

Introduction

Computer graphics is an important part of computer science and is widely used in visual simulation and interactive systems. In this project, a two-dimensional animated scene is created using the OpenGL graphics library. The main objective of this project is to understand object drawing, coordinate based positioning and basic animation techniques.

Scene: The scene represents a large suspension bridge built over an open sea. The environment is divided into several parts including the sky, bridge structure, road surface and sea. An orthographic coordinate system is used to place each element correctly and maintain proper size and balance. The bridge contains different components such as towers, suspension cables, road lanes, sidewalks, and lamp posts which together give the bridge a realistic appearance.

Object: To make the scene more interesting, several moving objects such as cars, a bus and a boat floating on the water are included. All objects are created using simple OpenGL primitives such as lines, quadrilaterals, polygons and circles. Each object is implemented in a separate function to keep the code organized and easy to understand. Reusable functions are also used for common shapes like wheels and circular parts.

Animation: Animation is an important part of this project. The vehicles move along the bridge in opposite directions to represent traffic flow. The boat moves across the water with a slight up and down motion to simulate floating. A sine function is used to generate wave motion on the water surface, making the animation look more natural. A timer function updates the display at regular intervals to ensure smooth movement.

Overall, this project demonstrates basic computer graphics concepts such as object modeling, transformations, animation and scene management. By combining simple drawing techniques with basic animation, the project creates a realistic and visually appealing bridge scene while improving understanding of OpenGL graphics programming.

Project Graph:

The project graph defines the overall layout of the scene and the coordinate system used to place different objects. An orthographic projection is used to create a two-dimensional view. This type of projection draws all objects without overlapping which makes it suitable for a structured and organized scene.

The coordinate system uses a fixed window size of 1400 units along the horizontal X-axis and 800 units along the vertical Y-axis. The origin is located at the bottom-left corner of the screen. Positive X values extend toward the right side while positive Y values extend upward.

