A bascule bridge scenary using FreeGLUT

A Project Submitted in Partial Fulfillment of the Requirements of Passing for the Course of

CSI 414 - Computer Graphics Sessional

by

MD Fardeen Ehsan Shawon
CSE 068 07941

Department of Computer Science and Engineering
STAMFORD UNIVERSITY BANGLADESH

July 2022

Abstract

Though GLUT with C++ codes aren't the best way of creating graphical applications now, but it was used widely earlier before the 21st century. As in 2022, GLUT is obsolete. GLUT's latest version was released in 1998 before it was abandoned. We'll be using FreeGLUT is this project, a free-software/open-source alternative to the OpenGL Utility Toolkit (GLUT) library. [1]

Acknowledgments

I want to express my gratitude to Meenakshi M. Raikar, whose book[2] was my initial source to learn OpenGL. My project idea was also taken from this book.

I would also want to thank the Networking pro[3] Youtube channel, whose content helped me a lot in the learning process of OpenGL.

Table of Contents

List of Figures

1 Introduction

This project contains a bascule bridge, a ship, cars, an ocean, and a sun/moon.

1.1 Bascule Bridge

A bascule bridge (also referred to as a drawbridge or a lifting bridge) is a moveable bridge with a counterweight that continuously balances a span, or leaf, throughout its upward swing to provide clearance for boat traffic. It may be single- or double-leafed. [4]

The name comes from the French term for balance scale, which employs the same principle. Bascule bridges are the most common type of movable span because they open quickly and require relatively little energy to operate while providing the possibility for unlimited vertical clearance for marine traffic.

1.2 Why choose this?

Initially, I opted for a more straightforward project containing some elements with basic shapes and movements. Later I chose bascule bridge, which seemed an innovative and uncommon topic as there was less chance of anyone else choosing this particular topic.

2 Project Aim

The aim is to create a desktop app that makes use of basic shapes and most of the basic libraries of FreeGLUT for C++. I've chosen a scenery that contains a bascule bridge, a ship, cars, an ocean, and a sun/moon.

3 Description

3.1 Background

The background consists of four elements, which are the sea, water, the sky and the sun/-moon.

3.1.1 The Sea

The Sea is drawn with a rectangle (GL_QUADS). The size and shape of the sea is always constant, but it changes color according to day/night mode.

3.1.2 Water

Water is drawn with some lines above the sea rectangle. The lines are drawn with a for loop. The water moves from left to right using the translate property of freeglut.

3.1.3 The Sky

The sky is also drawn with a rectangle (GL_QUADS). The size and shape of the sky is always constant, but it changes color according to day/night mode.

3.1.4 The Sun/Moon

The sun is drawn using the circle drawing algorithm. The algorithm is mentioned below. The color of the sun changes and turns into moon and vice versa according to day/might mode.

3.1.4.1 Algorithm for circle

```
glBegin(GL_TRIANGLE_FAN);
glColor3ub(0,0,0);
int x = 0, y = 0, triangleAmount = 40;
glVertex2f(x,y);
for(j=0;j<=triangleAmount;j++){
    glVertex2f(
        x+(radius*cos(j*twicePi/triangleAmount)),
        y+(radius*sin(j*twicePi/triangleAmount))
    );
}
glEnd();
```

The above code snippet represents a circle drawn using freeglut functions. This algorithm uses multiple triangles to represent a circle. Here x, and y denotes the coordinate of the center of the circle, and triangleAmount denotes how many triangles are to be drawn to complete the circle.

3.2 The Bridge

The bridge is drawn using multiple rectangles (GL_QUADS). The bridge mainly consists of 4 sections. The leftmost and rightmost sections are static all of the time. Both sections in the middle are dynamic, they move dynamically when a ship approaches or passes. When a ship approaches to pass the bridge, the sections move upwards to imitate the opening of a bascule bridge. When the ship passes through the bridge, the middle sections come downwards and be the regular bridge and allows the cars to passes through.

3.2.1 Traffic Lights

Both lanes of the bridge contain traffic lights to signal the cars to move or stop. When the bridge is opening, the lights are red, when the bridge is nearly closed, they turn into yellow, and upon closing the bridge completely, the lights turn green.

