Computer Graphics (UCS505)

Project on

Ship Under Bridge Simulation

Submitted By

Siddharth Banga	102103246
Ayush Kumar Singh	102103249
Paramdeep Singh Gill	102103258

B.E. Third Year – COE9

Submitted To:

Ms. Archana Kumari

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

Patiala – 147001

Table of Contents

Introduction

The "Simulation of Movable Bridge" project exemplifies the capabilities of OpenGL in creating dynamic hardware simulations. Through OpenGL functions, users interact with hardware components, observing their behavior on-screen. This project aims to provide an immersive demonstration of a movable bridge simulation, complemented by a bustling cityscape where cars navigate roads and planes soar through the skies. Users control the simulation by toggling its operation with the 'S' or 's' key, and refreshing it with 'B' or 'b'.

Simulation Process

Upon activation with 'S' or 's', the movable bridge simulation begins. The train signal turns red, and the bridge ascends, facilitating the ship's traversal. Meanwhile, cars navigate city roads, and planes grace the skies. After the ship completes its passage, the bridge resets, and the train signal switches to green, allowing the train to cross. Users can pause the simulation with 'S' or 's' and restart it with 'B' or 'b', providing dynamic control over the simulation's flow.

Overall, the "Simulation of Movable Bridge" project offers an immersive and interactive exploration of dynamic hardware simulations, seamlessly integrating user interaction with captivating visual elements. Through its richly detailed environment and intuitive controls, users are invited to embark on a captivating journey through the intricate dynamics of bridge operations within a bustling urban landscape.

Computer Graphics Concepts Used

OpenGL is a cross language, cross platform application programming interface for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit to achieve hardware-accelerated rendering. As a software interface for graphics hardware, OpenGL's main purpose is to render two and three-dimensional objects into a frame buffer. These objects are described as sequences of vertices (which define geometric objects) or pixels (which define images). OpenGL performs several processing steps on this data to convert it to pixels to form the final desired image in the frame buffer.

To draw a primitive on the screen first we call glBegin, specifying what kind of primitive it is that you want to draw in the mode parameter of glBegin, and then list all vertices one by one (by sequentially calling glVertex3f) and finally call glEnd to let OpenGL know that we're done drawing a primitive.

2D-Transformations:

Transformation is a process of modifying and re-positioning the existing graphics. 2D Transformations take place in a two-dimensional plane. Transformations are helpful in changing the position, size, orientation, shape etc. of the object.'

Translation Function: A translation process moves every point a constant distance in a specified direction. It can be described as a rigid motion. A translation can also be interpreted as the addition of a constant vector to every point, or as shifting the origin of the coordinate system. You can translate a point in 2D by adding the translation coordinate (tx, ty) to the original coordinate X, Y to get the new coordinate X′, Y′.

Scaling: It is used to alter or change the size of objects. The change is done using scaling factors. There are two scaling factors, i.e., Sx in x direction, Sy in y-direction. If the original position is x and y. Scaling factors are Sx and Sy then the value of coordinates after scaling will be x1 and y1.

Rotation Function: glRotatef is an OpenGL function used for rotating objects in a 3D space. It takes parameters for angle and axis of rotation, allowing precise control over the orientation of rendered objects. This function is essential for creating dynamic visual effects and animations in graphics applications.

Animation: It is a method, where objects are manipulated to appear as moving, without the user input. It requires a form of infinite loop, which is usually achieved by a set of functions or class methods that describe the changes to each object in a small unit of time- called update or move.

RGB: The RGB Color Model is used for color representation. It is a color coordinate system having three primary colors (red, green and blue), each primary color having its intensity value ranging from 0 to 1. Mixing these three primary colors at varying intensities produces a variety of colors.

User Defined Functions

Function Name	Description
void **boat**(int s, int x, int y);	Draws a boat
void **circle**(int r, int a, int b, float color[]);	Draws a filled circle
void **drawA**(int x, int y, int height);	Draws letter A
void **drawBannerWithName**(int s);	Draws a banner with name on it
void **drawBase**(int h,int w, int x1, int y1,int x, int y2, flaot frontColo[], float sideCol[], float topCol[]);	Draws a 3d Cuboid
void **drawCar**(float c1, float c2);	Drwas a Car.
void **drawCircle**(int r, int a, int b, float color[]);	Draws a circle outline
void **drawCirclRoads**();	Draws circular roads
void **drawCity**();	Draws a city scene
void **drawD**(int x, int y, int height);	Draws letter D
void **drawFerrisWheel()**;	Draws a ferris wheel
void **drawFront**(int h, int w, int x, int y, int currentColor[3]);	Draws a front face of 3d object
void **drawH**(int x, int y, int height);	Draws letter H
void **drawHeadLight**(int x, int y, float color[]);	Draws headlight of the car
void **drawI**(int x, int y, int height);	Draws letter I
void **drawLeftBackBridge**();	Draws the static left back side of the bridge
void **drawLeftFrontBridge**();	Draws the static left front side of the bridge
void **drawM(**int x, int y, int height);	Draws letter M
void **drawP**(int x, int y, int height);	Draws letter P
void **drawR**(int x, int y, int height);	Draws letter R
void **drawRightBAckBridge**()	Draws the static right back side of the bridge
void **drawRightFrontBridge**();	Draws the static right front side of the bridge
void **drawS**(int x, int y, int height);	Draws letter S
void **drawSupportBeams**(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4, float color[]);	Draws the support for the static left and right bridge

void **drawT**(int x, int y, int height);	Draws letter T
void drawTrain(int x);	Draws a train which can be translated by passing x;
void **drawTriangle**s(int x1,int y1, int x2, int y2, int x3, int y3, float currentColor[]);	Draws a triangle
void **drawU**(int x, int y, int height);	Draws letter U
void **drawVertLog**(int h, int a1, int b1, int w float side[], float front[], int front2[]);	Draws a vertical log which has four faces front, left-side, right-side, back-side
void **drawWheel**(int x, int y, int z);	Draws a wheel of the at the given coordinates
Void **drawY**(int x, int y, int height);	Draws letter Y
void **getSide**(int h, int x1, int y1, int x2, int y2, float color[]);	Draws a side face of the 3d object(cuboid)
void **getTop**(int w, int x1,int y1, int x1, int y2, float color[]);	Draws a top face of the 3d object(cuboid).
void **lightPole**();	Draws a light pole which signals red which bridge is up and green when bridge is down.
void **movingBackBridge**();	Draws the back part of the moving bridge
void **movingFrontBridge**();	Draws the front part of the moving bridge.
void **myDisplay**();	Display function which is given to render all the objects
void **myInit()**;	Initializes OpenGL settings
void **myKeyboard**(unsigned char key, int x, int y);	To get user input through the keyboard and perform some actions.
void **ocean**();	Draws a ocean
void **railTrack**();	Draws a rail track on railway line
void **railway**();	Draws a railway line on which train will run
void **iimer**(int value);	To update the movement values of bridge, ship, train and water waves.
void **writeAyush**(int startX,int startY,int height);	Draws name ayush
void **writeParam**(int startX,int startY,int height);	Draws name param
void **writeSiddharth**(int startX,int startY,int height);	Draws name siddharth

