AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

Department of Computer Science

Faculty of Science & Technology (FST)

CSC 4118: COMPUTER GRAPHICS

2025-2026, Fall

Section: **J**; Group: **G**

Project Report

PROJECT TITLE

3-Scene Village Environment

Supervised by,

DIPTA JUSTIN GOMES

Group Member,

Name	ID	Contribution
NAYEM SIDDIKI	23-52361-2	Scene 1 A peaceful morning scene where the sun rises from behind the river
FARHAN CHOWDHURY	23-52314-2	Scene 2 Road behind crop field with a tractor, windmill etc
MD. TANBIN ISLAM	23-50657-1	Scene 3 A peaceful morning scene where the sun rises from behind the river

Submission Date: 22/01/2026

1. Introduction

Computer graphics plays a crucial role in modern visualization, simulation, and interactive system development. In this project, our goal is to design an animated 3-scene environment that reflects the natural beauty of a typical rural Bangladeshi village.

1.1. Explanation of the scenario

The project will illustrate **three different visual perspectives of village life**: (i) A peaceful morning scene where the sun rises from behind the river, (ii) Village Road view with natural elements, sunrise, night view and daily activities, and (iii) Road behind crop field with a tractor, windmill etc.

Details of scenario:

Scene 1 – River & Boat Village Scene

Purpose: Natural village environment with water movement

Features:

- **Day / Night Mode**
  - Day → Sun visible
  - Night → Moon + stars visible
- **Moving Boat**
  - Boat continuously moves on the river
  - Speed can be increased/decreased
- **Moving Clouds**
  - Clouds move continuously across the sky
- **Pause / Resume**
  - Entire animation can be paused and resumed
- **Static Village Objects**
  - Houses, trees, fence, straw hut, hills, grass, road
- **User Interaction**
  - Keyboard + mouse control for speed and mode changes

Scene 2 – Village Road, Houses & Life Scene

Purpose: Village lifestyle with perspective road and living elements

Features:

- **Day / Night Toggle**
  - Smooth transition of sun position
  - Night lighting inside house windows
- **Moving Clouds**
  - Multiple clouds move at different speeds
- **Flying Birds**
  - Birds fly continuously in the sky
- **Perspective Road**
  - Road becomes narrow towards the village (depth effect)

- **Lamp Posts with Light (Night)**
  - Lamps glow only at night
- **Cow Movement**
  - Cow visible in day mode
- **Natural Environment**
  - Trees, flowers, river waves, mountains
- **Keyboard Control**
  - Toggle day/night using key input

Scene 3 – Farm, Tractor & Weather Scene

Purpose: Agricultural village activity with weather effects

Features:

- **Day / Night Mode**
  - Day → Clear sky
  - Night → Moon + stars
- **Rain Mode (Day & Night)**
  - Cloudy sky
  - Falling rain with wind direction
  - Stars hidden during rain
- **Moving Clouds**
  - Clouds move continuously even in rain
- **Flying Birds**
  - Birds always flying across the sky
- **Windmill Animation**
  - Windmill blades rotate continuously
- **Tractor Movement**
  - Tractor moves horizontally on the road
  - Speed increase/decrease
  - Direction change
  - Stop / Resume movement
- **Night Headlight Effect**
  - Tractor headlight turns ON automatically at night
- **Static Objects**
  - Barn, trees, field, road
- **Realistic Weather Feel**
  - Rain overlay for realistic atmosphere

1.2 Motivation

The motivation behind choosing this theme is to digitally preserve rural environments that are gradually being replaced by urban growth. Many students work on cityscapes, cartoons, or abstract animations, but real rural scenes are rarely represented in computer graphics projects. At the same time, nature-based scenes allow us to explore fundamental and advanced graphics techniques such as 2D/3D transformations, object modeling, animation, layering, lighting, and user interaction.

1.3 Novelty of the project

i. **Integrated Multi-Mode Environment**: The project uniquely combines day, night, and rain modes within a single scene, allowing smooth visual transitions that simulate real-life environmental changes.

ii. **Interactive Scene Control**: Users can dynamically control scene behavior (tractor movement, speed change, direction, rain toggle, day/night switch) through keyboard inputs, making the environment interactive rather than static.

iii. **Real-Time Weather Simulation**: The rain system includes moving raindrops, cloud darkening, and sky color changes**,** creating a realistic weather effect in real time.

iv. **Context-Aware Lighting Effects**: Features like automatic tractor headlights at night and muted lighting during rain add realism and show contextual rendering logic.

v. **Algorithm-Based Drawing**: Core objects are drawn using computer graphics algorithms such as DDA line drawing and Midpoint Circle Algorithm instead of relying on built-in shapes, strengthening the academic value.

vi. **Continuous Background Animation**: Elements like windmill blades, clouds, and birds animate continuously, independent of user interaction, giving the scene a lively village atmosphere.

vii. **Humanized Scene Design**: The scene reflects a rural village lifestyle, combining natural elements (trees, sky, hills) with human activity (tractor, barn, windmill).

viii. **Educational Use of OpenGL Concepts**: The project demonstrates practical usage of OpenGL transformations, color handling, blending, and timer-based animation in a single cohesive application.

1.4 Background of the scenario

The background of the scenario represents a peaceful rural village environment inspired by everyday countryside life. The scene is designed to visually capture the harmony between nature and human activity, showing open fields, hills, trees, a barn, a windmill, and moving clouds under a wide sky. The background dynamically changes based on time and weather conditions, shifting between day and night while also supporting rainy and clear atmospheres. During daytime, the environment appears bright and lively, while nighttime introduces a calm mood with moonlight and subtle lighting effects. The inclusion of animated elements such as drifting clouds, flying birds, rotating windmill blades, and moving rain enhances the realism of the background. Overall, the scenario background serves as a foundation that brings together natural scenery and interactive village elements to create an engaging and lifelike visual simulation.

1.5 Background of the platform used

This project will be developed using the **OpenGL** graphics library with the **GLUT**/FreeGLUT toolkit. The programming language will be **C++**, and the development environment will be **Code::Blocks** on the **Windows operating system**. OpenGL provides low-level control over drawing operations, making it suitable for creating custom rural scenery with precise control over shapes, colors, and animations.

We will implement 2D graphics but apply layered depth to mimic 3D-like visuals. Basic primitives such as polygons, lines, triangles, quads, and circles will be used to draw natural objects. Transformations like translation, rotation, and scaling will be used to animate elements such as the rising sun, moving clouds, birds, vehicles, walking villagers, and floating boats. Timer functions (glutTimerFunc) will control animation speed, while double buffering will ensure smooth rendering.

Each scene will be designed in separate drawing functions to maintain modularity. A keyboard listener will allow scene switching. Colors will be selected to simulate morning light, natural greenery, and water reflections. To create river waves and flowing animation, we will use sine-wave-based motion or periodic vertical/horizontal shifting.

The project will also use hierarchical animation for complex objects such as boats with oars or moving people. Overall, the environment enables clean and efficient rendering of three detailed village scenes with smooth interactive control.

