Village Scenario with Animation

Submitted By

Student Name	Student ID
Abdullah Al Noman	221-15-5387
Nushrat Jahan Mila	221-15-5758
Abrar Hameem Bornil	221-15-5331

LAB PROJECT REPORT

This Report Presented in Partial Fulfillment of the course

CSE422: Computer Graphics Lab in the Department of Computer Science and Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

Dhaka, Bangladesh

13/12/2025

DECLARATION

We hereby declare that this lab project has been done by us under the supervision of **Kridita Ray**, **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:

____

Kridita Ray
Lecturer
Department of Computer Science and Engineering
Daffodil International University

Submitted by

[illegible] ______________________ Abdullah Al Noman ID: 221-15-5387 Dept. of CSE, DIU	
mila ______________________ Nushrat Jahan Mila ID: 221-15-5758 Dept. of CSE, DIU	Bornil ______________________ Abrar Hameem Bornil ID: 221-15-5331 Dept. of CSE, DIU

COURSE & PROGRAM OUTCOME

The following course have course outcomes as following:

Table 1: Course Outcome Statements

CO's	Statements
CO1	**Understand** computer graphics system and implement graphics primitives for drawing a graphics scene.
CO2	**Apply** appropriate OpenGL programming techniques, resources and modern engineering and IT tools to solve graphics programming issues including different shapes, 2D and 3D transformation
CO3	**Perform** effectively as an individual or a member or a leader of diverse teams through proper documentation and initialization of project work
CO4	**Create** a project by explaining complex computer engineering activities with the computer engineering community by performing effective communication through effective reports, design documentation, make effective presentations and give and receive clear instructions.

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

Mapping Course Outcome (COs) with the Teaching-Learning and Assessment Strategy:

CO's	Corresponding PO number	Domain Level/ Learning Taxonomy	Level of Knowledge Profile (K)	Complex Engineering Problem (EP)	Complex Engineering Activities (EA)
CO1	PO1	C1, C2	K1-K4	EP1	
CO2	PO5	C3, C4	K1-K4, K6	EP1, EP3	
CO3	PO9	C4, A2	K6	EP1, EP3	
CO4	PO10	C6, P3, A2	K4	EP3, EP5	EA3

The mapping justification of this table is provided in section **4.2.1**, **4.2.2** and **4.2.3**.

Table of Contents

Chapter 1
Introduction

This chapter provides an overview of the Computer Graphics project, including its objectives, scope, and significance. It establishes the foundation by explaining the project's theme, visual concept, and the goals it aims to achieve through graphical modeling and animation.

1.1 Introduction

Computer Graphics plays a vital role in modern visualization and real-time rendering applications. Using OpenGL, graphical scenes can be created with precise geometric modeling, 2D transformations, dynamic animations, and efficient rendering techniques. This project, "Village Scenario with Animation," is developed using C++, OpenGL, and GLUT, where various graphical objects such as houses, trees, sun, river, boat, human figure, bird, clouds, and natural elements are designed from scratch using primitive shapes. The scene simulates a lively rural environment featuring animated movements such as a moving sun, flowing river, walking human, sailing boat, flying bird, swaying grass, moving clouds, and tree sway effects. The project demonstrates strong use of computer graphics algorithms, real-time rendering strategies, transformation techniques, object-oriented design, and creativity to build an engaging animated 2D environment that represents a realistic village setting.

1.2 Motivation

The motivation behind developing this project comes from the desire to create a visually appealing and naturally interactive environment using basic Computer Graphics principles. Village life holds cultural significance, offering a peaceful and scenic atmosphere. Representing that beauty through OpenGL animation allows us to practice algorithm implementation, understand rendering pipelines, and creatively build a complete scene using low-level graphical primitives. Additionally, OpenGL-based projects enhance understanding of how animations, transformations, pixel plotting algorithms, and timed rendering loops work—making it an ideal choice for a project in the Computer Graphics course.

