 AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

408/1, Kuratoli, Khilkhet, Dhaka 1229, Bangladesh

|  |  |  |
| --- | --- | --- |
| Project Title: Chronoscape: A Journey Through Time |  |  |
| Assignment No: 01 |  | Date of Submission: 28/06/25 |
| Course Title: Computer Graphics |  |  |
| Course Code: CSC 4118 |  | Section: E |
| Semester: 2024-25 |  | Course Teacher: Dipta Justin Gomes |

**Declaration and Statement of Authorship:**

1. I/we hold a copy of this Assignment/Case-Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case-Study is my/our original work and no part of it has been copied from any other student’s work or

from any other source except where due acknowledgement is made.

1. No part of this Assignment/Case-Study has been written for me/us by any other person except where such collaborationhas been authorized by the concerned teacher and is clearly acknowledged in the assignment.
2. I/we have not previously submitted or currently submitting this work for any other course/unit.
3. This work may be reproduced, communicated, compared and archived for the purpose of detecting plagiarism.
4. I/we give permission for a copy of my/our marked work to be retained by the Faculty for review and comparison, including review by external examiners.
5. I/we understand thatPlagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a formofcheatingandisaveryseriousacademicoffencethatmayleadtoexpulsionfromtheUniversity. Plagiarized material can be drawn from, and presented in, written, graphic and visual form, including electronic data, and oral presentations. Plagiarism occurs when the origin of them arterial used is not appropriately cited.
6. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or to copy my/our work.

*\* Student(s) must complete all details except the faculty use part.*

\*\* Please submit all assignments to your course teacher or the office of the concerned teacher.

Group Name/No.: G

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Name** | **ID** | **Program** | **Signature** |
| 1 | Chinmoy Guha | 22-48056-2 | BSc CSE |  |
| 2 | Suvra Chakraborty | 22-48067-2 | BSc CSE |  |
| 3 | Mohammad Ansar uddin | 22-47975-2 | BSc CSE |  |
| 4 | Md. Foyez Ahmed | 23-50848-1 | BSc CSE |  |

|  |  |  |
| --- | --- | --- |
| ***Faculty use only*** | | |
| FACULTYCOMMENTS | **Marks Obtained** |  |
|  |
|  |
|  | **Total Marks** |  |
|  |
|  |

**Abstract**

This project, *Chronoscape: A Journey Through Time*, is a computer graphics simulation that visually represents the natural transitions of a full day—from morning to night. Developed using OpenGL and GLUT in C++, the simulation is divided into four distinct scenes: morning, afternoon, evening, and night. Each scene captures the atmosphere of its respective time period through dynamic animations and color transitions, such as a rising sun, moving clouds, glowing streetlights, and twinkling stars. The project highlights key graphics programming concepts, including transformations, lighting effects, modular scene design, and real-time user interaction. Our aim is to deliver a visually engaging and educational experience that reflects the passage of time through simple yet expressive 2D animations. While the current version focuses on basic rendering and interaction, the system is designed with scalability in mind for potential enhancements like real-time synchronization and weather effects. Through this project, we demonstrate our understanding of fundamental graphics principles and our ability to build an interactive visual narrative using code.

***Keywords****:* computer graphics, OpenGL, animation, time simulation, interactive scenes

**Introduction**

In the realm of computer graphics, visual storytelling plays a crucial role in engaging users and conveying complex ideas through simple animations and effects. Our project, *Chronoscape: A Journey Through Time*, aims to represent the passage of a full day using dynamic visual scenes. The concept is built around dividing the day into four distinct parts—morning, afternoon, evening, and night—each rendered with unique colors, animations, and elements to reflect the specific mood and environmental characteristics of that time.

The inspiration for this project came from observing how many modern games and applications simulate time changes to enhance realism and immersion. These visual transitions not only serve aesthetic purposes but also create a sense of progression. We wanted to recreate this experience on a smaller scale using OpenGL, applying our understanding of computer graphics principles in a meaningful and creative way.