2. Related Works

1. Computer Graphics Project – Village View

URL: https://github.com/radwanromy/Computer-Graphics-Project-Village-View

Brief Description:

This project presents a 2D village scene implemented using OpenGL. The work focuses on drawing a static rural environment that includes houses, trees, fields, hills, and other elements commonly found in a village setting. Unlike many basic graphics examples that only draw simple shapes, this repository shows a complete composition where multiple objects come together to form a coherent scene. The elements are drawn using OpenGL primitives such as lines, polygons, and circles, and the coordinates are carefully calculated to align objects proportionally. Although the scene lacks advanced animation or interactivity, it demonstrates careful planning of scene layout and layering. The project is useful for understanding how to structure a complete scene from multiple graphical objects by modularizing code into functions that represent different scene components. While this work does not include dynamic elements such as moving objects or environment effects, it serves as a good reference for how a base village environment can be built using fundamental OpenGL drawing routines.

2. Computer Graphics Project (Zubair Ahmed)

URL: https://github.com/zubair-ahmed-official/Computer-Graphics-Project

Brief Description:

This repository focuses on multiple graphics assignments compiled into a single project, likely created as part of a course or semester work. The project includes various graphical scenes, each demonstrating different aspects of OpenGL programming. Typical components include shape rendering, line drawing, circle generation, and simple animations. Although the repository's documentation is limited, the code indicates an emphasis on implementing graphics primitives and understanding coordinate transformations. Some scenes may include basic movement or interaction using keyboard input, but it does not present a fully cohesive environment. Compared to a dedicated village simulation, this project serves more as a collection of graphics experiments rather than a single narrative scene. It provides a range of small modules that touch on different graphics topics, making it useful for studying individual algorithms but less effective as a unified animation demonstration.

3. Village View – Computer Graphics Project (Abzana Sultan Ira)

URL: https://github.com/AbzanaSultanIra/Village-view--Computer-Graphics-project

Brief Description:

This project develops a 2D animated scene of a village using OpenGL. The work goes beyond static rendering by introducing motion elements such as moving clouds, flying birds, and possibly toggles for day and night. The environment includes grass, river or ground layers, and multiple objects placed to create depth and foreground-background relationships. The usage of animation techniques such as incremental object movement and conditional rendering reflects a deeper engagement with dynamic scene modeling compared to purely static scenes. Although the implementation remains in 2D and focuses on fundamental graphics functions, the integration of motion and environmental variation makes this project closer in spirit to real-time simulation. The project appears to organize its drawing logic into separate modules, which makes the code more reusable and clearer for learning purposes. Despite the absence of advanced physics or 3D rendering, this work demonstrates the effective combination of drawing, animation, and interaction logic within a single visual environment.

4. Modern Village View – Computer Graphics

URL: https://github.com/abdurtutul/Modern-Village-View-Computer-Graphics

Brief Description:

The *Modern Village View* project presents a 2D graphical representation of a village environment using OpenGL and C++. It focuses on rendering a complete composition that includes houses, trees, roads, fields, and other scenic elements. The code demonstrates the use of OpenGL primitives such as polygons, lines, and filled shapes to construct complex objects from basic shapes. The project also uses coordinate transformation and layering techniques to position elements relative to each other, giving a cohesive look to the scene. Unlike simple shape-drawing examples, this repository attempts to combine several graphical components into a structured environment, which helps in understanding scene design and modular code organization. However, the project primarily remains a **static scene** without extensive animation or user interaction. There is limited use of animation or dynamic event handling, which means objects remain fixed once drawn. Despite this, the project serves as a good example of foundational graphics programming, demonstrating how a visual environment can be built from basic drawing routines and emphasizing clean organization of scene elements.

5. Village_View

URL: https://github.com/Rezwan009/Village_View

Brief Description:

The *Village_View* repository focuses on creating a village scene using OpenGL with emphasis on graphical composition and shading. This project structures the environment by placing multiple object groups such as buildings, nature elements, and decorative objects in a coordinated layout. It also includes some animation behavior, such as moving objects or changing colors over time. The code highlights transformation functions and incremental object updates, which make the scene appear more lively compared to static renderings. User interaction may be supported through basic keyboard controls for toggling certain scene states. Compared to the *Modern Village View*, this project shows **more dynamic behavior**, making it closer in spirit to real-time simulation while still focusing on 2D rendering. The modular design and combination of multiple scene objects mimic real-time scene building and help in understanding graphics design patterns. Integration of animation elements such as movement and color change distinguishes this project from more static examples.

6. Computer Graphics (jahidul39306)

URL: https://github.com/jahidul39306/Computer-Graphics

Brief Description:

This repository offers a collection of computer graphics implementations in OpenGL, demonstrating various rendering techniques. It includes examples of shape drawing, transformations, and potentially interactive scenes, making it a collection of multiple small graphics demonstrations. While it does not concentrate specifically on a village environment, it provides foundational graphics elements that could be repurposed for larger simulations. The project demonstrates core OpenGL drawing functions, object positioning, and basic input handling. As a result, it serves as a reference for understanding individual graphics modules rather than an integrated environment. Compared to the other two repositories, this project is more academic in nature, focusing on discrete graphics tasks rather than complete scene design or animation. It highlights foundational concepts and practical usage of graphics primitives in isolated contexts, which can serve as building blocks for environmental simulation.

7. Computer Graphics Project – Natural Village

URL: https://github.com/HrithikMojumdar/Computer-Graphics-Project-Natural-Village-

Brief Description:

This project presents a comprehensive 2D village scene using OpenGL. It focuses on drawing multiple graphical components such as trees, houses, fields, rivers, and other natural elements to simulate a realistic rural environment. The codebase emphasizes the organization of scene elements into modular drawing functions, allowing each object to be rendered independently and then composed into a complete view. Although the project primarily consists of static rendering, it demonstrates careful planning of object coordinates and relative positioning, contributing to an aesthetically pleasing scene layout. The use of basic shapes and primitives such as polygons, circles, and transformation functions helps reinforce foundational computer graphics principles. However, compared to fully animated projects, this repository lacks significant dynamic behavior and interactivity. Despite this, it serves as a solid demonstration of scene composition fundamentals and shows how multiple objects can be combined to portray a detailed village landscape.

8. Computer_Graphics_Project

URL: https://github.com/eiemon12/Computer_Graphics_Project

Brief Description:

The *Computer_Graphics_Project* repository is an example of a varied graphics assignment set implemented in OpenGL. It includes several graphical functions and scene rendering modules that highlight basic drawing techniques such as line drawing, circle rendering, and shape transformations. The project may also feature simple keyboard interaction to trigger changes in the scene, providing limited interactivity. Unlike elaborate environment simulations, this repository appears to focus more on demonstrating individual graphics algorithms rather than integrating them into a unified environment. While it offers useful modules for learning the fundamentals of OpenGL rendering, the project lacks cohesive scene structure and extensive animation, making it more suitable as a reference for basic graphics techniques than as a complete visual experience.

9. Seasons Scenario in City and Village

URL: https://github.com/mddaudhossainsupto/Seasons-Scenario-In-City-And-Village

Brief Description:

This project explores dynamic scene rendering by depicting seasonal changes in both city and village environments. It includes multiple scenarios that illustrate different weather and environmental conditions such as summer, monsoon, autumn, and possibly winter. The work employs animation to simulate elements like moving clouds, falling rain, or changing sun positions. The dynamic nature of seasonal simulation sets this project apart from static renderings, as it demonstrates how environmental factors affect overall scene aesthetics and mood. However, the complexity of animation and user interaction may be limited or scripted, making this project closer in scope to weather-focused simulations rather than fully interactive environments. It serves as a compelling example of how seasonal variations can be incorporated in graphical scenes, offering insights into time and weather simulation techniques.

