Project Title: Development of a Two-Dimensional Animated Bridge Environment Using OpenGL Graphics Library.

1. Introduction

Computer graphics is an important field of computer science that deals with creating and displaying visual content using computers. It is widely used in areas such as animation, simulation, games and interactive applications. In this project, a two-dimensional animated scene is developed using the OpenGL graphics library to understand the basic concepts of computer graphics.

Scene: The scene shows a large suspension bridge at night built over the sea. An orthographic coordinate system is used to place all objects correctly and maintain proper size and alignment. The environment is divided into several parts including the night sky, sea, bridge structures and road surface. The night atmosphere is created using a dark sky gradient, a crescent moon, randomly placed stars and a moving shooting star effect.

Objects: The bridge is designed with different components such as road lanes, sidewalks, towers, suspension cables, vertical support cables, lamp posts and lane markings. These elements are drawn using basic OpenGL shapes like lines, quadrilaterals, polygons, points and circles. Darker colors are used to represent nighttime while lights on the bridge and vehicles are kept bright to improve visibility and realism.

Animation: To make the scene more dynamic, several animated objects are added. Cars and buses move across the bridge in opposite directions to simulate traffic flow. A large cargo ship moves across the sea below the bridge and gently moves up and down to represent floating on water. The sea waves are animated using a sine function which makes the water motion look more natural. A timer function is used to update the scene regularly and ensure smooth animation.

Overall, this project demonstrates basic computer graphics concepts such as object drawing, coordinate-based positioning, transformations, animation, and scene management. By combining simple drawing methods with basic animation techniques, the project creates a realistic and visually appealing night-time bridge scene and helps improve understanding of OpenGL graphics programming.

2. Working Procedure

Initially a brainstorming process was done to decide the overall idea of the project. Different scene concepts were considered and finally a night-time bridge scene was selected because it allows the use of both static and moving objects and creates a visually attractive environment.

After the idea was finalized, the scene layout was planned to use a graph-based coordinate system. The positions of major elements such as the sky, bridge and sea were fixed at this stage. This planning helped in understanding object placement and maintaining proper balance in the scene before starting the coding part.

Once the design was completed, the implementation phase was started gradually. Background elements like the night sky, moon and stars were drawn first. Then the sea and bridge structure were added according to the planned layout.

After that moving objects such as cars, bus, cargo ship and shooting star were included in the scene. Simple animation was applied to these objects to create motion and improve realism. Vehicles were moved along the bridge, waves were added to the sea and the ship was made to float smoothly.

Finally the project was tested several times and small adjustments were made to improve smoothness and visual appearance. In this way, the project was completed step by step, following a clear process from idea development to final implementation.

3. Project Graph Explanation

The project graph defines the overall layout of the scene and the coordinate system used to place and animate different objects. An orthographic projection is used to create a two-dimensional view of the scene. This projection keeps object sizes constant and prevents perspective distortion, which makes it suitable for a clear and well-organized graphical layout.

The coordinate system uses a fixed window size of 1400 units on the X-axis and 800 units on the Y-axis. The origin (0,0) is located at the bottom-left corner of the screen. The X-axis increases from left to right, and the Y-axis increases from bottom to top. All objects in the scene are positioned using these coordinate values.