Our main motivation behind choosing this project was to blend technical learning with artistic freedom. Each team member is responsible for developing one of the four scenes, allowing us to contribute individual creativity while working towards a unified outcome. The project also gives us the opportunity to explore essential graphics techniques such as shape rendering, transformations, lighting, and scene control using user input.

The primary goal of this project is to build an interactive and visually appealing simulation that effectively communicates the progression of time. We aim to implement smooth transitions, environment-specific details, and basic animations to make each scene feel distinct. Additionally, we want the system to be user-controllable, enabling viewers to move between different times of day manually.

By working on Chronoscape, we aim to strengthen our grasp of fundamental computer graphics concepts while also experimenting with scene composition, user interaction, and animation timing. The project strikes a balance between educational value and creative expression, making it both a technical challenge and a rewarding learning experience.

**Related Work**

The idea for *Chronoscape: A Journey Through Time* was inspired by visual elements commonly seen in simulation games and weather-based applications. Games like *Stardew Valley*, *Animal Crossing*, and *The Legend of Zelda: Breath of the Wild* influenced our approach to dynamic time-of-day transitions, where lighting, colors, and environment behavior change depending on the in-game time. These examples showed how simple visual effects can significantly enhance user immersion.

In terms of development, we referred to several OpenGL and GLUT tutorials that demonstrate basic 2D rendering, transformation matrices, and animation techniques. Online resources like learnopengl.com, and interactive YouTube guides provided practical knowledge about how to manage real-time drawing using glutDisplayFunc(), animation with glutTimerFunc(), and user input through glutKeyboardFunc().

Although we did not use complex third-party libraries, OpenGL itself is a foundational graphics API that supports efficient rendering and scene control. GLUT was used for window management and input handling, allowing us to focus more on design and animation logic than on low-level system details.

We also looked at examples of simple OpenGL projects like “Animated Solar System” or “Day-Night Cycle” demos often used in academic settings, which helped us understand how to structure scenes modularly and apply gradual lighting changes. These references served as both technical and conceptual guides throughout the development process.

**Tools, Technologies & Libraries *(200-300 words)***

Our project *Chronoscape: A Journey Through Time* was developed primarily using **C++**, chosen for its efficiency and compatibility with graphics libraries like OpenGL. The project makes extensive use of **OpenGL**, a cross-platform graphics API that allows for direct control over rendering pipelines, transformations, and drawing primitives. To manage window creation, input handling, and animations, we used **GLUT (OpenGL Utility Toolkit)**, which simplifies the process of setting up and interacting with the graphics context.

The development environment for this project was **Code::Blocks**, an open-source IDE that offers good support for C++ and OpenGL development. All work was carried out on **Windows** operating systems, which provided a stable and well-documented environment for configuring OpenGL and GLUT. Code::Blocks allowed us to organize the project into manageable source files, and its built-in debugger was useful for testing animation behavior and real-time scene transitions.

We opted to create all visual elements—such as the sun, moon, trees, buildings, and characters—using **basic geometric primitives** (quads, circles, lines) instead of importing prebuilt assets. This kept the focus on manual rendering and transformations, reinforcing our understanding of graphics programming fundamentals. Simple **color gradients** and shape transformations were used to simulate dynamic lighting and movement.

While we did not use external textures, models, or sound files in the current version, we can design the system in a way that can accommodate future enhancements like **ambient background sounds** or **bitmap textures**. The minimal reliance on external assets allowed us to prioritize core functionality and scene composition using code-driven techniques.