10. Day–Night Cycle Graphics Project

URL: https://github.com/kneerose/day-night

Brief Description:

The *Day–Night* repository focuses on the implementation of a scene that transitions between day and night states. It highlights the use of color gradients, lighting changes, star rendering, and possibly basic animation to convey time progression. This project explores the visual impact of time change on scene elements, such as sky color and object silhouettes. While it may include basic user interaction or automated transitions, the scope remains centered on time-based visual variation. This work provides a clear example of graphics concepts like shading and color manipulation, making it a valuable reference for understanding how visual environments evolve with time.

3. Implementation

3.1 The key features of the project

a) **Interactive Day–Night Cycle:** The project supports a real-time **day and night transition**. Users can toggle between day and night using a keyboard key.

- During **daytime**, the scene shows a bright sky, sunlight, green fields, and clear visibility.
- During **nighttime**, the sky becomes dark, stars and the moon appear, and artificial lights (tractor headlights, house windows) become active.

b) **Rain Mode with Realistic Weather Effects:** A **rain mode** can be turned on or off independently of day or night.

- When rain is enabled, the sky becomes cloudy and darker.
- Slanted raindrops fall continuously with wind effect.
- A semi-transparent overlay gives a realistic rainy atmosphere.
- Rain works in **both day and night modes**, enhancing realism.

c) **Animated Natural Elements:** Several environmental elements are continuously animated:

- **Clouds** move slowly across the sky at different speeds.
- **Birds** fly continuously, giving life to the sky.
- **Windmill blades** rotate continuously regardless of other scene states.
- **River waves** simulate flowing water using sinusoidal motion.

d) **Controllable Tractor with Headlight System:** The tractor is an interactive object with multiple controls:

- Move / stop using a key toggle.
- Speed up and slow down incrementally.
- Direction change (left/right movement).
- Speed reset to default.
- At night, the **tractor headlight automatically turns on** when the tractor is moving.

e) **Persistent Background Activity:** Even when the tractor is stopped:

- Clouds continue moving.
- Birds keep flying.
- Windmill keeps rotating.
  This separation of animation logic improves realism and scene continuity.

f) **Dynamic Scene Coloring:** The scene colors adapt dynamically based on:

- Day or night state.
- Rain mode state.
  Objects such as trees, hills, fields, sky, and clouds adjust brightness and tone accordingly.

g) **Structured Graphics Algorithms:** The project uses classic computer graphics algorithms:

- **DDA Line Algorithm** for drawing birds and raindrops.
- **Midpoint Circle Algorithm** for wheels and circular shapes.
  These algorithms demonstrate fundamental graphics concepts.

h) **Modular and Human-Readable Code Structure**

- Code is divided into logical sections (sky, rain, tractor, windmill, etc.).
- Uses human-readable formatting and full conditional blocks.
- Avoids shorthand expressions for clarity and academic readability.
- Suitable for learning, extension, and presentation.

i) **Keyboard-Based User Interaction:** The scene responds to multiple keyboard inputs:

- Toggle day/night
- Toggle rain mode
- Control tractor movement and speed
- Reset values and exit program
  This makes the project fully interactive and user-driven.

j) **Realistic Village Environment Representation:** The combined features create a complete **animated rural village scene**, integrating:

- Nature
- Weather
- Human activity
- Time-based changes

This makes the project suitable for **Computer Graphics coursework**, demonstrations, and further expansion.

3.2 Key Technologies Used in the Project

i. **OpenGL (Open Graphics Library):** OpenGL is the core graphics technology used to render all visual elements of the project. It provides low-level control over drawing shapes, colors, transformations, and animations.

In this project, OpenGL is used to:

- Draw basic geometric primitives such as points, lines, polygons, and circles
- Render complex objects like houses, trees, tractors, windmills, clouds, and hills
- Apply transformations such as translation, rotation, and scaling
- Manage color blending and transparency for lighting and rain effects

ii. **GLUT (OpenGL Utility Toolkit):** GLUT is used to manage window creation and user interaction.

Its role in the project includes:

- Creating and managing the display window
- Handling keyboard input for controlling day/night mode, rain mode, and tractor movement
- Handling timer-based animation using callback functions
- Managing the rendering loop efficiently

iii. **C++ Programming Language:** C++ is used as the primary programming language for implementing the project logic.

C++ enables:

- Efficient execution of real-time animations
- Structured and modular program design
- Use of functions for reusable drawing and animation logic
- Memory-efficient handling of multiple animated objects

iv. **Computer Graphics Algorithms:** The project explicitly implements classical graphics algorithms to demonstrate theoretical understanding.

a) DDA Line Algorithm

Used for:

- Drawing bird wings
- Rendering slanted raindrops

This algorithm incrementally plots points between two endpoints, ensuring smooth and consistent lines.

b) Midpoint Circle Algorithm

Used for:

- Drawing tractor wheels
- Creating circular outlines

This algorithm efficiently plots symmetric points of a circle using integer arithmetic.

v. **Timer-Based Animation System:** The animation system is built using GLUT timer callbacks.

This approach allows:

- Continuous movement of clouds, birds, windmill blades, rain, and tractor
- Frame-rate–independent animation control
- Separation of animation logic from drawing logic

vi. **Alpha Blending and Transparency:** Alpha blending is used to create realistic visual effects.

Applications include:

- Tractor headlight glow at night
- Rain fog overlay
- Light diffusion during rainy and night scenes

This is achieved using OpenGL's blending functions.

vii. **Trigonometric Functions:** Mathematical functions such as sine and cosine are used to simulate natural motion.

They are applied to:

- River wave animation
- Circular shape generation
- Smooth rotational movement

viii. **Event-Driven Interaction:** The project follows an event-driven programming model.

Events include:

- Keyboard key presses
- Timer events for animation updates
- Window resizing events

This model ensures responsive and interactive scene control.

ix. **Modular Rendering Architecture:** Each scene component (sky, rain, tractor, windmill, etc.) is rendered through separate functions.

Benefits include:

- Easy debugging and maintenance
- Clear logical separation
- Easy future feature expansion

x. **Real-Time Simulation Environment:** By combining OpenGL rendering, animation, and interaction, the project functions as a real-time simulation of a rural village environment with dynamic weather and time changes.

