VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgaum-590018.

COMPUTER GRAPHICS AND VISUALIZATION
(18CSL67)

MINI PROJECT REPORT

on

""VERTICAL LIFT BRIDGE SIMULATION "

Submitted in partial fulfillment for the requirements of the VI Semester degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

For The Academic Year
2020-21

By

ANSU SINGH
(1DB18CS015)

Under The Guidance Of
Mrs. Rohini B.R
ASSOCIATE PROFESSOR,
Dept. of CSE

Department of Computer Science and Engineering

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, MysoreRoad, Bengaluru - 560 074.

ACKNOWLEDGEMENT

I am here by submitting Computer Graphics And Visualization Mini Project entitled **“‘VERTICAL LIFT BRIDGE SIMULATION”**, as per the scheme of Visvesvaraya Technological University, Belgaum. In this connection, I would like to express my deep sense of gratitude to my beloved institution Don Bosco Institute of Engineering and also I would like to express my sincere gratitude and indebtedness to **Dr. Hemadri Naidu T, Principal, DBIT, Bengaluru.**

I would like to express my sincere gratitude to **Prof. Umashankar B S, Professor and Head of Dept. of Computer Science and Engineering, DBIT** for providing a congenial environment to work in and carryout my mini project.

I would like to express the deepest sense of gratitude to thank my Project Guide Mrs. Rohini B.R, Associate professor, **Dept. of Computer Science and Engineering, DBIT** for her constant help and support extended towards me during the course of the project.

Finally I am very much thankful to all the teaching and non teaching members of the Department of Computer Science and Engineering, my seniors, friends and my parents for their constant encouragement, support and help throughout completion of report.

ANSU SINGH

(1DB18CS015)

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Bengaluru – 560 074.

DEPARTMENT OF COMPUTER SCINECE AND ENGINEERING

CERTIFICATE

This is to certify that the mini project report entitled **"VERTICAL LIFT BRIDGE SIMULATION"** is a bonafide work carried out by **ANSU SINGH (1DB18CS015)** in partial fulfillment of award of Degree of **Bachelor of Engineering in Computer Science and Engineering** of Visvesvaraya Technological University, Belagavi, during the academic year 2020-2021. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated. The mini project has been approved as it satisfies the academic requirements associated with the degree mentioned.

Signature of guide

……

Rohini B.R
Associate professor
Dept. of CSE,
DBIT, Bengaluru.

Signature of HOD

……

Prof. B S Umashankar
Head of Dept.,
Dept. of CSE,
DBIT, Bengaluru.

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Bengaluru – 560 074.

DECLARATION

I, **ANSU SINGH (1DB18CS015)**, student of sixth semester B.E, Department of Computer Science and Engineering, Don Bosco Institute of Technology, Kumbalagodu, Bangalore, declare that the mini project work entitled **"VERTICAL LIFT BRIDGE SIMULATION"** has been carried out by me and submitted in partial fulfillment of the course requirements for the award of degree in **Bachelor of Engineering** in **Computer Science and Engineering** of **Visvesvaraya Technological University, Belgaum** during the academic year **2020-21**. The matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

Place: Bangalore **ANSU SINGH**
Date: **1DB18CS015**

TABLE OF CONTENTS

CHAPTERS				Pg. No
1.			INTRODUCTION	
2.			2.1 SOFTWARE REQUIREMENTS 2.2 HARDWARE REQUIREMENTS	
3.			OVERVIEW 3.1 Computer Graphics. 3.2 OpenGL Overview	
4.			ANALYSIS	
5.			IMPLEMENTATION	
6.			TESTING	
7.			SNAPSHOTS	
8.			CONCLUSION	
9.			APPENDIX	
10.			BIOGRAPHY	

ABSTRACT

The project "VERTICAL LIFT BRIDGE SIMULATION" is used to demonstrate the use of OpenGL functions that is defined in OpenGL. Our objective is to develop a simple project to show the simulation of vertical lift bridge.

Here the user must enter the necessary parameters in order to perform the simulation of vertical lift bridge. The OpenGL can be used for interaction with the hardware. Then the result will be displayed on the screen.

This project we are working is under linux platform and are closed using C programming language with underlying tool-OpenGL which gives rich and highly usable 3D graphics.

The same program can be run on different computer and the graphics will be the same on the two machines. We make use of the "GL/glut" to implement the project. "GL/glut" is the library that gives robust framework to create good graphical effects

1.INTRODUCTION

Computer graphics are graphics created using computers and more generally, the representation and manipulation of image data by a computer.

The development of computer graphics has made computers easier to interact with, and better for understanding and interpreting many types of data. Developments in computer graphics have a profound impact on many types of media and have revolutionized animation, movies and the video game industry.

2.1 Software Requirements

1.Operating System : Windows 10
2.Compiler Used : DEV-C++
3.Language Used : C++ language
4.Editor : DEV-C++

2.2 Hardware Requirements

1.Main processor : INTEL CORE i5
2.Processor Speed : 800 MHz
3.RAM Size : 8GB DDR4
4.Keyboard : Standard qwerty serial or PS/2 keyboard
5.Mouse : Standard serial or PS/2 mouse
6.Compatibility : AT/T Compatible
7.Cache memory : 256 KB
8.Diskette drive A : 1.44 MB, 3.5 inches

3. OVERVIEW

3.1 Computer Graphics.

Computer Graphics is concerned with all aspects of producing pictures or images. The term computer graphics includes almost everything on computers that is not text or sound. The term Computer Graphics has several meanings:

- the representation and manipulation of pictorial data by a computer
- the various technologies used to create and manipulate such pictorial data
- the images so produced, and
- the sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content, see study of computer graphics

Today computers and computer-generated images touch many aspects of our daily life. Computer imagery is found on television, in newspapers, in weather reports, and during surgical procedures. A well-constructed graph can present complex statistics in a form that is easier to understand and interpret. Such graphs are used to illustrate papers, reports, theses, and other presentation material. A range of tools and facilities are available to enable users to visualize their data, and computer graphics are used in many disciplines.

3.2 OpenGL Overview

OpenGL (OpenGLGraphics Library) is a cross-language, multiplatform API for rendering 2D and 3D computer graphics. The API is typically used to interact with a GPU, to achieve hardware-accelerated rendering. OpenGL was developed by Silicon Graphics Inc (SGI) from 1991 and released in January 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, flight simulation, and videogames. OpenGL is managed by the non-profit technology consortium Khronos Group.

While OpenGL provides comprehensive, efficient, and portable support for 3D output, it does not support other features typical of real-life 3D applications. These include:

- graphical input, e.g. mouse, spaceball
- window operations, e.g. create, re-size
- menu system

The GLUT system, developed by Mark Kilgard and enhanced for Windows by Nate Robins, addresses some of these requirements, but the provided menu system is rather rudimentary.

The basic implementation strategy is to enable a GLUT process to launch an independent Tk script. Thus, the built-in event loops of these two systems can operate as usual and the resulting programming style (registering call-backs for given events) is unchanged.The independent processes then send messages back and forth. To the GLUT process, the Tcl process appears to be just another input device (like the mouse and keyboard), and the API is written to conform to this model.

4. ANALYSIS AND DESIGN

ANALYSIS:

Swing, bascule, and vertical lift movable bridge options were considered. For this site, a vertical lift bridge similar to the existing bridge with a machinery platform supported at height was ideal, as it was evident that the best durability would be achieved by removing vulnerable elements of the bridge as far from grade level and water level as possible. A swing bridge option was not carried forward as a viable option as the pier for the swing span would be located in the middle of the channel thus reducing its effective width. A bascule bridge was considered unfavourable at this site as it requires a counterweight system for the moving parts to be slung below the stationary side spans which in turn requires a higher clearance above water and higher approaches.

