Title

Interactive 2.5D Cargo Port Simulation

Submitted By

NAME	ID
AYESHA KHATUN	22-49442-3
TURJO DAS DIP	22-48558-3
SHAMMI AKTER RIMI	22-49232-3
AHANAF TANVIR SHOVON	22-49221-3

American International University-Bangladesh (AIUB)

Computer Graphics & CSC4118

Dipta Justin Gomes

June 28, 2025

Abstract

The Interactive 2.5D Cargo Port Simulation project presents a real-time animated model of an industrial dockyard. Developed using C++ and OpenGL, the simulation illustrates a functional port environment with animated ships, cranes, trucks, containers, and buildings. It also integrates a day-night cycle and weather effects to reflect dynamic environmental changes. Designed to demonstrate key computer graphics principles like matrix transformations, motion, and layering, the project emphasizes modular design, timed animations, and foundational graphics programming. Although currently non-interactive, the simulation framework supports future expansions such as user-controlled views, physics, and sound enhancement. This project serves as a practical educational tool for visualizing industrial logistics through 2.5D graphics.

Keywords: OpenGL, Cargo Port, 2.5D Graphics, Simulation, C++, Animation, Real-time Rendering

Introduction

Cargo ports are essential to global logistics, enabling the transfer of goods across regions. As technology advances, visualizing such industrial systems has become crucial for education, planning, and training. Our project, the Interactive 2.5D Cargo Port Simulation, was developed for the CSC4118 Computer Graphics course at AIUB to explore real-time rendering and animation using C++ and OpenGL.

This simulation replicates a busy port environment, showcasing animated ships, cranes, trucks, containers, and buildings. The inclusion of environmental dynamics like a changing sky, moving clouds, and day-night transitions adds realism. The modular design helps manage complexity, and OpenGL's transformation functions allow precise control over animations.

The motivation for choosing this topic was to apply theoretical knowledge in a practical setting, simulating industrial logistics visually and interactively. Cargo ports are ideal for such demonstrations due to their dynamic and structured nature. By implementing this simulation, we strengthened our understanding of scene management, matrix operations, modular programming, and visual storytelling.

The report includes a review of related works, tools and technologies used, implementation details, results, and future development possibilities. It highlights the educational value of using computer graphics for visual simulations in industrial domains.

Related Work

We drew inspiration from tutorials, simulation games, and technical documentation. The NeHe OpenGL tutorials (NeHe, 2000) played a foundational role in teaching transformation, animation logic, and object rendering using OpenGL. These lessons shaped how we approached modeling and animating objects in our project.

We also took conceptual ideas from simulation games such as Port Simulator 2012 (Valve Corporation, 2012) and classic titles like Transport Tycoon Deluxe. These games visualize complex logistics in simplified environments and influenced our scene design and feature scope.

From a technical standpoint, GLUT (Kilgard, 1996) was essential for window management and timed animations. While we used OpenGL's legacy immediate mode (glBegin, glEnd, etc.), the modular structure supports potential migration to modern programmable pipelines like GLSL or shader-based rendering. Other inspirations included community forums and previous academic projects shared in GitHub repositories or graphics research articles.

Tools, Technologies & Libraries

The simulation was developed in C++ using OpenGL for rendering and GLUT for window management and timing. We chose C++ for its speed and control, and OpenGL's immediate mode for simplicity in drawing shapes and handling transformations.

The project was built in Code::Blocks IDE on Windows 10, offering easy OpenGL integration and debugging tools. We used matrix transformation functions such as glPushMatrix, glTranslatef, and glScalef to animate and position objects in the scene. Color transitions and animation states were handled with global variables and custom update functions.

Environmental sounds (e.g., rain, ship engines, truck movement) were implemented using .wav audio files and simple sound playback libraries. These sounds add depth and immersion to the visual scenes. Future versions may adopt advanced libraries like OpenAL or SDL_mixer for enhanced audio handling.