3.3 List of Key Functions Used in the Project

i. `main()`

- Entry point of the program
- Initializes GLUT and OpenGL environment
- Creates the display window
- Registers callback functions
- Starts the main rendering loop

ii. `display()`

- Core rendering function
- Draws the complete village scene frame by frame
- Calls all individual drawing functions (sky, trees, tractor, rain, etc.)
- Handles layering of objects and visual effects

iii. `Timer(int value)`

- Controls all animations in the scene
- Updates positions of clouds, birds, rain drops, windmill blades, and tractor
- Ensures smooth real-time motion using a fixed refresh rate

iv. `keyboard(unsigned char key, int x, int y)`

- Handles user keyboard input
- Toggles day/night mode
- Toggles rain mode
- Controls tractor movement, speed, and direction

v. `reshape(int width, int height)`

- Handles window resizing
- Maintains correct aspect ratio
- Updates viewport and rain distribution when the window size changes

vi. `setOrtho2D()`

- Sets up the 2D orthographic projection
- Converts the window into a 2D coordinate system
- Ensures consistent drawing across different screen sizes

vii. `drawSkyAndHills()`

- Draws sky background based on day, night, or rain mode
- Renders stars, moon, clouds, and hills

viii. `drawFieldAndRoad()`

- Renders ground layers including grass, road, and field
- Adjusts colors according to weather and time

ix. `drawCloud(float x, float y, float scale)`

- Draws individual clouds
- Used repeatedly to create animated cloud movement
- Changes color based on weather conditions

x. `drawBird(float x, float y, float scale)`

- Draws a single bird using line primitives
- Used to simulate flying birds across the sky

xi. `drawBirds()`

- Calls multiple `drawBird()` functions
- Manages formation of birds in the sky

xii. `drawTree(float x, float y, float scale)`

- Draws trees using trunk and foliage
- Used for foreground and background vegetation

xiii. `drawBarn()`

- Draws the barn structure in the village
- Includes walls, roof, and door

xiv. `drawWindmillTower()`

- Draws the windmill base using DDA line algorithm

xv. `drawWindmillBlades(float cx, float cy)`

- Draws rotating windmill blades

- Uses rotation transformations

xvi. `drawWindmill()`

- Combines windmill tower and blades
- Manages complete windmill rendering

xvii. `drawWheel(int cx, int cy, int rOuter, int rInner)`

- Draws tractor wheels
- Uses Midpoint Circle Algorithm

xviii. `drawTractorShapeLocal()`

- Draws the tractor body using polygons
- Contains local coordinate–based design

xix. `drawTractor(float x, float y)`

- Positions the tractor in the scene
- Handles direction flipping and headlights

xx. `drawHeadlightBeamLocal()`

- Creates headlight light cone using alpha blending
- Active during night movement

xxi. `initRain()`

- Initializes rain particle positions and speeds

xxii. `drawRain()`

- Renders animated rainfall
- Applies transparency for realistic rain effect

xxiii. `ddaLine(int x1, int y1, int x2, int y2)`

- Implements Digital Differential Analyzer algorithm
- Used for birds and rain streaks

xxiv. `midpointCircle(int xc, int yc, int r)`

- Implements Midpoint Circle algorithm
- Used for circular wheel outlines

xxv. Helper Functions

- `filledCircle()` – Draws filled circles

- `rectf()` – Draws filled rectangles
- `drawStar()` – Draws star shapes at night

4. Screenshot of the project

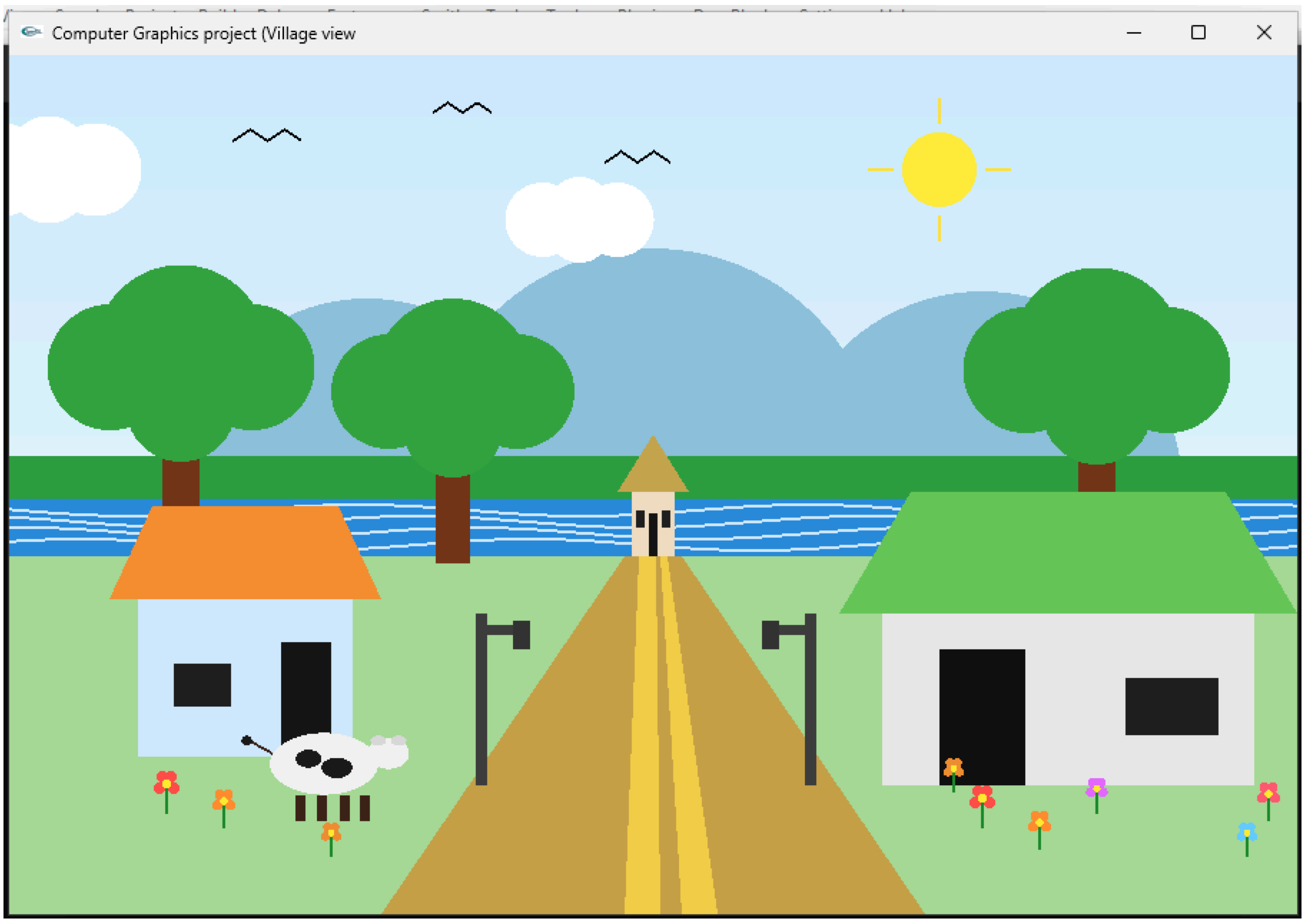

Figure 01: Village Road view with natural elements, sunrise, night view and daily activities in DAY MODE