1.3 Objectives

The main objectives of this project include:

- Implement graphics algorithms such as Bresenham's Line Drawing Algorithm for drawing geometric shapes.
- Apply 2D transformations including translation, rotation, scaling, reflection, and shear wherever necessary in the animated scene.
- Create a fully designed 2D animated village scenario using OpenGL primitives and modular functions.
- Demonstrate real-time animation using timer functions and movement logic (e.g., sun movement, human walking, river wave motion, cloud drift).

- Practice modular programming, clean rendering flow, and scene organization for better visualization and performance.

1.4 Feasibility Study

Several existing OpenGL-based animation projects and tutorials show implementations of cartoon scenarios, traffic scenes, fish tank animations, and landscape simulations. These works demonstrate the practical use of OpenGL primitives, scene rendering, and animation logic. However, many available examples are simplistic, lacking rich object detailing, multiple interacting animations, or natural effects such as wind sway, river waves, or layered object rendering. Some projects use only basic geometric shapes without focusing on realism or depth. This project expands on those ideas by implementing a comprehensive multi-object animated environment, integrating custom-designed objects (trees, houses, rivers, birds, humans, ground textures) and realistic animation effects, which enhances both educational and visual value.

1.5 Gap Analysis

Although multiple OpenGL scenes exist, most lack:

- Complex interactions between animated elements.
- Realistic environmental effects (wind sway, flowing water).
- Hierarchical object design with multiple layers.
- A scene integrating both nature (trees, sun, river) and human activity (walking figure, boat movement).

This project addresses these gaps by creating a rich, naturally animated rural environment that combines multiple animation techniques, detailed object modeling, and smooth scene flow.

1.6 Project Outcome

At the end of the project, the expected outcomes include:

- A fully functioning 2D animated village scene built entirely using OpenGL primitives.
- Real-time animations including.
- Moving sun with rising/setting effect
- Cloud drifting
- Human walking cycle
- Bird flying with wing flap
- Swaying trees and grass
- Flowing river
- Boat movement on water
- Well-structured and modularized C++ code for rendering and animation.
- Demonstration of practical knowledge in graphics algorithms, transformations, and animations.
- A visually attractive and smoothly animated final output that represents a peaceful village environment.

Chapter 2
System Architecture / Scene Description & Architecture

This chapter describes the architecture, structure, and visual design of the 2D animated village scene developed in this Computer Graphics project. It explains all graphical components, their interactions, transformations, and the modern tools used to build the OpenGL-based animated environment.

2.1 Requirement Analysis & Design Specification

2.1.1 Scene Overview

The project represents a complete animated village environment built using 2D OpenGL rendering. The scene contains natural and man-made components arranged to form a realistic landscape. The major segments of the scene include:

- A bright daytime sky with gradient color
- A moving sun with animated rays
- Multiple clouds drifting across the sky
- A flowing river with animated waves
- A boat floating over the river surface
- Grass, soil, and detailed ground elements
- Several trees with wind-sway animation
- A central main house and two small houses
- A walking human figure with leg and arm animation
- A flying bird following a wave path
- A fence and riverbank decorations

All objects are drawn using OpenGL primitives such as quads, triangles, polygons, triangle fans, and line loops. Layering order ensures proper depth perception and scene composition.

2.1.2 Object Components

Below are the major objects in the scene along with their geometric properties:

Table 2.1: Object Component

Object Name	Description	Geometric Structure
Sky	Background gradient for environment	Quads
Sun	Circular sun with rays	Triangle fan + line rays
Clouds	Fluffy clustered circles	Multiple triangle fans
Grounds	Soil and grass layer	Quads + triangle grass blades
Grass & Bushes	Animated grass swaying	Triangles, lines, small arcs

River	Flowing water with waves	Quads + dynamic line strips
Boat	Floating boat with sail	Polygons, quads, triangles
Main House	Large center house	Quads, polygons, lines
Small House	Village huts	Quads + triangle roofs
Trees	Multi-layered leaves + trunk	Triangle fans + quads
Bird	Flying V-shape bird	Lines
Human	Stick-like animated person	Lines, quads, triangle fan
Face	Wooden fence	Quads + lines

Each object is drawn in modular functions to maintain code readability and reduce redundancy.