Assets such as sound effects and custom-defined shapes (containers, cranes, ships, etc.) were coded manually, and weather effects were created using line drawing and animation functions.

System Design and Implementation

- **Scene Description**

The simulation presents a 2.5D cargo port environment that includes animated ships, cranes, containers, warehouses, buildings, streetlights, and atmospheric elements like the sun, moon, and drifting clouds. The port area is constructed using modular OpenGL functions, enabling hierarchical rendering and real-time animation. Elements such as cranes lift and lower containers, ships enter and exit the port, and streetlights illuminate during nighttime scenes. Each object is drawn using basic OpenGL primitives with custom color palettes and shading to simulate depth and motion. The entire environment dynamically transitions between day and night, adding realism and immersion.

- **User Interface**

The user interface is minimal and primarily visual. The simulation runs in fullscreen mode and automatically cycles through environmental changes such as lighting and sky transitions. While no graphical UI elements like buttons or menus are included in this version, the clear differentiation of scenes and animated cues provide intuitive visual feedback.

The current implementation relies on automatic timed animations, with no keyboard or mouse interactivity enabled. However, the GLUT framework has been used, allowing for future expansion into interactive controls, such as toggling views or manually operating the cranes and ships.

- **Challenges & Solutions**

One major challenge was managing complex object hierarchies and scene layering. To solve this, glPushMatrix and glPopMatrix were used extensively to isolate transformations. Another issue was achieving smooth, synchronized animations. This was addressed through frame-based updates and global variables tracking object states. Lastly, ensuring color consistency between day and night modes required defining distinct color schemes and dynamically applying them using a custom set_color() function.

Results & Demonstration

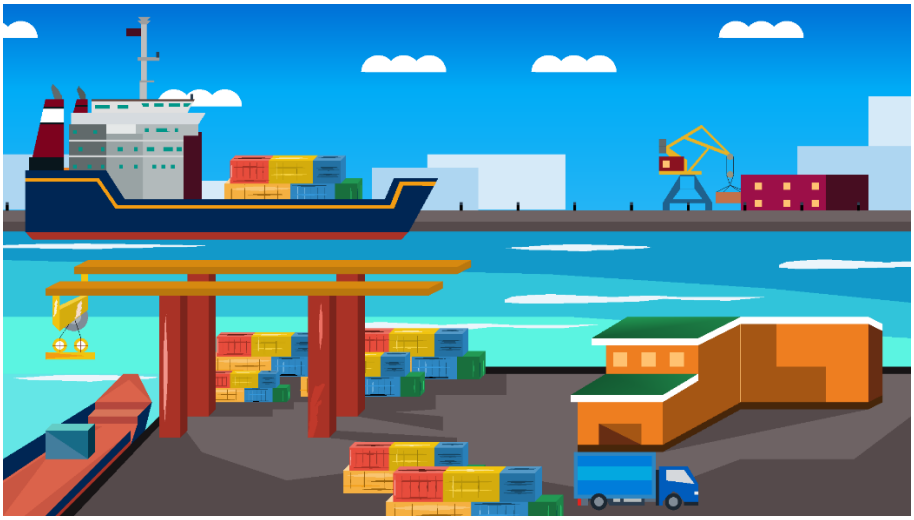

Fig 1. Port Morning View (Scenario 1)

Figure 1 presents the cargo port during a clear morning. The sky is bright, and the sun may be visible, casting natural daylight across the entire scene. Elements such as ships, cranes, containers, warehouses, and buildings are easily distinguishable due to high visibility. The ambient environment reflects a calm operational state, with standard animations active—such as moving clouds or slight water ripples—while no weather effects are applied. This serves as the baseline scenario, showcasing the static visuals and daytime ambiance of the port.