**System Design and Implementation *(200-300 words)***

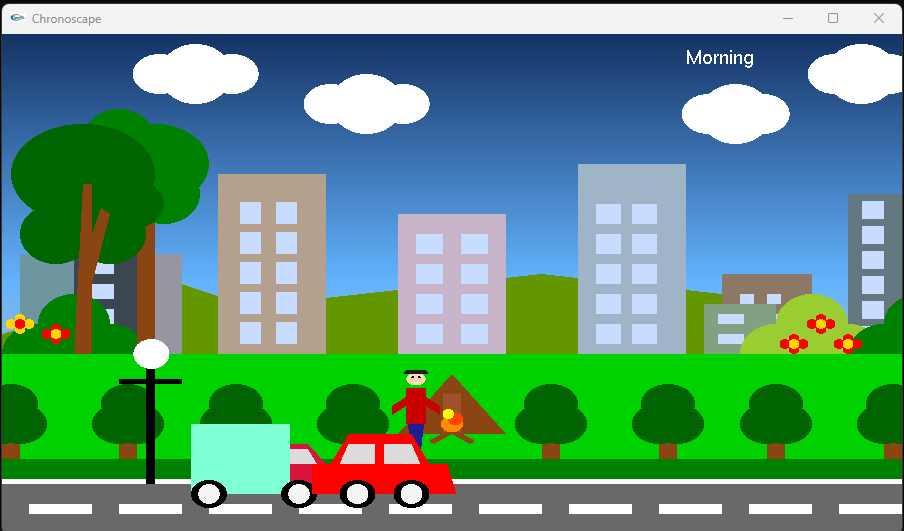
**Scene Description**  
The world of *Chronoscape* is divided into four distinct scenes—morning, afternoon, evening, and night, each designed to visually represent the unique atmosphere of that time of day. Every scene includes environmental elements such as the sun or moon, sky gradients, trees, clouds, and background objects like houses or streetlights. Animations bring the scenes to life, such as birds flying in the morning, cars driving in the afternoon, and stars twinkling at night. Each scene is coded as a separate module using OpenGL functions and can be displayed individually or transitioned between using keyboard input (arrow keys).

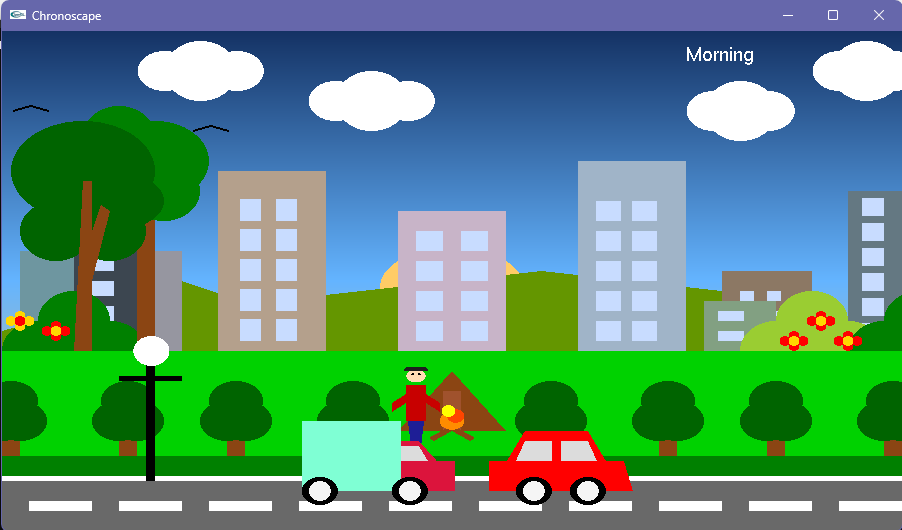
**User Interface**  
The user interface is minimal and non-intrusive, focusing entirely on visual storytelling. A small time label (e.g., "Morning", "Evening") appears on-screen to indicate the current scene. Visual cues like lighting changes and object movement help users interpret the time progression without the need for menus or buttons.