2.1.3 Sketch of project

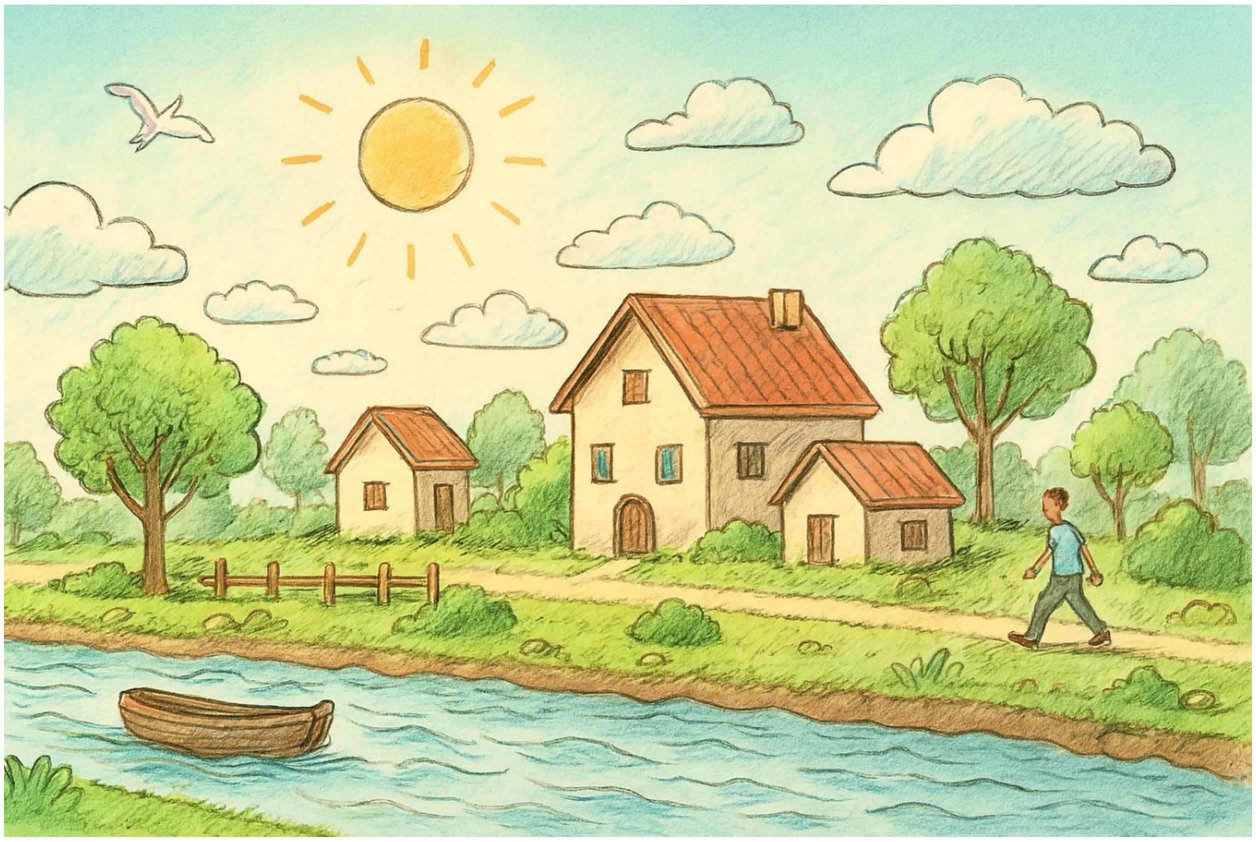

Figure 2.1: Sketch Diagram of this project

This sketch shows a peaceful village on a bright sunny day. The sky is light blue with soft clouds and a warm sun shining above. A bird is flying gently across the sky. In the middle of the scene, there is one big house with two smaller houses beside it, all with red roofs and simple village-style designs. The village is surrounded by green trees, bushes, and grass, giving it a calm, natural feeling.