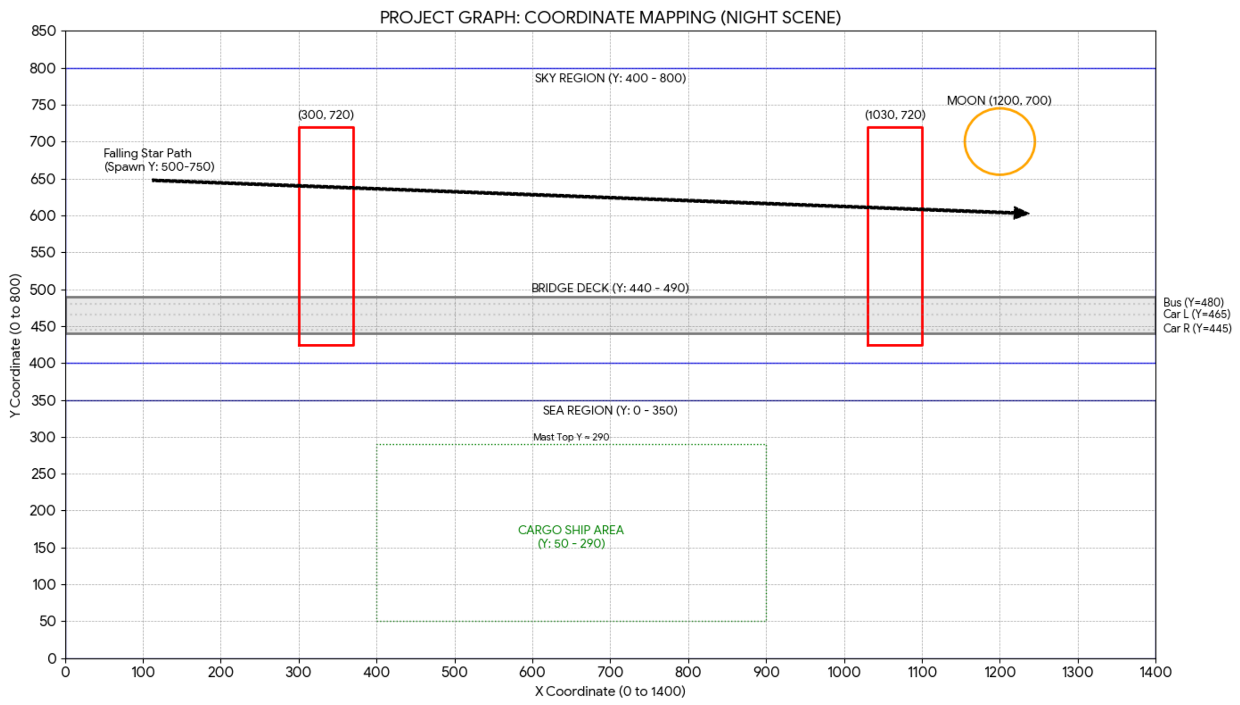


Figure-1: Detailed Coordinates of the Project

The scene is divided into several horizontal sections to maintain proper layering and visual balance. The night sky occupies the upper portion of the graph, approximately from Y = 400 to Y = 800. This region contains the sky gradient, stars, the crescent moon and a moving shooting star. These elements remain in the background and provide a realistic night-time environment.

The bridge structure is positioned in the middle part of the graph mainly between Y = 440 and Y = 490. This region includes the road surface, sidewalks, lane markings, towers, suspension cables, vertical support cables and lamp posts. Fixed Y-coordinates are used to keep the bridge straight and symmetric across the scene.

The sea is placed in the lower section of the graph extending from Y = 0 to Y = 350. Animated wave effects are created in this region using a sine function. A large cargo ship is positioned within the sea area and moves horizontally along the X-axis. The ship also has a small vertical motion to simulate floating on water.

Moving objects such as cars and buses are placed on the bridge road. Their Y-coordinates remain constant so that they stay aligned with the road surface, while their X-coordinates are continuously updated to create smooth motion in opposite directions. This clear separation of horizontal and vertical movement helps prevent overlapping and keeps the animation realistic.

By dividing the scene into clearly defined coordinate regions, the project graph ensures that all objects are properly aligned, well layered and visually balanced. This structured use of the coordinate system simplifies object placement, animation control and overall scene management in the OpenGL environment.

4. List of Objects

The following objects are implemented in the project. Each object is assigned a unique object ID to make the program organized, easy to understand, and simple to expand in the future.