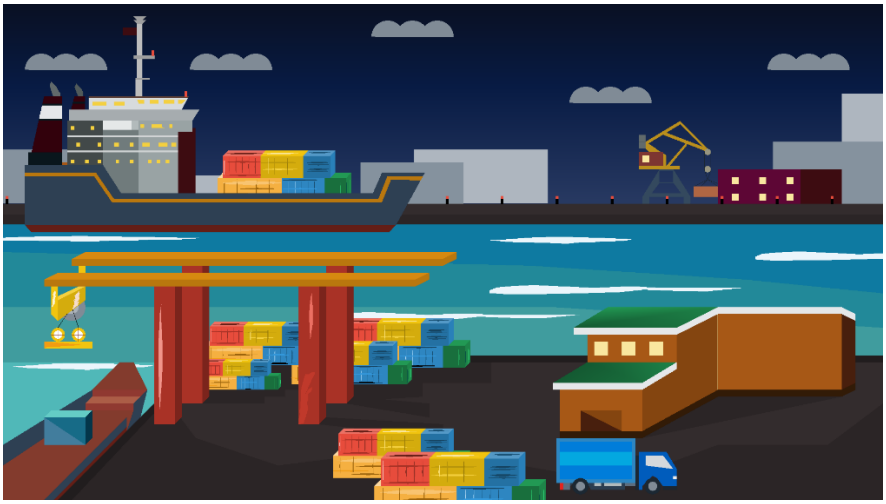

Fig 2. Port Night View (Scenario 1)

Figure 2 transitions the port into a nighttime environment. The sky adopts darker tones, with the possible presence of stars or the moon. Building and streetlights turn on, illuminating critical parts of the port such as roads and working areas. Despite the reduced visibility, key objects like cranes and containers remain highlighted due to artificial lighting. This figure emphasizes the simulation's lighting system and showcases the port's nighttime appearance using darker palettes and ambient effects.

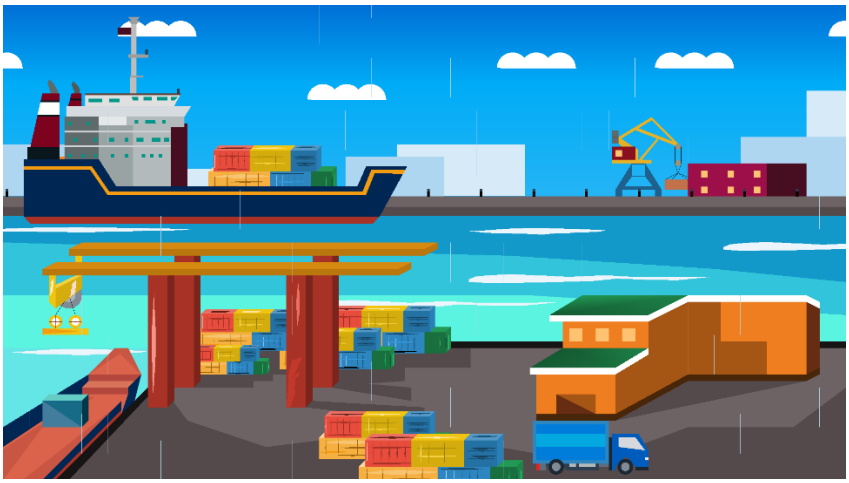

Fig 3. Port Morning View with Rain Effect (Scenario 1)

Figure 3 combines the bright daylight setting of the morning with a dynamic rain effect. Raindrops are animated as thin vertical lines falling across the scene, implemented using OpenGL line rendering. The scene may also feature subtle visual enhancements like rippling water or slightly muted colors to simulate a wet environment. This figure demonstrates the layering of weather effects on top of the base animation, adding realism to the otherwise static daylight setup.

Fig 4. Port Night View with Rain Effect (Scenario 1)

Figure 4 showcases the most complex environmental rendering in Scenario 1 by combining the darkness of nighttime with an active rainstorm. Rain lines fall continuously across the screen,

while artificial lights from the port infrastructure reflect off surfaces, enhancing the atmosphere. This scene reflects a heightened level of realism and demonstrates the simulation's capacity to manage multiple overlapping visual effects, including time-based lighting and weather dynamics.