3.3 The Ship

The ship is drawn with multiple elements including QUADS and POLYGONS. The ship approaches to the bridge with linear speed. When it reaches the bridge, it stops and allows

the bridge some time to open. When the bridge opens, it passes through. Upon passing through it goes further and shrinks to imitate the shrinking of an object while going further in real-life scenarios. The ship vanishes upon reaching a point and starts from the bottom again to act like another ship coming through.

3.4 The cars

The cars are drawn with multiple objects including QUADS, POLYGON, and circles. the cars move across the bridge while the traffic light is green. When the lights turn red, the cars stop moving. and when the lights turn yellow, the cars start moving again.

4 Methodology

4.1 Tools

Let's describe the tools I've used to complete this project.

4.1.1 Operating System

I've used PopOS!, a Linux distro in this project. There's no particular reason to choose this OS. I've used this OS just because it's my daily driver and my primary OS for development purposes.

It's worth mentioning that, some compatibility issues can occur while running the project on Windows OS because of the libraries used. The system should be prepared with the correct libraries to run this program in windows.

4.1.2 IDE

I've used CodeBlocks IDE as it's the best fit for both C/C++ and OpenGL development.

4.1.3 Library

I've used FreeGLUT [1], a free-software/open-source alternative to the OpenGL Utility Toolkit (GLUT) [5] library to complete this project.

4.2 GLUT

GLUT (pronounced like the glut in gluttony) is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing application programming interface (API) for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL programming. GLUT provides a portable

API so you can write a single OpenGL program that works on both Win32 PCs and X11 workstations. [5]

GLUT is designed for constructing small to medium sized OpenGL programs. While GLUT is well-suited to learning OpenGL and developing simple OpenGL applications, GLUT is not a full-featured toolkit so large applications requiring sophisticated user interfaces are better off using native window system toolkits like Motif. GLUT is simple, easy, and small. My intent is to keep GLUT that way.

4.3 FreeGLUT

4.3.1 What?

Freeglut [1] is a free-software/open-source alternative to the OpenGL Utility Toolkit (GLUT) library. GLUT was originally written by Mark Kilgard to support the sample programs in the second edition OpenGL 'RedBook'. Since then, GLUT has been used in a wide variety of practical applications because it is simple, widely available and highly portable.

GLUT (and hence freeglut) takes care of all the system-specific chores required for creating windows, initializing OpenGL contexts, and handling input events, to allow for trully portable OpenGL programs.

freeglut is released under the X-Consortium license.

4.3.2 Why?

The original GLUT library seems to have been abandoned with the most recent version (3.7) dating back to August 1998. Its license does not allow anyone to distribute modified library code. This is really unfortunate, since GLUT is getting old and really needs improvement. Also, GLUT's license is incompatible with some software distributions (e.g., XFree86).

4.3.3 Who?

freeglut was originally written by Pawel W. Olszta with contributions from Andreas Umbach and Steve Baker.

John F. Fay, John Tsiombikas, and Diederick C. Niehorster are the current maintainers of the freeglut project.

4.3.4 When?

Pawel started freeglut development on December 1st, 1999. The project is now virtually a 100

freeglut adds some additional features over the basic GLUT functionality, such as a larger set of predefined objects to use, the ability to run single iterations of the event loop, or exit from it gracefully, mousewheel input callbacks, optional OpenGL core/compatibility profile context creation, multitouch/multi-pointer input, and support for a larger and growing set of platforms, being just some of them.

5 Results

5.1 Source Code

The project Source Code can be found in *the GitHub repository* located at `https://github.com/fardeen-academic/CSI414-CGLab-FinalProject`

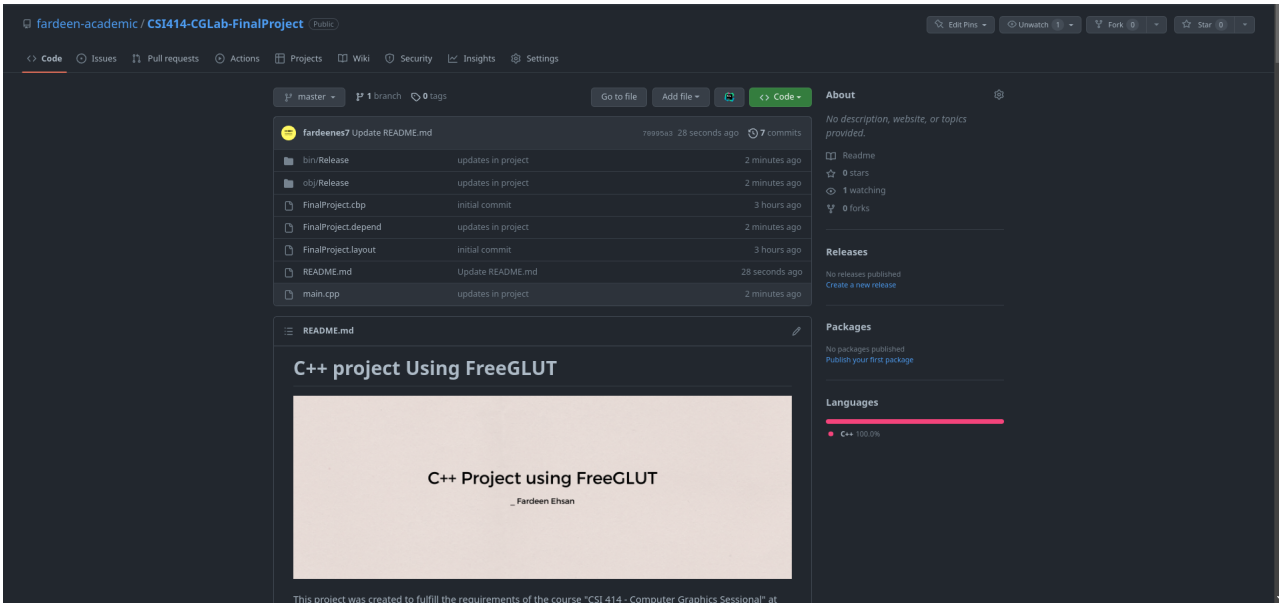

Figure 5.1: Screenshot Github Repository

5.2 Screenshots

I'm presenting some screenshots of the project output for better understanding of the outcome.

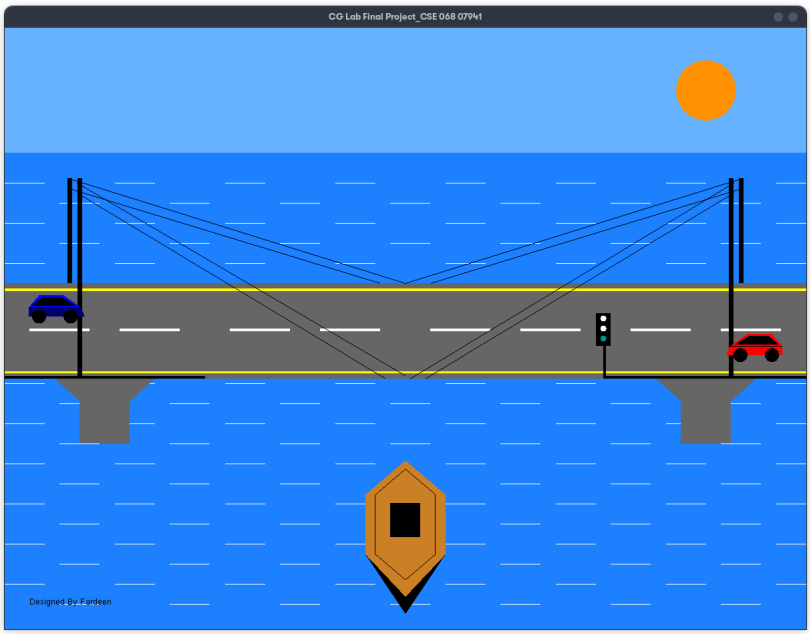

Figure 5.2: The Ship approaching the bridge

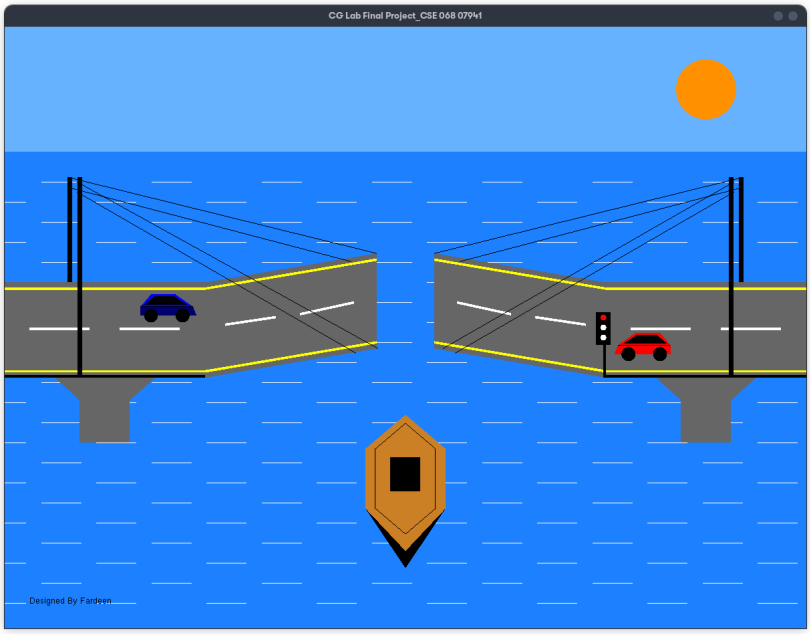

Figure 5.3: The Bridge is opening

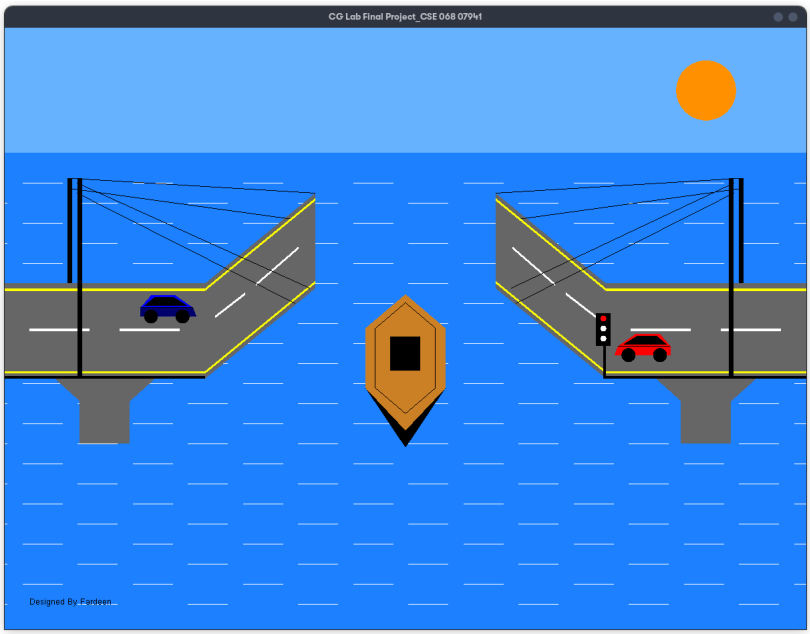

Figure 5.4: The bridge is opened and the ship is passing through

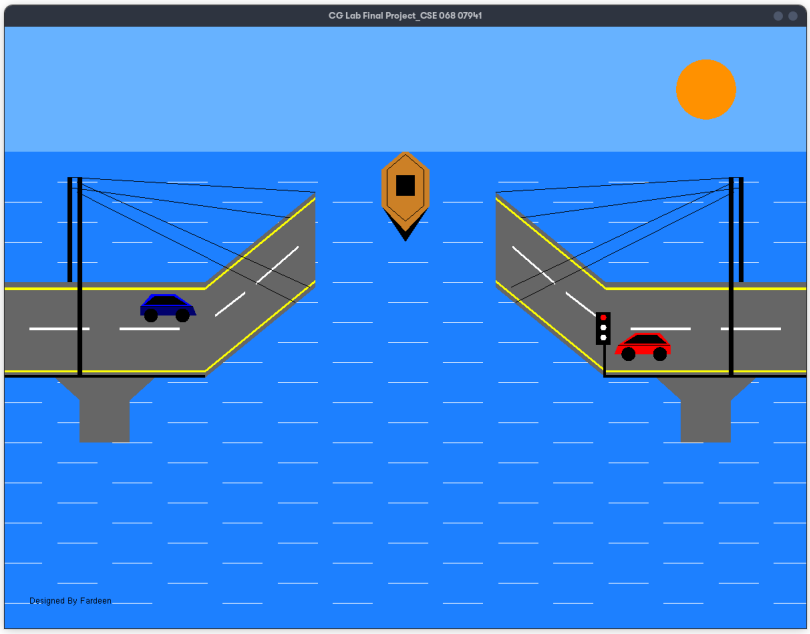

Figure 5.5: The ship has passed through and gone further

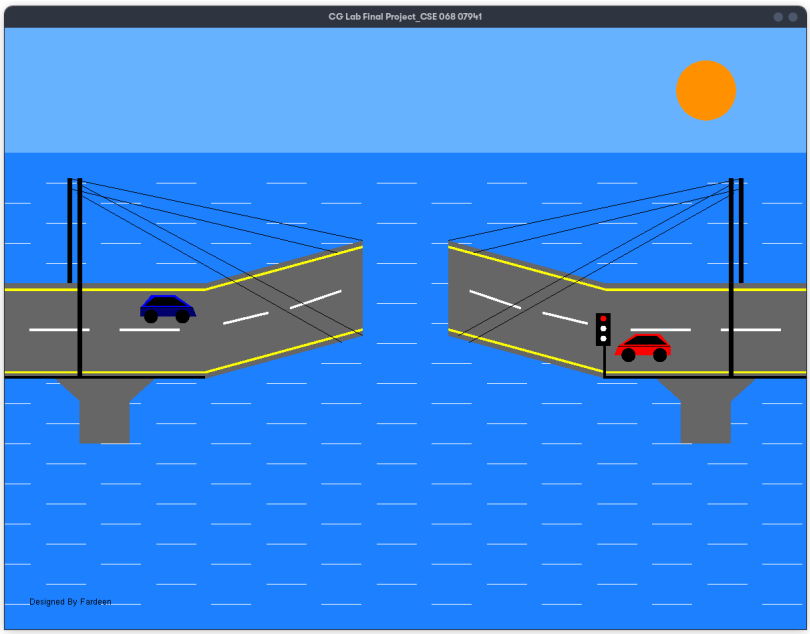

Figure 5.6: The bridge is closing

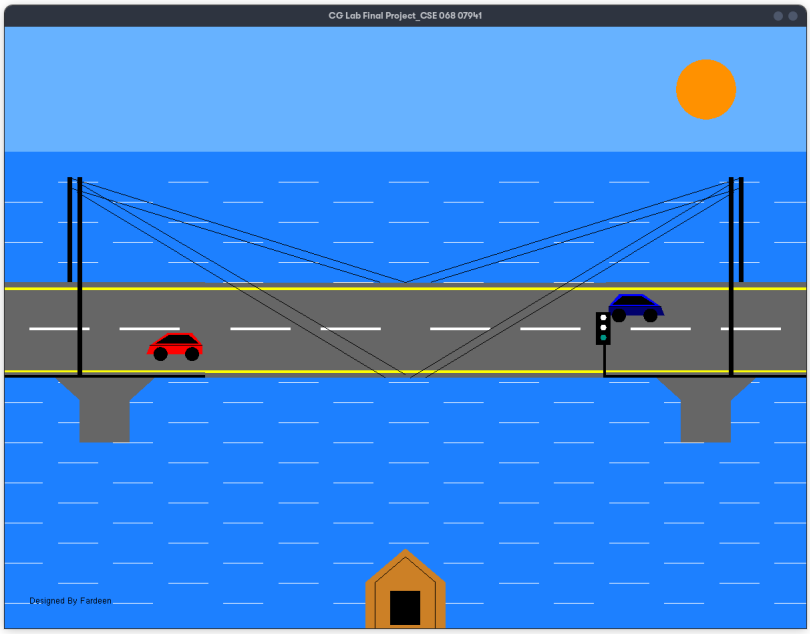

Figure 5.7: Another ship is now coming

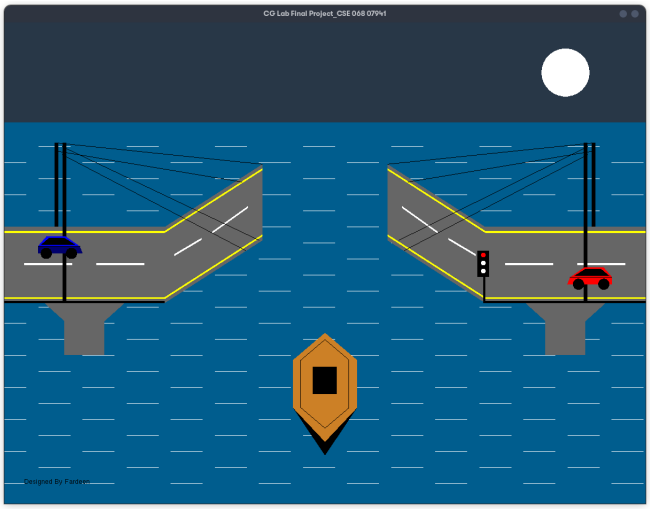

Figure 5.8: Night Mode

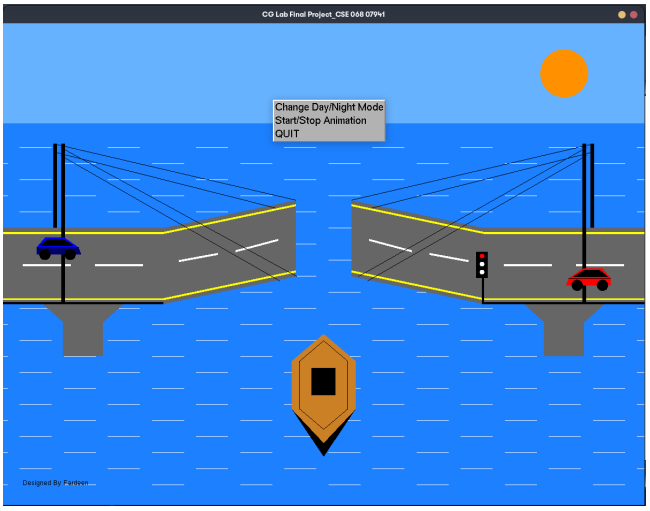

Figure 5.9: Menu to interact with the day/night mode and animation

A GIF of the project can be found on the GitHub repository mentioned in Source Code Section.

6 Future Works

This project can be enhanced in many ways, including but not limited to sunset and animation, more cars, and dynamic car movement (while the bridge opens, if a car is halfway to the traffic light, the car won't stop. And if a car is in the middle of the bridge, the bridge won't open).

But the change I'm interested to do in this project in the future is adding 3D support. By 3D support, I mean all of the elements in 3D. it would be fascinating to see the scenery in 3D.

7 Conclusion

This project was an enjoyable experience. There were some hardships and limitations in doing the project, but it was fun.

7.1 Limitations

There wasn't any mentionable limitation in this project, but It would be great to have a better way of drawing the elements instead of putting the pixels manually. Nonetheless, the experience was great.

References

[1] "The freeglut Project :: About — freeglut.sourceforge.net," https://freeglut.sourceforge.net/, [Accessed 23-Nov-2022].

[2] M. M. Raikar, *Computer Graphics Using Opengl*. Elsevier India, 2013.

[3] "Networking pro — networkingpro3049," https://www.youtube.com/@networkingpro3049, [Accessed 23-Nov-2022].

[4] "Bascule bridge - Wikipedia — en.wikipedia.org," https://en.wikipedia.org/wiki/Bascule_bridge, [Accessed 23-Nov-2022].

[5] T. K. G. Inc, "GLUT - The OpenGL Utility Toolkit — opengl.org," https://www.opengl.org/resources/libraries/glut/glut_downloads.php, [Accessed 23-Nov-2022].