A river flows in front of the village with light waves, and a small wooden boat floats on the water. Along the riverbank, there is a dirt path and some decorations like a wooden fence. A man is walking on the path, adding life and movement to the quiet countryside environment. Overall, the sketch shows a warm, happy, and simple village atmosphere.

2.2 Use of Modern Tools

This project uses various modern software tools and technologies to enhance development and visualization quality:

- OpenGL / FreeGLUT – Core graphics libraries for creating and rendering 2D objects, applying transformations, and handling animations.
- Visual Studio / Code::Blocks – IDEs used for writing, compiling, and debugging the C++ OpenGL program.
- GLUT / GLFW functions – For window creation, input handling, display looping, and timer-based animation control.
- C++ Programming Language – Primary language used to implement all rendering functions, algorithms, animation updates, and event-driven logic.
- Trigonometric Functions (sin, cos) – Applied for natural animation effects such as bird flight pattern, grass sway, and wave motion.
- Git/GitHub (optional) – For version control and code management.
- GIMP / Photoshop (optional) – Could be used for designing textures or visual elements if needed.

2.3 Algorithm Used

The project implements Bresenham's Line Drawing Algorithm, one of the fundamental computer graphics algorithms for rasterizing lines using incremental integer calculations.

Why we use Bresenham?

- It produces visually accurate and smooth lines.
- Does not require floating-point computation.
- Ideal for drawing fence posts, house edges, window frames, and geometric outlines.

In the project, the algorithm is used to render precise lines required for object edges and structural details such as door frames, fence outlines, rooftops, and more.

2.4 2D Transformations Applied

Several 2D transformations were applied in the scene to animate objects realistically:

1. Translation
Used for moving objects such as:

- Moving sun across the sky
- Clouds drifting left
- Human walking
- Boat floating
- Bird flying with wave motion

Example:
glTranslatef(boatX, -2.9f, 0.0f);

2. Rotation
Used for:

- Sun rays rotating around the sun
- Wind sway adjustment in trees
- Rotation is achieved using trigonometric functions and offset calculations.

3. Scaling
Used for:

- Trees of different sizes
- Small and large houses
- Bush variations

Scaling allows the same object-drawing code to produce differently sized versions.

4. Reflection / Shear

Not directly applied as independent operations but minor reflective effects appear in wave distortions and movement arcs.
Through these transformations, objects behave more naturally in animations.

2.5 Animation Logic

The animation system is driven by a timer function that updates object positions and triggers re-rendering at ~60 FPS.

Key Animation Techniques

1. Position-based Animation

Objects change their x/y coordinates gradually:

```
sunX += 0.01f;
cloudX -= 0.006f;
humanX += 0.012f;
boatX += 0.01f;
```

2. Natural Movement Using Sine Waves

- Bird flight path
- Grass sway
- Tree leaf movement
- River wave motion

```
float flap = sin(windOffset * 3) * 0.05f;
birdY = 3.3f + sin(birdX * 2) * 0.4f;
```

3. Timer Function for Smooth Animation

```
glutTimerFunc(16, update, 0);
```

This ensures continuous motion and smooth rendering.

4. Object Resetting

Objects wrap around the screen when reaching boundaries (e.g., clouds reappear on right side).

Chapter 3
Implementation and Results

This chapter explains the complete implementation process of the Village Scenario animation using OpenGL. It describes how each object was constructed, how animations were applied, and how all components were integrated to form the final animated scene. The outputs are also presented conceptually to reflect the results observed during execution.

3.1 Implementation

The implementation of the animated village scene was completed in C++ using OpenGL and GLUT. The entire program is structured into modular functions, with each function responsible for drawing a specific object or handling a specific animation behavior. The main stages of implementation are outlined below.