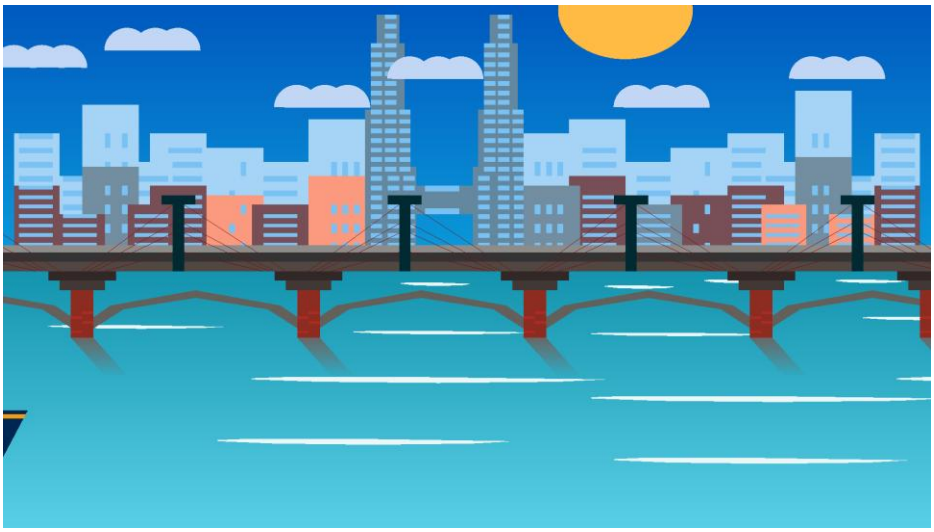

Fig 5. Morning View (Scenario 2)

Figure 5 introduces a redesigned port layout in a morning setting. The architectural elements, object placements, or visual styles differ from Scenario 1, showcasing the modular design of the simulation system. With the sun up and skies clear, the figure emphasizes clean lines, bright colors, and the adaptability of the rendering engine to support different port configurations while maintaining the same functional aesthetics.

Fig 6. Night View (Scenario 2)

Figure 6 continues with Scenario 2's layout but under nighttime conditions. The sky is darkened, and port lights are turned on to highlight essential areas and structures. Compared to the morning view, this scene employs cooler tones and emphasizes lighting effects more prominently. This figure highlights how the simulation handles lighting transitions across distinct layouts and maintains consistency in functionality despite visual differences.

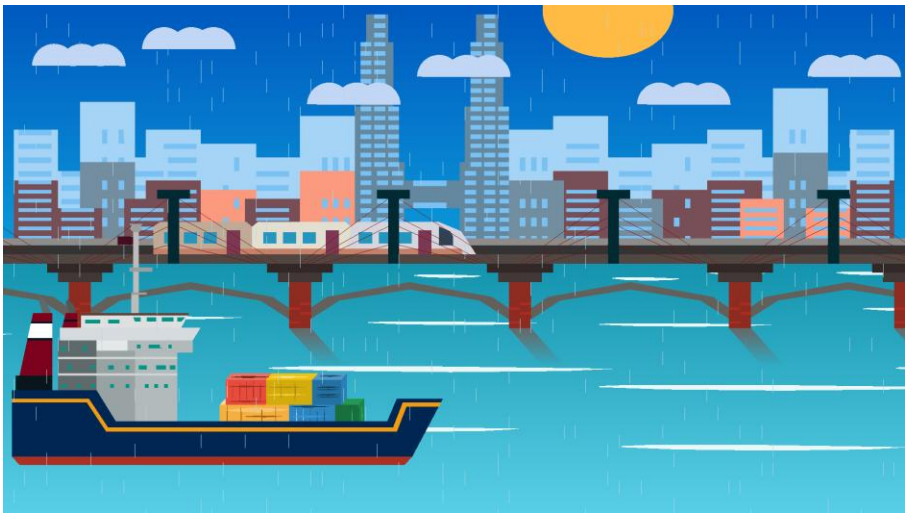

Fig 7. Morning View with Rain Effect (Scenario 2)

Figure 7 overlays the Scenario 2 morning scene with animated rainfall, combining a fresh port layout with weather effects. The simulation applies rain visuals uniformly across the scene while preserving the morning brightness. This demonstrates that the rain logic is modular and can be applied independently of scene design, reinforcing the environmental versatility of the simulation engine.