Design:

Design had to consider transportation, erection, and long-term durability. This was achieved by designing the tower members with sealed and welded tubular pipe members. Flange bolted connections were designed to allow the Contractor to fabricate the towers in manageable segments to be transported and handled on site without the need for field welding. This option allowed the Contractor to fabricate each of the towers in ten segments, assemble them on site, and erect them using a crane from a temporary work trestle. Bolted connections minimized the erection duration of the towers significantly, to a few days in comparison to field welding which

5. IMPLEMENTATION

```
include<GL/glut.h>
#include<string.h>
int i,flag=0,flag2=0,flagb=1,flags=0,flagt=0,flagp=0,flagw=1,flagx=0;
float a=0.0f , b=0.0f , c=0.0f , m=0.0f , n=0.0f, o=0.0f , p=0.0f , q=0.0f , r=0.0f , x=0.0f ,
y=0.0f , z=0.0f , a1=0.0 , a2=0.0 , a3=0.0,j;
void *currentfont;
void setFont(void *font){
currentfont=font;
}
void drawstring(char string[],float x1,float y1,float z1){
   int i , j;
j=strlen(string);
glRasterPos3f(x1,y1,z1);
    for (i=0;i<j;i++){
glutBitmapCharacter(currentfont, string[i]);
}
}
void screen1(){
   glClearColor(0.5,0.2,0.8,0.0);
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   setFont(GLUT_BITMAP_TIMES_ROMAN_24);
   glColor3f(1.0,1.0,1.0);
   char str1[]="DON BOSCO  INSTITUTE  OF  TECHNOLOGY";
   drawstring(str1,-0.5,0.9,0.0);
   glColor3f(0.9,0.9,0.9);
   char str2[]="DEPARTMENT  OF  COMPUTER  SCIENCE  AND  ENGINEERING";
   drawstring(str2,-0.7,0.8,0.0);
   glColor3f(0.8,0.8,0.8);
   char str3[]="A  MINI  PROJECT  ON";
   drawstring(str3,-0.3,0.7,0.0);
   glColor3f(1.0,0.7,0.7);
   char str4[]="-*-  VERTICAL  LIFT  BRIDGE  SIMULATION  -*-";
   drawstring(str4,-0.65,0.6,0.0);
   glColor3f(0.2,0.8,0.5);
   char str5[]="BY  :";
   drawstring(str5,-0.5,0.5,0.0);
```

```
    glColor3f(1.0,1.0,0.0);
    char str6[]="ANSU SINGH     (1DB18CS015)";
    drawstring(str6,-0.5,0.4,0.0);
     glColor3f(0.2,0.8,0.5);
    char str7[]="GUIDES  :";
    drawstring(str7,-0.5,0.1,0.0);
    glColor3f(1.0,1.0,0.0);
    char str8[]="Mr . B.S Umashankar  (Professor and Head, Dept of CSE)";
    drawstring(str8,-0.5,0.0,0.0);
    char str9[]="Mr . Rohini B.R (Asst.prof. Dept of CSE)";
    drawstring(str9,-0.5,-0.1,0.0);
    glColor3f(1.0,0.9,1.0);
    char str10[]="INSTRUCTIONS ";
    drawstring(str10,-0.3,-0.5,0.0);
    char str11[]="----------------";
    drawstring(str11,-0.35,-0.55,0.0);
    char str12[]=" * * *   PRESS     -ENTER -     TO  START  AND     -ESC-     TO EXIT   *
* *";
    drawstring(str12,-0.9,-0.7,0.0);
    char str13[]="PRESS     -S-     TO  START  ANIMATION   AND  PRESS    -T-     TO
STOP  ANIMATION";
    drawstring(str13,-1.05,-0.85,0.0);
    glFlush();
}
void screen3(){
    glClearColor(1.0,0.5,0.0,0.0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    setFont(GLUT_BITMAP_TIMES_ROMAN_24);
    glColor3f(0.0,0.0,0.0);
    char str1[]=" * * *  THANK YOU  * * * ";
    drawstring(str1,-0.2,0.2,0.0);
    char str2[]=" ------------------------- ";
    drawstring(str2,-0.3,0.1,0.0);
    glFlush();
}
void water(){
    glBegin(GL_QUADS);
    glColor3f(0.0,0.4,0.7);
    glVertex3f(-5.0,-0.415,5.0);
    glVertex3f(5.0,-0.415,5.0);
    glVertex3f(5.0,-0.415,-5.0);
    glVertex3f(-5.0,-0.415,-5.0);
    glEnd();
}
void lines(){
    float t1,t2;
    glBegin(GL_LINES);
    for(t1=0.0;t1<=10.0;t1+=0.4){
```

```
     for(t2=0.0;t2<=10.0;t2+=0.4){
       glColor3f(0.7,0.7,0.7);
       glVertex3f(-5.0+t2,-0.41,-4.5+t1);
       glVertex3f(-4.95+t2,-0.41,-4.5+t1);
     }
    }
    glEnd();
}
void base()
{
    float i;
    for(i=0.0;i<1.0;i+=0.8)  {
    glBegin(GL_QUADS);
    glColor3f(0.1,0.1,0.1);  //front
    glVertex3f(-0.5+i,-0.4,0.2);
    glVertex3f(-0.3+i,-0.4,0.2);
    glVertex3f(-0.3+i,-0.25,0.2);
    glVertex3f(-0.5+i,-0.25,0.2);
    glColor3f(0.3,0.3,0.3);  //left
    glVertex3f(-0.5+i,-0.4,0.2);
    glVertex3f(-0.5+i,-0.4,-0.2);
    glVertex3f(-0.5+i,-0.25,-0.2);
    glVertex3f(-0.5+i,-0.25,0.2);
     glColor3f(0.1,0.1,0.1);  //back
    glVertex3f(-0.5+i,-0.4,-0.2);
    glVertex3f(-0.3+i,-0.4,-0.2);
    glVertex3f(-0.3+i,-0.25,-0.2);
    glVertex3f(-0.5+i,-0.25,-0.2);
    glColor3f(0.3,0.3,0.3);  //right
    glVertex3f(-0.3+i,-0.4,0.2);
    glVertex3f(-0.3+i,-0.4,-0.2);
    glVertex3f(-0.3+i,-0.25,-0.2);
    glVertex3f(-0.3+i,-0.25,0.2);
    glColor3f(0.5,0.5,0.5);  //top
    glVertex3f(-0.5+i,-0.25,0.2);
    glVertex3f(-0.3+i,-0.25,0.2);
    glVertex3f(-0.3+i,-0.25,-0.2);
    glVertex3f(-0.5+i,-0.25,-0.2);
    glEnd();
    }
}
void earth(){
    float i;
    for(i=0.0;i<4.0;i+=3.5){
    glBegin(GL_QUADS);
    glColor3f(0.0,0.6,0.0);
    glVertex3f(-3.0+i,-0.4,0.17);  //front
    glVertex3f(-0.5+i,-0.4,0.17);
```

```
    glVertex3f(-0.5+i,-0.25,0.17);
    glVertex3f(-3.0+i,-0.25,0.17);
    glVertex3f(-3.0+i,-0.4,-0.17); //back
    glVertex3f(-0.5+i,-0.4,-0.17);
    glVertex3f(-0.5+i,-0.25,-0.17);
    glVertex3f(-3.0+i,-0.25,-0.17);
    glColor3f(0.0,0.5,0.0);   //top
    glVertex3f(-3.0+i,-0.25,0.17);
    glVertex3f(-0.5+i,-0.25,0.17);
    glVertex3f(-0.5+i,-0.25,-0.17);
    glVertex3f(-3.0+i,-0.25,-0.17);
    glEnd();
    }
}
void pillars(){
    glBegin(GL_QUADS);
    glColor3f(0.1,0.1,0.5);  //front
    glVertex3f(-0.35,-0.25,0.2);
    glVertex3f(-0.31,-0.25,0.2);
    glVertex3f(-0.31,0.2,0.2);
    glVertex3f(-0.35,0.2,0.2);
    glColor3f(0.1,0.1,0.3);   //left
    glVertex3f(-0.35,-0.25,0.2);
    glVertex3f(-0.35,-0.25,0.15);
    glVertex3f(-0.35,0.2,0.15);
    glVertex3f(-0.35,0.2,0.2);
    glColor3f(0.1,0.1,0.5);  //back
    glVertex3f(-0.35,-0.25,0.15);
    glVertex3f(-0.31,-0.25,0.15);
    glVertex3f(-0.31,0.2,0.15);
    glVertex3f(-0.35,0.2,0.15);
    glColor3f(0.1,0.1,0.3);  //right
    glVertex3f(-0.31,-0.25,0.2);
