interactive real-time application without relying on advanced game engines.

The game implements essential graphics techniques such as orthographic projection, geometric modeling, timer-based animation, and collision detection. Key gameplay elements including player control, enemy vehicle generation, scoring system, and game-over logic were effectively integrated. The use of the DDA line drawing algorithm for dashed lane markings further strengthens the educational value of the project.

Overall, the project fulfills its academic goals and provides practical experience in graphics programming, algorithm implementation, and event-driven application design.

5.2 Limitation

Despite its successful implementation, the project has certain limitations:

- The game uses basic shapes and solid colors, without textures or sprite images.
- There is no sound or background music, which limits the immersive experience.
- Enemy behavior is simple and does not include advanced artificial intelligence.
- The game supports only single-level gameplay with fixed difficulty.
- The application is limited to 2D graphics and does not utilize modern OpenGL features such as shaders.

These limitations were intentionally accepted to keep the project focused on core computer graphics concepts and academic requirements.

5.3 Future Work

The project can be extended in several ways to enhance functionality and realism:

- Addition of textures and sprite-based graphics for cars and road.

- Integration of sound effects and background music.
- Implementation of multiple levels with increasing difficulty.
- Introduction of power-ups, obstacles, and bonus items.
- Enhancement of enemy behavior using basic AI techniques.
- Extension of the game into a 3D version using modern OpenGL and shaders.

These improvements would make the application more engaging and provide opportunities to explore advanced computer graphics and game development concepts.

References

[1] Edward Angel and Dave Shreiner, Interactive Computer Graphics: A Top-Down Approach with OpenGL, 6th Edition, Pearson Education, 2012.

[2] Mark J. Kilgard, OpenGL Utility Toolkit (GLUT) Programming Interface, OpenGL.org.

[3] FreeGLUT Project Documentation, FreeGLUT API Reference Manual, Available online.

[4] Khronos Group, OpenGL Official Documentation, https://www.khronos.org/opengl/

[5] Sedgewick, Robert, Algorithms in C++, Addison-Wesley, for algorithmic concepts applied in collision detection and logic.