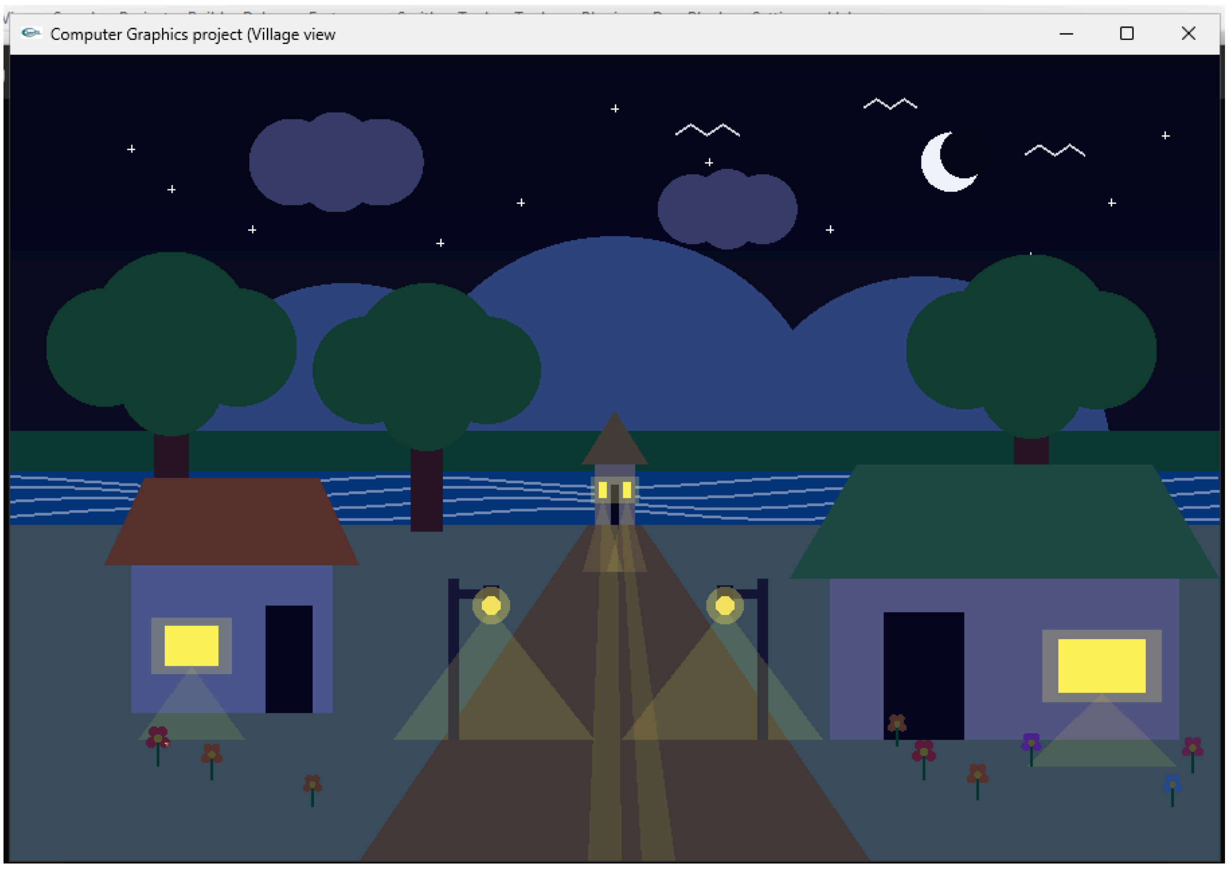

Figure 02: Village Road view with natural elements, sunrise, night view and daily activities in NIGHT MODE

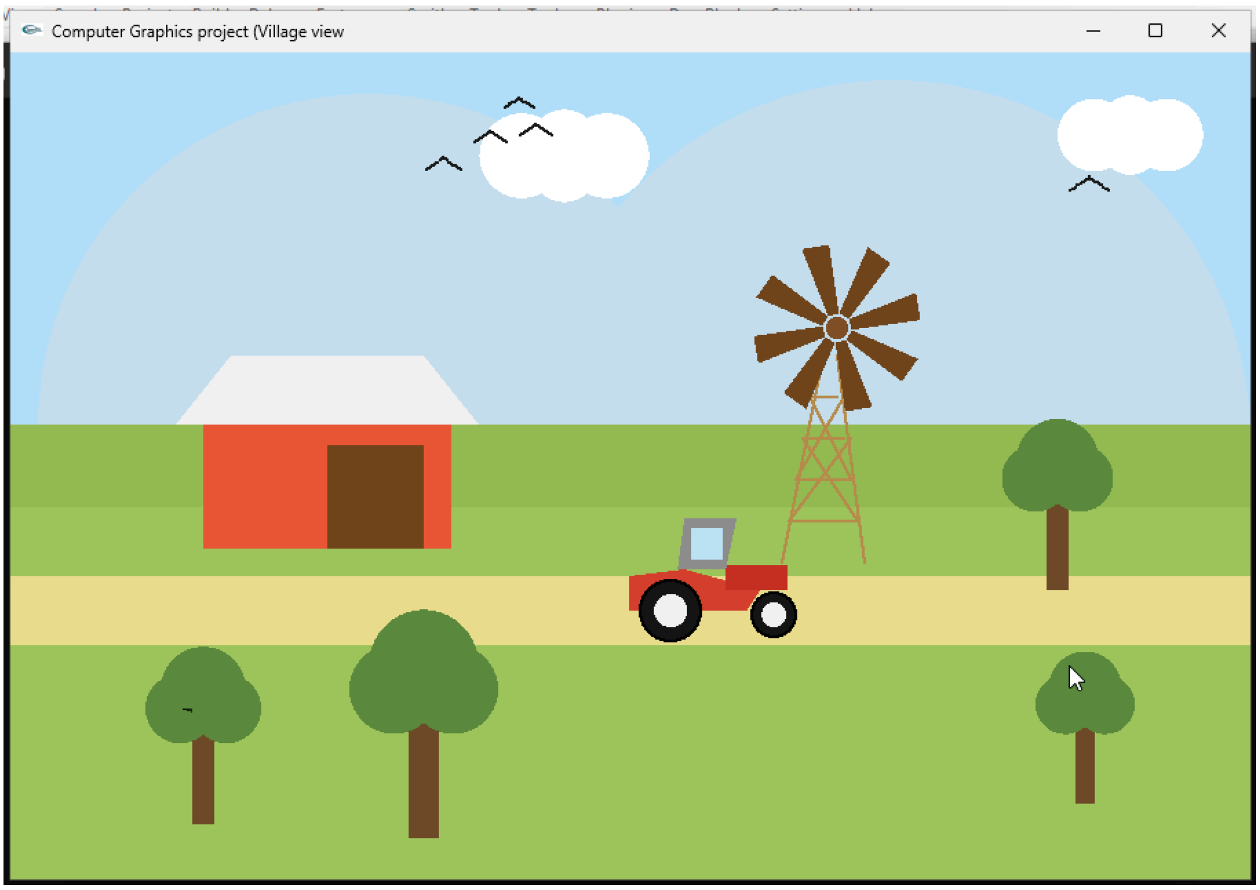

Figure 03: Road behind crop field with a tractor, windmill etc. in DAY MODE WITHOUT RAINING STATE