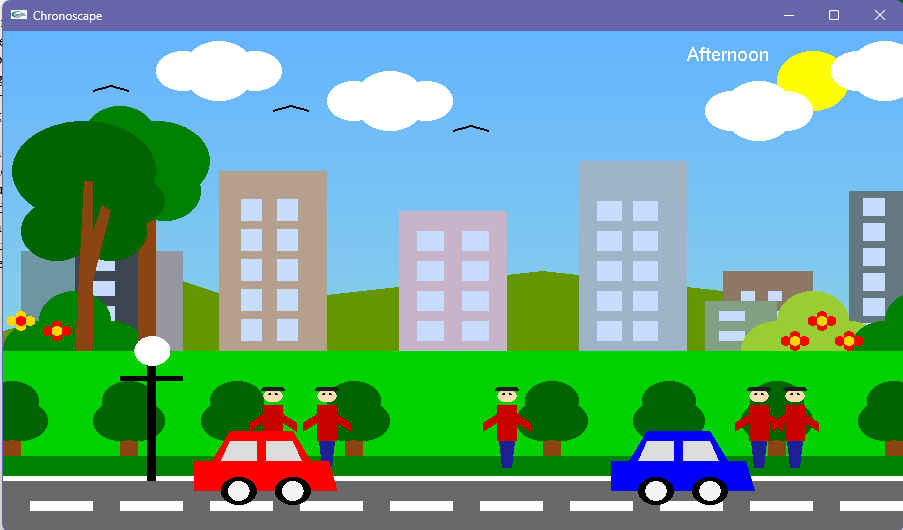
**Controls and Input Handling**  
The system supports keyboard interaction, allowing users to switch between scenes manually. Input is handled through the glutKeyboardFunc() method, which detects key presses and triggers the appropriate scene transitions. This simple control system helps users explore each part of the day freely. The arrow keys help to go back and forth between the scenes, and r key toggles rain.

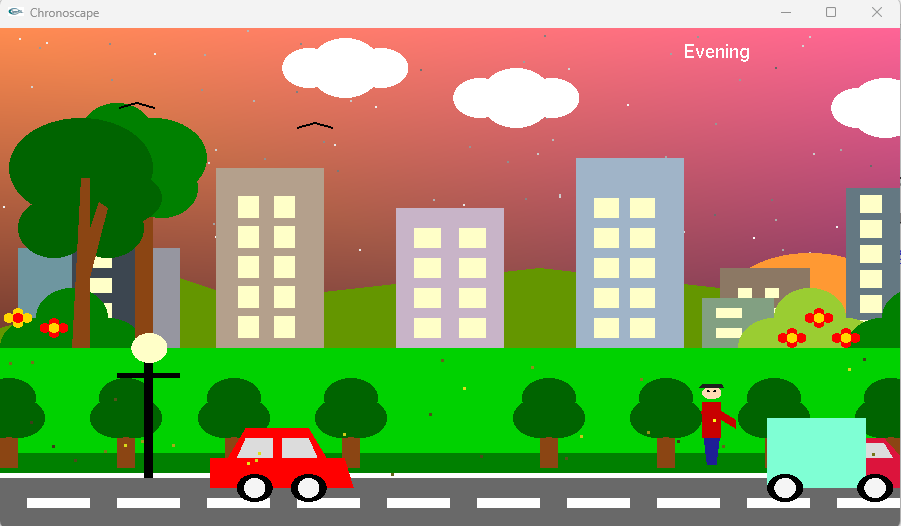
**Challenges and Solutions**  
One of the main challenges was synchronizing animations and transitions across scenes without causing delays or visual glitches. This was solved by using glutTimerFunc() to schedule updates independently of frame rendering. Another issue was maintaining consistency in object scaling and positioning across different scenes. We addressed this by standardizing coordinate boundaries and reusing base functions for drawing common elements. These approaches helped keep the project modular, efficient, and visually coherent.