1. Object Construction Using Primitives

Each object in the scene was created using basic OpenGL primitives:

- GL_QUADS for houses, ground, sky, trunks, doors
- GL_TRIANGLES / POLYGON for roofs, sails, grass blades
- GL_TRIANGLE_FAN for sun, clouds, tree leaves, human head
- GL_LINE_LOOP / GL_LINE_STRIP for outlines, waves, bird wings
- GL_LINES for fence rails, window grids, leg/arm movement

This modular breakdown allowed efficient rendering and clarity in the code. For example:

- The sky uses gradient quads
- The ground uses quads + animated triangle grass blades
- The river uses quads with sinusoidal line strips
- The houses use structured combinations of quads, polygons, and tree fans
- The tree has multiple leaf layers for depth
- The human uses lines and a triangle fan
- The bird uses lines with wing movement

All shapes were drawn relative to the center, allowing easy translation and scaling.

2. Animation Handling

Animation was achieved by updating global variables such as:

sunX, sunY – sun movement
cloudX, cloud2X – cloud drift
humanX – walking animation
boatX – floating boat movement

birdX, birdY – flying bird
windOffset – sway for trees and grass
riverFlow – wave motion

These variables are updated inside the update() function, which runs approximately at 60 FPS using:

```
glutTimerFunc(16, update, 0);
```

This ensures smooth transitions between frames.

3. Drawing Order and Layering

Objects are drawn in a specific order to maintain depth:

- Sky
- Sun
- Clouds
- Bird
- Ground
- River
- Boat
- Trees
- Houses
- Fence
- Human

This ensures no object incorrectly overlaps another in the final rendering.

4. Use of Bresenham's Line Algorithm

The Bresenham algorithm was used to draw precise pixel-based lines, enhancing structural accuracy for:

- Fence outlines
- Windows
- Door frames
- Rooftop markings

This ensures straight lines without floating-point errors.

5. Transformation Integration

glTranslatef() was used extensively to animate objects by moving them along the X and Y axes. Scaling and rotation effects were added through trigonometric modifications (sin/cos) to simulate natural behavior.

