BULLET TRAIN SIMULATION

CG MINI PROJECT REPORT SUBMITTED BY

Harina Manoj
4NM17CS067
VII Semester, B Section

Harshit Shirsat
4NM17CS068
VII Semester, B section

JP Ritwik
4NM17CS069
VII Semester, B Section

Jasmine Glani Mathias
4NM17CS070
VII Semester, B section

UNDER THE GUIDANCE OF

Mr. Pradeep Kanchan
Assistant professor Gd. III
Department of Computer Science and Engineering

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

Bachelor of Engineering in Computer Science & Engineering

from

Visvesvaraya Technological University, Belagavi

N.M.A.M. INSTITUTE OF TECHNOLOGY

(An Autonomous Institution under VTU, Belgaum)
AICTE approved, (ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC
NITTE -574 110, Udupi District, KARNATAKA.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B.E. CSE Program Accredited by NBA, New Delhi from 1-7-2018 to 30-6-2021

NITTE
EDUCATION TRUST

N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India

(ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC
☎: 08258 - 281039 – 281263, Fax: 08258 – 281265

Department of Computer Science and Engineering

B.E. CSE Program Accredited by NBA, New Delhi from 1-7-2018 to 30-6-2021

CERTIFICATE

"Bullet Train Simulation" is a bonafide work carried out by Harina Manoj(4NM17CS067), Harshit Shirsat (4NM17CS068),JP Ritwik (4NM17CS069),Jasmine Glani Mathias (4NM17CS070) in partial fulfilment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering prescribed by Visvesvaraya Technological University, Belagavi during the year 2020- 2021.

It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report. The Mini project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

Signature of Guide

Signature of HOD

ACKNOWLEDGEMENT

We believe that our project will be complete only after we thank the people who have contributed to make this project successful.

First and foremost, our sincere thanks to our beloved principal, Dr. Niranjan N. Chiplunkar for giving us an opportunity to carry out our project work at our college and providing us with all the needed facilities.

We sincerely thank Dr. K.R. Udaya Kumar Reddy, Head of Department of Computer Science and Engineering, Nitte Mahalinga Adyantaya Memorial Institute of Technology, Nitte.

We express our deep sense of gratitude and indebtedness to our guide Mr. Pradeep Kanchan, Assistant Professor Gd. III, Department of Computer Science and Engineering, for his inspiring guidance, constant encouragement, support and suggestions for improvement during the course of our project.

We thank all the teaching and non-teaching staff members of the Computer Science and Engineering Department and our parents and friends for their honest opinions and suggestions throughout the course of our project.

Finally, we thank all those who have supported us directly or indirectly throughout the project and making it a grand success.

Harina Manoj
(4NM17CS067)

Harshit Shirsat
(4NM17CS068)

JP Ritwik
(4NM17CS069)

Jasmine Glani Mathias
(4NM17CS070)

ABSTRACT

In this project we use computer graphics to draw and simulate the running of a bullet train using OpenGL. The main theme behind the project is to use the basic concepts of the computer graphics to draw a package from the scratch. Concepts involved in this project are polygon drawing, color filling, the movement of objects and Bullet Train Simulation. What we will learn from this project is how to build a package from scratch and basics of computer graphics by programming in OpenGL. OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. This offer functions to create and manipulate render lighting, coloring, viewing the models.

This project also allows the user to change the views and do further functions with the movement of the train and other activities. Also deals with design and implementation inbuilt graphics functions and hardware and software requirement with results obtained and concludes the project.

Table of Contents

INTRODUCTION

1.1 Computer Graphics

To draw a picture, say a fish moving inside the water. Suddenly will get an idea to use paint, but the degree of accuracy, quality of image is not satisfied and become very sad. There is no need to worry, for every problem there will be a solution, so this problem of creating fish moving inside the water can be solved using COMPUTER GRAPHICS without any difficulties.

Computer Graphics become a powerful tool for the rapid and economical production of pictures. There is virtually no area in which Graphical displays cannot be used to some advantage so it is not surprising to find the use of CG so widespread.

Although early application in engineering & science had to rely on expensive & cumbersome equipment, advances in computer technology have made interactive computer graphics a practical tool.

Computer Graphics in a diverse area such as science, engineering, medicine, business, industry, government, art, entertainment, education and training.

Now it can be answered about computer graphics as generalized tool for drawing and creating pictures and simulates the real-world situations within a small computer window.

1.2 Applications of computer Graphics

Nowadays Computer Graphics used in almost all the areas ranges from science, engineering, medicine, business, industry, government, art, entertainment, education and training.

1.2.1 CG in the field of CAD

Computer Aided Design methods are routinely used in the design of buildings, automobiles, aircraft, watercraft, spacecraft computers, textiles and many other applications.

1.2.2 CG in presentation Graphics

Another major application area presentation graphics used to produce illustrations for reports or generate slides. Presentation graphics is commonly used to summarize financial, statistical, mathematical, scientific data for research reports and other types of reports.2D and 3D bar chart to illustrate some mathematical or statistical report.

1.2.3 CG in computer Art

CG methods are widely used in both fine art and commercial art applications. Artists use a variety of computer methods including special purpose hardware, artist's paintbrush program (lumena), other pain packages, desktop packages, maths packages, animation packages that provide facility for designing object motion. Ex: cartoons design is an example of computer art which uses CG.

1.2.4 Entertainment

Computer graphics methods are now commonly used in making motion pictures, music, videos, games and sounds. Sometimes graphics objects are combined with the actors and live scenes.

1.2.5 Education and Training

Computer generated models of physical financial, economic system is often as education aids. For some training application special systems are designed. Ex: specialized system is simulator for practice sessions or training of ship captain, aircraft pilots and traffic control.

1.2.6 Image Processing

Although the methods used in CG image processing overlap, the 2 areas are concerned with fundamentally different operations. In CG a computer is used to create picture. Image processing on the other hand applies techniques to modify existing pictures such as photo scans, TV scans.

1.2.7 User Interface

It is common for software packages to provide a graphical interface. A major component of a graphical interface is a window manager that allows a user to display multiple window area. Interface also displays menus, icons for fast selection and processing.

1.3 Open GL

OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation. It is also used in video games, where it competes with Direct3D on Microsoft Windows platforms (see OpenGL vs. Direct3D). OpenGL is managed by a non-profit technology consortium, the Khronos Group.

Most of the application will be designed to access OpenGL directly through functions in three libraries. Functions in the main GL (or OpenGL in windows) library have names that begin with the letters gl and are stored in a library usually referred to as GL (or OpenGL in windows). The second is the **OpenGL Utility Library** (GLU). This library uses only GL functions but contains code for creating common objects and simplifying viewing. Allfunctions in GLU can be created from the core GL library but application programmers prefer not to write the code repeatedly. The GLU library is available in all OpenGL implementations; functions in the GLU library begin with letters glu.

To interface with the window system and to get input from external devices into the programs, need at least one more system-specific library that provides the “glue” between the window system and OpenGL. For the X window system, this library is functionality that should be expected in any modern windowing system.

OpenGL serves two main purposes:

1. Hide complexities of interfacing with different 3D accelerators by presenting a single, uniform interface.
2. Hide differing capabilities of hardware platforms by requiring support of full OpenGL feature set for all implementations (using software emulation if necessary).
OpenGL's basic operation is to accept primitives such as points, lines and polygons, and convert them into pixels. This is done by a graphics pipeline known as the OpenGL state machine.

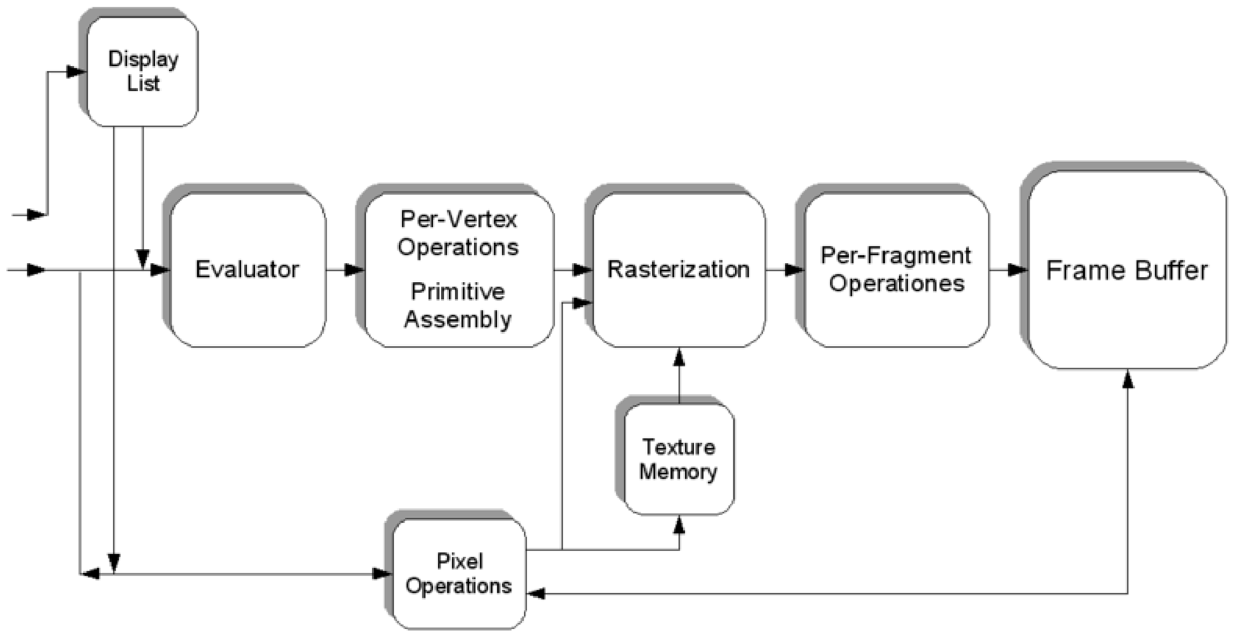


Figure 1.1 OpenGL Graphics Pipeline

A brief description of the process in the graphics pipeline could be:
1. Evaluation, if necessary, of the polynomial functions which define certain inputs, like NURBS surfaces, approximating curves and the surface geometry.
2. Vertex operations, transforming and lighting them depending on their material. Also clipping non visible parts of the scene in order to produce the viewing volume.
3. Rasterization or conversion of the previous information into pixels. The polygons are represented by the appropriate color by means of interpolation algorithms.
4.Per-fragment operations, like updating values depending on incoming and previously stored depth values, or color combinations, among others.
5. Lastly, fragments are inserted into the Frame buffer.

Most OpenGL commands either issue primitives to the graphics pipeline, or configure how the pipeline processes these primitives. OpenGL is a low-level, procedural API, requiring the programmer to dictate the exact steps required to render a scene. This contrasts with descriptive (aka scene graph or retained mode) APIs, where a programmer only needs to describe a scene and can let the library manage the details of rendering it. OpenGL's low-level design requires programmers to have a good knowledge of the graphics pipeline, but also gives a certain amount of freedom to implement novel rendering algorithms.

Fig 1.2 shows the organization of the libraries for an X Window System environment. For this window system, GLUT will use GLX and the X libraries. The application program, however, can use only GLUT functions and thus can be recompiled with the GLUT library for other window systems.

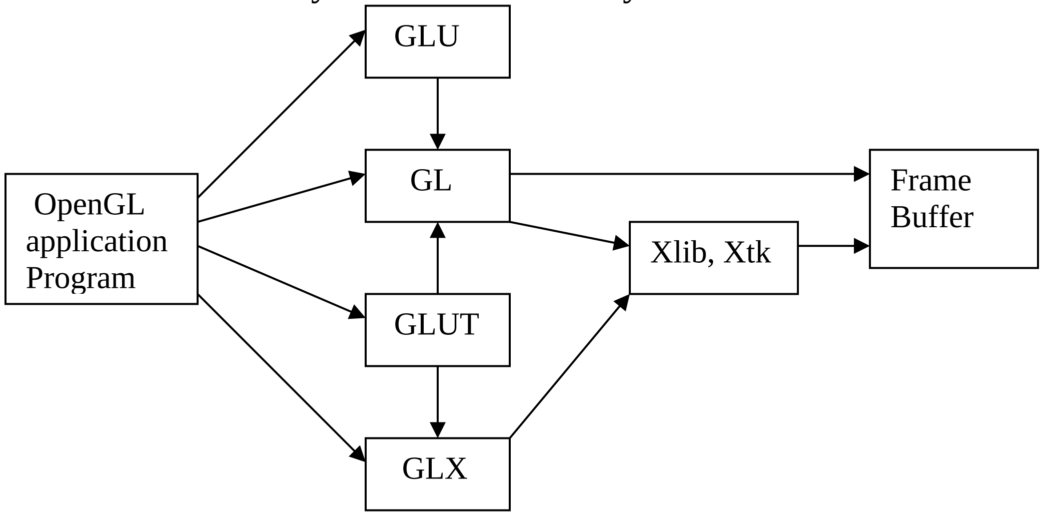


Fig 1.2 Library organization of OpenGL

1.4.1 Advantages of OpenGL

1. Industry standard

An independent consortium, the OpenGL Architecture Review Board, guides the OpenGL specification. With broad industry support, OpenGL is the only truly open, vendor-neutral, multiplatform graphics standard.

2. Stable

OpenGL implementations have been available for more than seven years on a wide variety of platforms. Additions to the specification are well controlled, and proposed updates are announced in time for developers to adopt changes. Backward compatibility requirements ensure that existing applications do not become obsolete.

3 .Reliable and portable

All OpenGL applications produce consistent visual display results on any OpenGL API-compliant hardware, regardless of operating system or windowing system.

4. Evolving

Because of its thorough and forward-looking design, OpenGL allows new hardware innovations to be accessible through the API via the OpenGL extension mechanism. In this way, innovations appear in the API in a timely fashion, letting application developers and hardware vendors incorporate new features into their normal product release cycles.

5. Scalable

OpenGL API-based applications can run on systems ranging from consumer electronics to PCs, workstations, and supercomputers. As a result, applications can scale to any class of machine that the developer chooses to target.

6. Easy to use

OpenGL is well structured with an intuitive design and logical commands. Efficient OpenGL routines typically result in applications with fewer lines of code than those that make up programs generated using other graphics libraries or packages. In addition, OpenGL drivers encapsulate information about the underlying hardware, freeing the application developer from having to design for specific hardware features.

7. Well-documented

Numerous books have been published about OpenGL, and a great deal of sample code is readily available, making information about OpenGL inexpensive and easy to obtain.

1.4 Problem Statement

The problem statement is to digitalize India by the introduction of BULLET TRAIN for economy and growth in our country in comparison to other countries like China, Japan, France etc. So here we demonstrate the working model simulation of a Bullet train.

1.5 Objective of the project

- The interactive demo of Bullet Train.
- Graphical approach towards understanding the Bullet Train design.

IMPLEMENTATION

4.1 Graphic functions

4.1.1 void glBegin(glEnum mode);

Initiates a new primitive of type mode and starts the collection of vertices. Values of mode include GL_POINTS, GL_LINES and GL_POLYGON.

4.1.2 void glEnd( );

Terminates a list of vertices.

4.1.3 void glColor3f[i f d] (TYPE r, TYPE g, TYPE b);

Sets the present RGB colors. Valid types are int (i), float (f) and double (d). The maximum and minimum values of the floating-point types are 1.0 and 0.0, respectively.

4.1.4 void glClearColor(GLclampf r,GLclampf g,GLclampf b,GLclampf a);

Sets the present RGBA clear color used when clearing the color buffer. Variables of GLclampf are floating-point numbers between 0.0 and 1.0.

4.1.5 int glutCreateWindow(char *title);

Creates a window on the display. The string title can be used to label the window. The return value provides a reference to the window that can be used where there are multiple windows.

4.1.6 void glutInitWindowSize(int width, int height);

Specifies the initial height and width of the window in pixels.

4.1.7 void glutInitWindowPosition(int x, int y);

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

4.1.8 void glutInitDisplayMode(unsigned int mode);

Request a display with the properties in mode. The value of mode is determined by the logical OR of operation including the color model (GLUT_RGB, GLUT_INDEX) and buffering (GLUT_SINGLE, GLUT_DOUBLE);

4.1.9 void glFlush( );

Forces any buffered any OpenGL commands to execute.

4.1.10 void glutInit (int argc, char **argv);

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

4.1.11 void glutMainLoop( );

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

4.1.12 void glutDisplayFunc(void (*func) (void));

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

4.1.13 gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);

Defines a two-dimensional viewing rectangle in the plane Z=0;

4.1.14 void glClear(GL_COLOR_BUFFER_BIT);

To make the screen solid and white.

4.1.15 void MouseFunc(myMouse);

It is used for the implementation of mouse interface. Passing the control to void myMouse(int button,int state,int x,int y);

4.1.16 void KeyboardFunc(key);

It is used for the implementation of keyboard interface. Passing control to void key(unsigned char key,int x,int y);

4.1.17 void translate[fd](TYPE x,TYPE y,TYPE z);

Alters the current matrix by displacement of (x,y,z).Type is either GLfloat or GLdouble.

4.1.18 void glPushMatrix(); void glPopMatrix();

Pushes to and pops from the matrix stack corresponding to current matrix mode.

4.1.19 void glLoadMatrix[fd](TYPE *m);

Loads the 16 element array of TYPE GLfloat or GLdouble as a current matrix.

4.2 USER DEFINED FUNCTIONS

4.2.1 Keyboard Function

```
void mykey(unsigned char key, int x, int y)
{
	switch (key)
	{
	case'l':
	case'L':reinit();
		sceneid--;
		break;
	case'k':
	case'K':reinit();
		sceneid++;
		break;
	}
}
void menu(int id)
{
	int n = 0;
	while (n<1)
	{
		switch (id)
		{
		case 1:
			st = (st + 1) % 2;//stop train
			break;
		case 2://nightmode
			nig = (nig + 1) % 2;
			break;
		case 3:break;
		case 4:	exit(0);
			break;
		}
		n++;
	}
}
```

4.2.2 Mouse Function

```
void mouse(int btn, GLint state, GLint x, int y)
{
	if (btn == GLUT_LEFT_BUTTON&&state == GLUT_DOWN)
		printf("x=%d y = %d", x, y);
}
```

4.2.3 Display Function

```
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	switch (sceneid)
	{
	case 0: scene0();
```

```
			break;
		case 1: scene1();
			break;
		case 2: scene2();
			break;
		case 3: scene3();
			break;
		case 4: scene4();
			break;
		}
		glutSwapBuffers();
}
```

4.2.4 Init Function

```
void myinit()
{
		glClearColor(1.0, 1.0, 1.0, 1.0);
		glClear(GL_COLOR_BUFFER_BIT);
		glColor3f(1.0, 0.0, 0.0);
		glMatrixMode(GL_PROJECTION);
		glLoadIdentity();
		gluOrtho2D(0, 499, 0, 499);
}
```

4.2.5 Train and Other Functions

```
void train(int x)
{
		int y;
		if(sceneid==3||sceneid==4)
			y=50;
		else
			y=150;
		glColor3ub(240, 240, 241);
		circle(0, 360, x, y + 15, 15);
		glBegin(GL_POLYGON);			//MOVING TRAIN // ENGINE //
		glColor3ub(240, 240, 241);
		glVertex2f(x, y);
		glVertex2f(x + 165, y);
		glVertex2f(x + 165, y + 100);
		glVertex2f(x + 25, y + 100);
		glVertex2f(x, y + 50);
		glEnd();
		glBegin(GL_LINES);
		glVertex2f(x + 5, y + 30);
		glVertex2f(x - 5, y + 40);
		glEnd();
		glBegin(GL_POLYGON);				//COACH 1//
		glColor3ub(240, 240, 241);
		glVertex2f(x + 175, y);
		glVertex2f(x + 282, y);
		glVertex2f(x + 282, y + 100);
		glVertex2f(x + 175, y + 100);
```

```
        glEnd();
        glColor3f(0.0, 0.0, 1.0);
        glBegin(GL_POLYGON);                //COACH 1 WINDOW //
        glColor3f(0, 1, 1);
        glVertex2f(x + 245, y + 60);
        glVertex2f(x + 270, y + 60);
        glVertex2f(x + 270, y + 85);
        glVertex2f(x + 245, y + 85);
        glEnd();
        glBegin(GL_POLYGON);                        //ENGINE WINDOW //
        glColor3f(0, 1, 1);
        glVertex2f(x + 25, y + 50);
        glVertex2f(x + 50, y + 50);
        glVertex2f(x + 50, y + 85);
        glVertex2f(x + 25, y + 85);
        glEnd();
        glColor3ub(0, 0, 0);                                //bogey
        glBegin(GL_POLYGON);
        glVertex2f(x + 165, y + 5);
        glVertex2f(x + 175, y + 5);
        glVertex2f(x + 175, y + 95);
        glVertex2f(x + 165, y + 95);
        glEnd();
        glBegin(GL_POLYGON);                //COACH 2
        glColor3ub(240, 240, 241);
        glVertex2f(x + 293, y);
        glVertex2f(x + 400, y);
        glVertex2f(x + 400, y + 100);
        glVertex2f(x + 293, y + 100);
        glEnd();
        glBegin(GL_POLYGON);                //COACH2 WINDOW
        glColor3f(0, 1, 1);
        glVertex2f(x + 340, y + 60);
        glVertex2f(x + 365, y + 60);
        glVertex2f(x + 365, y + 85);
        glVertex2f(x + 340, y + 85);
        glEnd();
        glBegin(GL_POLYGON);                   //DOOR 1//
        glColor3f(0.8, 0.78, 0.78);
        glVertex2f(x + 180, y + 20);
        glVertex2f(x + 200, y + 20);
        glVertex2f(x + 200, y + 70);
        glVertex2f(x + 180, y + 70);
        glEnd();
        glBegin(GL_POLYGON);                 //DOOR 2//
        glColor3f(0.8, 0.78, 0.78);
        glVertex2f(x + 300, y + 20);
        glVertex2f(x + 320, y + 20);
        glVertex2f(x + 320, y + 70);
        glVertex2f(x + 300, y + 70);
        glEnd();
}
void electricwire()
```

```
{
        int y;
        if(sceneid==2||sceneid==3)
                y=200;
        else
                y=300;
        glLineWidth(2);
        glBegin(GL_LINES);
        glColor3f(0, 0, 0);
        glVertex2f(-1000, y);
        glVertex2f(5000, y);
        glEnd();
        glBegin(GL_LINES);
        glColor3f(0, 0, 0);
        glVertex2f(-1000, y+30);
        glVertex2f(5000, y+30);
        glEnd();
}
void track()
{
int y;
if(sceneid==2||sceneid==3)
        y=35;
else
        y=135;
        glLineWidth(2);
        glBegin(GL_LINES);
        glColor3f(0, 0, 0);
        glVertex2f(-1000, y);
        glVertex2f(5000, y);
        glEnd();
        glBegin(GL_LINES);
        glColor3f(0, 0, 0);
        glVertex2f(-1000, y+10);
        glVertex2f(5000, y+10);
        glEnd();
        glPointSize(8);
        glBegin(GL_LINES);
        glColor3f(0, 0, 0);
        for (i = -3000; i <= 5000; i = i + 10)
        {
                glVertex2f(i, y);
                glVertex2f(i, y+10);
        }
        glEnd();
}
void sky()
{
        if (nig == 0)
        {
                glBegin(GL_POLYGON);
                glColor3f(0.4, 0.6, 1);
                glVertex2f(0, 230);
```

```
                        glVertex2f(0, 700);
                        glVertex2f(700, 700);
                        glVertex2f(700, 230);
                        glEnd();
                        int l;
                        glColor3f(1.0, 1.0, 0.0);    //SUN
                        for (l = 0; l <= 35; l++)
                        {
                                circle_draw(100 + sun, 450, l);
                        }
                }
        void road()
        {
                glColor3f(0.2, 0.7, 0.3);                                   //road
                glBegin(GL_POLYGON);
                glVertex2f(0, 200);
                glVertex2f(0, 12);
                glVertex2f(10000, 12);
                glVertex2f(10000, 200);
                glEnd();
                glColor3f(0.3, 0.5, 0.2);                     //lower road
                glBegin(GL_POLYGON);
                glVertex2f(550, 180);
                glVertex2f(600, 180);
                glVertex2f(600, 280);
                glVertex2f(550, 280);
                glEnd();
                for (i = -3000; i <= 5000; i = i + 20)
                {
                        glVertex2f(i + 5, 200);
                        glVertex2f(i + 10, 230);
                }
                glEnd();
                glColor3f(0.0, 0.0, 0.0);
                glRasterPos3f(320,190,0);
                cityname("Bangalore");
                glRasterPos3f(320,170,0);
                cityname("City Jn.");
                glColor3ub(200, 233, 240);                                       //poles
                glBegin(GL_POLYGON);
                glVertex2f(25, 25);
                glVertex2f(25, 280);
                glVertex2f(35, 280);
                glVertex2f(35, 25);
                glEnd();
                glLineWidth(0.5);
                glColor3f(0.0, 0.0, 0.0);
                glLineWidth(5);
                glBegin(GL_LINES);
                glVertex2f(35, 275);
                glVertex2f(65, 230);
                glEnd();
                glBegin(GL_LINES);
```

```
		glVertex2f(35, 230);
		glVertex2f(65, 230);
		glEnd();
		glBegin(GL_LINES);
		glVertex2f(35, 230);
		glVertex2f(65, 200);
		glEnd();
}
void overhead()
{
		int y;
		if(sceneid==2||sceneid==3)
			y=200;
		else
			y=300;
		glLineWidth(1);
		glColor3f(0, 0, 0);
		glVertex2f(0, y);
		glVertex2f(1000, y);
		glEnd();
		glBegin(GL_LINES);
		glColor3f(0, 0, 0);
		glVertex2f(0, y+30);
		glVertex2f(10000, y+30);
		glEnd();
}
void hill()
{
		glColor3f(0.5f, 0.3f, 0.0f);
		glBegin(GL_TRIANGLE_STRIP);
		glVertex2f(100, 400);
		glVertex2f(280, 230);
		glVertex2f(0, 230);
		glEnd();
		}
void scene1()
{
		glClearColor(0.0, 0.0, 0.0, 0.0);
		glClear(GL_COLOR_BUFFER_BIT);
		glFlush();
}
void scene2()
{
		glClearColor(1.0, 1.0, 1.0, 1.0);
		glClear(GL_COLOR_BUFFER_BIT);
		sky();
		hill();
		electricwire();
		road();
		track();
		overhead();
		glFlush();
}
```

```
void scene3()
{
        glPushMatrix();
        sky();
        hill();
        electricwire();
        road();
        track();
        overhead();
        glColor3f(1.0, 0.0, 0.0);
        glPointSize(3.0);
        if (!st)
                glTranslated(-xx, 0, 0);
        train(50);
        train(750);
        train(1450);
        train(2150);
        train(2850);
        glColor3f(0.0, 0.0, 0.0);
        glPopMatrix();
}
void scene4()
{
        int l;
        glClearColor(0.4, 0.6, 1,1.0);
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(0.0,0.0,0.0);                 //cable
        glLineWidth(4);
        glBegin(GL_LINE_LOOP);
        glVertex2f(250,200);
        glVertex2f(200,200);
        glVertex2f(400,700);
        glVertex2f(450,700);
        glEnd();
        glLineWidth(2);
        glBegin(GL_LINES);
        glVertex2f(257,343);
        glVertex2f(307,341);
        glEnd();
        glColor3f(0.2, 0.7, 0.3);                       //land
        glBegin(GL_POLYGON);
        glVertex2f(0,0);
        glVertex2f(700,0);
        glVertex2f(700,100);
        glVertex2f(0,100);
        glEnd();
        glColor3f(0.5,0.3,0.0);                                 //side track
        glBegin(GL_POLYGON);
        glVertex2f(180,100);
        glVertex2f(140,0);
        glVertex2f(370,0);
        glVertex2f(310,100);
        glEnd();
```

```
glColor3f(1,1,1);
glColor3f(0.0,0.0,0.0);
glLineWidth(4);
glBegin(GL_LINE_LOOP);//track
glVertex2f(240,100);
glVertex2f(200,0);
glVertex2f(300,0);
glVertex2f(260,100);
glEnd();
glColor3f(0.0,0.0,0.0);                    //top part
glBegin(GL_POLYGON);
glVertex2f(240,260);
glVertex2f(240,280);
glVertex2f(260,280);
glVertex2f(260,260);
glEnd();
glLineWidth(2);
glBegin(GL_LINE_LOOP);
glVertex2f(252,280);
glVertex2f(257,343);
glVertex2f(307,341);
glEnd();
glColor3ub(240, 240, 241);                         //body
glBegin(GL_POLYGON);
glVertex2f(230,100);
glVertex2f(180,170);
glVertex2f(180,220);
glVertex2f(180,250);
glVertex2f(250,265);
glVertex2f(315,250);
glVertex2f(315,220);
glVertex2f(315,170);
glVertex2f(270,100);
glEnd();
        glColor3ub(225, 230, 225);             //nose
for(l=0;l<25;l++)
        circle_draw(250,110,l);
glColor3ub(50,50,50);                              //lights
glBegin(GL_POLYGON);
glVertex2f(215,100);
glVertex2f(205,100);
glVertex2f(195,120);
glVertex2f(205,120);
glEnd();
glBegin(GL_POLYGON);
glVertex2f(295,100);
glVertex2f(285,100);
glVertex2f(295,120);
glVertex2f(305,120);
glEnd();
glColor3f(1.0,1.0,0.0);
for(l=0;l<4;l++)
{
```

```
                circle_draw(208,105,l);
                circle_draw(292,105,l);
        }
        glColor3f(0, 1, 1);
        glBegin(GL_POLYGON);                    //windows
        glVertex2f(185,190);
        glVertex2f(185,240);
        glVertex2f(310,240);
        glVertex2f(310,190);
        glEnd();
        glLineWidth(3);
        glColor3ub(240, 240, 241);
        glBegin(GL_LINES);
        glVertex2f(247,190);
        glVertex2f(247,240);
        glEnd();
        glColor3f(0.0,0.0,1.0);
        glBegin(GL_LINES);
        glVertex2f(200,140);
        glVertex2f(297,140);
        glEnd();
        glBegin(GL_LINES);
        glVertex2f(197,145);
        glVertex2f(300,145);
        glEnd();
        glColor3ub(200, 233, 240);                      //polef
        glBegin(GL_POLYGON);
        glVertex2f(120,50);
        glVertex2f(140,50);
        glVertex2f(140,380);
        glEnd();
        glColor3f(0.0,0.0,0.0);
       glBegin(GL_LINES);
        glVertex2f(140,320);
        glVertex2f(295,310);
        glEnd();
}
```

CONCLUSION

We have successfully completed the animation of Bullet Train with several stationary and moving components with the OpenGL tool implemented in CodeBlocks. OpenGL supports enormous flexibility in the design and the use of OpenGL graphics programs. The presence of many built in classes methods take care of much functionality and reduce the job of coding as well as makes the implementation simpler. We have implemented the project making it user-friendly and error free as possible.

As specified by the problem statement we have designed a working model for the simulation of the Bullet Train as expected, without the concept of wheels, and with the use of magnetic energy the train is able to reach at high speeds without any hindrance.

REFERENCES

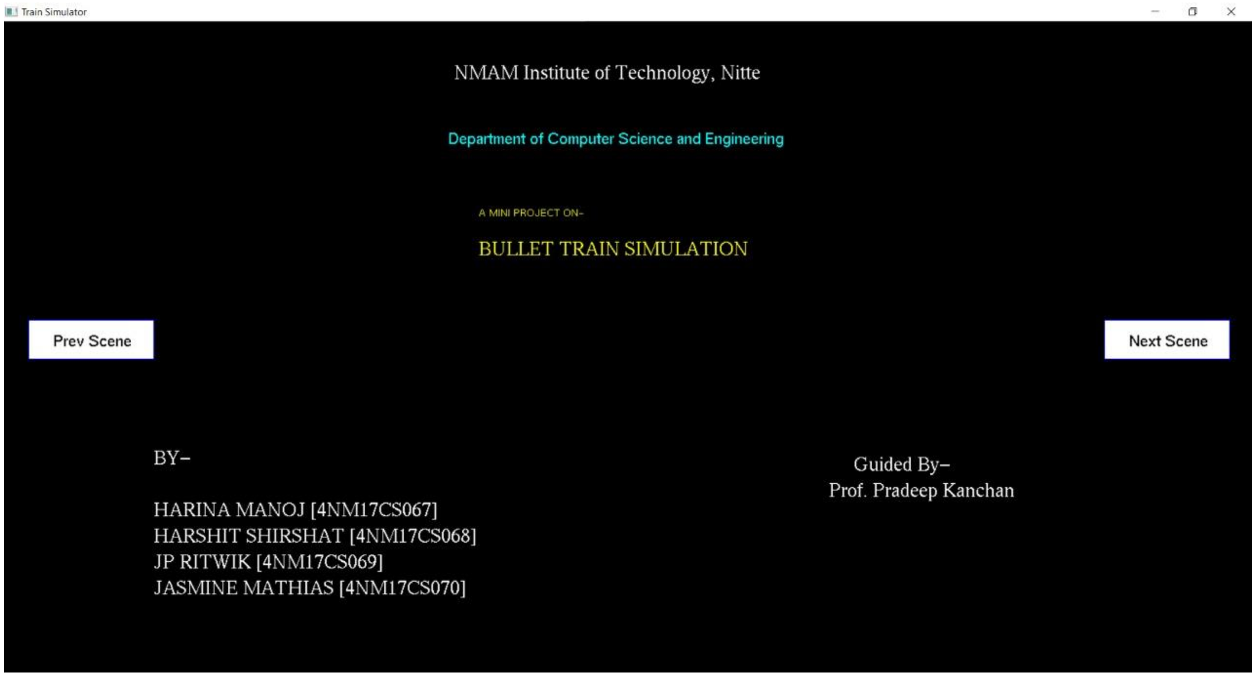
https://docs.microsoft.com/en-us/windows/win32/opengl/gl-functions

https://www.opengl.org/resources/libraries/glut/spec3/spec3.html

https://lazyfoo.net/tutorials/OpenGL/index.php

https://www.opengl.org/resources/libraries/

APPENDIX



NMAM Institute of Technology, Nitte

Department of Computer Science and Engineering

A MINI PROJECT ON-

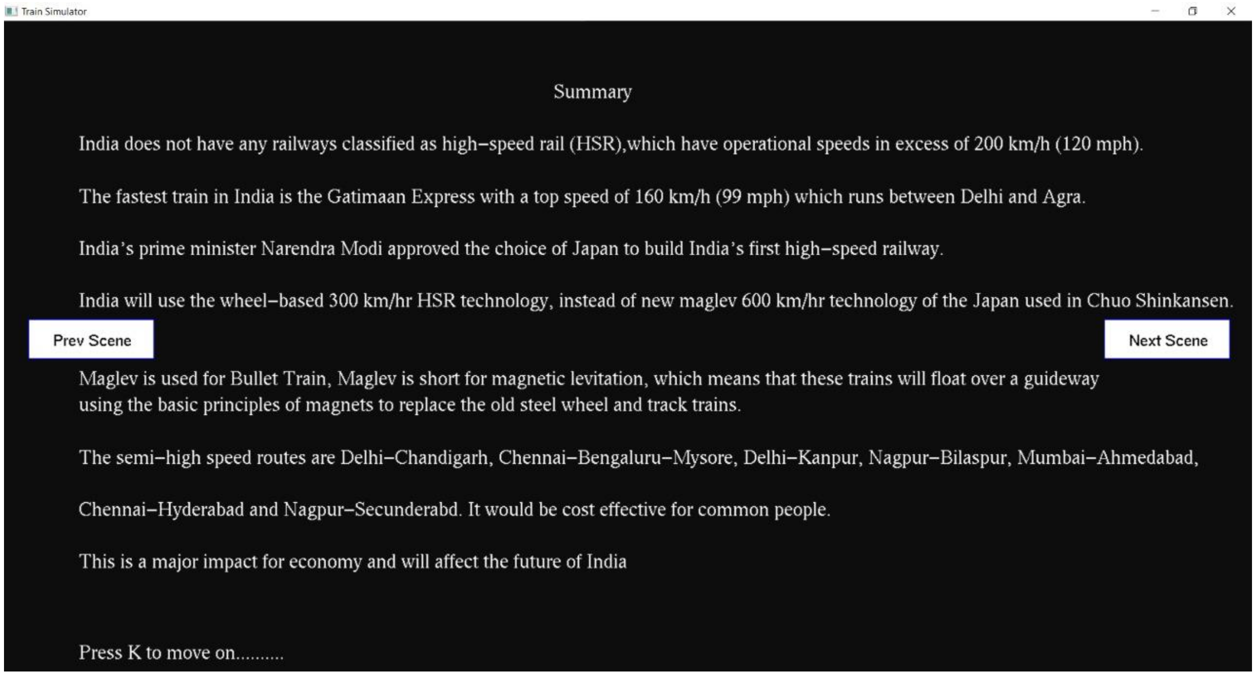
BULLET TRAIN SIMULATION

BY–

HARINA MANOJ [4NM17CS067]
HARSHIT SHIRSHAT [4NM17CS068]
JP RITWIK [4NM17CS069]
JASMINE MATHIAS [4NM17CS070]

Guided By–
Prof. Pradeep Kanchan

Figure1: NAME USN front window.



Summary

India does not have any railways classified as high–speed rail (HSR),which have operational speeds in excess of 200 km/h (120 mph).

The fastest train in India is the Gatimaan Express with a top speed of 160 km/h (99 mph) which runs between Delhi and Agra.

India's prime minister Narendra Modi approved the choice of Japan to build India's first high–speed railway.

India will use the wheel–based 300 km/hr HSR technology, instead of new maglev 600 km/hr technology of the Japan used in Chuo Shinkansen.

Maglev is used for Bullet Train, Maglev is short for magnetic levitation, which means that these trains will float over a guideway
using the basic principles of magnets to replace the old steel wheel and track trains.

The semi–high speed routes are Delhi–Chandigarh, Chennai–Bengaluru–Mysore, Delhi–Kanpur, Nagpur–Bilaspur, Mumbai–Ahmedabad,

Chennai–Hyderabad and Nagpur–Secunderabd. It would be cost effective for common people.

This is a major impact for economy and will affect the future of India

Press K to move on..........

Figure 2: Summary about the train and the way to procced.

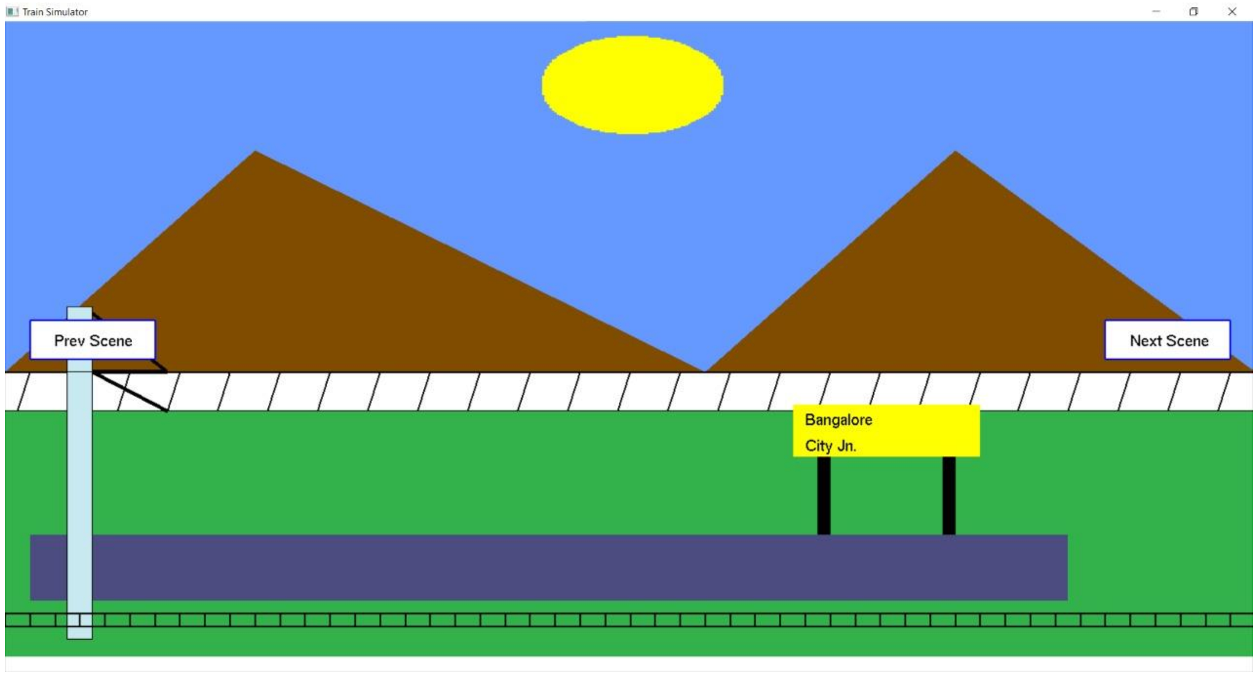


Figure 3: Day mode waiting for train to pass as we click k or K.

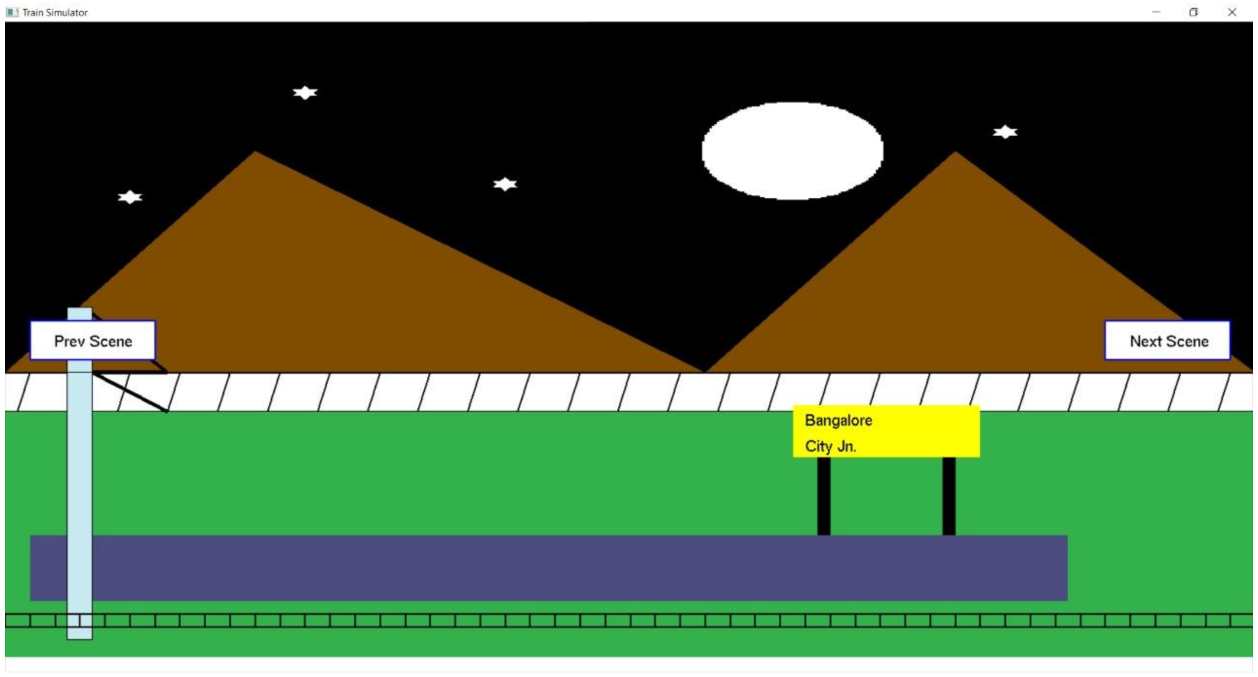


Figure 4: Night mode waiting for train to appear as we click k or K.

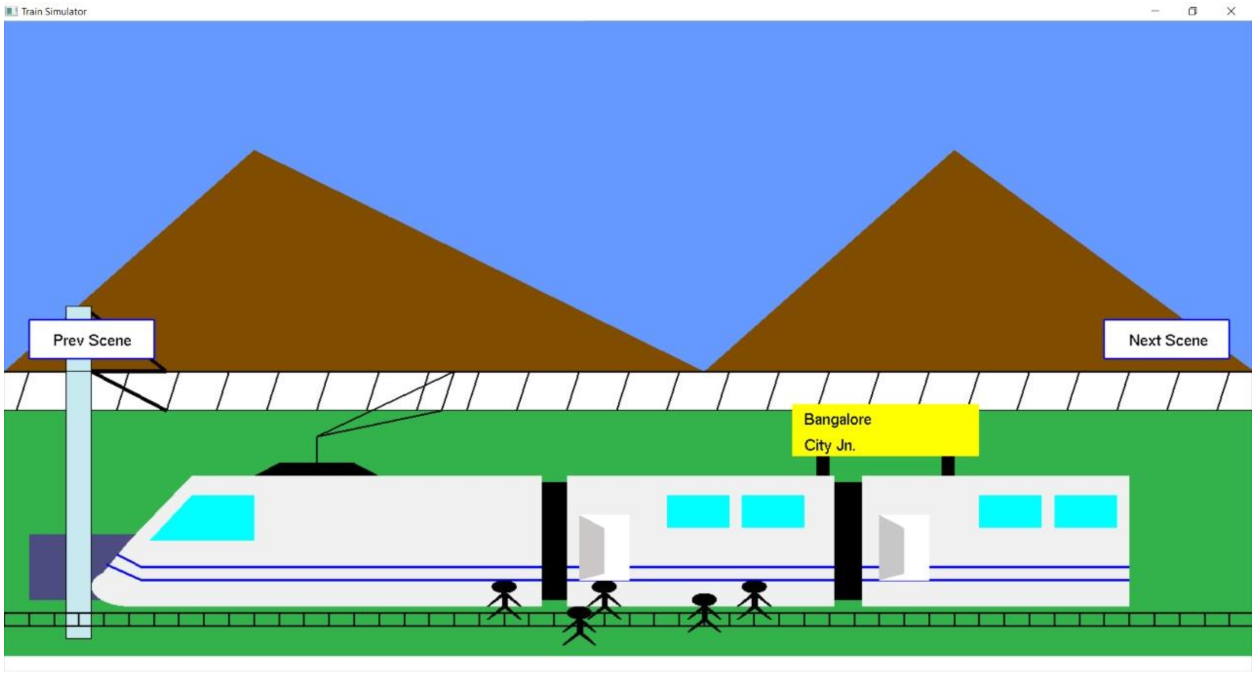


Figure 5: Day mode where train has stopped for a while and will resume as we click start train with mouse click.

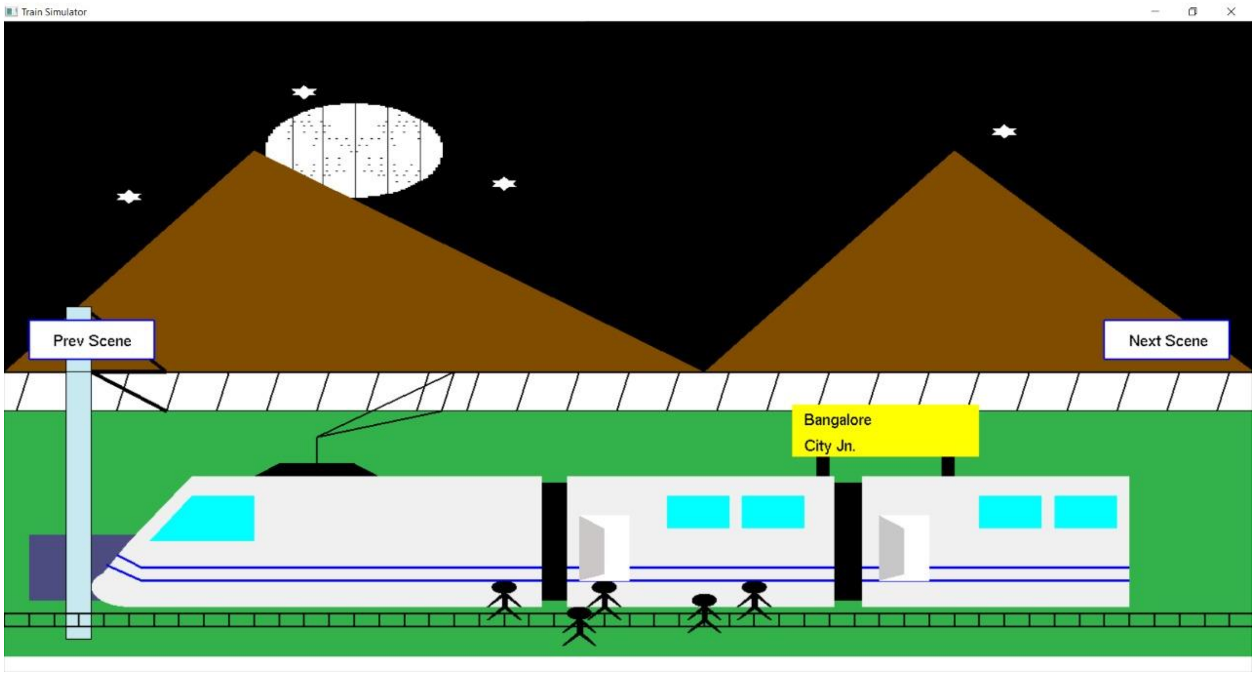


Figure 6: Night mode where train has stopped for a while and will resume as we click start train with mouse click.

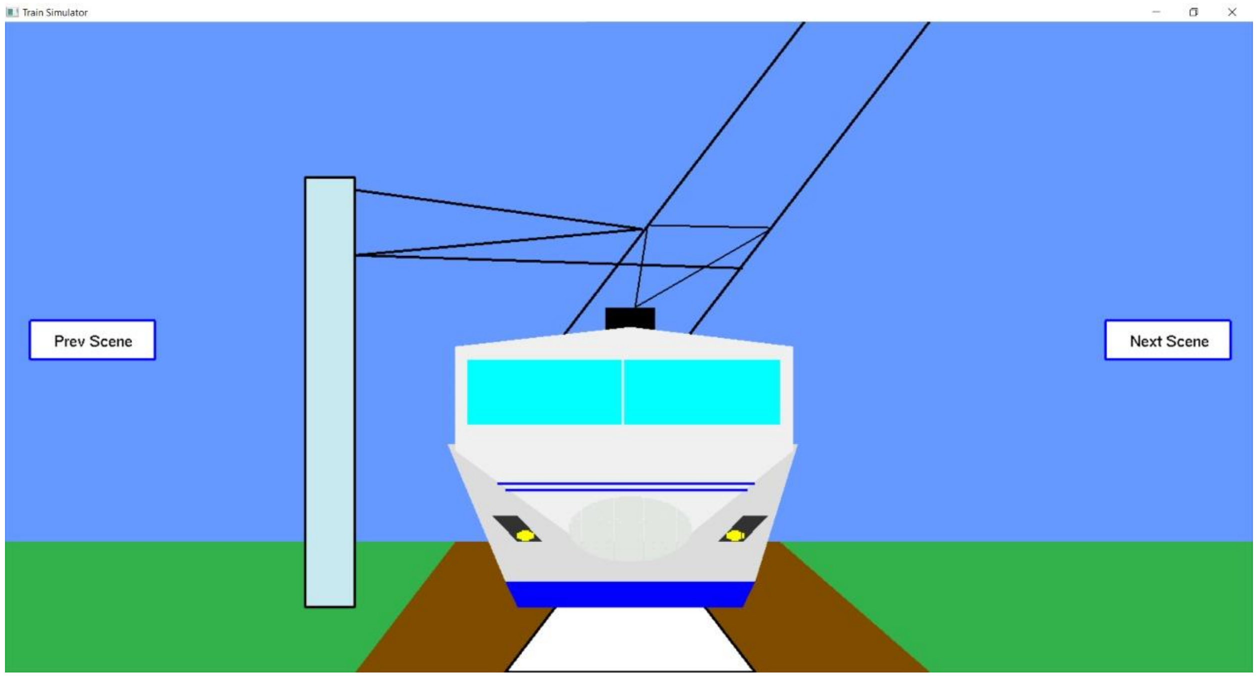


Figure 7: Front View of Train

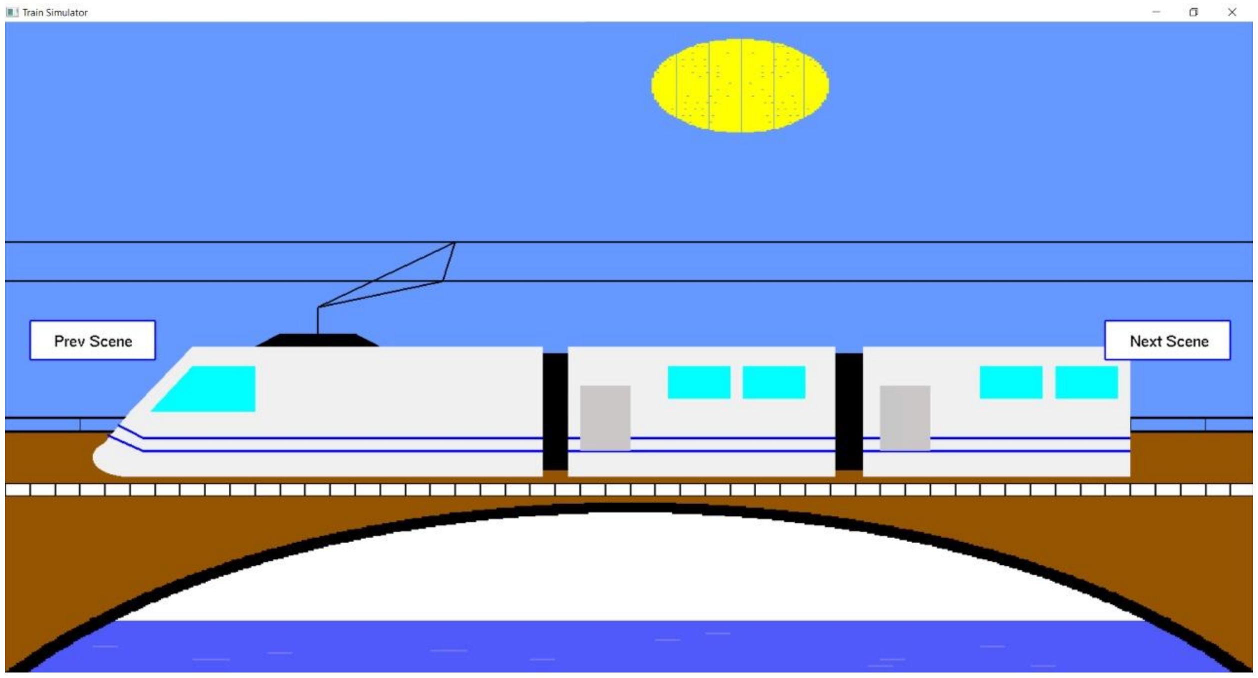


Figure 8: Day mode with scene changed as Bridge.

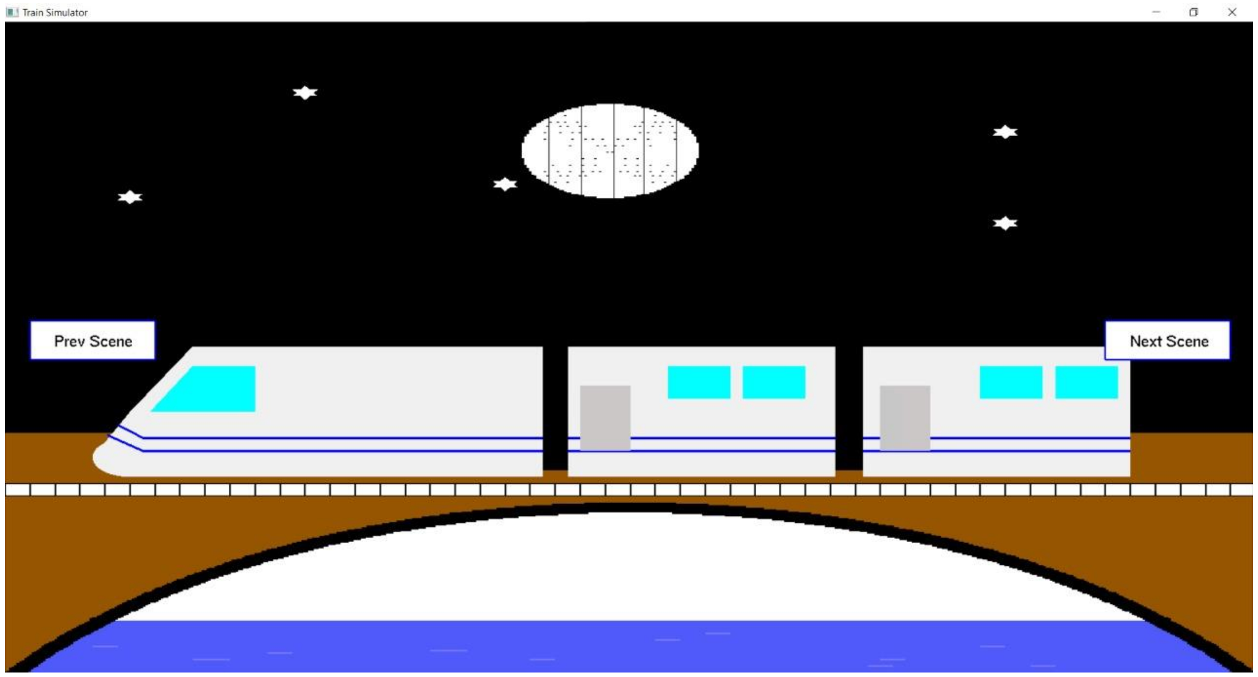


Figure 9: Night mode with scene changed as Bridge.