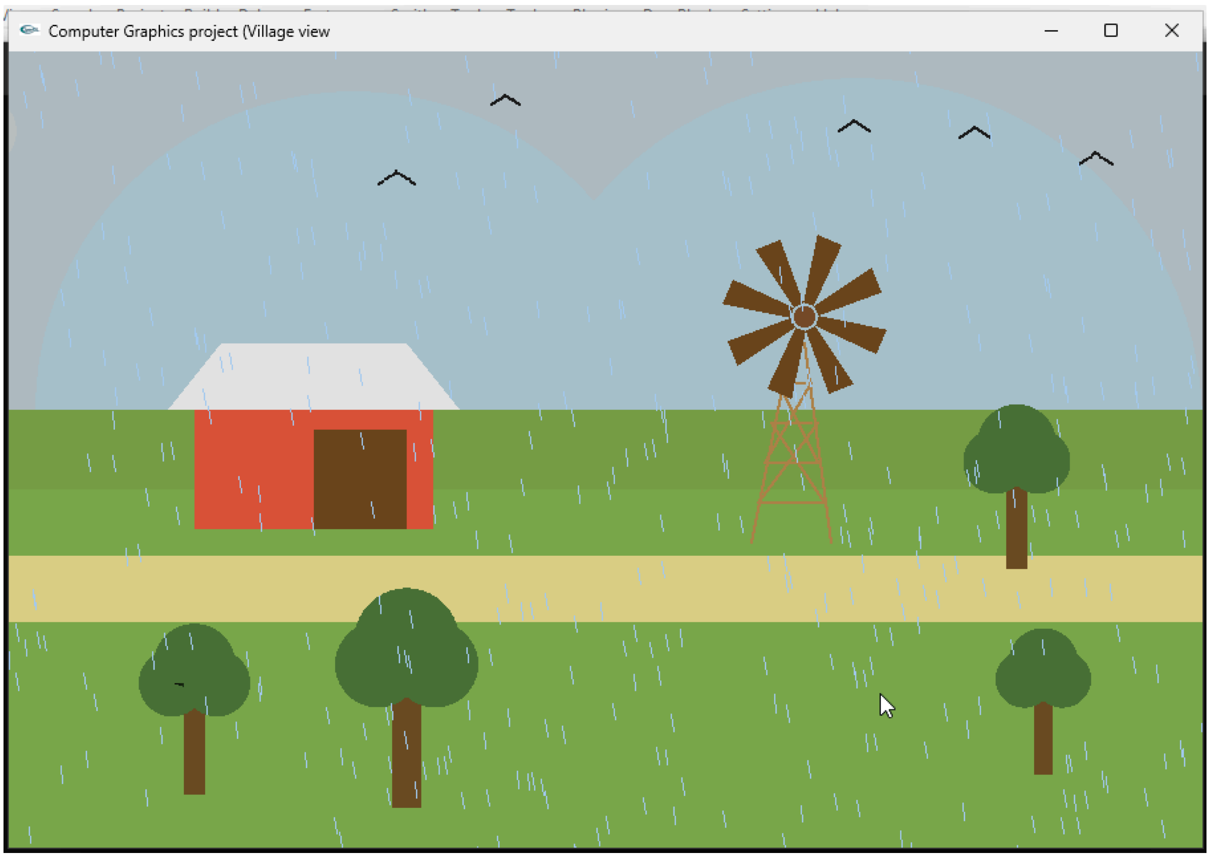

Figure 04: Road behind crop field with a tractor, windmill etc. in DAY MODE WITH RAINING STATE

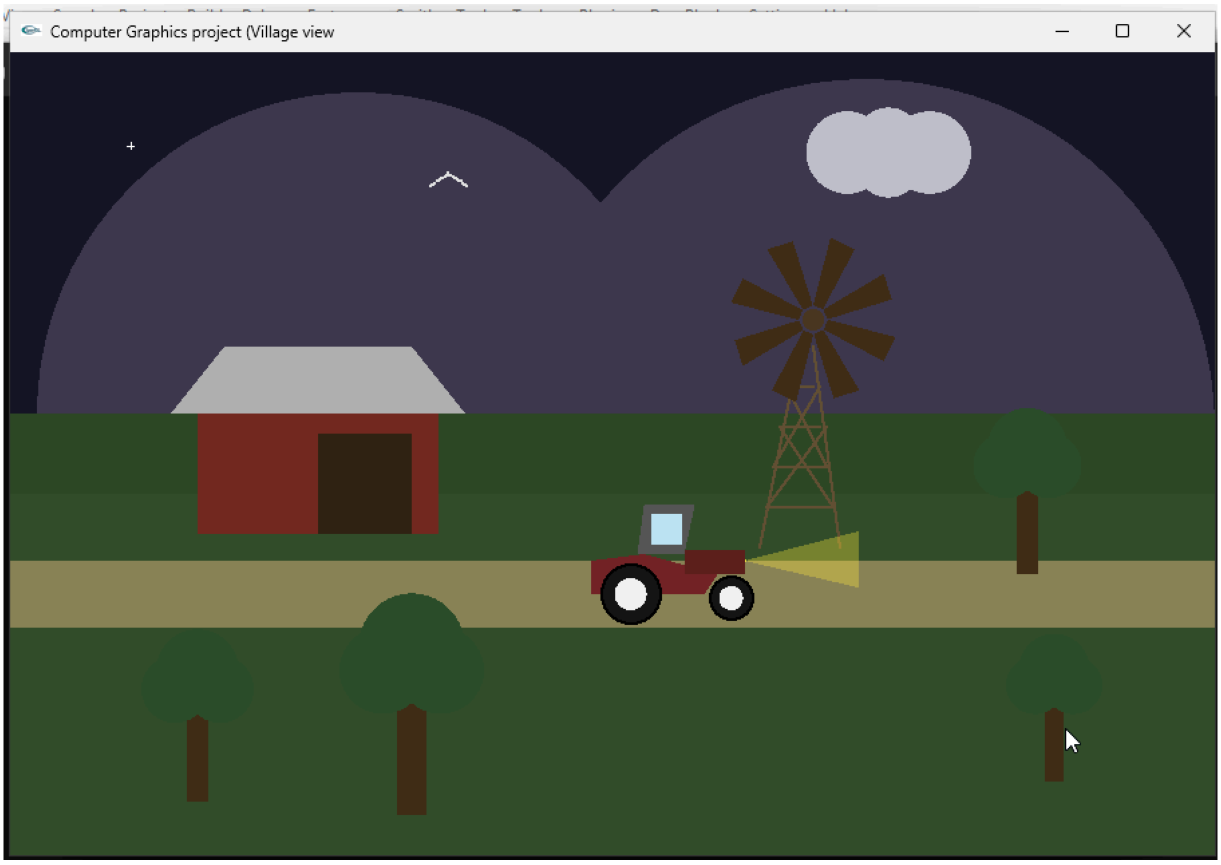

Figure 05: Road behind crop field with a tractor, windmill etc. in NIGHT MODE WITHOUT RAINING STATE