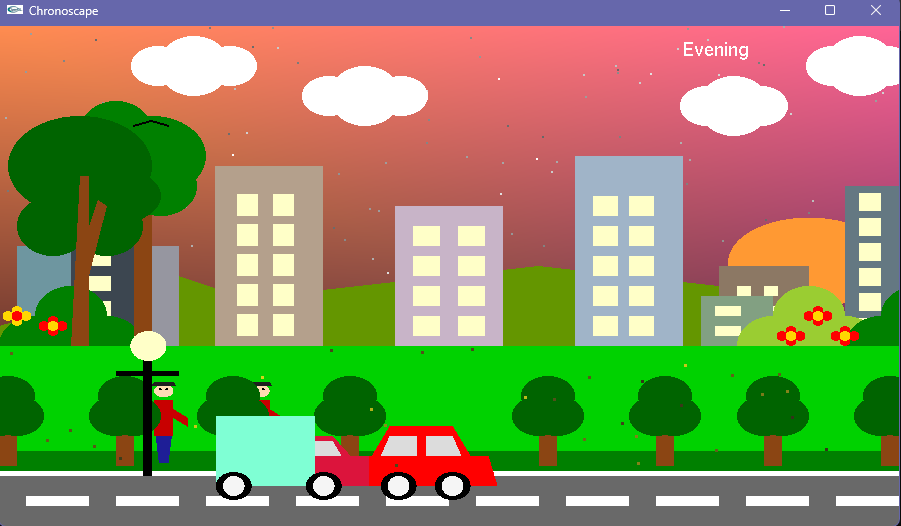
**Results & Demonstration**

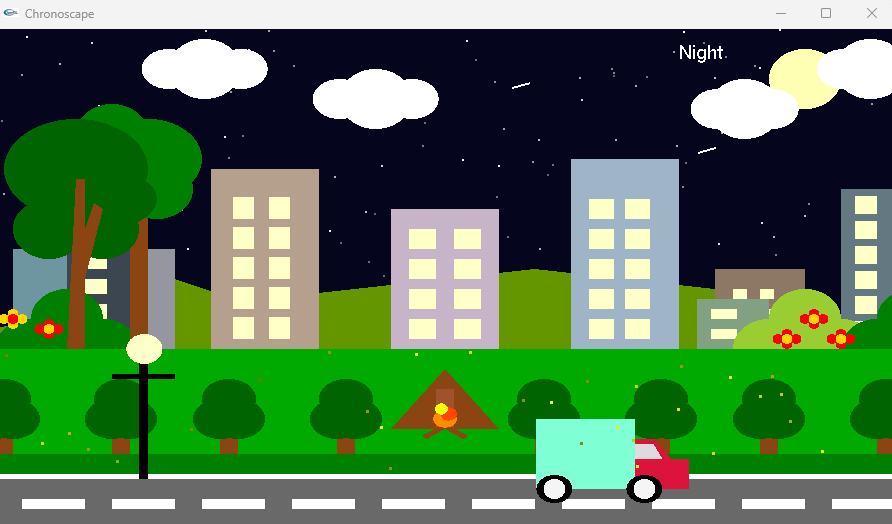
****

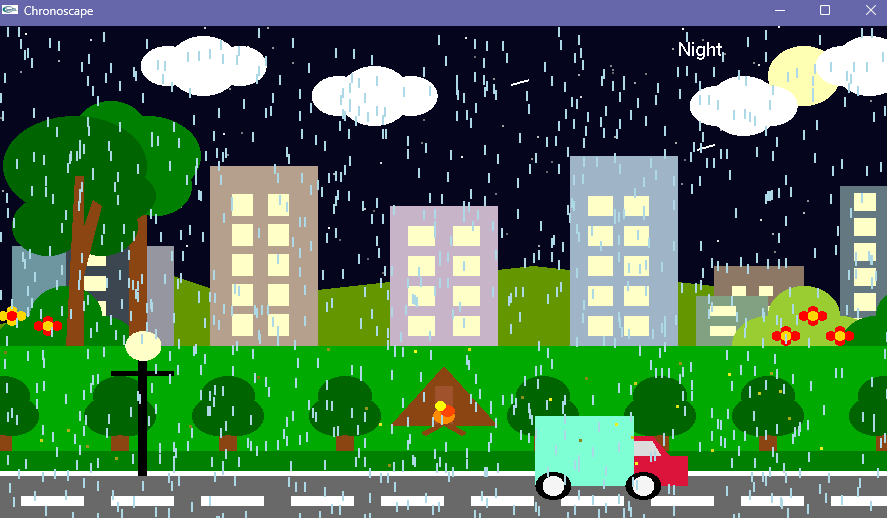
****

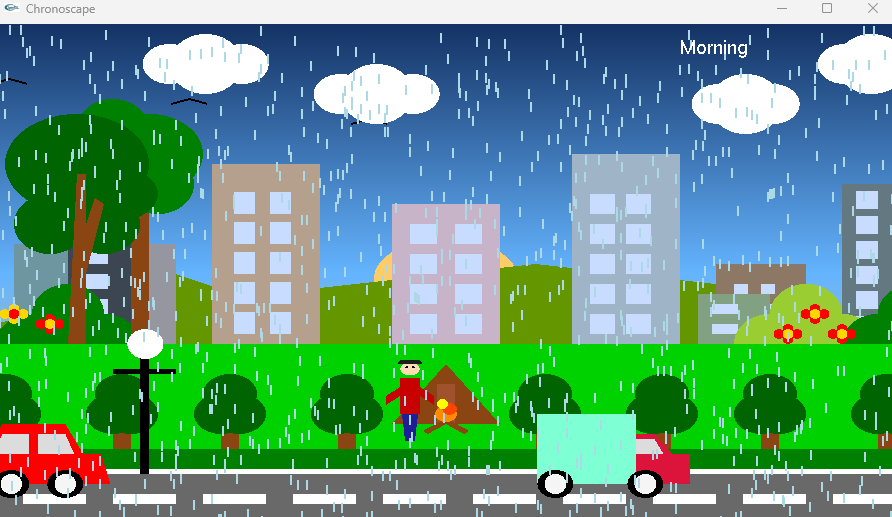
****

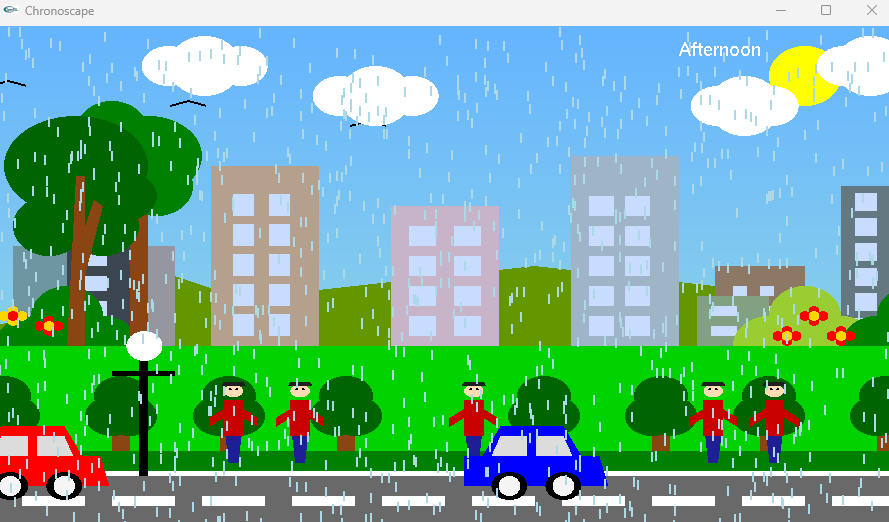
****

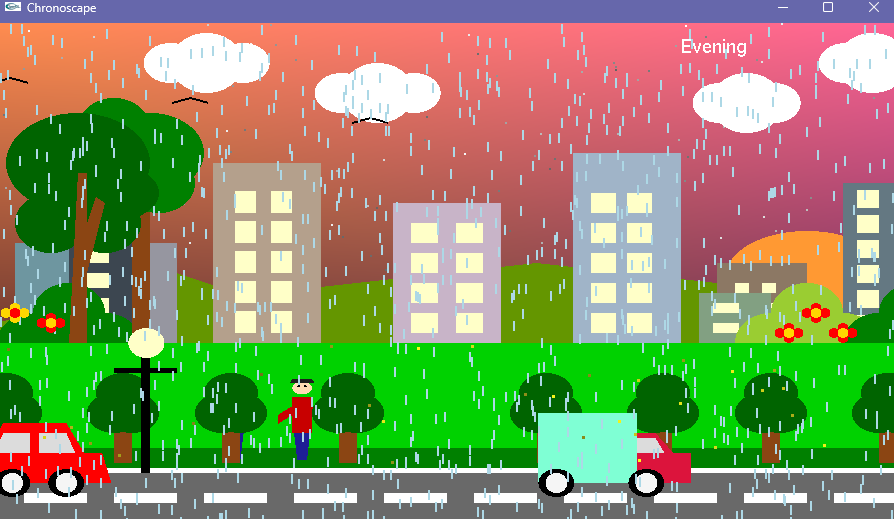
****

****

****

****

****

****

**Conclusion**

In this project, we successfully designed and implemented a dynamic 2D animated environment using OpenGL and GLUT. Our main achievement was creating a multi-scene animation that smoothly transitions between night, morning, and afternoon modes. Each scene includes animated elements such as cars, humans, birds, clouds, and environmental features like trees, hills, buildings, and a campfire. We added interactive controls to switch between scenes using the arrow keys and toggle rain using the keyboard. We also implemented natural effects like rain, moving clouds, flying birds, and flickering stars and fireflies to bring the environment to life.