Serial No	Object ID	Object Name
1	OBJ_01	Night Sky
2	OBJ_02	Stars
3	OBJ_03	Shooting Star
4	OBJ_04	Moon
5	OBJ_05	Sea
6	OBJ_06	Waves
7	OBJ_07	Bridge
8	OBJ_08	Bridge Towers
9	OBJ_09	Suspension Cables
10	OBJ_10	Lamp Posts
11	OBJ_11	Cargo Ship
12	OBJ_12	Car (Left Direction)
13	OBJ_13	Car (Right Direction)
14	OBJ_14	Bus
15	OBJ_15	Wheels

The project consists of multiple graphical objects that together form a complete and realistic night-time scene. All objects are created using basic OpenGL primitive shapes such as lines, quadrilaterals, polygons, points and circles. The objects are arranged based on their role in the environment and their position in the coordinate system.

The night sky acts as the background of the scene and occupies the upper portion of the display. A gradient color is used to represent the dark night atmosphere. This object remains static and does not change during animation. The stars are small objects randomly placed in the sky region. They add visual depth and realism to the night scene. Some stars slightly change brightness to create a flickering effect.

The shooting star is a temporary animated object that appears at random intervals. It moves diagonally across the sky with a glowing head and a fading tail. This object adds dynamic visual interest to the background. The moon is drawn as a crescent shape using overlapping circles. It is located in the sky region and remains static throughout the animation.

The sea is positioned in the lower part of the stage and represents the water below the bridge. The sea serves as the base region for water-related animation. The waves are animated elements of the sea. They are generated using a sine function to create continuous wave motion which makes the water surface look natural.

The bridge is the central structural object of the project. It includes the road surface, sidewalks, lane markings and support elements. The bridge remains static and provides a platform for vehicle movement. The bridge towers are vertical structures that support the suspension cables. They improve the realistic appearance of the bridge and help define its shape. The suspension cables connect the bridge towers and the road surface. Curved lines are used to represent real suspension behavior. The lamp posts are placed along the bridge. They include bright light points to simulate street lighting and enhance the night-time environment.

The cargo ship is a large moving object placed on the sea. It moves horizontally across the scene and has a small vertical motion to simulate floating on water. This object increases realism and adds motion to the lower part of the scene.

The cars represent small vehicles traveling on the bridge. Two separate car objects are used to show traffic moving in opposite directions. Their motion demonstrates basic animation and transformation techniques. The bus is a larger vehicle object that moves along the bridge road. Its size and design distinguish it from the cars and add variety to the traffic flow. The wheels are reusable circular objects used by cars and the bus. Using a separate wheel object improves code modularity and avoids repetition.

Overall, the object list includes both static and animated elements. Together these objects create a visually balanced and realistic night-time bridge scene while demonstrating object-based design and animation principles in computer graphics.

5. List of Functions

The following functions are implemented in the project. Each function is associated with one or more objects, which allows modular design, easier management, and reusability.

SL#	Function ID	Function Name	Object ID
1	OBJ_FUNC_01	drawSky()	OBJ_01 (Night Sky)
2	OBJ_FUNC_02	drawStars()	OBJ_02 (Stars)
3	OBJ_FUNC_03	drawFallingStar()	OBJ_03 (Shooting Star)
4	OBJ_FUNC_04	drawMoon()	OBJ_04 (Moon)
5	OBJ_FUNC_05	drawSea()	OBJ_05 (Sea)
6	OBJ_FUNC_06	drawSea()	OBJ_06 (Waves)
7	OBJ_FUNC_07	drawBridge()	OBJ_07 (Bridge)
8	OBJ_FUNC_07	drawBridge()	OBJ_08 (Bridge Towers)
9	OBJ_FUNC_07	drawBridge()	OBJ_09 (Suspension Cables)
10	OBJ_FUNC_07	drawBridge()	OBJ_10 (Lamp Posts)
11	OBJ_FUNC_08	drawCargoShip()	OBJ_11 (Cargo Ship)
12	OBJ_FUNC_09	drawCar()	OBJ_12 (Car – Left Direction)
13	OBJ_FUNC_09	drawCar()	OBJ_13 (Car – Right Direction)
14	OBJ_FUNC_10	drawBus()	OBJ_14 (Bus)
15	OBJ_FUNC_11	drawCircle()	OBJ_15 (Wheels / Circular Shapes)

Each object in the project is represented by a separate function. This modular design improves clarity, allows reuse, and makes it easier to modify or expand the program.