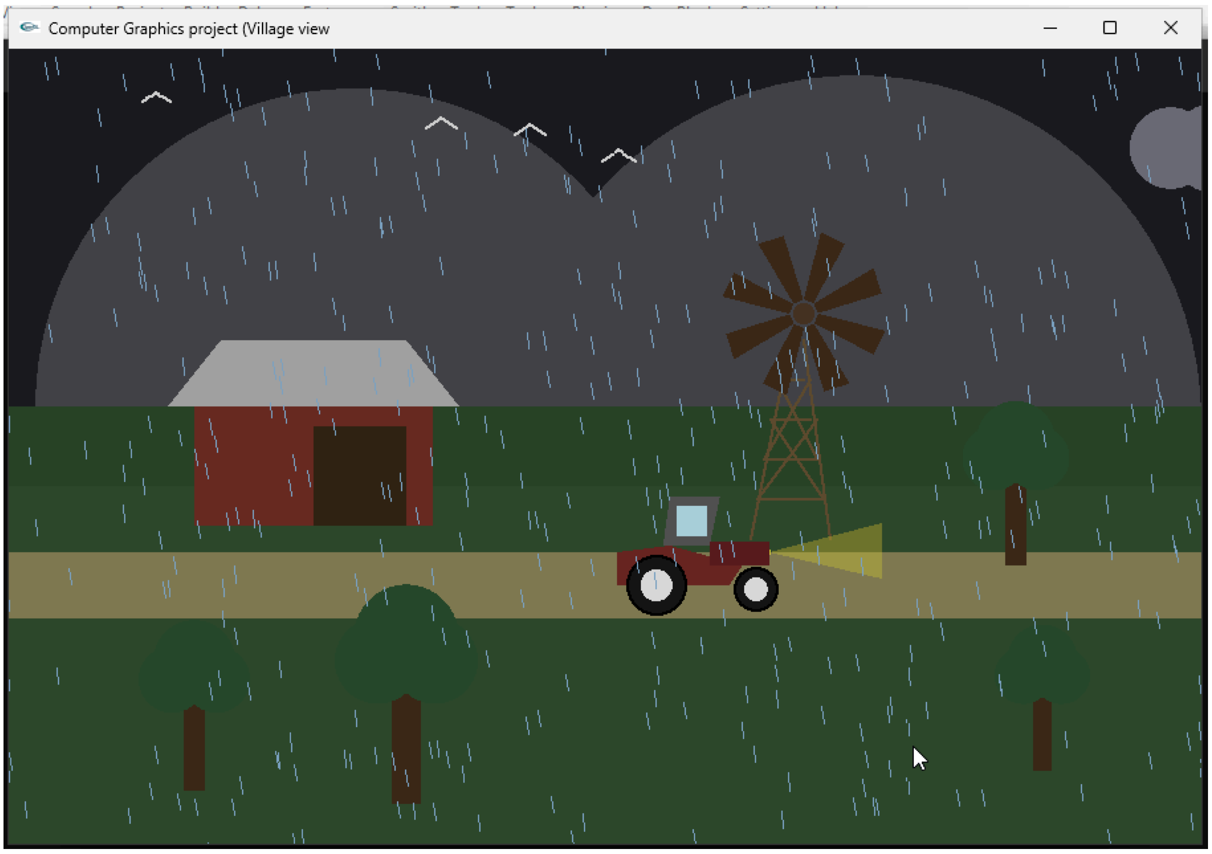

Figure 06: Road behind crop field with a tractor, windmill etc. in NIGHT MODE WITH RAINING STATE

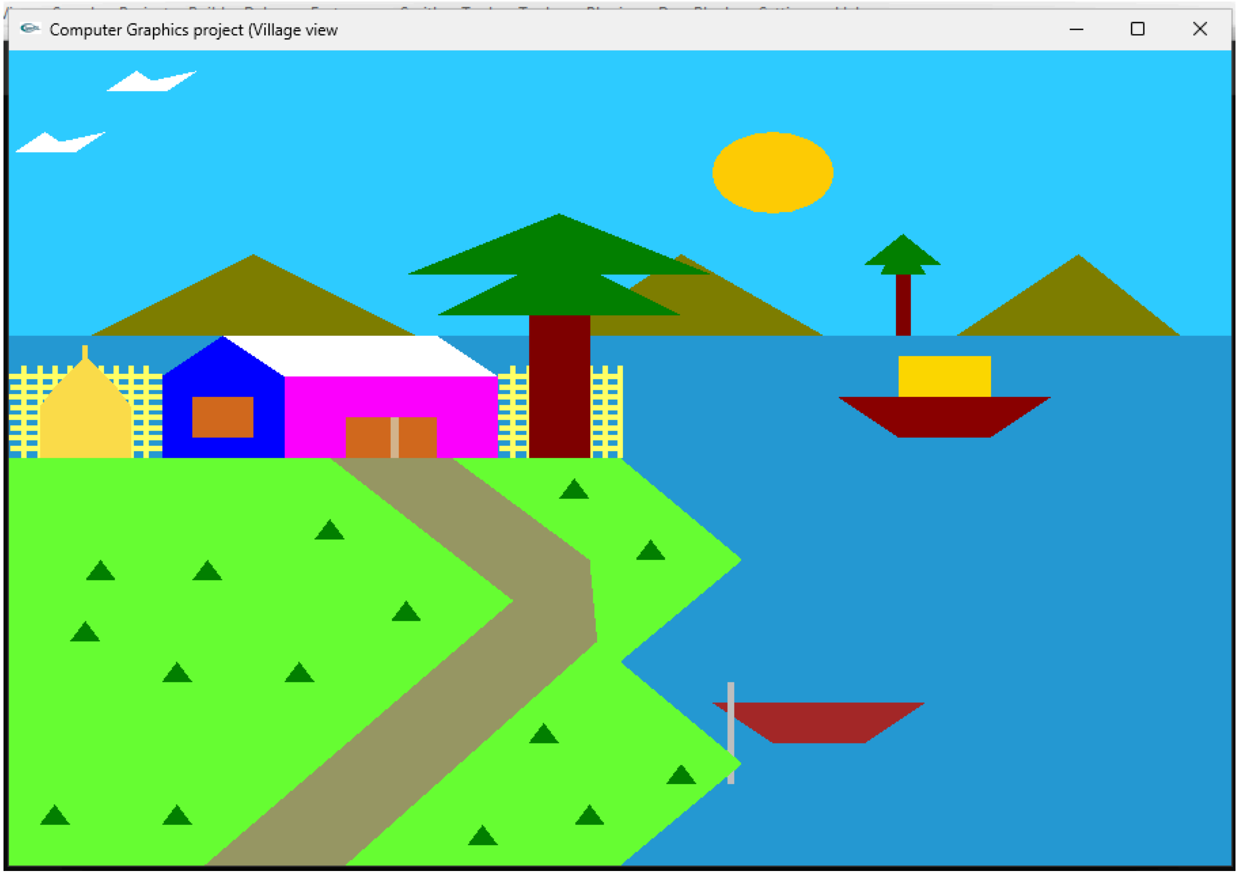

Figure 07: A peaceful morning scene where the sun rises from behind the river in DAY MODE