3.2 Output

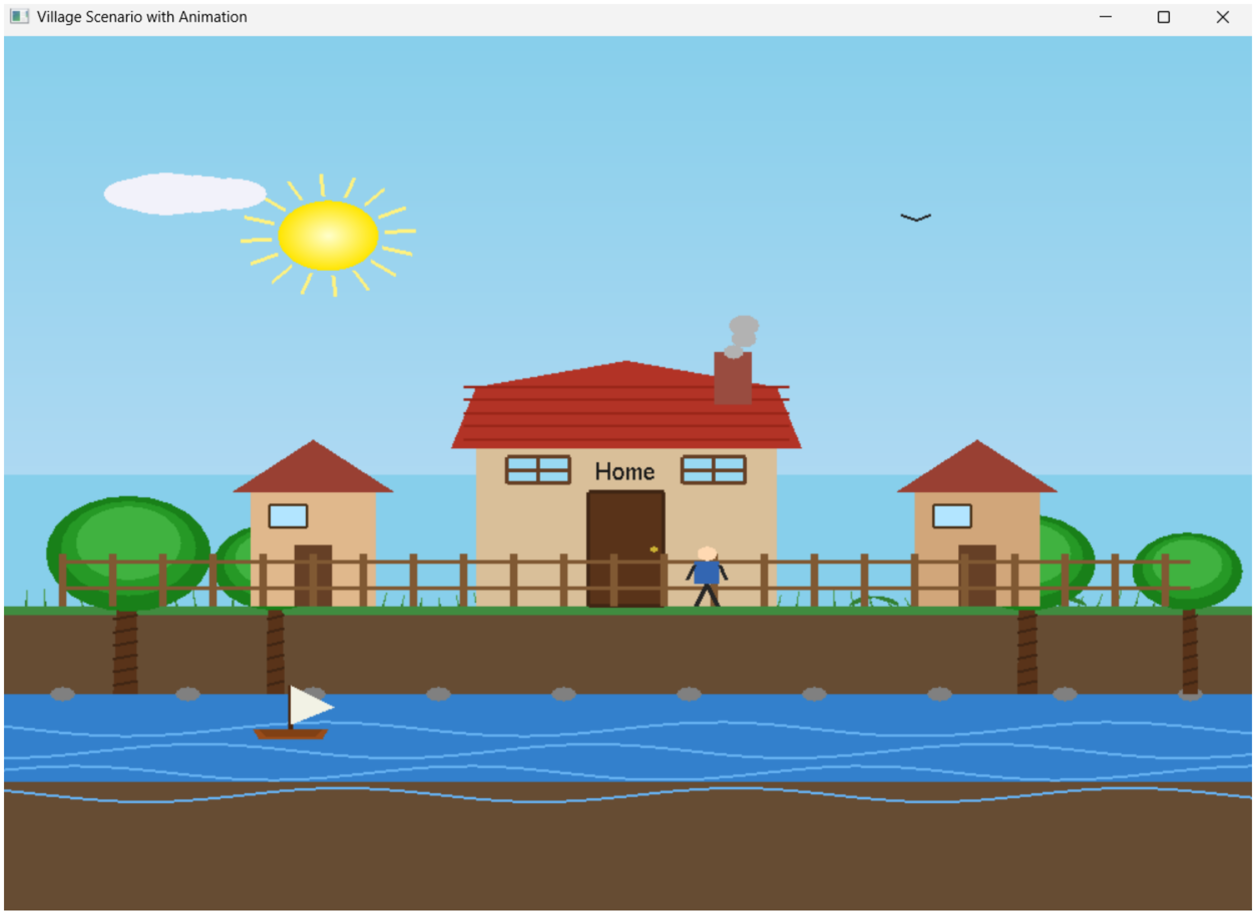

Figure 3.1: Village Scenario

The output of the program is a fully animated 2D village environment rendered on an OpenGL window. The key visible outputs are:

- A bright blue sky with gradient shading
- A rotating and moving sun following an arc-like sunrise–sunset path
- Fluffy clouds drifting at different speeds
- A flowing river with wave animation
- A wooden boat smoothly floating on the water
- Green grass swaying in the wind
- Bushes and trees with natural leaf movement
- A detailed main house labeled “Mila’s Home”
- Two small houses on both sides
- A human figure walking continuously across the village
- A flying bird with animated wing flaps
- A wooden fence marking the boundary

The final result visually represents a peaceful and dynamic rural environment, where natural and man-made elements coexist harmoniously through smooth animations.

3.3 Discussion

The project successfully demonstrates the power of OpenGL in creating interactive and realistic 2D animated scenes. Key strengths observed include:

1. **Smooth and consistent animation**

The timer-driven update logic ensures near 60 FPS animation, making all moving components appear fluid.

2. **Realistic environmental effects**

- Use of sine-based motion added natural behavior:
- Trees sway dynamically
- Grass bends with wind
- River waves oscillate realistically
- Birds fly with wing flapping

3. **Modular and organized code structure**

Each object having its own function helped in debugging and extending the scene easily.

4. **Application of graphics algorithms & transformations**

- Bresenham Line Algorithm for accurate line drawing
- Translation for moving entities
- Scaling for trees and houses
- Trigonometric rotation effects for sun rays

5. **Creative and visually appealing design**

The village scenario is rich in detail and atmosphere, producing an engaging visual output that feels lively and immersive.

Chapter 4
Engineering Standards and Mapping

This chapter discusses the engineering standards followed throughout the project and maps the project activities with the corresponding Course Outcomes (COs) and Program Outcomes (POs). It highlights how the project meets academic guidelines, ethical considerations, and sustainable development aspects required in Computer Graphics and engineering practice.