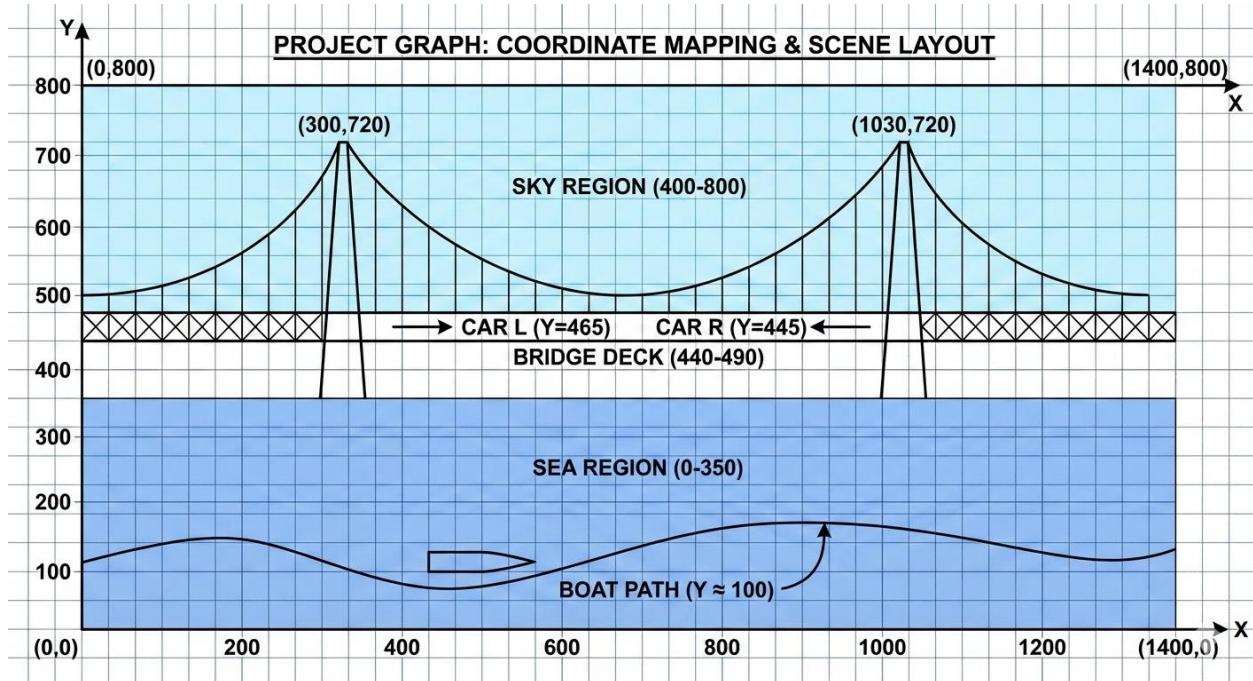


Figure-1: Detailed Coordinates of the Project

The scene is divided into multiple horizontal sections to maintain clarity and proper layering. The sky occupies the upper portion of the graph ranging approximately from $Y = 400$ to $Y = 800$. This area serves as the background of the scene. Below the sky, the bridge structure is positioned between $Y = 440$ and $Y = 490$. The bridge includes the road surface, sidewalks, suspension towers, cables and lamp posts which are aligned using consistent coordinate values to ensure symmetry.

The sea is placed in the lower section of the graph extending from $Y = 0$ to $Y = 350$. This area contains animated wave effects generated using mathematical functions. The boat is positioned within the sea region and moves horizontally across the X-axis while maintaining a small vertical oscillation because of waves.

Moving vehicles such as cars and buses are placed on the bridge road. Their Y-coordinates remain constant to keep them aligned with the road surface, while their X-coordinates are continuously updated to create motion in both directions. This separation of vertical and horizontal movement helps maintain realism and prevent objects overlapping.

By organizing the scene into clearly defined coordinate regions, the project graph ensures that all objects remain properly aligned and visually balanced. This structured graphical layout simplifies object placement, animation control and overall scene management.

List of Objects:

Here is the list of the objects that are implemented on the project:

Serial No	Object ID	Object Name
1	OBJ_01	Sky
2	OBJ_02	Sea
3	OBJ_03	Bridge
4	OBJ_04	Bridge Tower
5	OBJ_05	Suspension Cables
6	OBJ_06	Lamp Posts
7	OBJ_07	Boat
8	OBJ_08	Car (Left Direction)
9	OBJ_09	Car (Right Direction)
10	OBJ_10	Bus
11	OBJ_11	Waves
12	OBJ_12	Wheels

The project consists of multiple graphical objects that together form a complete and realistic scene. Each object is assigned a unique object ID to simplify identification, organization, and future expansion of the program. The objects are designed using OpenGL primitive shapes and are arranged according to their role in the environment.

The sky object serves as the background of the scene. It occupies the upper portion of the display and provides visual depth through color variation. This object does not involve animation and remains static throughout the execution of the program.

The sea object is placed in the lower section of the scene. It represents the water body beneath the bridge and includes animated wave patterns. The sea acts as a base region for floating objects and adds motion to the environment. The waves are animated elements of the sea object. They are generated using mathematical functions and are responsible for creating continuous water motion.

The bridge is the central structural object of the project. It includes the road surface, sidewalks, suspension towers, cables, arches and lamp posts. This object connects both sides of the scene and

provides a platform for vehicle movement. The bridge remains static but plays a key role in positioning animated objects.

The bridge towers are vertical structural components of the bridge. They support the suspension cables and contribute to the realistic appearance of the bridge. These towers are drawn as part of the bridge structure but are treated as separate logical objects for design clarity.

The suspension cables connect the bridge towers and the road surface. They are drawn using curved line segments to represent real suspension behavior. Lamp posts are decorative and functional elements placed along the bridge. They improve visual detail and represent real-world infrastructure commonly found on large bridges.

The boat is a moving object placed on the sea. It travels horizontally across the scene and includes a slight vertical motion to simulate floating. The boat adds dynamic behavior and increases the realism of the water environment.

The cars represent light vehicles traveling on the bridge. Two separate car objects are used to simulate traffic moving in opposite directions. Their continuous motion demonstrates basic object 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 it contributes to the variety of traffic in the scene. The wheels are reusable circular components used by vehicle objects. A separate wheel object improves modularity and avoids repeated code for circular shapes.

Overall, the list of objects represents a combination of static and dynamic elements. A structured and visually balanced scene is created by using these objects that demonstrate object-based design and animation principles in computer graphics.