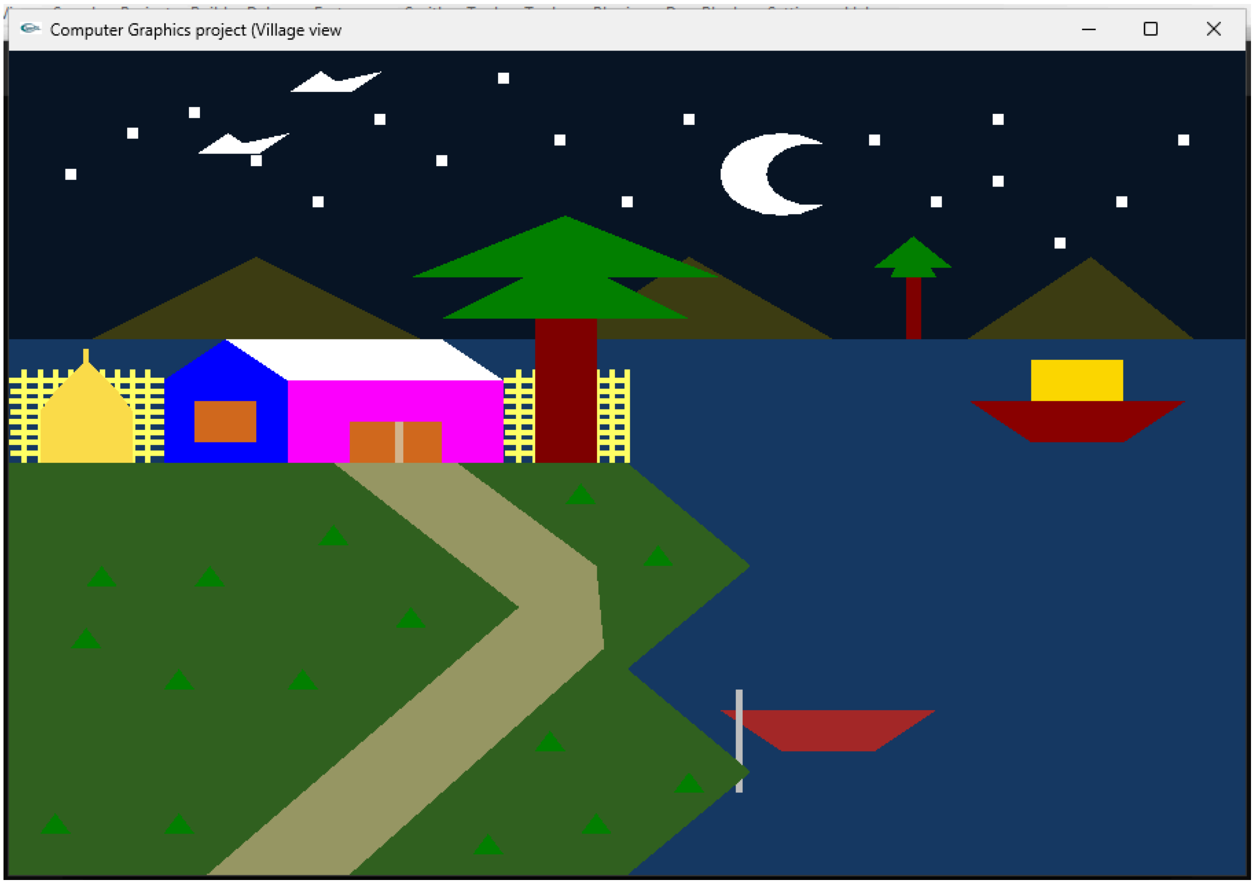

Figure 07: A peaceful morning scene where the sun rises from behind the river in NIGHT MODE

5. Code availability

Link: https://github.com/NayemSiddiki02/Computer-Graphics

6. Future Work of the Project

i. **Advanced Weather Effects:** Additional weather conditions such as fog, storm clouds, lightning, and snowfall can be introduced to make the environment more realistic.

ii. **Seasonal Environment Changes**: The village scene can be enhanced by adding seasonal variations like summer, rainy season, autumn, and winter, with changes in tree colors, ground textures, and lighting.

iii. **Multiple Vehicle Types:** The tractor can be replaced or supplemented with other vehicles such as cars, buses, or bicycles with different motion behaviors.

iv. **Day–Night Transition Animation**: A smoother transition between day and night using gradual color interpolation and dynamic light intensity can improve visual realism.

v. **Sound Integration**: Environmental sounds such as wind, rain, engine noise, birds, and night insects can be added to increase immersion.

vi. **Camera Movement and Zooming**: Introducing camera panning, zooming, and scene rotation would allow users to explore the environment interactively.

vii. **Texture Mapping**: Applying texture mapping on roads, buildings, trees, and vehicles would significantly enhance visual detail.

viii. **Shadow and Lighting Models**: Real-time shadows and basic lighting models (ambient, diffuse) can be implemented for better depth perception.

ix. **Object Interaction**: Interactive elements such as opening doors, switching lights manually, or reacting to mouse clicks can make the scene more engaging.

x. **Optimized Rendering Performance:** The project can be optimized using better data structures, object grouping, and reduced draw calls for smoother performance.

xi. **3D Scene Extension:** The current 2D village scene can be extended into a 3D environment using perspective projection and depth buffering.

xii. **AI-Based Movement:** Vehicles, birds, or animals can follow intelligent paths using basic AI algorithms instead of fixed trajectories.

7. Conclusion

7.1 Limitations of the Project

1. **2D Graphics Only**: The project is limited to a 2D environment, so depth perception and realistic spatial movement are restricted.
2. **Basic Lighting Model**: The scene does not use advanced lighting techniques such as dynamic shadows, reflections, or global illumination.
3. **No Texture Mapping**: Most objects are drawn using solid colors instead of texture images, which reduces visual realism.
4. **Limited Interactivity**: User interaction is mostly keyboard-based; mouse-driven or object-level interaction is minimal.
5. **Simple Animations**: Movements like clouds, birds, windmill rotation, and tractor motion follow fixed paths and speeds without physics-based behavior.
6. **No Physics Simulation**: Real-world physics such as gravity, collision detection, and object response are not implemented.
7. **Single Scene Environment**: The project displays only one village scene without multiple maps or environments.
8. **Performance Dependency on Hardware**: Rendering performance may vary depending on system hardware, especially on lower-end machines.
9. **No Audio Support**: The project does not include background music or environmental sound effects.
10. **Limited Scalability**: Adding many more objects or effects may require significant code restructuring and optimization.

7.2 Conclude with a brief summary

In conclusion, the project successfully demonstrates an interactive village scene that combines graphical drawing techniques with real-time animation and user control. By integrating dynamic elements such as day–night transitions, weather changes, moving objects, and lighting effects, the scenario creates a realistic and engaging visual experience. The project highlights the effective use of computer graphics principles to simulate a natural rural environment, making it a solid example of how simple graphics algorithms and event-driven programming can be used to build an immersive and meaningful scene.

8. References

1. radwanromy. (n.d.). *Computer Graphics Project – Village View*. GitHub. https://github.com/radwanromy/Computer-Graphics-Project-Village-View

2. zubair-ahmed-official. (n.d.). *Computer Graphics Project*. GitHub. https://github.com/zubair-ahmed-official/Computer-Graphics-Project

3. AbzanaSultanIra. (n.d.). *Village View – Computer Graphics Project*. GitHub. https://github.com/AbzanaSultanIra/Village-view--Computer-Graphics-project

4. abdurtutul. (n.d.). *Modern Village View – Computer Graphics*. GitHub. https://github.com/abdurtutul/Modern-Village-View-Computer-Graphics

5. Rezwan009. (n.d.). *Village_View*. GitHub. https://github.com/Rezwan009/Village_View

6. jahidul39306. (n.d.). *Computer Graphics*. GitHub. https://github.com/jahidul39306/Computer-Graphics

7. HrithikMojumdar. (n.d.). *Computer graphics project – Natural village* [GitHub repository]. GitHub. https://github.com/HrithikMojumdar/Computer-Graphics-Project-Natural-Village-

8. Eiemon12. (n.d.). *Computer graphics project* [GitHub repository]. GitHub. https://github.com/eiemon12/Computer_Graphics_Project

9. Hossain, M. D. (n.d.). *Seasons scenario in city and village* [GitHub repository]. GitHub. https://github.com/mddaudhossainsupto/Seasons-Scenario-In-City-And-Village

10. Kneerose. (n.d.). *Day-night* [GitHub repository]. GitHub. https://github.com/kneerose/day-night