4.1 Impact on Society, Environment and Sustainability

4.1.1 Impact on Life

The project depicts a calm, natural village environment, helping users appreciate the beauty of rural life through computer graphics visualization. Such interactive animations can be used for:

- Educational purposes to demonstrate natural phenomena (sun movement, river flow, wind effects).
- Simulation-based learning, enabling students to understand how graphical elements behave in motion.
- Entertainment and visualization tools supporting creative industries, game design, and animation design.
- The project enhances visual understanding and stimulates interest in graphical modeling and animation techniques.

4.1.2 Impact on Society & Environment

This project visually represents a peaceful village ecosystem, promoting:

- Environmental awareness through realistic animations of nature (trees, birds, river).
- Cultural appreciation by portraying typical rural scenes found in Bangladesh and other regions.
- Creative learning for students exploring graphics fields.

While the project has no direct environmental impact, it encourages developers to think about nature-friendly designs, sustainable environments, and the importance of rural ecosystems. It also demonstrates how technology can be used to recreate natural environments in digital form.

4.1.3 Ethical Aspects

In developing this project, the following ethical principles were followed:

- All graphic objects (trees, houses, birds, sun, river, etc.) were manually designed, ensuring the content is original and free from plagiarism.
- No copyrighted or sensitive assets were used.
- The project avoids any offensive, unsafe, or culturally insensitive elements.

- Proper acknowledgment of algorithms, concepts, and references was maintained to uphold academic honesty.
- The project follows clean, modular coding practices, enabling clarity, understanding, and responsible engineering communication.

These aspects ensure compliance with ethical standards in digital content creation and engineering education.

4.1.4 Sustainability Plan

The project promotes long-term sustainability in the following ways:

Reusability of Code

Each object is written in separate modular functions, making the code easily extendable for future projects.

Scalability

New objects or animations (vehicles, animals, additional houses, weather changes) can be added without altering the existing structure.

Maintaining Lightweight Performance

Since no textures or heavy assets are used, the project runs smoothly on low-end systems, supporting sustainable computing.

Educational Longevity

Students can continue using this project as a base for learning transformations, animations, and graphical rendering for years to come.

4.2 Complex Engineering Problem

The project involves solving multiple computational and design-related challenges classified under Complex Engineering Problems (EP). These include:

- Rendering multiple animated objects with real-time updates
- Using geometric primitives to construct detailed environmental components
- Applying mathematical algorithms such as Bresenham's line algorithm
- Handling multiple interacting animations (sun, bird, boat, human, river waves)
- Implementing natural movement patterns using sine functions
- Managing depth layering and visual composition
- Such tasks require analytical thinking, algorithmic design, and technical problem-solving.

4.2.1 Mapping of Program Outcome

Below is the mapping of the project activities with DIU's Program Outcomes (POs) as defined in your template.

Table 4.1: Justification of Program Outcomes

POs	Justification of Mapping (Project Perspective)
PO1	The project applies mathematical and computational knowledge (coordinates, geometry, trigonometry, transformations) to create and animate graphical objects using OpenGL. This meets PO1 through the use of algorithms, pixel plotting, and real-time rendering techniques.
PO5	The project uses modern graphics tools such as OpenGL, GLUT, C++ programming, and advanced rendering logic to develop a functional real-time animated environment. This satisfies PO5 by applying modern engineering and software tools.
PO9	Teamwork, documentation, code modularization, and shared responsibilities are reflected in the systematic development of multiple graphical components. These activities align with PO9.
PO10	The project requires proper documentation, structured report writing, and effective explanation of graphics concepts and animation logic. This supports PO10 through communication and presentation of technical details.