    glVertex3f(-0.31,-0.25,0.15);
    glVertex3f(-0.31,0.2,0.15);
    glVertex3f(-0.31,0.2,0.2);
    glColor3f(0.1,0.1,0.5);  //front
    glVertex3f(-0.35,-0.25,-0.2);
    glVertex3f(-0.31,-0.25,-0.2);
    glVertex3f(-0.31,0.2,-0.2);
    glVertex3f(-0.35,0.2,-0.2);
    glColor3f(0.1,0.1,0.3);   //left
    glVertex3f(-0.35,-0.25,-0.2);
    glVertex3f(-0.35,-0.25,-0.15);
    glVertex3f(-0.35,0.2,-0.15);
    glVertex3f(-0.35,0.2,-0.2);
    glColor3f(0.1,0.1,0.5);  //back
    glVertex3f(-0.35,-0.25,-0.15);
```

```
glVertex3f(-0.31,-0.25,-0.15);
glVertex3f(-0.31,0.2,-0.15);
glVertex3f(-0.35,0.2,-0.15);
glColor3f(0.1,0.1,0.3);  //right
glVertex3f(-0.31,-0.25,-0.2);
glVertex3f(-0.31,-0.25,-0.15);
glVertex3f(-0.31,0.2,-0.15);
glVertex3f(-0.31,0.2,-0.2);
glColor3f(0.2,0.2,0.5);//top
glVertex3f(-0.35,0.2,0.3);
glVertex3f(-0.31,0.2,0.3);
glVertex3f(-0.31,0.25,0.3);
glVertex3f(-0.35,0.25,0.3);
glColor3f(0.3,0.3,0.5);
glVertex3f(-0.35,0.2,0.3);
glVertex3f(-0.35,0.2,-0.3);
glVertex3f(-0.35,0.25,-0.3);
glVertex3f(-0.35,0.25,0.3);
glColor3f(0.2,0.2,0.5);
glVertex3f(-0.35,0.2,-0.3);
glVertex3f(-0.31,0.2,-0.3);
glVertex3f(-0.31,0.25,-0.3);
glVertex3f(-0.35,0.25,-0.3);
glColor3f(0.3,0.3,0.5);
glVertex3f(-0.31,0.2,0.3);
glVertex3f(-0.31,0.2,-0.3);
glVertex3f(-0.31,0.25,-0.3);
glVertex3f(-0.31,0.25,0.3);
glColor3f(0.3,0.3,0.6);
glVertex3f(-0.35,0.25,0.3);
glVertex3f(-0.31,0.25,0.3);
glVertex3f(-0.31,0.25,-0.3);
glVertex3f(-0.35,0.25,-0.3);
//top piller
glColor3f(0.1,0.1,0.5);  //front
glVertex3f(-0.35,0.25,0.2);
glVertex3f(-0.31,0.25,0.2);
glVertex3f(-0.31,0.35,0.2);
glVertex3f(-0.35,0.35,0.2);
glColor3f(0.1,0.1,0.3);   //left
glVertex3f(-0.35,0.25,0.2);
glVertex3f(-0.35,0.25,0.15);
glVertex3f(-0.35,0.35,0.15);
glVertex3f(-0.35,0.35,0.2);
glColor3f(0.1,0.1,0.5);  //back
glVertex3f(-0.35,0.25,0.15);
glVertex3f(-0.31,0.25,0.15);
glVertex3f(-0.31,0.35,0.15);
```

```
glVertex3f(-0.35,0.35,0.15);
glColor3f(0.1,0.1,0.3);  //right
glVertex3f(-0.31,0.25,0.2);
glVertex3f(-0.31,0.25,0.15);
glVertex3f(-0.31,0.35,0.15);
glVertex3f(-0.31,0.35,0.2);
glColor3f(0.1,0.1,0.5);  //front
glVertex3f(-0.35,0.25,-0.2);
glVertex3f(-0.31,0.25,-0.2);
glVertex3f(-0.31,0.35,-0.2);
glVertex3f(-0.35,0.35,-0.2);
glColor3f(0.1,0.1,0.3);   //left
glVertex3f(-0.35,0.25,-0.2);
glVertex3f(-0.35,0.25,-0.15);
glVertex3f(-0.35,0.35,-0.15);
glVertex3f(-0.35,0.35,-0.2);
glColor3f(0.1,0.1,0.5);  //back
glVertex3f(-0.35,0.25,-0.15);
glVertex3f(-0.31,0.25,-0.15);
glVertex3f(-0.31,0.35,-0.15);
glVertex3f(-0.35,0.35,-0.15);
glColor3f(0.1,0.1,0.3);  //right
glVertex3f(-0.31,0.25,-0.2);
glVertex3f(-0.31,0.25,-0.15);
glVertex3f(-0.31,0.35,-0.15);
glVertex3f(-0.31,0.35,-0.2);
glEnd();
//right side
glBegin(GL_QUADS);
glColor3f(0.1,0.1,0.5);  //front
glVertex3f(0.35,-0.25,0.2);
glVertex3f(0.31,-0.25,0.2);
glVertex3f(0.31,0.2,0.2);
glVertex3f(0.35,0.2,0.2);
glColor3f(0.1,0.1,0.3);   //left
glVertex3f(0.35,-0.25,0.2);
glVertex3f(0.35,-0.25,0.15);
glVertex3f(0.35,0.2,0.15);
glVertex3f(0.35,0.2,0.2);
glColor3f(0.1,0.1,0.5);  //back
glVertex3f(0.35,-0.25,0.15);
glVertex3f(0.31,-0.25,0.15);
glVertex3f(0.31,0.2,0.15);
glVertex3f(0.35,0.2,0.15);
glColor3f(0.1,0.1,0.3);  //right
glVertex3f(0.31,-0.25,0.2);
glVertex3f(0.31,-0.25,0.15);
glVertex3f(0.31,0.2,0.15);
```

```
glVertex3f(0.31,0.2,0.2);
glColor3f(0.1,0.1,0.5);  //front
glVertex3f(0.35,-0.25,-0.2);
glVertex3f(0.31,-0.25,-0.2);
glVertex3f(0.31,0.2,-0.2);
glVertex3f(0.35,0.2,-0.2);
glColor3f(0.1,0.1,0.3);   //left
glVertex3f(0.35,-0.25,-0.2);
glVertex3f(0.35,-0.25,-0.15);
glVertex3f(0.35,0.2,-0.15);
glVertex3f(0.35,0.2,-0.2);
glColor3f(0.1,0.1,0.5);  //back
glVertex3f(0.35,-0.25,-0.15);
glVertex3f(0.31,-0.25,-0.15);
glVertex3f(0.31,0.2,-0.15);
glVertex3f(0.35,0.2,-0.15);
glColor3f(0.1,0.1,0.3);  //right
glVertex3f(0.31,-0.25,-0.2);
glVertex3f(0.31,-0.25,-0.15);
glVertex3f(0.31,0.2,-0.15);
glVertex3f(0.31,0.2,-0.2);
glColor3f(0.2,0.2,0.5);//top
glVertex3f(0.35,0.2,0.3);
glVertex3f(0.31,0.2,0.3);
glVertex3f(0.31,0.25,0.3);
glVertex3f(0.35,0.25,0.3);
glColor3f(0.3,0.3,0.5);
glVertex3f(0.35,0.2,0.3);
glVertex3f(0.35,0.2,-0.3);
glVertex3f(0.35,0.25,-0.3);
glVertex3f(0.35,0.25,0.3);
glColor3f(0.2,0.2,0.5);
glVertex3f(0.35,0.2,-0.3);
glVertex3f(0.31,0.2,-0.3);
glVertex3f(0.31,0.25,-0.3);
glVertex3f(0.35,0.25,-0.3);
glColor3f(0.3,0.3,0.5);
glVertex3f(0.31,0.2,0.3);
glVertex3f(0.31,0.2,-0.3);
glVertex3f(0.31,0.25,-0.3);
glVertex3f(0.31,0.25,0.3);
glColor3f(0.3,0.3,0.6);
glVertex3f(0.35,0.25,0.3);
glVertex3f(0.31,0.25,0.3);
glVertex3f(0.31,0.25,-0.3);
glVertex3f(0.35,0.25,-0.3);
//top piller
glColor3f(0.1,0.1,0.5);  //front
```

```
    glVertex3f(0.35,0.25,0.2);
    glVertex3f(0.31,0.25,0.2);
    glVertex3f(0.31,0.35,0.2);
    glVertex3f(0.35,0.35,0.2);
    glColor3f(0.1,0.1,0.3);   //left
    glVertex3f(0.35,0.25,0.2);
    glVertex3f(0.35,0.25,0.15);
    glVertex3f(0.35,0.35,0.15);
    glVertex3f(0.35,0.35,0.2);
    glColor3f(0.1,0.1,0.5);  //back
    glVertex3f(0.35,0.25,0.15);
    glVertex3f(0.31,0.25,0.15);
    glVertex3f(0.31,0.35,0.15);
    glVertex3f(0.35,0.35,0.15);
    glColor3f(0.1,0.1,0.3);  //right
    glVertex3f(0.31,0.25,0.2);
    glVertex3f(0.31,0.25,0.15);
    glVertex3f(0.31,0.35,0.15);
    glVertex3f(0.31,0.35,0.2);
    glColor3f(0.1,0.1,0.5);  //front
    glVertex3f(0.35,0.25,-0.2);
    glVertex3f(0.31,0.25,-0.2);
    glVertex3f(0.31,0.35,-0.2);
    glVertex3f(0.35,0.35,-0.2);
    glColor3f(0.1,0.1,0.3);   //left
    glVertex3f(0.35,0.25,-0.2);
    glVertex3f(0.35,0.25,-0.15);
    glVertex3f(0.35,0.35,-0.15);
    glVertex3f(0.35,0.35,-0.2);
    glColor3f(0.1,0.1,0.5);  //back
    glVertex3f(0.35,0.25,-0.15);
    glVertex3f(0.31,0.25,-0.15);
    glVertex3f(0.31,0.35,-0.15);
    glVertex3f(0.35,0.35,-0.15);
    glColor3f(0.1,0.1,0.3);  //right
    glVertex3f(0.31,0.25,-0.2);
    glVertex3f(0.31,0.25,-0.15);
    glVertex3f(0.31,0.35,-0.15);