- **drawSky()**: Draws the night sky background using a color gradient. This function creates a static sky object and remains unchanged during animation.
- **drawStars()**: Draws multiple stars as points randomly placed in the upper sky region. Some stars flicker slightly to simulate twinkling.
- **drawFallingStar()**: Draws a shooting star that moves diagonally across the sky. It includes a glowing head and a fading tail, adding dynamic visual effects to the scene.

- **drawMoon()**: Draws a crescent moon using overlapping circles. It is a static object in the sky region.
- **drawSea()**: Draws the sea as a dark water surface using quadrilaterals.
- **drawWaves()**: Creates animated wave patterns on the sea using a sine function. The waves move continuously to make the water look natural.
- **drawBridge()**: Draws the bridge structure, including road surface, sidewalks, towers, suspension cables, vertical hangers and lamp posts. Grouping these components ensures visual consistency and simplifies scene management.
- **drawCargoShip()**: Draws a large ship floating on the sea. It includes the hull, bridge, cargo containers, mast and small vertical motion to simulate floating while moving horizontally.
- **drawCar()**: Draws car objects moving on the bridge. Parameters control position, direction and color, allowing the same function to be reused for multiple cars traveling in different directions.
- **drawBus()**: Draws a larger vehicle moving on the bridge. The bus includes polygonal body shapes, circular wheels and lights distinguishing it from the cars and adding variety to traffic.
- **drawCircle()**: A utility function used to draw circles. It is mainly applied to wheels and other round components. Using this function reduces code repetition and improves reusability.

6. List of Animation Functions:

Here is the list of animation functions that are implemented on the project:

SL#	Animation Function	Animation Function ID
1	update()	ANIM_01
2	glutTimerFunc()	ANIM_02
3	glutPostRedisplay()	ANIM_03
4	glTranslatef()	ANIM_04
5	sin()	ANIM_05

update(): This is the main animation control function. It updates the positions of all moving objects, such as cars, bus, and cargo ship and also updates the wave angle for continuous sea motion. It handles the logic for the shooting star, including spawning, movement, and disappearance.

glutTimerFunc(): This function calls the update() function at fixed time intervals (approximately every 16 milliseconds). This ensures smooth and consistent animation by refreshing the scene around 60 times per second.

glutPostRedisplay(): Requests the display() function to be called again after each update. This ensures the updated positions of all animated objects are rendered on the screen.

glTranslatef(): Moves objects to a new position without altering their shape. It is used to show moving objects like cars, bus and cargo ship.

sin(): Generates smooth oscillating motion for natural effects. It is used to show wave motion on the sea and to create a floating effect for the cargo ship.

7. Contribution

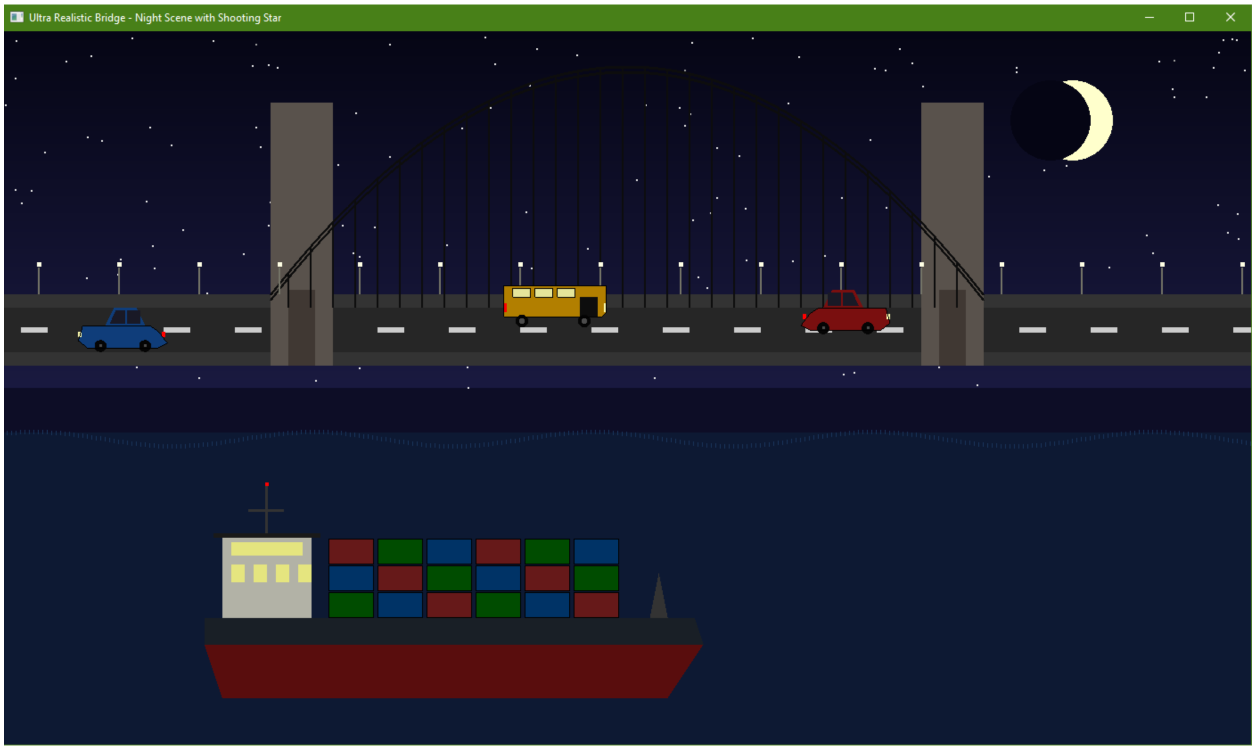
The following table shows the contributions of each project member. The tasks are divided based on the complexity of functions and effort required for implementing graphical objects and animation.

Member Name	Implemented Functions	Implemented Animation Functions	Percentage of Contribution
Md Zahedul Islam Tapu	drawBridge(), drawCargoShip(),drawStars()	update(), glTranslatef() (boat, vehicles, bridge translations), sin() (wave & floating motion), drawFallingStar() logic	40%
Nusrat Jahan Disha	drawSky(), drawSea(),drawMoon()	glutPostRedisplay() (refresh display), partial vehicle movement logic	30%
Sarker Lamia Yeamin	drawCar(), drawBus(), drawCircle()	glutTimerFunc() (timed animation), partial vehicle movement logic	30%

8. Conclusion

This project demonstrates the practical application of computer graphics in creating a realistic night-time bridge scene. Static and dynamic objects including sky, stars, moon, bridge, vehicles and cargo ship are combined to create a visually balanced environment. Modular functions and reusable components allow easy management, modification and expansion. Animation functions such as update(), glutTimerFunc(), and glTranslatef() ensure smooth motion while sine-based oscillations provide natural wave and floating effects. The project provides a strong understanding of 2D graphics programming, animation control and scene management while producing an interactive and visually appealing simulation.

Output Scenario of the Project:



Project Link:
https://drive.google.com/file/d/15QaWWuAdJdpKHK_3xEafxlx2vMN7_AjY/view?usp=sharing