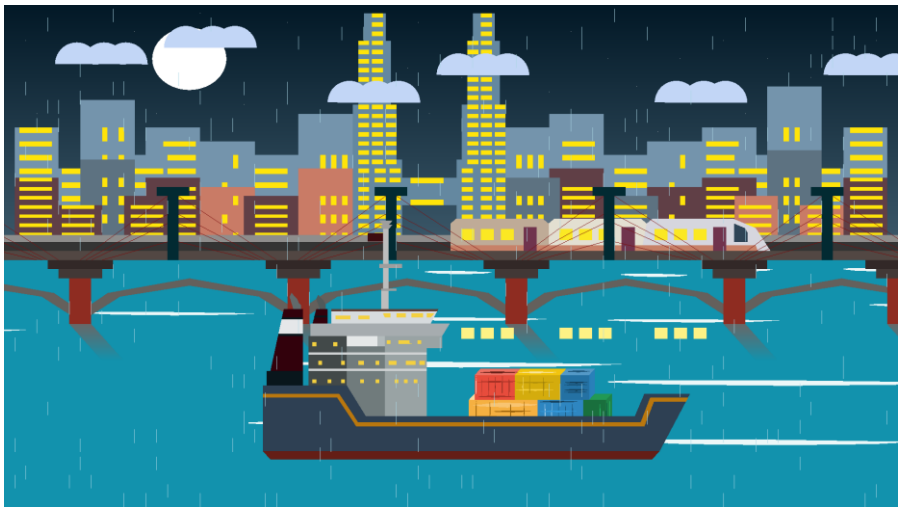

Fig 8. Night View with Rain Effect (Scenario 2)

Figure 8 is the most visually intensive in Scenario 2, merging the effects of nighttime and rainfall. Dim lighting, rain overlays, and object shadows work together to create a visually layered and immersive environment. The scene emphasizes depth and motion, reinforcing how well the system handles multiple overlapping effects, especially in varied layouts. It is a clear demonstration of the simulation's graphical complexity and realism.

Fig 9. Ship Moving (Scenario 1)

Figure 9 captures the animation of a ship moving either into or out of the port in Scenario 1. This motion is achieved through OpenGL transformations using variables like moveShip or moveShip3. The ship transitions smoothly across the water, simulating real docking or departure processes. This figure demonstrates animated object translation and is further enhanced by visual effects like rippling water or possibly sound cues for realism.

Fig 10. Truck Moving (Scenario 1)

Figure 10 illustrates the movement of a truck transporting cargo across the dockyard. The motion is driven by translation variables such as truckX, updating the truck's position frame by frame. This not only simulates logistical movement within the port but also reinforces multiple independent object animations working concurrently in the simulation environment.

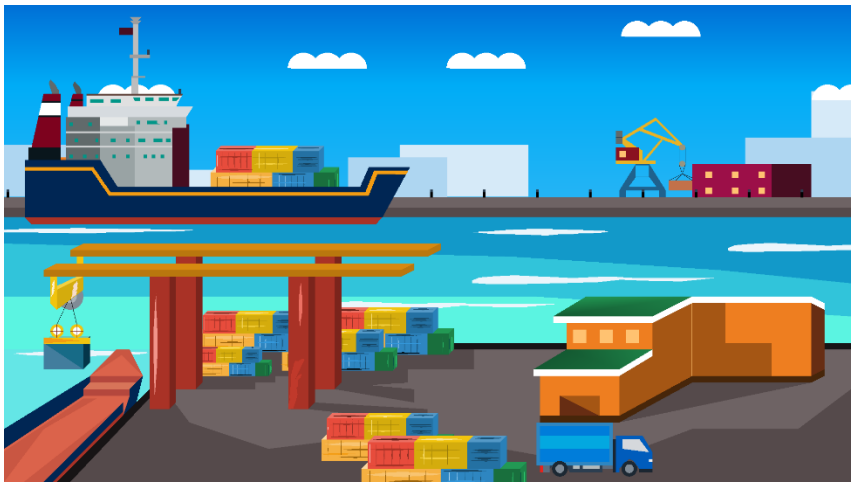

Fig 11. Container Pickup by Craine (Scenario 1)

Figure 11 shows a crane performing a container pickup sequence. The crane arm is animated to move downward, attach to the container, and begin lifting it. This is achieved using hierarchical transformation functions (glPushMatrix, glPopMatrix, and glTranslatef). The animation demonstrates object coordination and timing, representing realistic crane behavior in port operations.

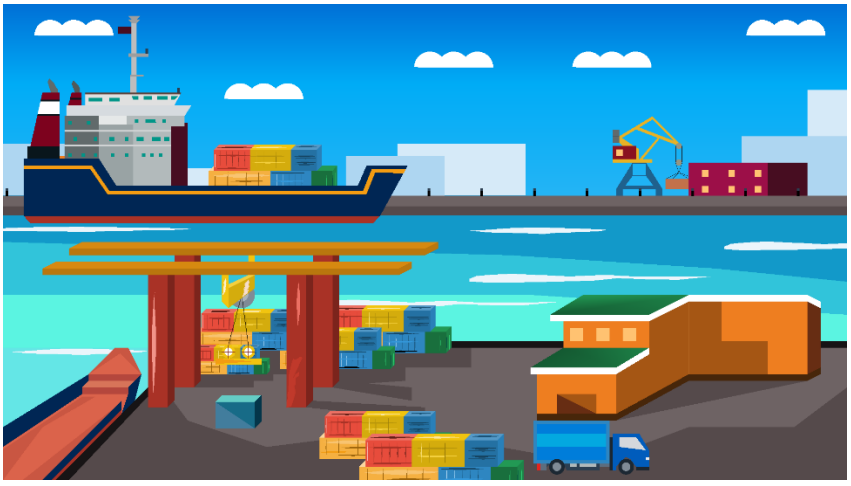

Fig 12. Container Release by Craine (Scenario 1)

Figure 12 continues from the previous figure, showing the crane lowering and releasing the container. The animation reflects the reverse motion of pickup—detaching and gently placing the container in a designated area. This figure confirms that the simulation supports continuous mechanical logic and smooth animation transitions, completing the lifecycle of cargo handling.

Conclusion

This project successfully demonstrates the application of computer graphics techniques to model a 2.5D cargo port. Using OpenGL and C++, we created a dynamic scene with moving objects, weather effects, and environmental transitions. Over 60 distinct animated features highlight the system's capability to manage multiple layers and modular rendering logic.

The simulation currently runs in automatic mode, with no real-time user interaction. While it achieves realism through timed animations and sound effects, it lacks features like camera movement, user controls, and physics simulation.

Future improvements could include:

- Keyboard and mouse control
- Physics-based object movement
- Improved lighting and shadow rendering
- Interactive GUI elements
- Multi-scene camera navigation

Overall, this simulation provides a foundational step into real-time 2.5D modeling and serves as a tool for both educational and visualization purposes in the study of computer graphics.

References

Kilgard, M. J. (1996). *OpenGL programming for the X Window System*. Addison-Wesley.

NeHe. (2000). *NeHe OpenGL tutorials*. http://nehe.gamedev.net

Transport Tycoon Forums. (n.d.). *Transport Tycoon Deluxe and OpenTTD fan community*. https://www.tt-forums.net/

Valve Corporation. (2012). *Port Simulator 2012* [Video game]. UIG Entertainment.