Despite these accomplishments, the project has a few limitations. The scenes are all rendered in 2D, which restricts the visual depth and realism of the environment. Most of the animations follow fixed paths and are not responsive to user input beyond scene switching. In addition, the code structure became quite lengthy, and we recognize the need for better modularity and reusability to enhance maintainability.

Looking ahead, there are several ways to extend this project. We could incorporate smoother transitions between scenes, add sound effects for a more immersive experience, or implement weather patterns such as thunderstorms or snow. Introducing simple artificial intelligence for characters or allowing user-controlled movement would add interactivity. Most importantly, converting the project into 3D using modern OpenGL would drastically improve visual appeal and open the door for more complex scene management and user engagement.

Through this project, we gained valuable experience with animation logic, real-time rendering, and scene design using OpenGL.

**References**

1. [https://www.gatevidyalay.com/2d-reflection-in-computer-graphics-definition-examples/https://en.wikipedia.org/wiki/Transformation\_matrix](https://www.gatevidyalay.com/2d-reflection-in-computer-graphics-definition-examples/https:/en.wikipedia.org/wiki/Transformation_matrix)
2. <https://www.gatevidyalay.com/2d-shearing-in-computer-graphics-definition-examples/>
3. <https://www.javatpoint.com/computer-graphics-2d-transformations>
4. Chapter 5: Foley, van Dam, Feiner, Hughes, Computer Graphics: principles and practice, Addison Wesley, Second Edition
5. Schaum's Outline of Theory & Problems of Computer Graphics.
6. Peter Shirley Steve Marschner , “Fundamental of computer graphics”, Third Edition.
7. Angel, E., & Shreiner, D. (2012). *Interactive computer graphics: A top-down approach with shader-based OpenGL* (6th ed.). Addison-Wesley.
8. Bourke, P. (1997). Drawing shapes with OpenGL. <http://paulbourke.net/dataformats/opengl/>
9. Gortler, S. J. (2012). *Foundations of 3D computer graphics*. The MIT Press.
10. OpenGL Architecture Review Board, Shreiner, D., Sellers, G., Kessenich, J., & Licea-Kane, B. (2013). *OpenGL programming guide: The official guide to learning OpenGL, Version 4.3* (8th ed.). Addison-Wesley.
11. Sommer, J. (2015). *Beginning OpenGL game programming* (2nd ed.). Cengage Learning PTR.
12. Stroustrup, B. (2013). *The C++ programming language* (4th ed.). Addison-Wesley.