4.2.2 Complex Problem Solving

Table 4.2: Mapping with Complex Problem Solving

EP Code	Explanation
EP1 – Depth of Knowledge	Designing multiple layered 2D objects, implementing Bresenham’s algorithm, managing movement and transformations require strong knowledge of graphics pipelines and math.
EP2 – Conflicting Requirements	Balancing realism, performance, and smooth animation required choosing optimized shapes, minimal vertex use, and efficient update logic.
EP3 – Depth of Analysis	Animations such as river waves, grass sway, and bird movement required analyzing motion equations (sin/cos) and timing functions.
EP4 – Familiarity of Issues	The project tackles real-world graphical problems such as layering, object overlapping, frame rates, and visual composition.
EP5 – Application of Codes	Bresenham’s algorithm, OpenGL rendering rules, and structured function-based programming follow standard graphics engineering practices.

EP6 – Stakeholder Involvement	The project meets academic expectations, educational needs, and programming standards for Computer Graphics labs.
EP7 – Interdependence	Multiple animated objects depend on shared variables (wind, timer, frame updates), demonstrating interdependent system design.

4.2.3 Complex Engineering Activities

Table 4.3: Mapping with Complex Engineering Activities

EA Code	Explanation
EA1 – Engineering Design Documentation	The project uses systematic methods to design, document, and present all graphical components and animations clearly.
EA2 – Presentation & Communication	Visual and oral presentation techniques can be used to demonstrate animation logic and rendering processes effectively to an audience.
EA3 – Technical Activity Execution	The project involves handling multiple animated activities simultaneously (boat, bird, clouds, human), meeting the criteria for complex engineering activities.

Chapter 5
Conclusion

This chapter summarizes the outcomes of the project, discusses its limitations, and highlights potential future improvements. The project successfully demonstrates the application of core computer graphics algorithms, 2D transformations, animation logic, and modular design using OpenGL.

5.1 Summary

The “Village Scenario with Animation” project creates a visually rich and naturally animated 2D environment using OpenGL and C++. Multiple graphical objects such as sky, sun, clouds, river, boat, houses, trees, grass, bird, fence, and a walking human were developed using basic geometric primitives.

Key achievements include:

- Successful implementation of Bresenham’s Line Drawing Algorithm for accurate and optimized line rendering.
- Application of 2D transformations such as translation, scaling, and rotation for dynamic animation.
- Use of real-time animation techniques through timer functions and frame updates.
- Creation of natural environmental effects like river flow, tree sway, grass movement, and bird flight patterns using sinusoidal functions.
- Structuring the code into modular functions for ease of development and readability.
- Delivering a complete, smooth, and visually appealing animated rural scene.

Overall, the project demonstrates strong understanding of computer graphics concepts and practical rendering techniques.

5.2 Limitation

Despite its successful implementation, the project has a few limitations:

- The scene is purely two-dimensional and does not include 3D depth, lighting, or advanced shading techniques.
- The animation speed depends on system performance, as there is no frame rate regulation beyond the timer function.
- No texture-mapping is used; all objects are drawn using simple shapes, which limits realism.
- User interaction is not included—keyboard or mouse controls could enhance engagement.
- Some animations (e.g., walking or flying) follow simple motion models and do not include advanced physics.

These limitations leave room for enhancement in future versions.

5.3 Future Work

Several improvements and expansions can be added to enhance the functionality and realism of the project:

1. Add User Interaction

- Keyboard controls to start/stop animation
- Mouse interaction for selecting or moving objects

2. Introduce 3D Graphics

- Convert the 2D village into a 3D environment using OpenGL 3.x+
- Add depth, lighting, texture, and perspective projection

3. Include Day–Night Cycle

- Gradual color transitions
- Moon, stars, and nighttime scene

4. Add More Animated Characters

- Animals (cows, birds, fish)
- Vehicles or people performing multiple actions

5. Add Sound Effects

- River flow sound, bird chirping, footsteps, etc.

6. Integrate Weather Effects

- Rain, wind gusts, fog, or snowfall using particle systems

7. Expand the Village Map

- Create roads, bridges, fields, crops, or marketplaces

Implementing these future improvements would make the project more interactive, realistic, and feature-rich.

References

[1] OpenGL Programming Guide, Addison Wesley.
[2] GeeksforGeeks OpenGL Tutorials.
[3] Stack Overflow discussions on animation logic.