    glVertex3f(0.31,0.35,-0.2);
    glEnd();
    //slops front
    glBegin(GL_QUADS);
    glColor3f(0.8,0.6,0.5);
    glVertex3f(-0.6,-0.25,0.17);
    glVertex3f(-0.5,-0.25,0.17);
    glVertex3f(-0.35,0.2,0.17);
    glVertex3f(-0.35,0.25,0.17);
    //BACK
```

```
    glVertex3f(-0.6,-0.25,-0.17);
    glVertex3f(-0.5,-0.25,-0.17);
    glVertex3f(-0.35,0.2,-0.17);
    glVertex3f(-0.35,0.25,-0.17);
    //right front
    glVertex3f(0.6,-0.25,0.17);
    glVertex3f(0.5,-0.25,0.17);
    glVertex3f(0.35,0.2,0.17);
    glVertex3f(0.35,0.25,0.17);
    //Back
    glVertex3f(0.6,-0.25,-0.17);
    glVertex3f(0.5,-0.25,-0.17);
    glVertex3f(0.35,0.2,-0.17);
    glVertex3f(0.35,0.25,-0.17);
    glEnd();
}
void bridge(){
    glBegin(GL_QUADS);
    glColor3f(0.1,0.2,0.3);
    glVertex3f(-0.3,-0.25,0.15);
    glVertex3f(0.3,-0.25,0.15);
    glVertex3f(0.3,-0.23,0.15);
    glVertex3f(-0.3,-0.23,0.15);
    glColor3f(0.3,0.2,0.3);
    glVertex3f(-0.3,-0.25,0.15);
    glVertex3f(-0.3,-0.25,-0.15);
    glVertex3f(-0.3,-0.23,-0.15);
    glVertex3f(-0.3,-0.23,0.15);
    glColor3f(0.1,0.2,0.3);
    glVertex3f(-0.3,-0.25,-0.15);
    glVertex3f(0.3,-0.25,-0.15);
    glVertex3f(0.3,-0.23,-0.15);
    glVertex3f(-0.3,-0.23,-0.15);
    glColor3f(0.3,0.2,0.3);
    glVertex3f(0.3,-0.25,0.15);
    glVertex3f(0.3,-0.25,-0.15);
    glVertex3f(0.3,-0.23,-0.15);
    glVertex3f(0.3,-0.23,0.15);
    glColor3f(0.3,0.3,0.4);
    glVertex3f(-0.3,-0.23,0.15);
    glVertex3f(0.3,-0.23,0.15);
    glVertex3f(0.3,-0.23,-0.15);
    glVertex3f(-0.3,-0.23,-0.15);
    glEnd();
    //pillers
    //left front
    glBegin(GL_QUADS);
    glColor3f(0.3,0.2,0.1);
```

```
glVertex3f(-0.3,-0.23,0.15);
glVertex3f(-0.28,-0.23,0.15);
glVertex3f(-0.28,0.1,0.15);
glVertex3f(-0.3,0.1,0.15);
glColor3f(0.3,0.2,0.3);
glVertex3f(-0.3,-0.23,0.15);
glVertex3f(-0.3,-0.23,0.12);
glVertex3f(-0.3,0.1,0.12);
glVertex3f(-0.3,0.1,0.15);
glColor3f(0.3,0.2,0.1);
glVertex3f(-0.3,-0.23,0.12);
glVertex3f(-0.28,-0.23,0.12);
glVertex3f(-0.28,0.1,0.12);
glVertex3f(-0.3,0.1,0.12);
glColor3f(0.3,0.2,0.3);
glVertex3f(-0.28,-0.23,0.15);
glVertex3f(-0.28,-0.23,0.12);
glVertex3f(-0.28,0.1,0.12);
glVertex3f(-0.28,0.1,0.15);
glEnd();
//right front
glBegin(GL_QUADS);
glColor3f(0.3,0.2,0.1);
glVertex3f(0.3,-0.23,0.15);
glVertex3f(0.28,-0.23,0.15);
glVertex3f(0.28,0.1,0.15);
glVertex3f(0.3,0.1,0.15);
glColor3f(0.3,0.2,0.3);
glVertex3f(0.3,-0.23,0.15);
glVertex3f(0.3,-0.23,0.12);
glVertex3f(0.3,0.1,0.12);
glVertex3f(0.3,0.1,0.15);
glColor3f(0.3,0.2,0.1);
glVertex3f(0.3,-0.23,0.12);
glVertex3f(0.28,-0.23,0.12);
glVertex3f(0.28,0.1,0.12);
glVertex3f(0.3,0.1,0.12);
glColor3f(0.3,0.2,0.3);
glVertex3f(0.28,-0.23,0.15);
glVertex3f(0.28,-0.23,0.12);
glVertex3f(0.28,0.1,0.12);
glVertex3f(0.28,0.1,0.15);
glEnd();
//left back
glBegin(GL_QUADS);
glColor3f(0.3,0.2,0.1);
glVertex3f(-0.3,-0.23,-0.15);
glVertex3f(-0.28,-0.23,-0.15);
```

```
glVertex3f(-0.28,0.1,-0.15);
glVertex3f(-0.3,0.1,-0.15);
glColor3f(0.3,0.2,0.3);
glVertex3f(-0.3,-0.23,-0.15);
glVertex3f(-0.3,-0.23,-0.12);
glVertex3f(-0.3,0.1,-0.12);
glVertex3f(-0.3,0.1,-0.15);
glColor3f(0.3,0.2,0.1);
glVertex3f(-0.3,-0.23,-0.12);
glVertex3f(-0.28,-0.23,-0.12);
glVertex3f(-0.28,0.1,-0.12);
glVertex3f(-0.3,0.1,-0.12);
glColor3f(0.3,0.2,0.3);
glVertex3f(-0.28,-0.23,-0.15);
glVertex3f(-0.28,-0.23,-0.12);
glVertex3f(-0.28,0.1,-0.12);
glVertex3f(-0.28,0.1,-0.15);
glEnd();
//right back
glBegin(GL_QUADS);
glColor3f(0.3,0.2,0.1);
glVertex3f(0.3,-0.23,-0.15);
glVertex3f(0.28,-0.23,-0.15);
glVertex3f(0.28,0.1,-0.15);
glVertex3f(0.3,0.1,-0.15);
glColor3f(0.3,0.2,0.3);
glVertex3f(0.3,-0.23,-0.15);
glVertex3f(0.3,-0.23,-0.12);
glVertex3f(0.3,0.1,-0.12);
glVertex3f(0.3,0.1,-0.15);
glColor3f(0.3,0.2,0.1);
glVertex3f(0.3,-0.23,-0.12);
glVertex3f(0.28,-0.23,-0.12);
glVertex3f(0.28,0.1,-0.12);
glVertex3f(0.3,0.1,-0.12);
glColor3f(0.3,0.2,0.3);
glVertex3f(0.28,-0.23,-0.15);
glVertex3f(0.28,-0.23,-0.12);
glVertex3f(0.28,0.1,-0.12);
glVertex3f(0.28,0.1,-0.15);
glEnd();
//top left
glBegin(GL_QUADS);
glColor3f(0.4,0.3,0.2);
glVertex3f(-0.3,0.1,0.15);
glVertex3f(-0.28,0.1,0.15);
glVertex3f(-0.28,0.13,0.15);
glVertex3f(-0.3,0.13,0.15);
```

```
glColor3f(0.5,0.3,0.2);
glVertex3f(-0.3,0.1,0.15);
glVertex3f(-0.3,0.1,-0.15);
glVertex3f(-0.3,0.13,-0.15);
glVertex3f(-0.3,0.13,0.15);
glColor3f(0.4,0.3,0.2);
glVertex3f(-0.3,0.1,-0.15);
glVertex3f(-0.28,0.1,-0.15);
glVertex3f(-0.28,0.13,-0.15);
glVertex3f(-0.3,0.13,-0.15);
glColor3f(0.5,0.3,0.2);
glVertex3f(-0.28,0.1,0.15);
glVertex3f(-0.28,0.1,-0.15);
glVertex3f(-0.28,0.13,-0.15);
glVertex3f(-0.28,0.13,0.15);
glEnd();
//top roght
glBegin(GL_QUADS);
glColor3f(0.4,0.3,0.2);
glVertex3f(0.3,0.1,0.15);
glVertex3f(0.28,0.1,0.15);
glVertex3f(0.28,0.13,0.15);
glVertex3f(0.3,0.13,0.15);
glColor3f(0.5,0.3,0.2);
glVertex3f(0.3,0.1,0.15);
glVertex3f(0.3,0.1,-0.15);
glVertex3f(0.3,0.13,-0.15);
glVertex3f(0.3,0.13,0.15);
glColor3f(0.4,0.3,0.2);
glVertex3f(0.3,0.1,-0.15);
glVertex3f(0.28,0.1,-0.15);
glVertex3f(0.28,0.13,-0.15);
glVertex3f(0.3,0.13,-0.15);
glColor3f(0.5,0.3,0.2);
glVertex3f(0.28,0.1,0.15);
glVertex3f(0.28,0.1,-0.15);
glVertex3f(0.28,0.13,-0.15);
glVertex3f(0.28,0.13,0.15);
glEnd();
//sides front
glBegin(GL_QUADS);
glColor3f(0.4,0.3,0.5);
glVertex3f(-0.28,-0.15,0.15);
glVertex3f(0.28,-0.15,0.15);
glVertex3f(0.28,-0.12,0.15);
glVertex3f(-0.28,-0.12,0.15);
glVertex3f(-0.28,-0.15,0.12);
glVertex3f(0.28,-0.15,0.12);
```

```
glVertex3f(0.28,-0.12,0.12);
glVertex3f(-0.28,-0.12,0.12);
//back
glVertex3f(-0.28,-0.15,-0.15);
glVertex3f(0.28,-0.15,-0.15);
glVertex3f(0.28,-0.12,-0.15);
glVertex3f(-0.28,-0.12,-0.15);
glVertex3f(-0.28,-0.15,-0.12);
glVertex3f(0.28,-0.15,-0.12);
glVertex3f(0.28,-0.12,-0.12);
glVertex3f(-0.28,-0.12,-0.12);
glEnd();
//top
glBegin(GL_QUADS);
glColor3f(0.4,0.3,0.5);
glVertex3f(-0.28,0.1,0.15);
glVertex3f(0.28,0.1,0.15);
glVertex3f(0.28,0.13,0.15);
glVertex3f(-0.28,0.13,0.15);
glVertex3f(-0.28,0.1,0.12);
glVertex3f(0.28,0.1,0.12);
glVertex3f(0.28,0.13,0.12);
glVertex3f(-0.28,0.13,0.12);
//back
glVertex3f(-0.28,0.1,-0.15);
glVertex3f(0.28,0.1,-0.15);
glVertex3f(0.28,0.13,-0.15);
glVertex3f(-0.28,0.13,-0.15);
glVertex3f(-0.28,0.1,-0.12);
glVertex3f(0.28,0.1,-0.12);
glVertex3f(0.28,0.13,-0.12);
glVertex3f(-0.28,0.13,-0.12);
glEnd();
//house
glColor3f(0.8,0.1,0.1);
glPushMatrix();
glTranslatef(0.0,0.25,0.0);
glutSolidCube(0.25);
glPopMatrix();
glBegin(GL_QUADS); //WINDOW
glColor3f(1.0,1.0,1.0);
glVertex3f(-0.05,0.18,0.16);
glVertex3f(0.05,0.18,0.16);
glVertex3f(0.05,0.25,0.16);
glVertex3f(-0.05,0.25,0.16);
glEnd();
glBegin(GL_TRIANGLES);  //ROOF
glColor3f(1.0,0.8,0.0);
```

```
    glVertex3f(-0.16,0.35,0.16);
    glVertex3f(0.16,0.35,0.16);
    glVertex3f(0.0,0.5,0.0);
    glVertex3f(-0.16,0.35,0.16);
    glVertex3f(-0.16,0.35,-0.16);
    glVertex3f(0.0,0.5,0.0);
    glVertex3f(-0.16,0.35,-0.16);
    glVertex3f(0.16,0.35,-0.16);
    glVertex3f(0.0,0.5,0.0);
    glVertex3f(0.16,0.35,0.16);
    glVertex3f(0.16,0.35,-0.16);
    glVertex3f(0.0,0.5,0.0);
    glEnd();
}
void track(){
    glBegin(GL_LINES);
    glColor3f(0.0,0.0,0.0);//left
    glVertex3f(-3.0,-0.23,0.12);
    glVertex3f(-0.3,-0.23,0.12);
    glVertex3f(-3.0,-0.23,0.1);
    glVertex3f(-0.3,-0.23,0.1);
    glVertex3f(-3.0,-0.23,-0.12);
    glVertex3f(-0.3,-0.23,-0.12);
    glVertex3f(-3.0,-0.23,-0.1);
    glVertex3f(-0.3,-0.23,-0.1);
    glVertex3f(3.0,-0.23,0.12);
    glVertex3f(0.3,-0.23,0.12);
    glVertex3f(3.0,-0.23,0.1);
    glVertex3f(0.3,-0.23,0.1);
    glVertex3f(3.0,-0.23,-0.12);
    glVertex3f(0.3,-0.23,-0.12);
    glVertex3f(3.0,-0.23,-0.1);
    glVertex3f(0.3,-0.23,-0.1);
    glEnd();
    glBegin(GL_LINES);
    glColor3f(0.0,0.0,0.0);
    for(j=0.0;j<=2.6;j+=0.1) {
        glVertex3f(-3.0+j,-0.23,0.1);
        glVertex3f(-3.0+j,-0.23,-0.1);
    }
    for(j=0.0;j<=3;j+=0.1){
        glVertex3f(0.3+j,-0.23,0.1);
        glVertex3f(0.3+j,-0.23,-0.1);
    }
    glEnd();
}
void ship(){
    glBegin(GL_QUADS);
```

```
glColor3f(0.8,0.8,0.8);   //base
glVertex3f(-0.2,-0.4,-3.5);
glVertex3f(0.2,-0.4,-3.5);
glVertex3f(0.2,-0.3,-3.5);
glVertex3f(-0.2,-0.3,-3.5);
glColor3f(0.8,0.8,1.0);
glVertex3f(-0.2,-0.4,-3.5);
glVertex3f(-0.2,-0.4,-4.8);
glVertex3f(-0.2,-0.3,-5.0);
glVertex3f(-0.2,-0.3,-3.5);
glColor3f(0.8,0.8,0.8);
glVertex3f(-0.2,-0.4,-4.8);
glVertex3f(0.2,-0.4,-4.8);
glVertex3f(0.2,-0.3,-5.0);
glVertex3f(-0.2,-0.3,-5.0);
glColor3f(0.8,0.8,1.0);
glVertex3f(0.2,-0.4,-3.5);
glVertex3f(0.2,-0.4,-4.8);
glVertex3f(0.2,-0.3,-5.0);
glVertex3f(0.2,-0.3,-3.5);
glColor3f(1.0,0.8,1.0);
glVertex3f(-0.2,-0.3,-3.5);
glVertex3f(0.2,-0.3,-3.5);
glVertex3f(0.2,-0.3,-5.0);
glVertex3f(-0.2,-0.3,-5.0);
glColor3f(1.0,0.0,0.7);
glVertex3f(-0.18,-0.3,-3.7);
glVertex3f(0.18,-0.3,-3.7);
glVertex3f(0.18,-0.2,-3.7);
glVertex3f(-0.18,-0.2,-3.7);
glColor3f(1.0,0.0,0.5);
glVertex3f(-0.18,-0.3,-3.7);
glVertex3f(-0.18,-0.3,-4.8);
glVertex3f(-0.18,-0.2,-4.8);
glVertex3f(-0.18,-0.2,-3.7);
glColor3f(1.0,0.0,0.7);
glVertex3f(-0.18,-0.3,-3.7);
glVertex3f(0.18,-0.3,-4.8);
glVertex3f(0.18,-0.2,-4.8);
glVertex3f(-0.18,-0.2,-3.7);
glColor3f(1.0,0.0,0.5);
glVertex3f(0.18,-0.3,-3.7);
glVertex3f(0.18,-0.3,-4.8);
glVertex3f(0.18,-0.2,-4.8);
glVertex3f(0.18,-0.2,-3.7);
glColor3f(1.0,0.1,0.8);
glVertex3f(-0.18,-0.2,-3.7);
glVertex3f(0.18,-0.2,-3.7);
```

```
glVertex3f(0.18,-0.2,-4.8);
glVertex3f(-0.18,-0.2,-4.8);
glEnd();
//front
glBegin(GL_TRIANGLES);
glColor3f(0.5,0.5,0.7);
glVertex3f(-0.2,-0.4,-3.5);
glVertex3f(-0.2,-0.3,-3.5);
glVertex3f(0.0,-0.15,-2.2);
glColor3f(0.5,0.8,0.7);
glVertex3f(-0.2,-0.4,-3.5);
glVertex3f(0.2,-0.4,-3.5);
glVertex3f(0.0,-0.15,-2.2);
glColor3f(0.5,0.5,0.7);
glVertex3f(0.2,-0.4,-3.5);
glVertex3f(0.2,-0.3,-3.5);
glVertex3f(0.0,-0.15,-2.2);
glEnd();
//TOP PILLARS
glBegin(GL_QUADS);
glColor3f(1.0,0.8,0.1);
glVertex3f(-0.05,-0.2,-3.8);
glVertex3f(0.05,-0.2,-3.8);
glVertex3f(0.05,0.1,-3.8);
glVertex3f(-0.05,0.1,-3.8);
glColor3f(1.0,0.8,0.2);
glVertex3f(-0.05,-0.2,-3.8);
glVertex3f(-0.05,-0.2,-4.0);
glVertex3f(-0.05,0.1,-4.0);
glVertex3f(-0.05,0.1,-3.8);
glColor3f(1.0,0.8,0.1);
glVertex3f(-0.05,-0.2,-4.0);
glVertex3f(0.05,-0.2,-4.0);
glVertex3f(0.05,0.1,-4.0);
glVertex3f(-0.05,0.1,-4.0);
glColor3f(1.0,0.8,0.2);
glVertex3f(0.05,-0.2,-3.8);
glVertex3f(0.05,-0.2,-4.0);
glVertex3f(0.05,0.1,-4.0);
glVertex3f(0.05,0.1,-3.8);
//back
glColor3f(1.0,0.5,0.1);
glVertex3f(-0.05,-0.2,-4.2);
glVertex3f(0.05,-0.2,-4.2);
glVertex3f(0.05,0.2,-4.2);
glVertex3f(-0.05,0.2,-4.2);
glColor3f(1.0,0.6,0.2);
glVertex3f(-0.05,-0.2,-4.2);
```

```
    glVertex3f(-0.05,-0.2,-4.5);
    glVertex3f(-0.05,0.2,-4.5);
    glVertex3f(-0.05,0.2,-4.2);
    glColor3f(1.0,0.5,0.1);
    glVertex3f(-0.05,-0.2,-4.5);
    glVertex3f(0.05,-0.2,-4.5);
    glVertex3f(0.05,0.2,-4.5);
    glVertex3f(-0.05,0.2,-4.5);
    glColor3f(1.0,0.6,0.2);
    glVertex3f(0.05,-0.2,-4.2);
    glVertex3f(0.05,-0.2,-4.5);
    glVertex3f(0.05,0.2,-4.5);
    glVertex3f(0.05,0.2,-4.2);
    glEnd();
}
void train(){
    glBegin(GL_QUADS);   //engine
    glColor3f(0.8,0.6,0.4);
    glVertex3f(1.0,-0.23,0.1);
    glVertex3f(1.15,-0.23,0.1);
    glVertex3f(1.15,-0.14,0.1);
    glVertex3f(1.0,-0.14,0.1);
    glColor3f(0.5,0.5,0.8);
    glVertex3f(1.0,-0.23,0.1);
    glVertex3f(1.0,-0.23,-0.1);
    glVertex3f(1.0,-0.14,-0.1);
    glVertex3f(1.0,-0.14,0.1);
    glColor3f(0.8,0.6,0.4);
    glVertex3f(1.0,-0.23,-0.1);
    glVertex3f(1.15,-0.23,-0.1);
    glVertex3f(1.15,-0.14,-0.1);
    glVertex3f(1.0,-0.14,-0.1);
    //FRONT
    glColor3f(0.0,0.1,0.9);
    glVertex3f(1.0,-0.14,0.1);
    glVertex3f(1.15,-0.05,0.1);
    glVertex3f(1.15,-0.05,-0.1);
    glVertex3f(1.0,-0.14,-0.1);
    glColor3f(0.8,0.6,0.2);
    glVertex3f(1.02,-0.12,0.1);
    glVertex3f(1.13,-0.05,0.1);
    glVertex3f(1.13,-0.05,-0.06);
    glVertex3f(1.02,-0.12,-0.06);
    glEnd();
    //side
    glBegin(GL_TRIANGLES);
    glColor3f(0.0,0.0,0.0);
    glVertex3f(1.0,-0.14,0.1);
```

```
    glVertex3f(1.15,-0.14,0.1);
    glVertex3f(1.15,-0.05,0.1);
    glVertex3f(1.0,-0.14,-0.1);
    glVertex3f(1.15,-0.14,-0.1);
    glVertex3f(1.15,-0.05,-0.1);
    glEnd();
    //bogies
    glBegin(GL_QUADS);
    for(j=0.0;j<2;j+=0.27){
    glColor3f(0.5,0.0,0.1);
    glVertex3f(1.15+j,-0.23,0.1);
    glVertex3f(1.4+j,-0.23,0.1);
    glVertex3f(1.4+j,-0.05,0.1);
    glVertex3f(1.15+j,-0.05,0.1);
    glColor3f(0.5,0.0,0.5);
    glVertex3f(1.15+j,-0.23,0.1);
    glVertex3f(1.15+j,-0.23,-0.1);
    glVertex3f(1.15+j,-0.05,-0.1);
    glVertex3f(1.15+j,-0.05,0.1);
    glColor3f(0.5,0.0,0.1);
    glVertex3f(1.15+j,-0.23,-0.1);
    glVertex3f(1.4+j,-0.23,-0.1);
    glVertex3f(1.4+j,-0.05,-0.1);
    glVertex3f(1.15+j,-0.05,-0.1);
    glColor3f(0.5,0.0,0.5);
    glVertex3f(1.4+j,-0.23,0.1);
    glVertex3f(1.4+j,-0.23,-0.1);
    glVertex3f(1.4+j,-0.05,-0.1);
    glVertex3f(1.4+j,-0.05,0.1);
    glColor3f(0.8,0.3,0.5);
    glVertex3f(1.15+j,-0.05,0.1);
    glVertex3f(1.4+j,-0.05,0.1);
    glVertex3f(1.4+j,-0.05,-0.1);
    glVertex3f(1.15+j,-0.05,-0.1);
    }
    glEnd();
}
void aero(){
    glBegin(GL_POLYGON);
    glColor3f(0.9,0.9,.9);
    glVertex3f(-3.2,0.7,-0.8);
    glVertex3f(-3.0,0.67,-0.8);
    glVertex3f(-2.4,0.67,-0.8);
    glVertex3f(-2.4,0.73,-0.8);
    glVertex3f(-3.2,0.73,-0.8);
    glVertex3f(-3.2,0.7,-0.65);
    glVertex3f(-3.0,0.67,-0.65);
    glVertex3f(-2.4,0.67,-0.65);
```

```
glVertex3f(-2.4,0.73,-0.65);
glVertex3f(-3.2,0.73,-0.65);
glEnd();
glBegin(GL_QUADS);
glColor3f(0.7,0.7,.7);
glVertex3f(-3.2,0.7,-0.65);
glVertex3f(-3.2,0.7,-0.8);
glVertex3f(-3.2,0.73,-0.8);
glVertex3f(-3.2,0.73,-0.65);
glColor3f(0.7,0.7,.7);
glVertex3f(-2.4,0.67,-0.65);
glVertex3f(-2.4,0.67,-0.8);
glVertex3f(-2.4,0.73,-0.8);
glVertex3f(-2.4,0.73,-0.65);
glColor3f(0.7,0.8,0.8);
glVertex3f(-3.2,0.73,-0.65);
glVertex3f(-2.4,0.73,-0.65);
glVertex3f(-2.4,0.73,-0.8);
glVertex3f(-3.2,0.73,-0.8);
//wings
glColor3f(0.8,0.2,0.5);
glVertex3f(-2.8,0.7,-0.7);
glVertex3f(-2.62,0.7,-0.7);
glVertex3f(-2.75,0.7,-0.2);
glVertex3f(-2.85,0.7,-0.2);
glVertex3f(-2.8,0.7,-0.8);
glVertex3f(-2.62,0.7,-0.8);
glVertex3f(-2.75,0.7,-1.3);
glVertex3f(-2.85,0.7,-1.3);
glVertex3f(-3.2,0.73,-0.725);
glVertex3f(-3.1,0.73,-0.725);
glVertex3f(-3.18,0.82,-0.725);
glVertex3f(-3.25,0.82,-0.725);
glEnd();
glBegin(GL_TRIANGLES);
glColor3f(1.0,0.5,0.2);
glVertex3f(-2.4,0.73,-0.8);
glVertex3f(-2.4,0.73,-0.65);
glVertex3f(-2.2,0.68,-0.725);
glColor3f(1.0,0.7,0.4);
glVertex3f(-2.4,0.67,-0.65);
glVertex3f(-2.4,0.73,-0.65);
glVertex3f(-2.2,0.68,-0.725);
glColor3f(1.0,0.5,0.2);
glVertex3f(-2.4,0.68,-0.65);
glVertex3f(-2.4,0.68,-0.65);
glVertex3f(-2.2,0.68,-0.725);
glColor3f(1.0,0.7,0.4);
```

```
    glVertex3f(-2.4,0.69,-0.8);
    glVertex3f(-2.4,0.73,-0.8);
    glVertex3f(-2.2,0.68,-0.725);
    glEnd();
    char str[]=" AIR  CIT . . .";
    setFont(GLUT_BITMAP_HELVETICA_18);
    glColor3f(0.0,0.0,0.0);
    drawstring(str,-3.05,0.69,-0.65);
}
void lighthouse(){
    glBegin(GL_QUADS);
    glColor3f(0.8,0.2,0.0);
    glVertex3f(0.2,-0.42,-5.01);
    glVertex3f(0.35,-0.42,-5.01);
    glVertex3f(0.3,-0.1,-5.01);
    glVertex3f(0.25,-0.1,-5.01);
    glColor3f(1.0,1.0,1.0);
    glVertex3f(0.23,-0.1,-5.01);
    glVertex3f(0.32,-0.1,-5.01);
    glVertex3f(0.32,0.0,-5.01);
    glVertex3f(0.23,0.0,-5.01);
    glColor3f(1.0,1.0,1.0);
    glVertex3f(0.215,-0.3,-5.0);
    glVertex3f(0.33,-0.3,-5.0);
    glVertex3f(0.315,-0.2,-5.0);
    glVertex3f(0.23,-0.2,-5.0);
    glEnd();
    glBegin(GL_TRIANGLES);
    glColor3f(1.0,0.2,0.2);
    glVertex3f(0.2,0.0,-5.0);
    glVertex3f(0.35,0.0,-5.0);
    glVertex3f(0.27,0.05,-5.0);
    glEnd();
}
void signal(){
    glBegin(GL_QUADS);
    glColor3f(0.1,0.2,0.1);
    glVertex3f(0.7,-0.25,-0.17);
    glVertex3f(0.73,-0.25,-0.17);
    glVertex3f(0.73,0.15,-0.17);
    glVertex3f(0.7,0.15,-0.17);
    glColor3f(0.1,0.1,0.2);
    glVertex3f(0.67,0.15,-0.17);
    glVertex3f(0.76,0.15,-0.17);
    glVertex3f(0.76,0.3,-0.17);
    glVertex3f(0.67,0.3,-0.17);
    glEnd();
}
```

```
void light(){
   if(b>0.0)
      glColor3f(1.0,0.0,0.0);
   else
         glColor3f(0.0,1.0,0.0);
   if(p<-3.5)
       glColor3f(1.0,0.0,0.0);
   glPushMatrix();
   glTranslatef(0.715,0.25,-0.17);
   glutSolidSphere(0.03,10,10);
   glPopMatrix();
}
void new1(){
   glTranslatef(a,b,c);
   bridge();
}
void new2(){
   glTranslatef(m,n,o);
   ship();
}
void new3(){
   glTranslatef(p,q,r);
   train();
}
void new4(){
   glTranslatef(x,y,z);
   aero();
}
void new5(){
   glTranslatef(a1,a2,a3);
   lines();
}
void update(int value){
   if(flagx==1){
    if(flagb==1){
      b+=0.02f;
      if(b>0.5){
         flagb=2;
         flags=1;
      }
    }
    if(flags==1){
      o+=0.07f;
      if(o>2.0)
         flagp=1;
      if(o>6.0){
         flagb=0;
      }
```

```
    }
    if(flagb==0){
      b-=0.02f;
      if(b<0.01){
          flagb=1;
          flagt=1;
      }
    }
    if(flagt==1){
        p-=0.05f;
    }
    if(flagp==1){
      x+=0.035;
    }
     if(flagw==1){
      a1+=0.006;
     }
   }
   glutPostRedisplay();
   glutTimerFunc(100,update,0);
}
void display(){
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glClearColor(0.0,0.5,1.0,0.0);
   glPushMatrix();
   glRotatef(20.0,0.25,0.5,0.0);
   base();
   glPopMatrix();
   glPushMatrix();
   glRotatef(20.0,0.25,0.5,0.0);
   pillars();
   glPopMatrix();
   glPushMatrix();
   glRotatef(20.0,0.25,0.5,0.0);
   earth();
   glPopMatrix();
   glPushMatrix();
   glRotatef(20.0,0.25,0.5,0.0);
   track();
   glPopMatrix();
   glPushMatrix();
   glRotatef(20.0,0.25,0.5,0.0);
   glPopMatrix();
   glPushMatrix();
   glRotatef(20.0,0.25,0.5,0.0);
   new1();
   glPopMatrix();
   glPushMatrix();
```

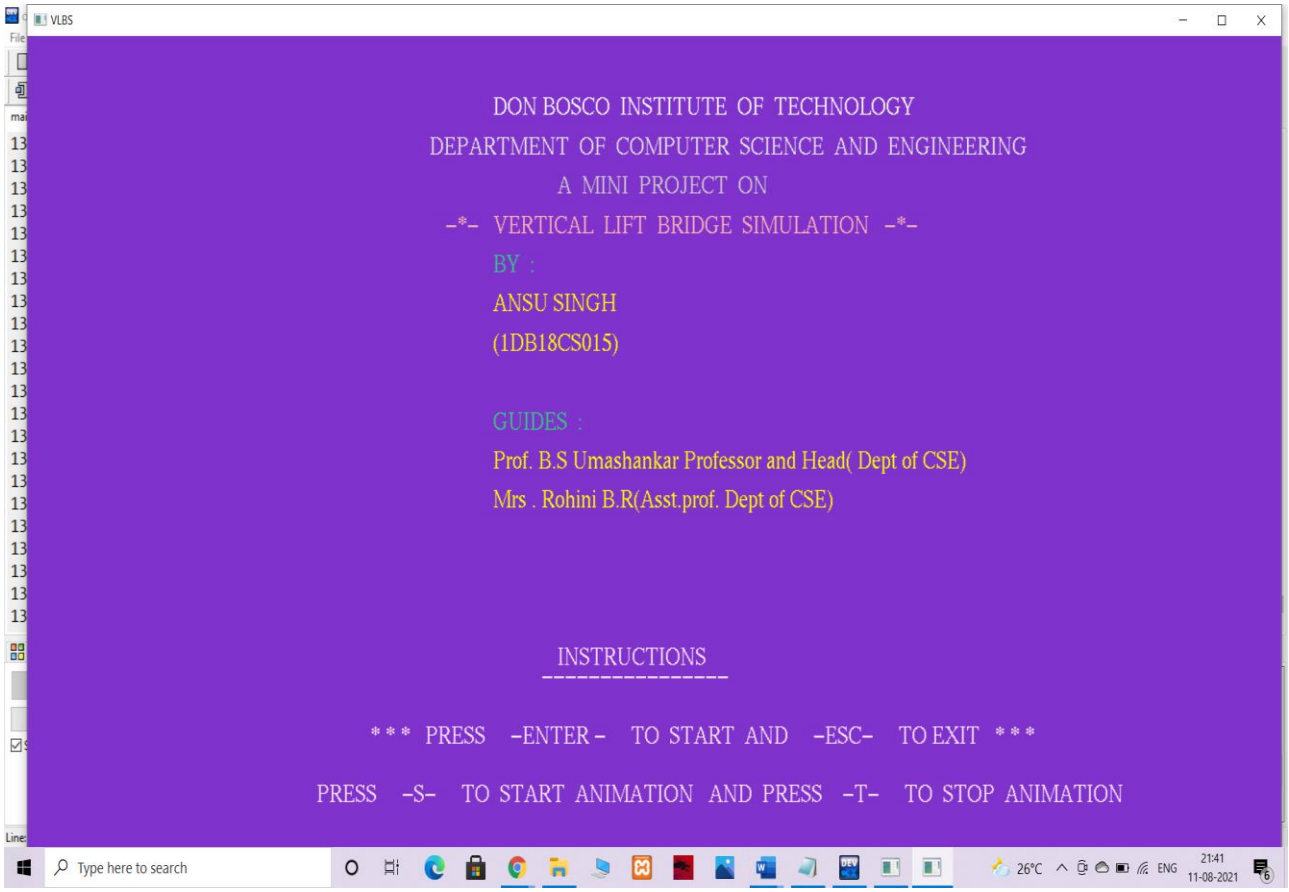
```
    glRotatef(20.0,0.25,0.5,0.0);
    new2();
    glPopMatrix();
    glPushMatrix();
    glRotatef(20.0,0.25,0.5,0.0);
    new3();
    glPopMatrix();
    glPushMatrix();
    glRotatef(20.0,0.25,0.5,0.0);
    new4();
    glPopMatrix();
    glPushMatrix();
    glRotatef(20.0,0.25,0.5,0.0);
    water();
    glPopMatrix();
    glPushMatrix();
    glRotatef(20.0,0.25,0.5,0.0);
    new5();
    glPopMatrix();
    glPushMatrix();
    glColor3f(1.0,1.0,0.0);
    glTranslatef(1.2,0.9,-5.1);
    glutSolidSphere(0.08,20,20);
    glPopMatrix();
    glPushMatrix();
    glRotatef(20.0,0.25,0.5,0.0);
    lighthouse();
    glTranslatef(0.28,-0.05,-5.0);
    glColor3f(0.0,0.0,0.0);
    glutSolidSphere(0.02,20,20);
    glPopMatrix();
    glPushMatrix();
    glRotatef(20.0,0.25,0.5,0.0);
    signal();
    glPopMatrix();
    glPushMatrix();
    glRotatef(20.0,0.25,0.5,0.0);
    light();
    glPopMatrix();
    glFlush();
}
void mydisplay(){
    if(flag==0)
        screen1();
     if(flag==1)
      display();
    if(p<-6.0)
      screen3();
```

```
    if(p<-6.8)
      exit(0);
}
void mykeyboard(unsigned char key,int x,int y){
    switch(key){
        case 13 :flag=1;break;
        case 83 :if(flag==1)
                  flagx=1;break;
        case 115:if(flag==1)
                   flagx=1;break;
        case 84 :flagx=0;break;
       case 116 :flagx=0;break;
        case 27:exit(0);
    }
}
void reshape(int w,int h){
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if(w<=h)
        glOrtho(-1.1,1.1,1.1*(GLfloat)h/(GLfloat)w,1.1*(GLfloat)h/(GLfloat)w,-10.0,10.0);
    else
        glOrtho(-1.1*(GLfloat)w/(GLfloat)h,1.1*(GLfloat)w/(GLfloat)h,-1.1,1.1,-10.0,10.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
int main(int argc,char **argv){
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(1500,1000);
    glutCreateWindow("VLBS");
    glClearColor(0.0,0.0,0.0,0.0);
    glEnable(GL_DEPTH_TEST);
    glutReshapeFunc(reshape);
    glutDisplayFunc(mydisplay);
    glutKeyboardFunc(mykeyboard);
    glutTimerFunc(200,update,0);
    glutMainLoop();
    return 0;
}
```

6.TESTING

- Press enter key to start and esc to exit
- After enter key press S key to start animation. T key to stop animation.
- In animation bridge is LIFTING up by pressing S key And SHIP CROSSING THE BRIDGE
- After ship crosses the bridge, the bridge come down. And train start crossing the birdge.

7.SNAPSHOTS

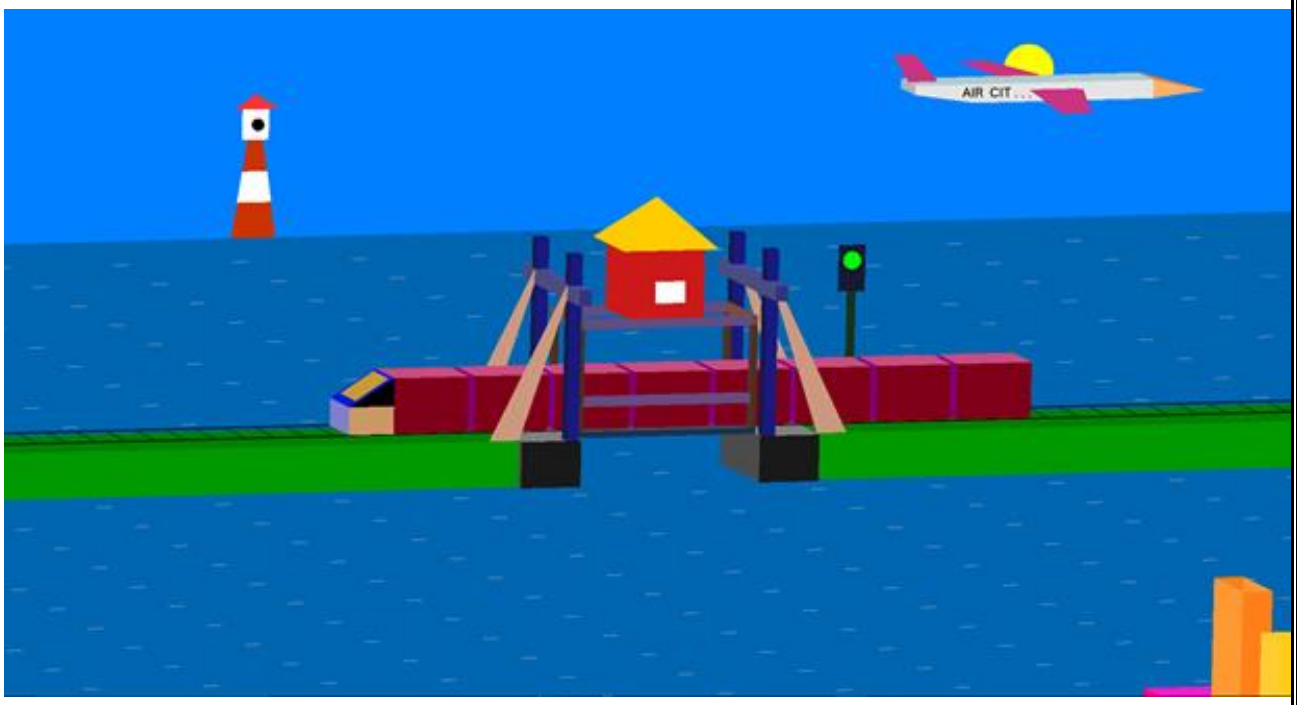


WELCOME SCREEN

LIFTING OF THE BRIDGE

SHIP CROSSING THE BRIDGE

LOWERING OF THE BRIDGE



TRAIN CROSSING THE BRIDGE

EXIT SCREEN

8.CONCLUSION

Designing and implementing project in graphics is a great experience. We understood and analyzed about the concepts of OpenGL which is very useful for our future.

"VERTICAL LIFT BRIDGE SIMULATION" is developed to provide a GUI. An attempt has been made to develop an openGL package which meets necessary requirements of the user.

The development of the mini project has given us a good exposure to openGL by which we have learnt some of the techniques which help in the development of interactive application

9.APPENDIX

glutPostRedisplay(void):

Request that the image be redrawn**.**

glutSwapBuffers(void):

Swap the front and back buffers(used in double buffering).

glClearColor(GLclampf R, GLclampf G, GLclampf B, GLclampf A):

Specifies the background color to be used by glClear() , when clearing the buffer.

glClear(GLbitfield mask):

Clears one or more of the existing buffers. The mask is logical or ("|") of any of the following : GL_COLOR_BUFFER_BIT,GL_DEPTH_BUFFER_BIT.

glFlush():

OpenGL normally saves or "buffers" drawing commands for greater efficiency. This forces the image to be redrawn , in case the next update will be far in the future.

glBegin(GLenum mode)…glEnd(void):

Specify the starting and ending of vertices.

glVertex*(…):

Specify the coordinates of a vertex.

glNormal*(...):

Specify the surface normal for subsequent vertices. The normal should be of unit length after the modelview transformation is applied. Call glEnable (GL_NORMALIZE) to have OpenGL do normalization automatically.

glColor*(…):

Specify the color of subsequent vertices . This is normally used if lighting is not enabled . Otherwise , colors are defined by glMaterial*() , defined below under lighting .

glLineWidth(GLfloat width):

Sets the line width for subsequent line drawing .

glRasterPos*(...):

Set the current raster position (in 3D window coordinates) for subsequent glutBitmapCharacter () and other pixel copying operations.

glutBitmapCharacter (void *font, int character):

Draw the given character of the given font at the current raster position and advance the current raster position. Fonts include GLUT_BITMAP_9_BY_15, GLUT_BITMAP_8_BY_13, GLUT_BITMAP_TIMES_ROMAN_10, GLUT_BITMAP_TIMES_ROMAN_24. When a character is drawn, the raster position is advanced. So to draw a string, initialize the raster position (using glRasterPos*()) and then call this on each character of the string.

glViewport (GLint x, GLint y, GLsizei width, GLsizei height):

Sets the current viewport, that is, the portion of the window to which everything will be clipped. Typically this is the entire window: glViewport (0, 0, win width, win height), and hence this should be called whenever the window is reshaped.

glMatrixMode(GLenum mode):

Set the current matrix mode to one of GL_MODELVIEW , GL_PROJECTION, or GL_TEXTURE. The default is GL_MODELVIEW.

glLoadIdentity (void):

Set the current matrix to the identity.

glPushMatrix (void):

Make a copy of the current matrix and push it onto the stack.

glPopMatrix (void):

Pop the top of the current matrix.

glLoadMatrixf (const GLfloat *M):

Sets the current matrix to *M*.

glColor3f (float, float, float):

This function will set the current drawing color.

gluOrtho(GLdouble left, GLdouble right, GLdouble bottom,

GLdoubletop,GLdouble zNear,Gldouble zfar):

The ortho function multiplies the current matrix by an orthography matrix.

glClear( ):

Takes a single argument that is the bitwise OR of several values indicating which buffer is to be cleared.

glClearColor ():

Specifies the red, green, blue, and alpha values used by **glClear** to clear the color buffers.

glLoadIdentity( ):

The current matrix with the identity matrix.

glMatrixMode(mode):

Sets the current matrix mode, mode can be GL_MODELVIEW, GL_PROJECTION or GL_TEXTURE.

void glutInit (int *argc, char**argv):

Initializes GLUT, the arguments from main are passed in and can be used by the application.

void glutInitDisplayMode (unsigned int mode):

Requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model and buffering.

void glutInitWindowSize (int width, int height):

Specifies the initial position of the top-left corner of the window in pixels..

void glutDisplayFunc (void (*func) (void)):

Register the display function func that is executed when the window needs to be redrawn.

glutPostReDisplay ( ) :

Which requests that the display callback be executed after the current callback returns.

void glutMainLoop ():

Cause the program to enter an event-processing loop. It should be the last statement in main function.

glFlush():

Forces and buffers any openGL commands to execute

glPushMatrix():

Make a copy of current matrix and push it onto the stack.

glPopMatrix():

Pop the top of the current matrix.

glutBitmapCharacter(void* font,int character):

Write a character to output.

glRasterpos3f(x,y,z):

To set the coordinates of raster text.

10.Bibliography

Books:

- **Edward Angel**, "Interactive Computer Graphics",5th edition,Pearson Education,2005

Websites:

1. **http://www.opengl.org**
2. http://www. Github.com
3. http://www.youtube.com
4. www.openglproject.in