Code

```
#include <GL/glut.h>
#include <GL/gl.h>
#define GL_SILENCE_DEPRECATION
#include<iostream>
using namespace std;
float tmp[] = { 0.2,0.2,0.2 },tmp2[] = { 0,1,0 },bridgeY = 0.0,shipX=-50.0,shipY=-
50.0,bridgeWidth = 300.0,trainX=0.0,
bridgeColor[] = { 0.3, 0.2, 0.05 }, frontColor[] = { 0.7, 0.5, 0.2 }, supportColor[] =
{ 0.5, 0.5, 1 }, sideColor[] = {0.5,0.3,.2},
greyColor[] = { 0.4,0.4,0.4 }, redColor[] = {1,0,0},greenColor[]={0,1,0},
blackColor[]={0,0,0}, c1 = 0.0,c2=0.0,c3=0.0,c4=0,planeX=0,planeY=0;
bool isShipGone = false,isBridgeUp=false,start=false;
int sides = 360,pos1=90,pos2=45,pos3=270;
const float radius = 600.0f;
const float PI = 3.14159265f;



void myInit(void) {
    glClearColor(0,0.0,0.2, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0, 1500,00,1000,1500,-2000);//viewport
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}



void drawCircle(int r, int a, int b, float color[]) {
    glColor3f(color[0], color[1], color[2]);
    glBegin(GL_LINE_LOOP);
    for (int i = 0; i < sides; i++) {
        float angle = 2.0f * PI * i / sides;
        float x = a + r * cos(angle);
        float y = b + r * sin(angle);
        glVertex2f(x, y);
    }
    glEnd();
}
```

```
void circle( int r,int a,int b, float color[]) {
    glColor3f(color[0],color[1],color[2]);
    glBegin(GL_POLYGON);
    for (int i = 0; i < sides; i++) {
        float angle = 2.0f * PI * i / sides;
        float x =a+ r * cos(angle);
        float y =b+ r * sin(angle);
        glVertex2f(x, y);
    }
    glEnd();
}


void drawFront(int h, int w, int x, int y, float curentColor[3]) {
    glColor3f(curentColor[0], curentColor[1], curentColor[2]);
    glBegin(GL_QUADS);
    glVertex2f(x, y);
    glVertex2f(x + w, y);
    glVertex2f(x + w, y + h);
    glVertex2f(x, y + h);
    glEnd();
}


void getTop(int w, int x1, int y1, int x2, int y2,float color[]) {
    glColor3f(color[0],color[1],color[2]);
    glBegin(GL_QUADS);
    glVertex2f(x1, y1);
    glVertex2f(x2, y2);
    glVertex2f(x2 + w, y2);
    glVertex2f(x1 + w, y1);
    glEnd();

}


void getSide(int h, int x1, int y1, int x2, int y2,float color[]) {
    glColor3f(color[0], color[1], color[2]);
    glBegin(GL_QUADS);
    glVertex2f(x1, y1);
    glVertex2f(x2, y2);
    glVertex2f(x2, y2 + h);
    glVertex2f(x1, y1 + h);
    glEnd();

}
```

```
void drawBase(int h, int w, int x1, int y1, int x2, int y2,float frontCol[], float
sideCol[],float topCol[]) {
    glBegin(GL_QUADS);
    glColor3f(0, 0.5, 0.2);
    getTop(w, x1, y1 + h, x2, y2 + h,topCol);
    getSide(h, x1, y1, x2, y2,sideCol);
    drawFront(h, w, x1, y1, frontCol);
    glEnd();
}
```

```
void drawTriangles(int x1, int y1, int x2, int y2, int x3, int y3, float
curentColor[3]) {
    glColor3f(curentColor[0], curentColor[1], curentColor[2]);
    glBegin(GL_TRIANGLES);
    glVertex2f(x1, y1);
    glVertex2f(x2, y2);
    glVertex2f(x3, y3);
    glEnd();

}
```

```
void drawSupportBeams(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4,
float color[3]) {
    glColor3f(color[0], color[1], color[2]);
    glBegin(GL_QUADS);
    glVertex2f(x1, y1);
    glVertex2f(x2, y2);
    glVertex2f(x3, y3);
    glVertex2f(x4, y4);
    glEnd();
}
```

```
void drawVertLog(int h, int a1, int b1, int w, float side[],float front[],float
front2[]) {
    getSide(h, a1 + w - 3, b1 + 3, a1 + w, b1,side);
    drawFront(h, w, a1 - 3, b1 + 3, front2);
    getSide(h, a1 - 3, b1 + 3, a1, b1,side);
    drawFront(h, w, a1, b1, front);
}


void ocean() {
    glBegin(GL_QUADS);
    glColor3f(0.0, 0.4, 0.6);
    glVertex2f(0, 0);
    glVertex2f(1500, 0);
    glVertex2f(1500, 700);
    glVertex2f(0, 680);
    glEnd();

}




void railway() {
    getTop(650, 0, 250, -50, 280, bridgeColor);//left-top-base
    float col[] = { .4,0.4,0.4 }, col2[] = { 0.25,0.25,0.25 }, col3[] = { 0.6,0.6,0.6
};
    getTop(50, 600, 250, 550, 280, col);//left-top-concrete-base
    float front[] = { 0.56, 0.39, 0.05 };
    drawFront(60, 650, 0, 190, col2);//left-front-wall
    getTop(1000, 850, 250, 800, 280, bridgeColor);//right-top-base
    getTop(50, 850, 250, 800, 280, col);//right-top-concrete-base
    getSide(60, 850, 190, 800, 220,col3);//right-side-face
    drawFront(60, 1000, 850, 190, col2);//right-front-wall

}
```

```
void railTrack() {
    glLineWidth(2);

    glBegin(GL_LINES);
    for (int i = 0; i < 12; i++) {
        glColor3f(0, 0, 0);
            glVertex2f(i * 50 + 2, 255 );
            glVertex2f((i + 1) * 50, 255 );
            glVertex2f(i * 50 + 2 - 30, 275 );
            glVertex2f((i + 1) * 50 - 30, 275 );
            glColor3f(1, 1, 1);
            glVertex2f(i * 50 + 2, 255 );
            glVertex2f(i * 50 + 2 - 30, 275 );
    }
    for (int i = 0; i < 13; i++) {
        glColor3f(0, 0, 0);
        glVertex2f( 900+ i * 50 + 2, 255);
        glVertex2f(900 + (i + 1) * 50, 255);
        glVertex2f(900 + i * 50 + 2 - 30, 275);
        glVertex2f(900 + (i + 1) * 50 - 30, 275);
        glColor3f(1, 1, 1);
        glVertex2f(900 + i * 50 + 2, 255);
        glVertex2f(900 + i * 50 + 2 - 30, 275);
    }
    glColor3f(0, 0, 0);
    glVertex2f(600, 255);
    glVertex2f(645, 255);
    glVertex2f(550, 275);
    glVertex2f(610, 275);
    glColor3f(1, 1, 1);
    glVertex2f(590, 255);
    glVertex2f(560, 275);
    glVertex2f(645, 255);
    glVertex2f(610, 275);
    glColor3f(0, 0, 0);
    glVertex2f(840, 255);
    glVertex2f(900, 255);
    glVertex2f(810, 275);
    glVertex2f(900, 275);
    glColor3f(1, 1, 1);
    glVertex2f(840, 255);
    glVertex2f(810, 275);
    glEnd();
}
```

```
void boat(int s, int x, int y) {
    float col1[] = {0.7,0.7,0.7}, col2[] = { 0.647f, 0.165f, 0.165f }, col3[] =
{0.4,0.4,0.4};
    drawBase(25 * s, 55 * s, 180 * s + x, 200 * s + y, 130 * s + x, 250 * s +
y,col1,col3,col2);    //bottom-base
    drawBase(18 * s, 50 * s, 180 * s + x, 228 * s + y, 136 * s + x, 272 * s +
y,col1,col3,col2);//second-base
    drawBase(14 * s, 45 * s, 180 * s + x, 248 * s + y, 142 * s + x, 286 * s +
y,col1,col3,col2);//top-base
    //top-base-cuboid
    drawFront(70 * s, 20 * s, 162 * s + x, 290 * s + y,col3);
    col3[2] = 0.6;
    getSide(70 * s, 190 * s + x, 280 * s + y, 182 * s + x, 290 * s + y,col3);
    getSide(70 * s, 170 * s + x, 280 * s + y, 162 * s + x, 290 * s + y,col3);
    drawFront(70 * s, 20 * s, 170 * s + x, 280 * s + y, col1);
    //boat-front-design
    drawTriangles(235 * s + x, 200 * s + y, 235 * s + x, 225 * s + y, 265 * s + x, 218
* s + y, col1);
    float tmpColor[] = { 0,0,0 };
    drawTriangles(180 * s + x, 200 * s + y, 235 * s + x, 200 * s + y, 265 * s + x, 218
* s + y, tmpColor);
    drawTriangles(180 * s + x, 200 * s + y, 180 * s + x, 225 * s + y, 265 * s + x, 218
* s + y, col3);
}


void drawRightBackBridge() {
    tmp2[1] = 0.2;
    int a1 = 850,b1 = 252;
    drawSupportBeams(920, 280, 970, 280, 808, 525, 808, 545, supportColor);//side-
support
    a1 = 810;b1 = 276;
    drawVertLog(250, a1, b1, 15,sideColor,bridgeColor,tmp);//lower-big-log
    drawVertLog(40, 812, 540,15, sideColor, bridgeColor, tmp);//upper-smaller-log

}

void drawRightFrontBridge() {
    tmp2[1] = 0.2;
    int a1 = 850,b1 = 252;
    drawSupportBeams(1000, 250, 1055, 250, 848, 490, 848, 520, supportColor);//side-
support
    drawVertLog(250, a1, b1, 15, sideColor, bridgeColor, tmp);//lower-big-log
    drawVertLog(40, 850, 510, 15, sideColor, bridgeColor, tmp);//upper-smaller-log
    drawBase(20, 20, 855, 485, 785, 542, sideColor,frontColor,tmp);//log
}
```

```
void drawLeftBackBridge() {
    int a1 = 595,b1 = 276;
    drawVertLog(250, a1, b1, 15, sideColor, bridgeColor, tmp);//lower-big-log
    drawSupportBeams(420, 280, 475, 280, 592, 525, 592, 545, supportColor);//side-
support
    drawVertLog(40, 595, 540, 15, sideColor, bridgeColor, tmp);;//upper-smaller-log
}


void drawLeftFrontBridge() {
    float col[] = { .4,0.4,0.4 }, col2[] = { 0.25,0.25,0.25 }, col3[] = { 0.6,0.6,0.6
};
    int a1 = 633, b1 = 252;
    drawVertLog(40, 633, 510, 15, sideColor, bridgeColor, tmp);//upper-smaller-log
    a1 = 633; b1 = 252;
    drawVertLog(250, a1, b1, 15, sideColor, bridgeColor, tmp);//lower-big-log
    drawBase(20, 20, 635, 485, 570, 545,  sideColor,frontColor , tmp);//log-joining-
both-ver-log
    drawSupportBeams(470, 250, 525, 250, 631, 490, 631, 510, supportColor);//side-
support
}

void movingBackBridge() {
    float col[] = { .4,0.4,0.4 }, col2[] = { 0.25,0.25,0.25 }, col3[] = { 0.6,0.6,0.6
};
    drawBase(5, 202, 645, 250,605, 275, col2, col3, col);//railway-base
    //railway-track
    glBegin(GL_LINES);
    for (int i = 0; i < 4; i++) {
        glColor3f(0, 0, 0);
        glVertex2f(645+ i * 50 + 2, 255 + i * 0.01);
        glVertex2f(645+ (i + 1) * 50, 255 + (i + 1) * 0.01);
        glVertex2f(646+i * 50 + 2 - 30, 275 + i * 0.01);
        glVertex2f(646+ (i + 1) * 50 - 30, 275 + (i + 1) * 0.01);
        glColor3f(1, 1, 1);
        glVertex2f(645+ i * 50 + 2, 255 + i * 0.01);
        glVertex2f(645+ i * 50 + 2 - 30, 275 + i * 0.01);
    }
    //back-logs
    float  frontTmp[] = { 0.25, 0.0, 0.0 } ,sideTmp[] = { 0.5, 0.0, 0.0 };
    drawVertLog(150, 788, 275, 20, sideTmp, frontTmp, bridgeColor);
    drawBase(15, 160, 628, 410, 625, 413, col3, sideTmp, bridgeColor);
    drawVertLog(150, 608, 275, 20, sideTmp, frontTmp, bridgeColor);
    glEnd();
}
```

```
void movingFrontBridge() {
    float  frontTmp[] = { 0.25, 0.0, 0.0 }, col3[] = { 0.6,0.6,0.6 }, sideTmp[] = {
0.5, 0.0, 0.0 };
    drawVertLog(150, 828, 255, 20, sideTmp, frontTmp, bridgeColor);//left-front-
vertical-log
    drawBase(15, 160, 668, 390, 665, 393,col3, sideTmp, bridgeColor);//support-log
    drawVertLog(150, 648, 255, 20, sideTmp, frontTmp,bridgeColor);//right-vertical-log
}


void drawTrain(int x) {
    glPushMatrix();
    drawBase(70, 100, 1950 + x, 255, 1920 + x, 275, frontColor, sideColor,
tmp);//sixth-box
    drawFront(35, 40, 1960 + x, 280, blackColor);
    drawFront(35, 40, 2005 + x, 280, blackColor);
    drawBase(70, 100, 1840 + x, 255, 1810 + x, 275, frontColor, sideColor,
tmp);//fifth-box
    drawFront(35, 40, 1850 + x, 280, blackColor);
    drawFront(35, 40, 1895 + x, 280, blackColor);
    drawBase(70, 100, 1730 + x, 255, 1700 + x, 275, frontColor, sideColor,
tmp);//fourth-box
    drawFront(35, 40, 1740 + x, 280, blackColor);
    drawFront(35, 40, 1785 + x, 280, blackColor);
    drawBase(70, 100, 1620 + x, 255, 1590 + x, 275, frontColor, sideColor,
tmp);//third-box
    drawFront(35, 40, 1630 + x, 280, blackColor);
    drawFront(35, 40, 1675 + x, 280, blackColor);
    drawBase(70, 100, 1510 + x, 255, 1480 + x, 275, frontColor, sideColor,
tmp);//second-box
    drawFront(35, 40, 1520 + x, 280, blackColor);
    drawFront(35, 40, 1565 + x, 280, blackColor);
    drawBase(70, 100, 1400 + x, 255, 1370 + x, 275, frontColor, sideColor,
tmp);//first-box
    drawFront(35, 40, 1410 + x, 280, blackColor);
    drawFront(35, 40, 1455 + x, 280, blackColor);
    //train-engine
    drawFront(35, 60, 1340 + x, 255, redColor);
    getSide(35, 1340 + x, 255, 1305 + x, 275,sideColor);
    glColor3f(1, 1, 1);
    glBegin(GL_QUADS);
    glVertex2f(1340 + x, 290);
    glVertex2f(1305 + x, 310);
    glVertex2f(1370 + x, 345);
    glVertex2f(1400 + x, 325);
    glEnd();
    drawTriangles(1340 + x, 290, 1400 + x, 290, 1400 + x, 325, greyColor);
```

```
    glColor3f(0.7, 1.0, 1.0);
    glBegin(GL_QUADS);
    glVertex2f(1345 + x, 300);
    glVertex2f(1323 + x, 313);
    glVertex2f(1367 + x, 340);
    glVertex2f(1390 + x, 325);
    glEnd();
    glPopMatrix();
}


void drawCircleRoads() {
    float baseColor[] = { 0.3f, 0.3f, 0.3f };
    //lower-cricle-for-depth
    circle(radius, 0, -60, baseColor);
    float topColor[] = { 0.35f, 0.35f, 0.35f };
    //base-circle
    circle(radius, 0, 0, topColor);
    glColor3f(0, 0, 0);
    //base-border-for-road
    glBegin(GL_LINE_LOOP);
    for (int i = 0; i < sides; i++) {
        float angle = 2.0f * PI * i / sides;
        float x = radius * cos(angle);
        float y = radius * sin(angle);
        glVertex2f(x, y);

    }
    glEnd();
    //city-circle
    float cityGroundColor[] = { 0.6f, 0.6f, 0.6f };
    circle(radius - 130, 0, 0, cityGroundColor);
    //road-lines
    glColor3f(1.0f, 1.0f, 1.0f);
    glBegin(GL_LINES);
    sides = 600;
    for (int i = 0; i < sides; i++) {
        float angle = 2.0f * PI * i / sides;
        float x = (radius - 60) * cos(angle);
        float y = (radius - 60) * sin(angle);
        glVertex2f(x, y);
        i += 3;
    }
    glEnd();

}
```

```
void drawWheel(int x, int y,int z) {
    for (float i = -1.0; i <= 5.0; i += 0.1) {
        //right-back
        glPushMatrix();
        glRotatef(90, 0, 1, 0);
        glTranslatef(x, y, -i);
        circle(8, x, y, blackColor);
        glPopMatrix();
    //right-front
        glPushMatrix();
        glRotatef(90, 0, 1, 0);
        glTranslatef(x, y+45, -i);
        circle(8, x, y, blackColor);
        glPopMatrix();
        //left-front
        glPushMatrix();
        glRotatef(90, 0, 1, 0);
        glTranslatef(x, y + 45, 54-i);
        circle(8, x, y, blackColor);
        glPopMatrix();
        //left-back
        glPushMatrix();
        glRotatef(90, 0, 1, 0);
        glTranslatef(x, y, 54 - i);
        circle(8, x, y, blackColor);
        glPopMatrix();
    }
}



void drawHeadLight(int x, int y, float color[]) {
    glBegin(GL_POLYGON);
    glColor3f(color[0], color[1], color[2]);
    glVertex3f(x, y, -16);
    glVertex3f(x - 3, y, -14);
    glVertex3f(x - 3, y, -12);
    glVertex3f(x, y, -8);
    glVertex3f(x + 3, y, -12);
    glVertex3f(x + 3, y, -14);
    glEnd();
}
```

```
void drawCar(float c1,float c2) {
    glEnable(GL_DEPTH_TEST);
    glBegin(GL_QUADS);
    //right-side
    glColor3f(c1*0.5,c2*0.5,0);
    glVertex3f(50, 00, -5);
    glVertex3f(50, 0, -20);
    glVertex3f(50, 100, -20);
    glVertex3f(50, 100, -5);
    //back-panel-base
    glColor3f(c1*1, c2*1, 0);
    glVertex3f(0, 0, -20);
    glVertex3f(0, 100, -20);
    glVertex3f(50, 100, -20);
    glVertex3f(50, 0, -20);
    //left-side
    glColor3f(c1 * 0.5, c2 * 0.5, 0);
    glVertex3f(0, 0, -5);
    glVertex3f(0, 0, -20);
    glVertex3f(0, 100, -20);
    glVertex3f(0, 100, -5);
    //back-face
    glColor3f(c1 * 0.5, c2 * 0.5, 0);
    glVertex3f(0, 0, -5);
    glVertex3f(0, 0, -20);
    glVertex3f(50, 0, -20);
    glVertex3f(50, 0, -5);
    //front-face
    glColor3f(c1 * 0.4, c2 * 0.4, 0);
    glVertex3f(0, 100, -5);
    glVertex3f(0, 100, -20);
    glVertex3f(50, 100, -20);
    glVertex3f(50, 100, -5);
    glEnd();
    //left-window
    glBegin(GL_POLYGON);
    glColor3f(c1 * 0.2, c2 * 0.2, 0.2);
    glVertex3f(0, 10, -20);
    glVertex3f(4, 10, -25);
    glVertex3f(10, 20, -30);
    glVertex3f(10, 70, -30);
    glVertex3f(4, 80, -25);
    glVertex3f(0, 80, -20);
    glEnd();
```

```
    //right-window
    glBegin(GL_POLYGON);
    glVertex3f(50, 10, -20);
    glVertex3f(46, 10, -25);
    glVertex3f(40, 20, -30);
    glVertex3f(40, 70, -30);
    glVertex3f(46, 80, -25);
    glVertex3f(50, 80, -20);
    glEnd();
    //top
    glBegin(GL_POLYGON);
    glColor3f(c1 * 0.5, c2 * 0.5, 0);
    glVertex3f(10, 20, -30);
    glVertex3f(10, 70, -30);
    glVertex3f(40, 70, -30);
    glVertex3f(40, 20, -30);
    glEnd();
    //back-window
    glBegin(GL_POLYGON);
    glColor3f(0.7, 1.0, 1.0);
    glVertex3f(10, 20, -30);
    glVertex3f(40, 20, -30);
    glVertex3f(46, 10, -25);
    glVertex3f(4, 10, -25);
    glEnd();
    //front-window
    glBegin(GL_POLYGON);
    glColor3f(0.7, 1.0, 1.0);
    glVertex3f(10, 70, -30);
    glVertex3f(40, 70, -30);
    glVertex3f(46, 80, -25);
    glVertex3f(4, 80, -25);
    glEnd();
    //bonet-left-triangle
    glBegin(GL_POLYGON);
    glColor3f(c1 * 0.6, c2 * 0.6, 0);
    glVertex3f(0, 100, -20);
    glVertex3f(0, 80, -20);
    glVertex3f(4, 80, -25);
    glEnd();
    //bonet-right-triangle
    glBegin(GL_POLYGON);
    glVertex3f(46, 80, -25);
    glVertex3f(50, 80, -20);
    glVertex3f(50, 100, -20);
    glEnd();
    //bonet
```

```
    glBegin(GL_POLYGON);
    glColor3f(c1 * 1, c2 * 1, 0);
    glVertex3f(50, 100, -20);
    glVertex3f(0, 100, -20);
    glVertex3f(4, 80, -25);
    glVertex3f(46, 80, -25);
    glEnd();
    float headLightColor1[] = {1,1,1};
    float headLightColor2[] = {0.3,0.3,0.3};
    drawHeadLight(8, 101,headLightColor1);
    drawHeadLight(42, 101,headLightColor1);
    drawHeadLight(8, -1,headLightColor2);
    drawHeadLight(42, -1, headLightColor2);
    drawWheel(4,15,-1);
    glDisable(GL_DEPTH_TEST);

}


void drawCity() {
    float col1[] = { 0.3,0.3,0.3 }, col2[] = { 0.647f, 0.65f, 0.65f }, col3[] = {
0.4,0.4,0.4 };
    int x1 = 900, y1 = 1000;
    glPushMatrix();
    glRotatef(65.0f, 1.0f, 0.0f, 0.0f);// Rotate around the x-axis by 60 so that roads
looks 3d
    glTranslatef(1750, 1300, 0);//translate to the position
    drawCircleRoads();//roads
    //first-car-translation-and-rotation-logic
    glPushMatrix();
    glTranslatef((radius - 75)*cos(c1), (radius - 75) * sin(c1), 0);
    glPushMatrix();
    glRotatef(c1 * 180 / PI, 0, 0, 1);
    drawCar(1,0);
    glPopMatrix();
    glPopMatrix();
    //second-car-translation-and-rotation-logic
    glPushMatrix();
    glTranslatef(-(radius - 85) * cos(c2), (radius - 85) * sin(c2), 0);
    glPushMatrix();
    glRotatef(-c2 * 180 / PI, 0, 0, 1);
    drawCar(0.5,0.5);
    glPopMatrix();
    glPopMatrix();
    //second-car-translation-and-rotation-logic
    glPushMatrix();
    glTranslatef(-(radius - 85) * cos(c3), (radius - 85) * sin(c3), 0);
```

```
    glPushMatrix();
    glRotatef(-c3 * 180 / PI, 0, 0, 1);
    drawCar(0.7, 0.2);
    glPopMatrix();
    glPopMatrix();
    glPopMatrix();
    //buildings
    drawBase(400, 60, 1400, 560, 1380, 610, col3, col2, tmp);
    drawBase(200, 50, 1340, 500, 1320, 550, col3,col2, tmp);
    drawBase(250, 70, 1430, 430, 1410, 480, col3, col2, tmp);

}

void drawFerrisWheel() {
    //support-base
    drawBase(20, 170, 1030, 420, 980, 490, frontColor, sideColor, tmp);
    glPushMatrix();
    glRotatef(70.0f, 0.0f, 1.0f, 0.0f);//ferris-wheel-view-logic
    glPushMatrix();
    glTranslatef(900, 750, 800);//ferris-wheel-initial-position
    float baseColor[] = { 1,1,1 };
    float topColor[] = { 0.8, 0.8, 0.8 };
    glLineWidth(3);
    drawCircle(radius - 450, 50, -50, topColor);//first-cricle
    //ferris-wheel-carts
    glColor3f(1, 0, 0);
    for (int i = 0; i < sides; i++) {
        float angle = 2.0f * PI * i / sides;
        float x = 20 + (radius - 450) * cos(c4 + angle);
        float y = -20 + (radius - 450) * sin(c4 + angle);
        i += 78;
        drawFront(50, 50, x - 50, y - 50, tmp);
        drawFront(50, 50, x - 30, y - 30, redColor);
    }
    drawCircle(radius - 450, 0, -50, baseColor);//second-circle
    float curColor[] = { 0,0,0 };
    glPopMatrix();
    glPopMatrix();
}
```

```
void lightPole() {
    float col[] = { .4,0.4,0.4 }, col2[] = { 0.25,0.25,0.25 }, col3[] = { 0.6,0.6,0.6
};
    drawBase(150, 15, 1050, 275, 1045, 280,col2 , sideColor, blackColor);
    drawBase(80, 35, 1040, 425, 1035, 430, col3, sideColor, blackColor);
    glPushMatrix();
    glRotatef(30, 0, 1, 0);
    //red-light
    circle(15, 1223, 480, bridgeY == 0 ? greyColor : redColor);
    //green-light
    circle(15, 1223, 445,bridgeY!=0? greyColor : greenColor);
    glPopMatrix();
}

void drawA(int x, int y, int height) {
    glBegin(GL_LINE_STRIP);
    glVertex2f(x, y);
    glVertex2f(x, y + height);
    glVertex2f(x + height / 2, y + height);
    glVertex2f(x + height / 2, y + height / 2);
    glVertex2f(x, y + height / 2);
    glVertex2f(x + height / 2, y + height / 2);
    glVertex2f(x + height / 2, y);
    glEnd();
}


void drawD(int x, int y, int height) {
    glBegin(GL_LINE_STRIP);
    glVertex2f(x, y);
    glVertex2f(x + height / 2, y+height/10);
    glVertex2f(x + height / 2, y+9*height / 10);
    glVertex2f(x, y + height);
    glVertex2f(x, y );
    glEnd();
}

void drawH(int x, int y, int height) {
    glBegin(GL_LINE_STRIP);
    glVertex2f(x, y);
    glVertex2f(x, y + height);
    glVertex2f(x, y + height / 2);
    glVertex2f(x + height / 2, y + height / 2);
    glVertex2f(x + height / 2, y + height);
    glVertex2f(x + height / 2, y);
    glEnd();
}
```

```
void drawI(int x, int y, int height) {
    glBegin(GL_LINE_STRIP);
    glVertex2f(x + height / 4, y);
    glVertex2f(x + 3 * height / 4, y);
    glVertex2f(x + height / 2, y);
    glVertex2f(x + height / 2, y + height);
    glVertex2f(x + height / 4, y + height);
    glVertex2f(x + 3 * height / 4, y + height);
    glEnd();
}

void drawM(int x, int y, int height) {
    glBegin(GL_LINE_STRIP);
    glVertex2f(x, y);
    glVertex2f(x, y + height);
    glVertex2f(x + height / 4, y + 5 * height / 8);
    glVertex2f(x + 2 * height / 4, y + height);
    glVertex2f(x + 2 * height / 4, y);
    glEnd();

}

void drawP(int x, int y, int height) {
    glBegin(GL_LINE_STRIP);
    glVertex2f(x, y);
    glVertex2f(x, y + height);
    glVertex2f(x + height / 2, y + height);
    glVertex2f(x + height / 2, y + height / 2);
    glVertex2f(x, y + height / 2);
    glEnd();

}

void drawR(int x, int y, int height) {
    glBegin(GL_LINE_STRIP);
    glVertex2f(x, y);
    glVertex2f(x, y + height);
    glVertex2f(x + height / 2, y+height);
    glVertex2f(x + height / 2, y + height / 2);
    glVertex2f(x, y + height/2);
    glVertex2f(x + height / 2, y );
    glEnd();

}
```

```
void drawS(int x, int y, int height) {
    glBegin(GL_LINE_STRIP);
    glVertex2f(x + height / 2, y + height);
    glVertex2f(x, y + height);
    glVertex2f(x, y + height / 2);
    glVertex2f(x + height / 2, y + height / 2);
    glVertex2f(x + height / 2, y);
    glVertex2f(x, y);
    glEnd();
}

void drawT(int x, int y, int height) {
    glBegin(GL_LINE_STRIP);
    glVertex2f(x, y + height);
    glVertex2f(x + height, y + height);
    glEnd();
    glBegin(GL_LINE_STRIP);
    glVertex2f(x + height / 2, y);
    glVertex2f(x + height / 2, y + height);
    glEnd();
}


void drawU(int x, int y, int height) {
    glBegin(GL_LINE_STRIP);
    glVertex2f(x, y + height);
    glVertex2f(x, y);
    glVertex2f(x + height / 2, y);
    glVertex2f(x + height / 2, y + height);
    glEnd();
}


void drawY(int x, int y, int height) {
    glBegin(GL_LINE_STRIP);
    glVertex2f(x + height / 2, y);
    glVertex2f(x + height / 2, y + height / 2);
    glVertex2f(x + height / 4, y + height);
    glVertex2f(x + height / 2, y + height / 2);
    glVertex2f(x + 3 * height / 4, y + height);
    glEnd();
}
```

```
void writeSiddharth(int startX, int startY, int height) {
    drawS(startX, startY, height);
    drawI(startX + 0.5 * height, startY, height);
    drawD(startX + 1.5 * (height), startY, height);
    drawD(startX + 2.5 * (height), startY, height);
    drawH(startX + 3.5 * (height), startY, height);
    drawA(startX + 4.5 * (height), startY, height);
    drawR(startX + 5.5 * (height), startY, height);
    drawT(startX + 6.2 * (height), startY, height);
    drawH(startX + 7.4* (height), startY+2, height);


}

void writeAyush(int startX, int startY, int height) {
    drawA(startX, startY, height);
    drawY(startX + 0.5 * height, startY, height);
    drawU(startX + 1.5 * (height), startY, height);
    drawS(startX + 2.5 * (height), startY, height);
    drawH(startX + 3.5 * (height), startY, height);
}


void writeParam(int startX, int startY, int height) {
    drawP(startX, startY, height);
    drawA(startX + 1* height, startY, height);
    drawR(startX + 2 * (height), startY, height);
    drawA(startX + 3 * (height), startY, height);
    drawM(startX + 4 * (height), startY, height);
}


void drawBannerWithName(int s) {
    glBegin(GL_QUADS);
    glColor3f(1, 1, 1);
    glVertex2f(25 * s, 15 * s);
    glVertex2f(25 * s, 36 * s);
    glVertex2f(230 * s, 40 * s);
    glVertex2f(230 * s, 20 * s);
    glEnd();
    glColor3f(0.8, 0.4, 0.1);
    writeAyush(65,40 ,20);
    writeSiddharth(170, 43, 20);
    writeParam(350, 46, 20);
}
```

```
void drawPlane(int s) {
    float sideCol[] = { 0.9,0.4,0 };
    getSide(3 * s, -33 * s, 17 * s, -40 * s, 32 * s,sideCol);
    glBegin(GL_QUADS);
    //right-front-wings
    glColor3f(1, 0, 0);
    glVertex2f(-33 * s, 20 * s);
    glVertex2f(-40 * s, 35 * s);
    glVertex2f(-35 * s, 35 * s);
    glVertex2f(0 * s, 20 * s);
    //plane-side-panel
    glColor3f(0.3, 0, 0);
    glVertex2f(-40*s, 12*s);
    glVertex2f(-40*s, 22*s);
    glVertex2f(23*s, 27*s);
    glVertex2f(23*s,23*s );
    //plane-top-panel
    glColor3f(0.6, 0, 0);
    glVertex2f(-40*s, 22*s);
    glVertex2f(-45*s, 25*s);
    glVertex2f(18*s, 28*s);
    glVertex2f(23*s, 27*s);
    glColor3f(0.4,0, 0);//plane-engline
    glVertex2f(-40 * s, 12 * s);
    glVertex2f(-40 * s, 22 * s);
    glVertex2f(-49 * s, 17 * s);
    glVertex2f(-49 * s, 13 * s);
    glColor3f(0.9, 0, 0);
    glVertex2f(-49 * s, 13 * s);
    glVertex2f(-49 * s, 17 * s);
    glVertex2f(-52 * s, 20 * s);
    glVertex2f(-52 * s, 16 * s);
    glColor3f(0.9, 0.9, 0.9);
    glVertex2f(-49 * s, 17 * s);
    glVertex2f(-40 * s, 22 * s);
    glVertex2f(-45 * s, 25 * s);
    glVertex2f(-52 * s, 20 * s);//left-front-wing
    glColor3f(1, 0, 0);
    glVertex2f(0*s, 0*s);
    glVertex2f(-30*s, 20*s);
    glVertex2f(-10*s, 20 * s);
    glVertex2f(05*s, 0*s);
    glEnd();
    drawBannerWithName(s);//banner-with-name
    //left-back-wing
    glBegin(GL_QUADS);
    glColor3f(1, 0, 0);
```

```
    glVertex2f(25* s, 15 * s);
    glVertex2f(10 * s, 25 * s);
    glVertex2f(20 * s, 25 * s);
    glVertex2f(28 * s, 15 * s);
    glColor3f(1, 0, 0);//back-top-wing
    glVertex2f(15 * s, 27 * s);
    glVertex2f(22 * s,  27* s);
    glVertex2f(22 * s, 40 * s);
    glVertex2f(19 * s, 40 * s);
    glEnd();
    //left-wing-depth
    drawFront(3*s, 5*s, 0*s, -3*s, supportColor);
    getSide(3*s, 0*s, -3*s,-30*s, 17*s,sideCol);
}


void myDisplay(void) {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glColor3f(0.0f, 1, 1);
    glPushMatrix();
    ocean();//ocean
    glPushMatrix();
    glTranslatef(1700-planeX, 1000-planeY, 0);
    drawPlane(2);//plane
    glPopMatrix();
    railway();//rail
    railTrack();
    glPopMatrix();
    //rail-static-bridge-right-portion
    drawRightBackBridge();
    if(shipX<=600)
    drawRightFrontBridge();
    //boat-translation-logic
    glPushMatrix();
    glTranslatef(shipX, -shipY, 0);
    boat(1.75, 230 ,300);
    glPopMatrix();
    //rail-moving-bridge-back-portion
    glPushMatrix();
    glTranslatef(0, bridgeY, 0);
    movingBackBridge();
    glPopMatrix();
    //rail-static-bridge-left-back-portion
    drawLeftBackBridge();
    //city-with-cars-and-building
    drawCity();
    drawFerrisWheel();
```

```
    lightPole();
    //train-with-engine
    drawTrain(200+trainX);
    //rail-static-bridge-right-front-portion
    if (shipX >= 600)
        drawRightFrontBridge();
    //rail-moving-bridge-front-portion
    glPushMatrix();
    glTranslatef(0, bridgeY, 0);
    movingFrontBridge();
    glPopMatrix();
    //rail-static-bridge-left-front-portion
    drawLeftFrontBridge();
    glPopMatrix();
    glFlush();
}

void timer(int value) {
    if (start) {
        c1 = 2 * PI * pos1 / sides;
        c2 = 2 * PI * pos2 / sides;
        c3 = 2 * PI * pos3 / sides;
        c4 = 2 * PI * pos2 / sides;
        pos1++;
        pos2 += 2;
        pos3 += 3;
        planeX += 3;
        planeY += 0.5;
        if (planeX > 2500) {
            planeX = 0;
            planeY = 0;
        }
        if (pos1 == sides) {
            pos1 = 0;
        }
        if (pos2 == sides) {
            pos2 = 0;
        }
        if (pos3 == sides) {
            pos3 = 0;
        }
        if (isBridgeUp) {
            if (shipX <= 1500) {
                shipX += 4;
                shipY += 4;
                if (shipX >= 360) {
                    isShipGone = true;
```

```
                }

            }
            else {
                shipX = -50;
                shipY = -50;
                bridgeY = 0;
                trainX = 0;
                isShipGone = false;
                isBridgeUp = false;
            }
        }
        if (isShipGone) {
            bridgeY -= 3;
            if (bridgeY - 3 <= 0) {
                bridgeY = 0;
                trainX -= 12;
            }
        }
        else if (bridgeY < 250) {
            bridgeY += 5;
        }
        else {
            isBridgeUp = true;
        }

    }
    glutPostRedisplay();
    glutTimerFunc(50, timer, 0);
}


void myKeyboard(unsigned char key, int x, int y)
{
    switch (key)
    {
    case 66: // B
        shipX = -50;
        shipY = -50;
        bridgeY = 0;
        trainX = 0;
        isShipGone = false;
        isBridgeUp = false;
        planeX = 0;
        planeY = 0;
        break;
    case 98: // b
```

```
        shipX = -50;
        shipY = -50;
        bridgeY = 0;
        trainX = 0;
        isShipGone = false;
        isBridgeUp = false;
        planeX = 0;
        planeY = 0;
        break;
    case 83:
        start=!start; // S
        break;
    case 115:
        start = !start; // s
        break;
    default:
        break;
    }
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(1500, 1000);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Ship Under Bridge Simulation");
    myInit();
    glutDisplayFunc(myDisplay);
    glutKeyboardFunc(myKeyboard);
    glutTimerFunc(50, timer, 0);
    glutMainLoop();

}
```

Screenshots

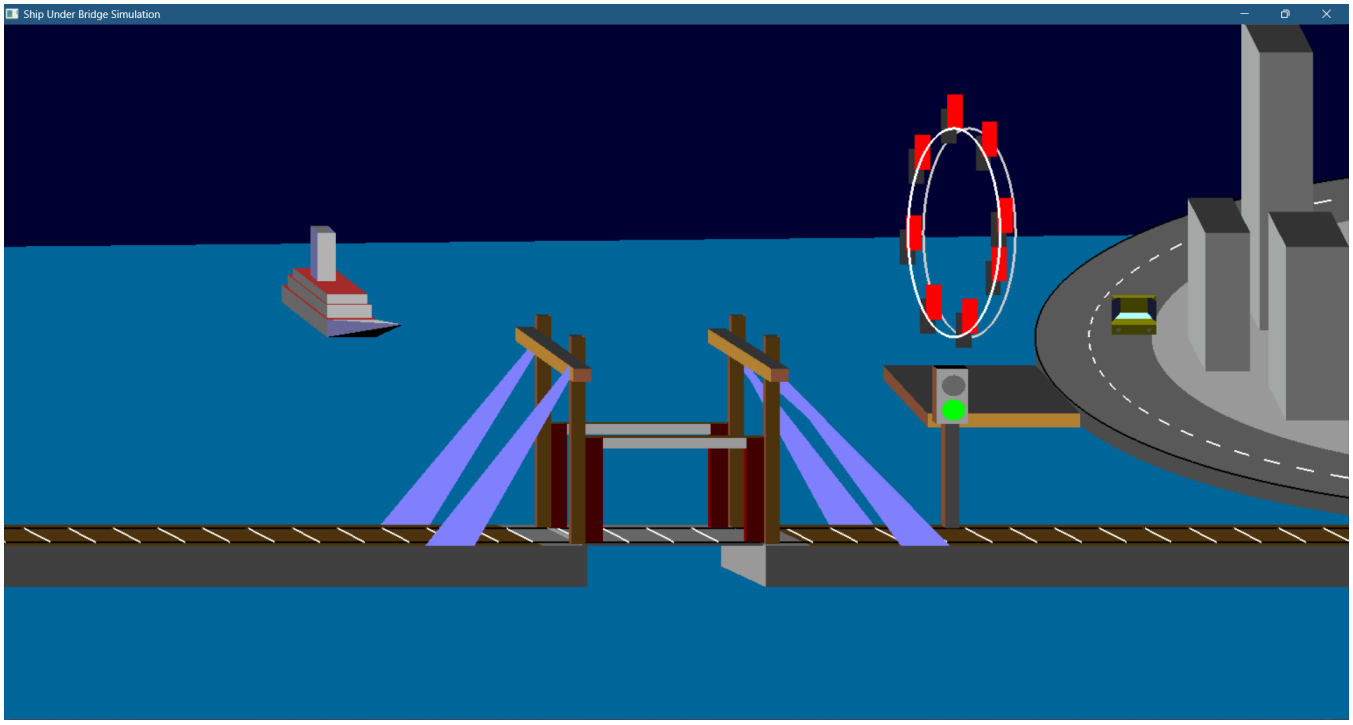

Fig1.1 Boat is at start position and bridge is down and after that bridge will move up

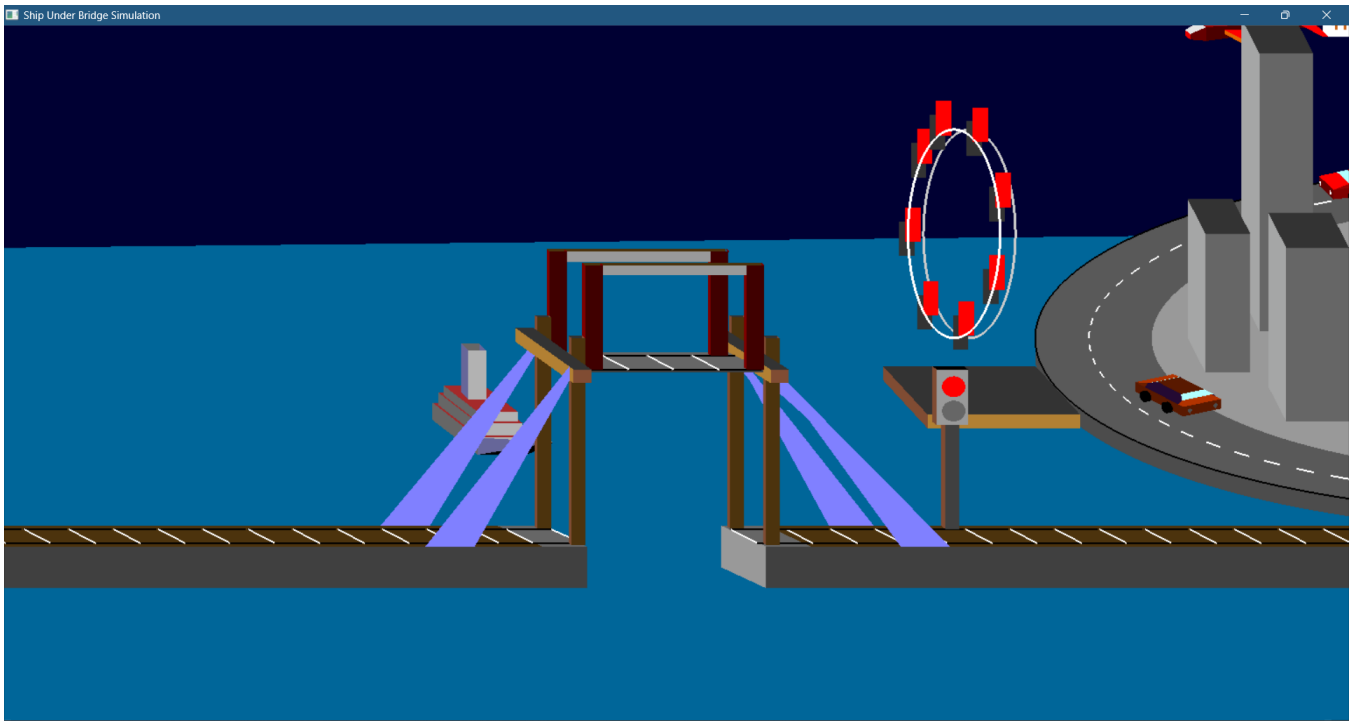

Fig 1.2 Boat starts moving under the bridge and the bridge is up and the signal is red

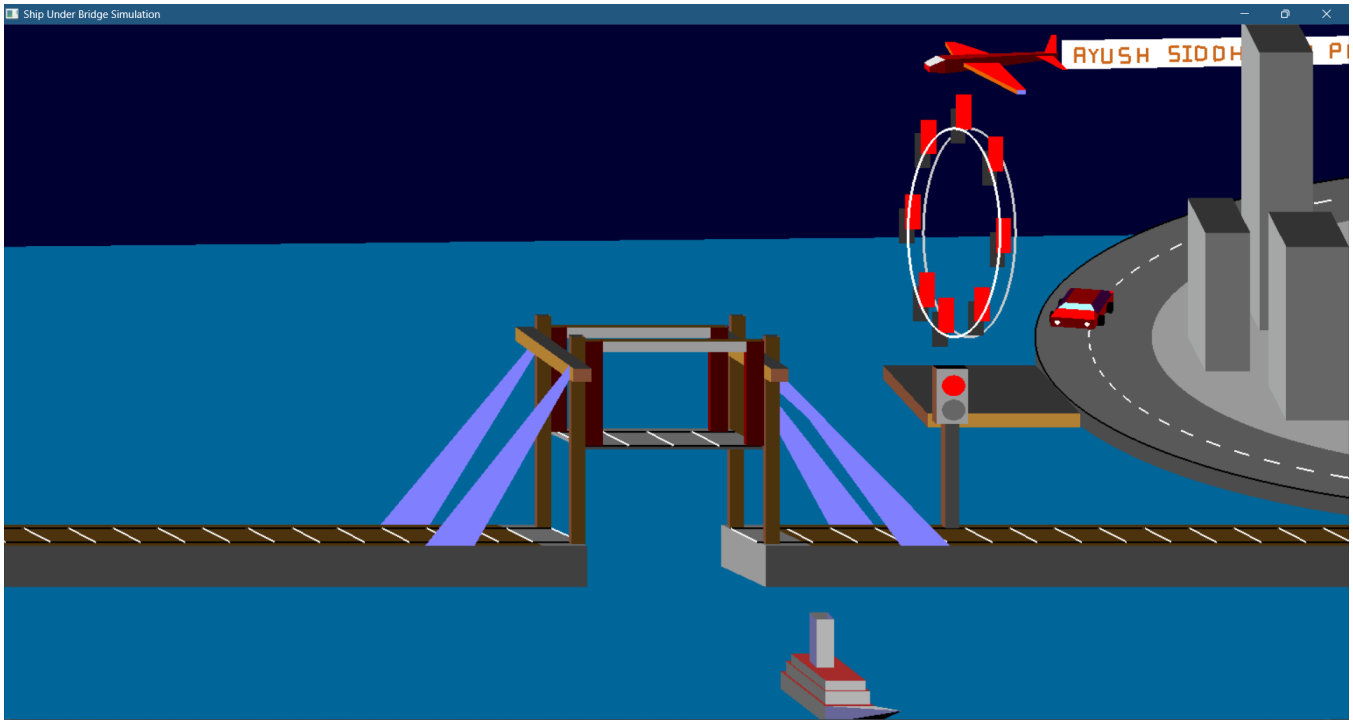

Fig 1.3 Boat crossed the bridge and now the bridge starts moving down again

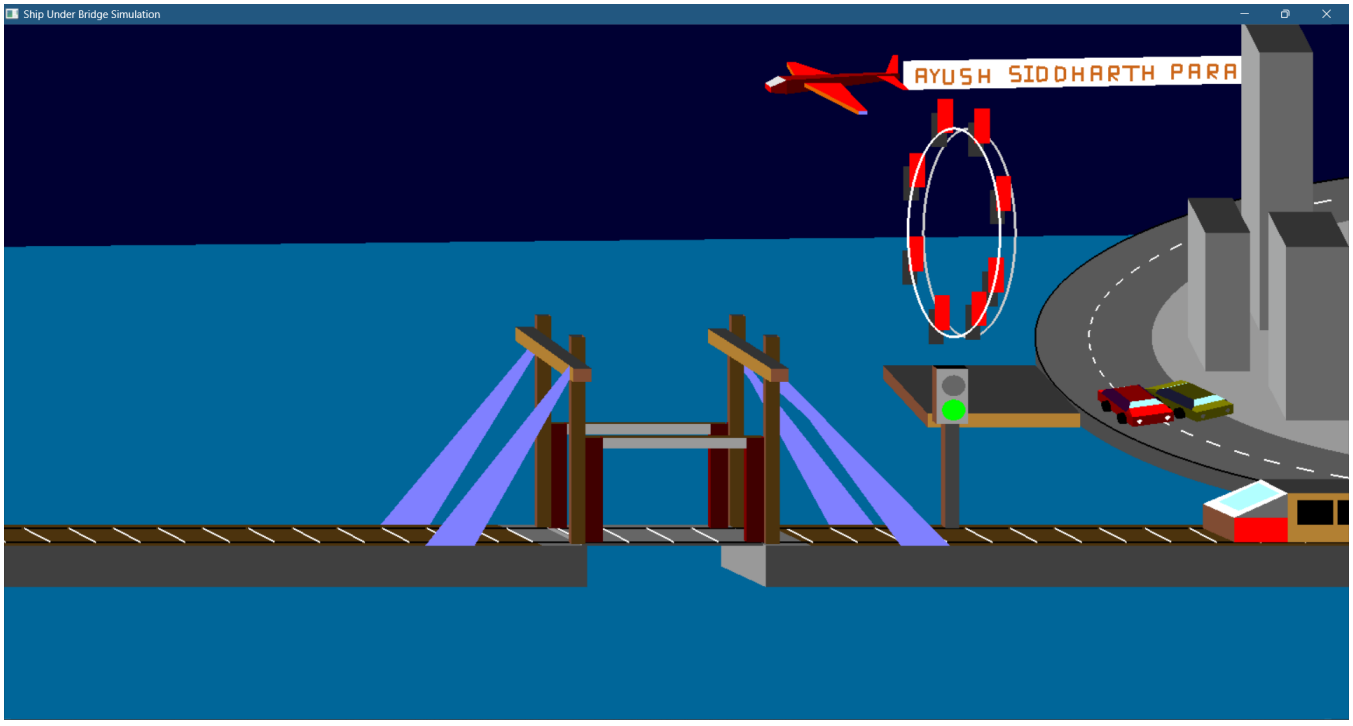

Fig 1.4 Train has arrived and the bridge is down, now train will move

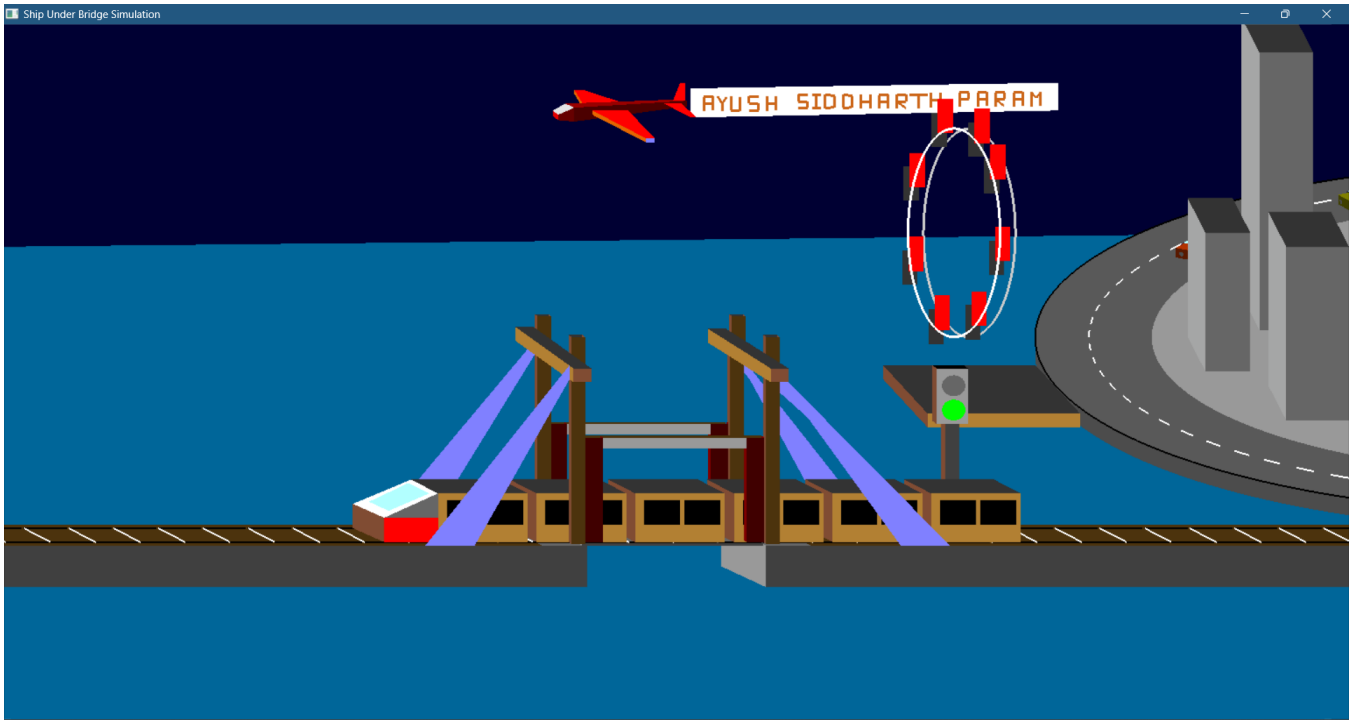

Fig 1.5 Train is crossing the bridge

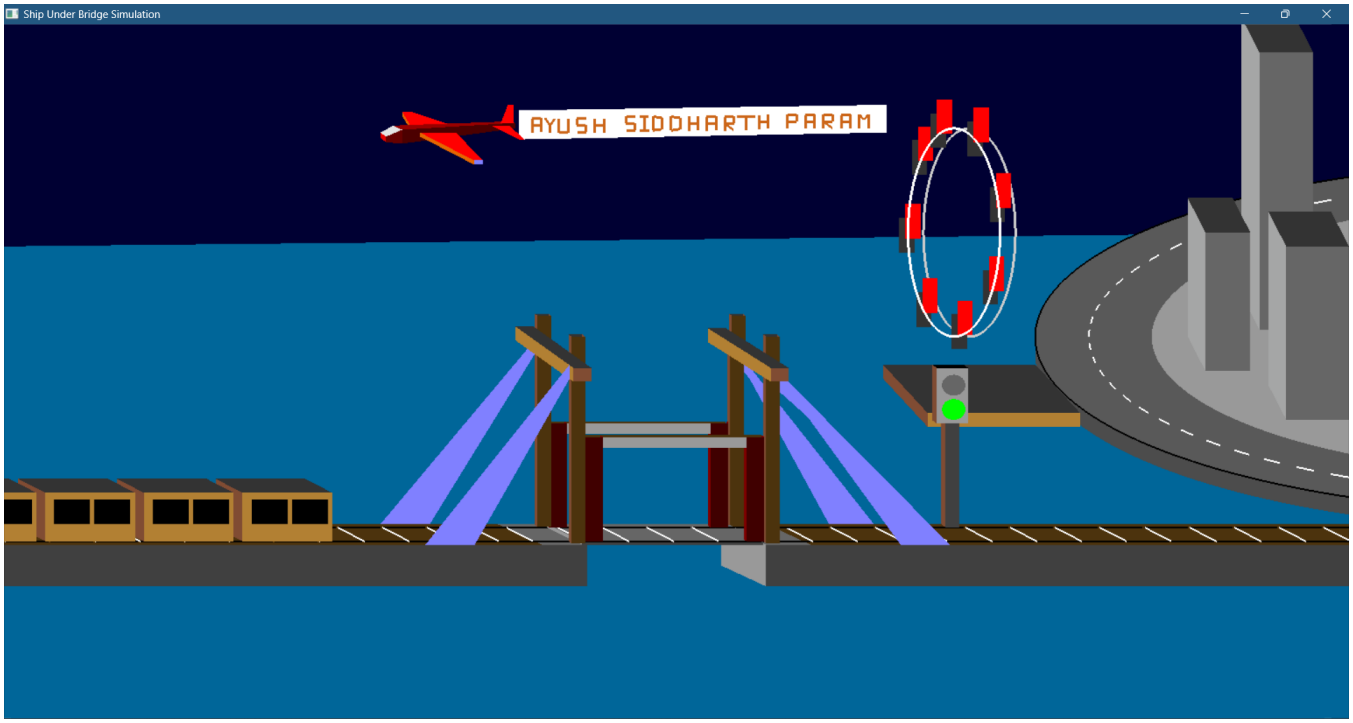

Fig 1.6 Train crossed the bridge

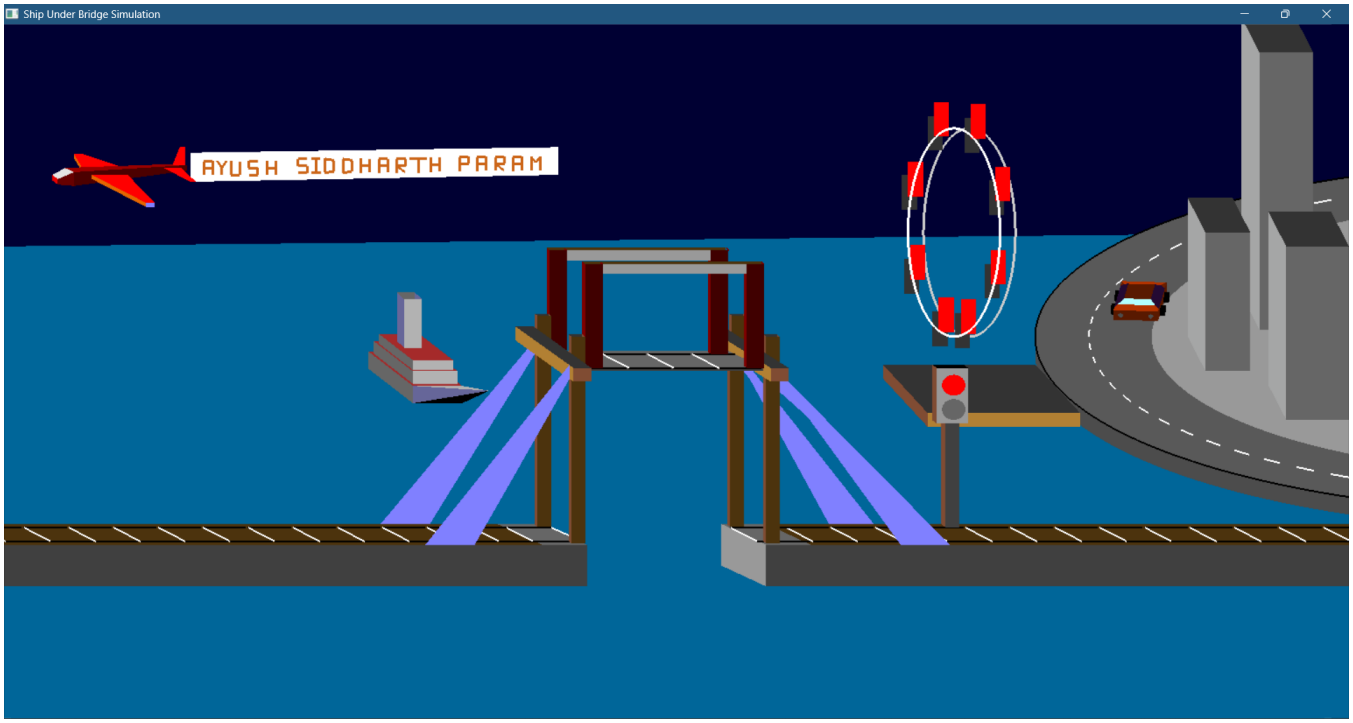

Fig 1.7 There is a city on right side in which cars are moving, a ferris wheel which is continuously rotating, a plane which is moving from right to left representing the name of team members.

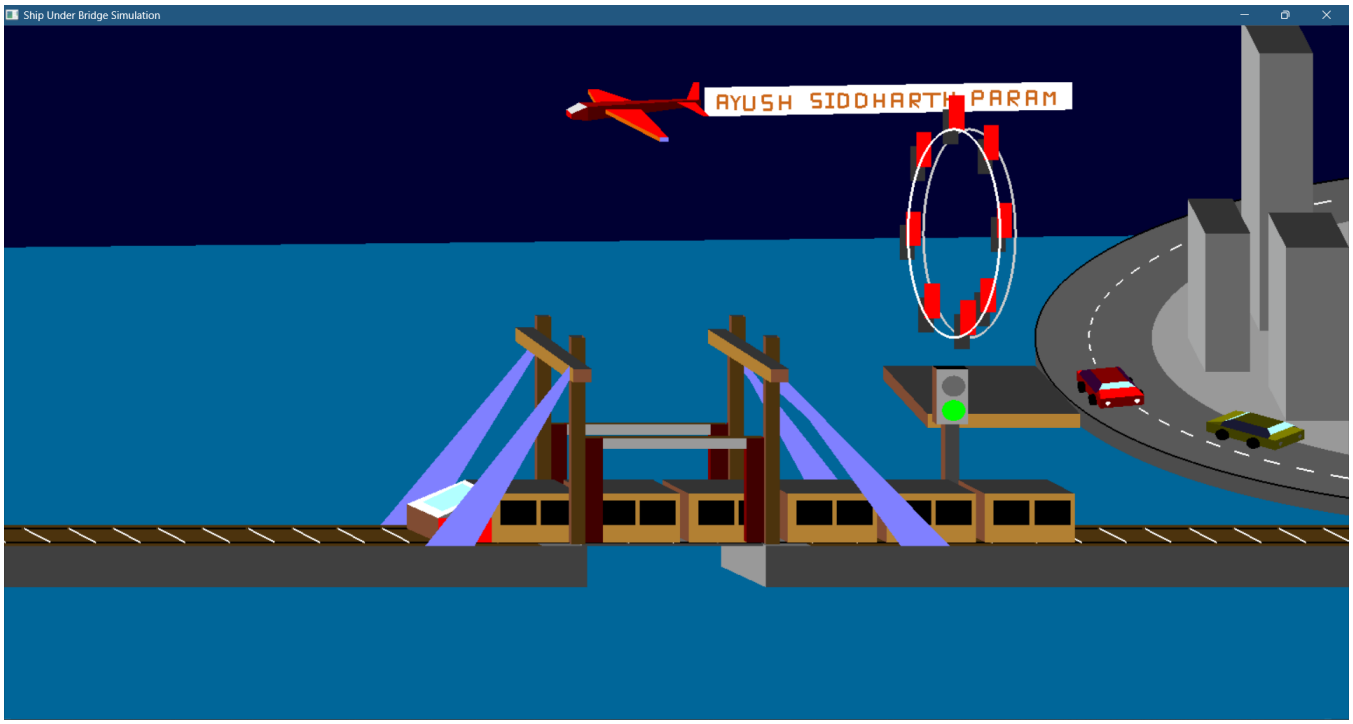

Fig 1.8 This simulation goes on forever.