List of Functions:

Here are the functions that are used associated with the objects:

SL#	Function ID	Function Name	Object ID
1	OBJ_FUNC_01	drawSky()	OBJ_01 (Sky)
2	OBJ_FUNC_02	drawSea()	OBJ_02 (Sea)
3	OBJ_FUNC_02	drawSea()	OBJ_11 (Waves)
4	OBJ_FUNC_03	drawBridge()	OBJ_03 (Bridge)
5	OBJ_FUNC_03	drawBridge()	OBJ_04 (Bridge Tower)
6	OBJ_FUNC_03	drawBridge()	OBJ_05 (Suspension Cables)
7	OBJ_FUNC_03	drawBridge()	OBJ_06 (Lamp Posts)
8	OBJ_FUNC_05	boat()	OBJ_07 (Boat)
9	OBJ_FUNC_06	drawCar()	OBJ_08 (Car – Left Direction)
10	OBJ_FUNC_06	drawCar()	OBJ_09 (Car – Right Direction)
11	OBJ_FUNC_07	drawBus()	OBJ_10 (Bus)
12	OBJ_FUNC_04	drawCircle()	OBJ_12 (Wheels / Circular Shapes)

Each object in the project is represented by specific function to maintain modularity and clarity in the program structure. This functional separation allows individual objects to be drawn, modified, and reused efficiently.

The drawSky() function is responsible for rendering the sky object. It draws a rectangular background using color gradients to create a realistic sky appearance. This function represents a static object and remains unchanged during animation.

The drawSea() function represents the sea object. It draws the water surface using quadrilateral shapes and includes animated wave effects. The wave motion is generated using a sine function, which is updated continuously to create a natural water movement.

The drawBridge() function is used to draw the main bridge structure. This function includes multiple sub-components such as the road surface, sidewalks, towers, suspension cables and lamp posts. Grouping these elements within a single function ensures structural consistency and simplifies scene management.

The boat() function represents the boat object. It draws the boat body, cabin, containers and mast. The function also applies translation and sinusoidal vertical movement to simulate floating on water while moving horizontally across the sea.

The drawCar() function is used to represent car objects moving in both directions on the bridge. Parameters are passed to the function to control position, direction and color. This approach allows the same function to be reused for multiple car objects with different behaviors.

The drawBus() function represents the bus object. It draws a larger vehicle using polygonal shapes and circular wheels. The bus moves horizontally along the bridge road simulating public transport within the scene.

The drawCircle() function is a reusable utility function used to draw circular shapes. It is mainly applied to vehicle wheels and other round components. This function improves code reusability and reduces repetition.

Overall, the use of separate functions for each object enhances program organization and readability. It also allows easy modification of individual objects without affecting the entire scene which is an important principle in computer graphics programming.

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

The update() function is the primary animation control function. It updates the position variables of moving objects such as the boat, cars, and bus and modifies the wave angle for continuous motion.

The glutTimerFunc() function is used to call the update function at fixed time intervals. This ensures smooth and consistent animation by refreshing the scene approximately sixty times per second.

The `glutPostRedisplay()` function requests the display function to be executed again after each update. This allows the updated positions of animated objects to be rendered on the screen.

The `glTranslatef()` function is used to move objects within the scene without changing their original shape. It is applied to animated objects such as vehicles and boat to control their position dynamically.

The `sin()` function is used to generate wave motion and floating effects. It provides smooth oscillation for sea waves and vertical boat movement, creating a natural animation effect.

Contribution:

The following table presents the individual contributions of each project member. The work has been divided based on the complexity of the tasks and the effort required to implement both the graphical objects and animation functions.

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

Conclusion:

The project successfully implements a realistic bridge scene with moving vehicles and a floating boat using OpenGL. Various objects including the sky, sea, bridge, vehicles and waves were designed and animated through dedicated functions integrating both static and dynamic elements. The work demonstrates the effective application of coordinate systems, object-oriented design and animation techniques in computer graphics. The resulting simulation combines multiple graphical components to create a visually clean and interactive scene highlighting the importance of structured programming and efficient scene management.

Screenshot of the Project's Output:

