Abstract

Train accidents occur as a result of human errors or mechanical faults in trains, in tracks, or in the signaling system. Major and costly train accidents occur due to head-on collision of trains running on the same track towards each other. Several schemes have been proposed by researchers in the past to detect the risk of possible collision and to take preventive measures. The aim of this project is to design a OpenGL using RFID, GPS, and an RF transmitter/receiver module to detect possible collisions and inform the drivers when trains travel on the same track. We are implementing it using different primitives available in OpenGL library and combining them together in a required manner. The project has been implemented by efficiently using the data structures to obtain the optimized results and also various functions and features that are made available by the OpenGL software package have been utilized effectively.

Contents:

1. Introduction

1.1 Introduction

Train accidents are usually highly costly in terms of lost or injured human lives, damaged mechanical parts, and cancellation of normal train services. Because of this, most railway companies invest large sums of money to detect and avoid such accidents at all costs. In general, train to train collisions can be classified into three: head-on (or head-to-head), rear-to-end, and side-to-side Collisions. Head-on collision is the most dangerous one and it occurs when two trains travel on the same track towards each other. It is very difficult to stop such collisions because of speed of moving trains, which needs a lead distance to stop. There have been many train accidents all over the world. As per the report from CNN IBN India dates Sept 2011 85% of the train accidents are due to human errors. So the best way of safety system is automatic collision detection and prevention system where in addition to detecting a possible collision, the system automatically shuts down (e.g. the trains stop) to prevent the collision.

In this OpenGL project, Train Collision Detection System (TCDS) uses RFID to detect the unique track ID that a train is travelling on, a GPS to know the exact positions and speeds of other nearby trains, a powerful RF transmitter/receiver module for communication between the trains, a buzzer to warn the drivers of any risk of collision, and a medium performance toolkit that is used to coordinate all activities in the system. OpenGL is a library for doing computer graphics. By using it, we can create interactive applications which render high-quality color images composed of geometric objects and images. OpenGL is window and operating system independent. As such, the part of our application which does rendering is platform independent. GLUT, the OpenGL Utility Toolkit is a library to make writing OpenGL programs regardless of windowing systems much easier.

1.2 Motivation

In 2014, there were 1928 persons killed or seriously injured in railway accidents in the European Union (EU). A total of 2213 significant train accidents were registered in the same year in EU. To solve all these kind of incident we came up with an idea of presenting a project using OpenGL which would demonstrate how the GPS and RFID could be used in order to avoid such train incidents.

1.3 Outline of the Project Report

The outcome of the project is that after it has been developed it will drastically decrease the train accident which is caused because of the Collision. As a result of that many human lives can be saved. Our objective is to develop a cg projects for collision detection of trains using GPS and RFIDs. Here we need to draw the following things -Trains - the bogies and the engine ,Train Tracks, Sky with surrounds (Environment),satellite generating signals.

After we finish the drawing our next aim would be to give motion to the train .As we are going to draw the trains so our track will be a straight not in zigzag manner. Satellite signals helps to avoid collisions lowering the risk of accidents, if properly followed.

2. PROBLEM DEFINITION AND REQUIREMENTS

2.1 Problem Definition

Prevention of Collision of Train using GPS & RFID using OpenGL functions.

2.2 Requirement Specification

- This software requires RAM that cannot be less than 520 MB.
- The processor cannot be less than 2 GHz speed.
- Hard disk of 20GB or more disk space.
- Operating System can be Linux , Windows XP or later
- Language tool that is to be used is OpenGL.
- Compiler that can be used is GNU/C++ compiler.

2.3 Functional Requirements

- Display the collision detection of trains using GPS.
- Displayed train's bogies should have only the authorized red color.
- Appropriate color trains should be placed at their respective tracks.
- It must provide facilities to change the trains positions appropriately based on the need of the user.

2.4 System Requirements

- Any GPU that is compatible with OpenGL 3.2. (Integrated graphic cards Intel HD 4000 or above).

3. Design and Implementation

3.1 User Interface Design

Mouse events:

- Click Event to Start the Demo of the Project.

Keyboard events:

- Press 1 to see how the collision can be avoided when the 2 trains are on the same track.
- Press 2 to see what happens when the trains are on different tracks.

.3.2 Various features

- Easy to understand.
- No complicated mouse or Keyboard events are used.

3.3 Graphic functions

- 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.

- void glEnd( );

Terminates a list of vertices.

- 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.

- 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.

- 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.

- void glutInitWindowSize(int width, int height);

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

- void glutInitWindowPosition(int x, int y);

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

- 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);

- void glFlush( );

Forces any buffered any OpenGL commands to execute.

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

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

- void glutMainLoop( );

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

- void glutDisplayFunc(void (*func) (void));

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

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

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

3.1.14 void glutBitmapCharacter(void *font, int char);

Renders the character with ASCII code char at the current raster position using the raster font given by font. Fonts include GLUT_BITMAP_TIMES_ROMAN_10 and GLUT_BITMAP_TIMES_ROMAN_8_Y_13.

- void glClear(GL_COLOR_BUFFER_BIT);

To make the screen solid and white.

- 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);

- void KeyboardFunc(key);

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

- 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.

- void glPushMatrix();

void glPopMatrix();Pushes to and pops from the matrix stack corresponding to current matrix mode.

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

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

3.3 Design

The main functions used in design are:

- void mountains();
- void sun();
- void menu();
- void mountains();
- void sky();
- void circle();
- void tree();
- void TRAINS();
- void mountains();
- void danger();
- void Basics_func()

4. SOURCE CODE

```
#include<stdio.h>
#include<stdlib.h>
#include<GL/glut.h>
#include <math.h>
#include <time.h>
#define M_PI 3.14
#define DEGREES_TO_RADIANS 3.14159/180.0

int disp=0;
void *currentfont;
int c=0,d=500,g=380,h=970,i=540,j=970,k=380,l=965,m=540,n=970,cc=0,dd=440;
GLfloat x=0.0;
int z=0,a;
int train1,train2,train3,basic;
int x11=0,y11=0,x12=0,y12=0,x21=0,y21=0,x22=0,y22=0;
int flag=0;
void mountains(int x,int y);

void setFont(void *font)
{
        currentfont=font;
}

void drawstring(float x,float y,float z,char *string)
{
        char *c;
        glRasterPos3f(x,y,z);

        for(c=string;*c!='\0';c++)
        {       glColor3f(0.0,1.0,0.0);
                glutBitmapCharacter(currentfont,*c);
        }
}

void displaytext()
{

glClear(GL_COLOR_BUFFER_BIT);
setFont(GLUT_BITMAP_HELVETICA_10);
        glColor3f(1,1,0);


        setFont(GLUT_BITMAP_HELVETICA_18);
        drawstring(420,650,0,"PROJECT ASSOCIATES");
        drawstring(220,550,0,"RAJAT R SHETTY");
        drawstring(220,500,0,"2GI15CS115");
        drawstring(675,550,0,"RAJESH KUMAR");
        drawstring(675,500,0,"      2GI15CS116");
```

```
	glColor3f(1,1,0);
	drawstring(412,450,0,"UNDER THE GUIDANCE OF");
	drawstring(210,320,0,"Prof.VIDYADHEESH PANDURANGI");
	drawstring(240,280,0,"");
	drawstring(625,320,0,"Prof. PANKAJA B.PATIL");
	drawstring(750,280,0,"");

	setFont(GLUT_BITMAP_TIMES_ROMAN_24);
	glColor3f(0,1,0);
	drawstring(350,750,0,"COLLISION DETECTION OF TRAINS");
	glColor3f(1,0,0);
	drawstring(340,850,0,"COMPUTER GRAPHICS MINI PROJECT");
	drawstring(450,800,0,"");

	setFont(GLUT_BITMAP_TIMES_ROMAN_24);
	glColor3f(1,0,1);
	drawstring(420,200,0,"CLICK TO START...");
	drawstring(320,120,0,"COMPUTER SCIENCE AND ENGINEERING");
	drawstring(330,60,0,"KLS GOGTE INSTITUTE TECHNOLOGY");
	glFlush();
}

void sun(GLfloat radius)
{
	int i;
	glBegin(GL_POLYGON);
	glColor3f(1.0f,1.f,1.0f);
	glVertex2f(0,0);

	for (i=0; i<361; i++)
	{
		glColor3f(.9f,0.8f,.0f);
		float degInRad = i * DEGREES_TO_RADIANS;
		glVertex2f(cos(degInRad)*radius,sin(degInRad)*radius);
	}

	glEnd();
}



void menu()
{
	glClear(GL_COLOR_BUFFER_BIT);
	setFont(GLUT_BITMAP_TIMES_ROMAN_24);
	glColor3f(1,0,0);
	drawstring(250,800,0,"-----------------------------------------------");
	glColor3f(0,1,0);
	drawstring(250,770,0,"1.COLLISION DETECTION REQUIRED -------- press 1");
	glColor3f(1,0,0);
```

```
        drawstring(250,740,0,"-----------------------------------------------");
        setFont(GLUT_BITMAP_HELVETICA_18);
        glColor3f(1,1,0);
        drawstring(250,710,0,"This case deals about two trains on the same track running at
high speeds.");
        glColor3f(1,1,0);
        drawstring(250,680,0,"This may be the case on a single line system or may be due to
poor visibility.");
        glColor3f(1,1,0);
drawstring(250,650,0,"Hence Anti-Collisision Device with the help of GPS technology
sends a warning");
        glColor3f(1,1,0);
        drawstring(250,620,0,"to approaching trains and an acknowledgement is sent by the
trains in return.");
        glColor3f(1,1,0);
drawstring(250,590,0,"Automatic brakes are applied and the train stops before collision");
        glColor3f(1,0,0);
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        drawstring(250,560,0,"-----------------------------------------------");


        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(1,0,0);
        drawstring(250,450,0,"-----------------------------------------------");
        glColor3f(0,1,0);
        drawstring(250,420,0,"2.COLLISION DETECTION NOT REQUIRED ---- press
2");
        glColor3f(1,0,0);
        drawstring(250,390,0,"-----------------------------------------------");
        setFont(GLUT_BITMAP_HELVETICA_18);
        glColor3f(1,1,0);
        drawstring(250,350,0,"In this case both the trains are on different tracks and there is
no danger detected");
        glColor3f(1,1,0);
        drawstring(250,320,0,"The trains are though connected through the GPS but the
brakes are not applied.");
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(1,0,0);
        drawstring(250,290,0,"-----------------------------------------------");
}


void sky()
{
                glBegin(GL_QUADS);
                glColor3f(0,1,1);
                glVertex2f(0,730);
                glColor3f(0,1,1);
                glVertex2f(999,730);
                glColor3f(1,1,1);
                glVertex2f(999,999);
```

```
                glColor3f(1,1,1);
                glVertex2f(0,999);
                glEnd();
}

void circle( float x, float y )
{
        float r=5.0,t=0;
        int n;

glBegin(GL_TRIANGLE_FAN);
        glColor3f(0,0,0);
     glVertex2f( x, y );
for( n = 0;n <= 90;++n) // increment by a fraction of the maximum
      {
        glVertex2f( x + sin( t * 2 * M_PI ) * r, y + cos( t * 2 * M_PI ) * r );
                        t = (float)n/90;
      }
glEnd();
}

void tree( int x, int y)
{
        glColor3f(0,0.9,0);

        glBegin(GL_TRIANGLES);
        glVertex2f(x,y);
        glVertex2f(x-10,y-15);
        glVertex2f(x+10,y-15);
        glEnd();
        glBegin(GL_TRIANGLES);
        glVertex2f(x,y-15);
        glVertex2f(x-20,y-30);
        glVertex2f(x+20,y-30);
        glEnd();

        glBegin(GL_TRIANGLES);
        glVertex2f(x,y-30);
        glVertex2f(x-30,y-45);
        glVertex2f(x+30,y-45);
        glEnd();

        glBegin(GL_TRIANGLES);
        glVertex2f(x,y-45);
        glVertex2f(x-40,y-60);
        glVertex2f(x+40,y-60);
        glEnd();

        glColor3f(0.7,0.3,0);
        glBegin(GL_POLYGON);
        glVertex2f(x-10,y-60);
```

```
        glVertex2f(x-5,y-65);
        glVertex2f(x+5,y-65);
        glVertex2f(x+10,y-60);
        glEnd();

        glBegin(GL_POLYGON);
        glVertex2f(x-5,y-65);
        glVertex2f(x-5,y-95);
        glVertex2f(x+5,y-95);
        glVertex2f(x+5,y-65);
        glEnd();

        glBegin(GL_POLYGON);
        glVertex2f(x-5,y-95);
        glVertex2f(x-10,y-100);
        glVertex2f(x+10,y-100);
        glVertex2f(x+5,y-95);;

        glEnd();

        glFlush();
}

void sleep(unsigned int seconds)
{
clock_t goal = seconds + clock();
while (goal > clock());
}


void TRAINS(int x1,int y1,int a,int b)
{               int i=0;

                glBegin(GL_QUADS);
                glColor3f(0,0.0,1.0);                   //ENGINE
                glVertex2f(x1,y1);              //lengh of engine=60;height of engine=30;
                glColor3f(0,0.0,1.0);
                glVertex2f(x1+60,y1);
                glColor3f(1.0,0.0,0.0);
                glVertex2f(x1+60,y1-30);
                glColor3f(0,0.0,0.0);
                glVertex2f(x1,y1-30);
                glEnd();
                circle(x1+10,y1-35);                    //wheels
                circle(x1+50,y1-35);

                while(i<3)
                {       glBegin(GL_QUADS);                      //BOGIES
                        glColor3f(1.0,0.0,0.0);                 //For right train a=795,b=510
                        glVertex2f(a,b);
                        glColor3f(1.0,0.0,0.0);
```

```
                glVertex2f(a+60,b);
                glColor3f(1.0,0.0,0.0);
                glVertex2f(a+60,b-20);
                glColor3f(1.0,0.0,0.0);
                glVertex2f(a,b-20);
                glEnd();
                a+=65;
                i++;
                circle(a-15,b-25);                  //wheels
                circle(a-55,b-25);
        }

}

void Basics_func()
{       int x=80,y=650,ii,xx=70,yy=350;

                sky();

                //Sun
        glPushMatrix();
        glTranslatef(720,690,0.0);
        sun(70);
        glPopMatrix();



                mountains(0,300);

                for(ii=0;ii<9;ii++)
                {
                        tree(x,y);
                        tree(xx,yy);
                        x+=110;
                        xx+=110;
                }



                /** Track **/
                glBegin(GL_LINES);

                glColor3f(0.0,0.0,0.0);
                glVertex2f(0,500);
                glVertex2f(1000,500);
                glVertex2f(0,499);
              glVertex2f(1000,499);

                glVertex2f(0,480);
                glVertex2f(1000,480);
```

```
        while(c!=1000)
        {
                glVertex2f(c,d);
                glVertex2f(c,d-20);
                c+=10;
        }
        glEnd();

        glBegin(GL_LINES);
        glColor3f(0.0,0.0,0.0);
        glVertex2f(0,440);
        glVertex2f(1000,440);
        glVertex2f(0,420);
        glVertex2f(1000,420);
        while(cc!=1000)
        {
                glVertex2f(cc,dd);
                glVertex2f(cc,dd-20);
                cc+=10;
        }
        glEnd();

/**     glBegin(GL_LINES);
        glColor3f(0.0,0.0,0.0);
        glVertex2f(468,500);
        glVertex2f(520,440);
        glVertex2f(460,480);
        glVertex2f(512,420);

        glVertex2f(468,500);
        glVertex2f(460,480);

glVertex2f(520,440);
glVertex2f(512,420);
glVertex2f(476,492);
glVertex2f(467,471);
glVertex2f(484,484);
glVertex2f(474,462);
glVertex2f(492,476);
glVertex2f(481,453);
glVertex2f(500,466);
glVertex2f(488,444);
glVertex2f(508,458);
glVertex2f(495,435);
glVertex2f(516,449);
glVertex2f(504,425);
glEnd();


        /**     satellite       **/
```

```
        glColor3f(0,0,0);
        setFont(GLUT_BITMAP_HELVETICA_18);
        drawstring(650,950,0,"satellite");



        glBegin(GL_POLYGON);                    //middle part in yellow
        glColor3f(1.0,1.5,0.0);
        glVertex2f(480,990);
        glVertex2f(510,990);
        glVertex2f(510,900);
        glVertex2f(480,900);
        glEnd();
        glBegin(GL_LINE_LOOP);          //black color surrounding middle part
        glColor3f(0.0,0.0,0.0);
        glVertex2f(480,990);
        glVertex2f(510,990);
        glVertex2f(510,900);
        glVertex2f(480,900);
        glEnd();

        glBegin(GL_LINE_LOOP);          //left panel of satellite
        glColor3f(0.0,0.0,1.0);
        glVertex2f(380,970);
        glVertex2f(450,970);
        glVertex2f(450,950);
        glVertex2f(480,950);
        glVertex2f(480,940);
        glVertex2f(450,940);
        glVertex2f(450,920);
        glVertex2f(380,920);
        glEnd();
while(g<=450)                           //vertical lines in left panel
{
//g=380;h=970
        glBegin(GL_LINES);
        glVertex2f(g,h);
        glVertex2f(g,h-50);
        glEnd();
        g+=5;
}
while(l>=920)                           //horizontal lines in left panel
{
//k=380;l=970
        glBegin(GL_LINES);
        glVertex2f(k,l);
        glVertex2f(k+70,l);
        glEnd();
        l-=7;

}
```

```
        glBegin(GL_LINE_LOOP);                                  //right panel of
satellite
                glColor3f(0.0,0.0,1.0);
                glVertex2f(610,970);
                glVertex2f(540,970);
                glVertex2f(540,950);
                glVertex2f(510,950);
                glVertex2f(510,940);
                glVertex2f(540,940);
                glVertex2f(540,920);
                glVertex2f(610,920);
        glEnd();
        while(i<=610)                                   //vertical lines in right panel
        {
        //i=540;j=970
                glBegin(GL_LINES);
                glVertex2f(i,j);
                glVertex2f(i,j-50);
                glEnd();
                i+=5;
        }
        while(n>=920)
        //horizontal lines in right panel
        {
        //m=540;n=970
                glBegin(GL_LINES);
                glVertex2f(m,n);
                glVertex2f(m+70,n);
                glEnd();
                n-=7;
        }


}

void mountains(int x,int y)
{
        glBegin(GL_POLYGON);
        glColor3f(0.5, 0.35, 0.05);
        glVertex2i(0+x,300+y);
        glVertex2i(100+x,450+y);
        glVertex2i(250+x,300+y);
        glVertex2i(500+x,450+y);
        glVertex2i(750+x,300+y);

        glEnd();
        glBegin(GL_TRIANGLES);
        glVertex2i(700+x,300+y);
```

```
        glVertex2i(900+x,470+y);
        glVertex2i(1000+x,300+y);
        glEnd();

}

void myinit()
{       glClearColor(0.0,0.0,0.0,0.0);//background color
        gluOrtho2D(0,999,0,999);
        glPointSize(2.0);
        glLineWidth(2.0);

        basic = glGenLists(1);
                glNewList(basic, GL_COMPILE);
                Basics_func();
        glEndList();

        train1 = glGenLists(1);
                glNewList(train1, GL_COMPILE);
                TRAINS(850,520,915,510);
        glEndList();

        train2 = glGenLists(1);
                glNewList(train2, GL_COMPILE);
                TRAINS(85,520,-110,510);
        glEndList();

        train3 = glGenLists(1);
                glNewList(train3, GL_COMPILE);
                TRAINS(280,460,85,450);
        glEndList();

}

void danger()
{
        x11+=16;
        y11-=20;
        glTranslatef(x11,y11,0);
                glBegin(GL_POLYGON);
                glColor3f(1,0,0);
                glVertex2f(495,900);
                glVertex2f(510,880);
                glVertex2f(520,880);
                glVertex2f(505,900);
                glEnd();
        glColor3f(0,0,0);
        setFont(GLUT_BITMAP_HELVETICA_18);
        drawstring(520,880,0,"WARNING");

        x21-=40;
```

```
	y21-=10;
	glTranslatef(x21,y21,0);
		glBegin(GL_POLYGON);
		glColor3f(1,0,0);
		glVertex2f(495,900);
		glVertex2f(505,900);
		glVertex2f(495,870);
		glVertex2f(485,870);
		glEnd();
	glColor3f(0,0,0);
	setFont(GLUT_BITMAP_HELVETICA_18);
	drawstring(410,870,0,"WARNING");

}


void ack()
{
	glTranslatef(x12,y12,0);
		glBegin(GL_POLYGON);
		glColor3f(0,0.5,0);
		glVertex2f(815,540);
		glVertex2f(825,540);
		glVertex2f(840,520);
		glVertex2f(830,520);
		glEnd();
	x12-=20;
	y12+=20;
	glColor3f(0,0,0);
	setFont(GLUT_BITMAP_HELVETICA_18);
	drawstring(815,545,0,"Acknowledgement");

	glTranslatef(x22,y22,0);
		glBegin(GL_POLYGON);
		glColor3f(0,0.5,0);
		glVertex2f(90,540);
		glVertex2f(100,540);
		glVertex2f(95,520);
		glVertex2f(85,520);
		glEnd();
	x22+=50;
	y22+=10;
	glColor3f(0,0,0);
	setFont(GLUT_BITMAP_HELVETICA_18);
	drawstring(105,520,0,"Acknowledgement");


}


void draw(void)
```

```
{
        int count=0;

if(disp==0)
        {
        glClearColor(0,0,0,0);
        glClear(GL_DEPTH_BUFFER_BIT|GL_COLOR_BUFFER_BIT);
        displaytext();
        }
if(disp==1)
        {
        glClearColor(0,0,0,0);
        glClear(GL_DEPTH_BUFFER_BIT|GL_COLOR_BUFFER_BIT);
        menu();
        }
if(disp==3)
        {
        glClearColor(0.0,0.5,0,0);
        glClear(GL_DEPTH_BUFFER_BIT|GL_COLOR_BUFFER_BIT);

        glPushMatrix();


        glPushMatrix();
                glCallList(basic);
        glPopMatrix();

        glPushMatrix();
                glTranslatef(x,0.0,0.0);
                glCallList(train1);
                glFlush();
        glPopMatrix();

        glPushMatrix();
                glTranslatef(-x,0.0,0.0);
                sleep(150);
                glCallList(train3);
                glFlush();
        glPopMatrix();


if(z<250)
        {
        x=x-5.0;
        sleep(50);
        z++;
        }
if(z>=250)
        {
        glColor3f(0,0,0);
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
```

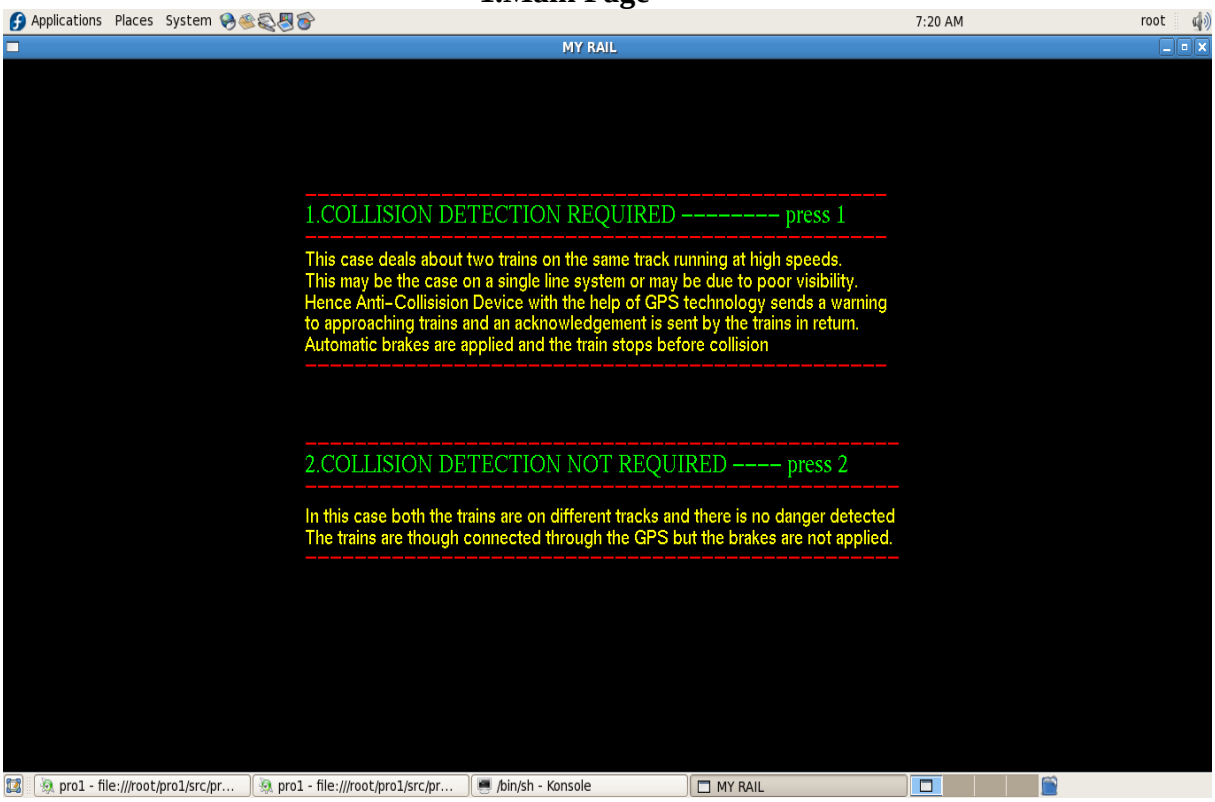
```
        drawstring(325,780,0,"!! DIFFERENT TRACKS, COLLISION NOT POSSIBLE
!!");
        }
glFlush();
glutPostRedisplay();
glPopMatrix();
}

if(disp==2)
{

        glClearColor(0.0,0.5,0,0);
        glClear(GL_DEPTH_BUFFER_BIT|GL_COLOR_BUFFER_BIT);
        glPushMatrix();
        glPushMatrix();
                glCallList(basic);
        glPopMatrix();

        glPushMatrix();
                glTranslatef(x,0.0,0.0);
                glCallList(train1);
                glFlush();
        glPopMatrix();

        glPushMatrix();
                glTranslatef(-x,0.0,0.0);
                glCallList(train2);
                glFlush();
        glPopMatrix();

        if(x11<=340&&y11>=-340&&x21>=-470&&y21>=-470)
        {
        glPushMatrix();
                danger();
                glFlush();
        glPopMatrix();
        }

        else if(x12>=-320&&y12<=320&&x22<=600&&y22<=600)
        {
        glPushMatrix();
                ack();
                glFlush();
        glPopMatrix();
        }

if(z<65)
        {
        x=x-5.0;
        sleep(100);
        z++;
```

```
        }
if(z>=65)
        {
        glColor3f(0,0,0);
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        drawstring(225,780,0,"!! DANGER DETECTED, TRAINS STOPPED BEFORE
COLLISION !!");
        glColor3f(0,0,0);
        drawstring(700,100,0,"Left Click to go back to menus");
        }

glPopMatrix();
}
glutPostRedisplay();
glFlush();
glutSwapBuffers();
}

void mouse(int btn,int state,int x,int y)
{
        if(btn==GLUT_LEFT_BUTTON&&state==GLUT_DOWN)
                disp=1;
                glutPostRedisplay();
 }

void mykey(unsigned char key,int x,int y)
{
        if(key=='1') disp=2;
        if(key=='2') disp=3;
}


int main(int argc, char **argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
        glutCreateWindow("MY RAIL");
        glutInitWindowPosition(500,500);
        glutInitWindowSize(1024,768);
        glutMouseFunc(mouse);
        glutKeyboardFunc(mykey);
        glutDisplayFunc(draw);
        glDisable(GL_DEPTH_TEST);
        myinit();
        glutMainLoop ();
        return(0);

}
```

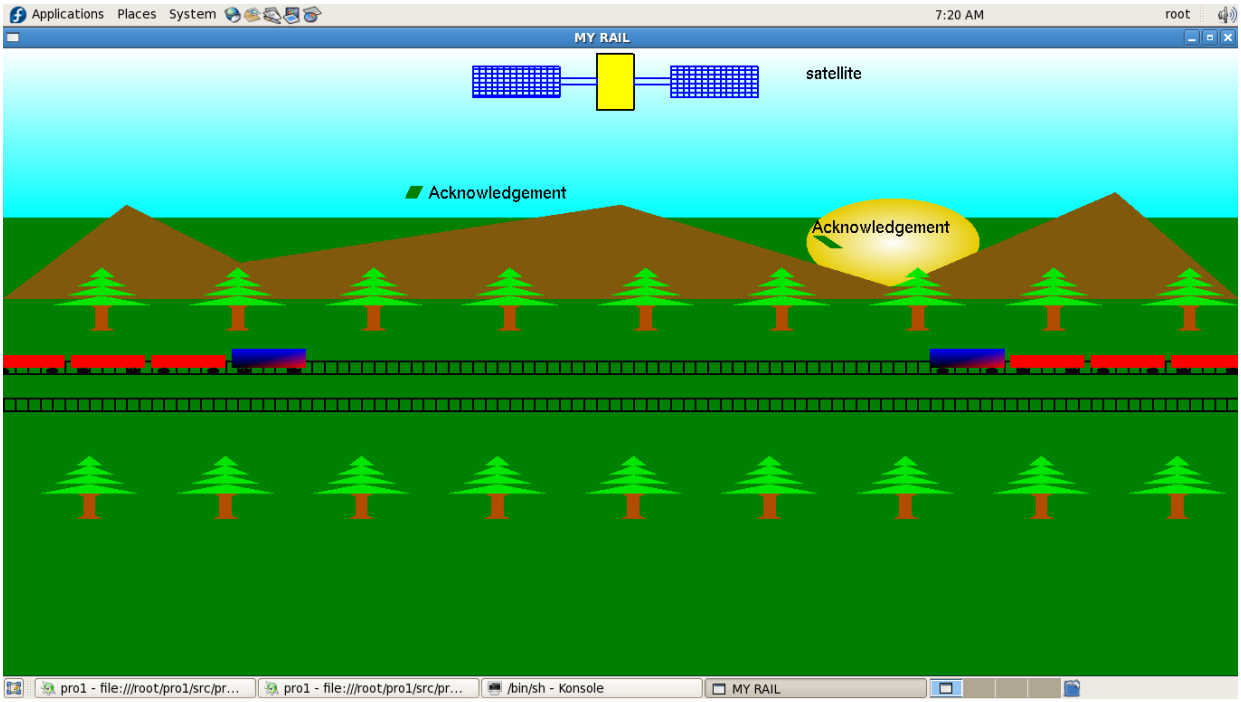
5. EXPERIMENTAL RESULTS AND ANALYSIS

5.1 SNAPSHOTS

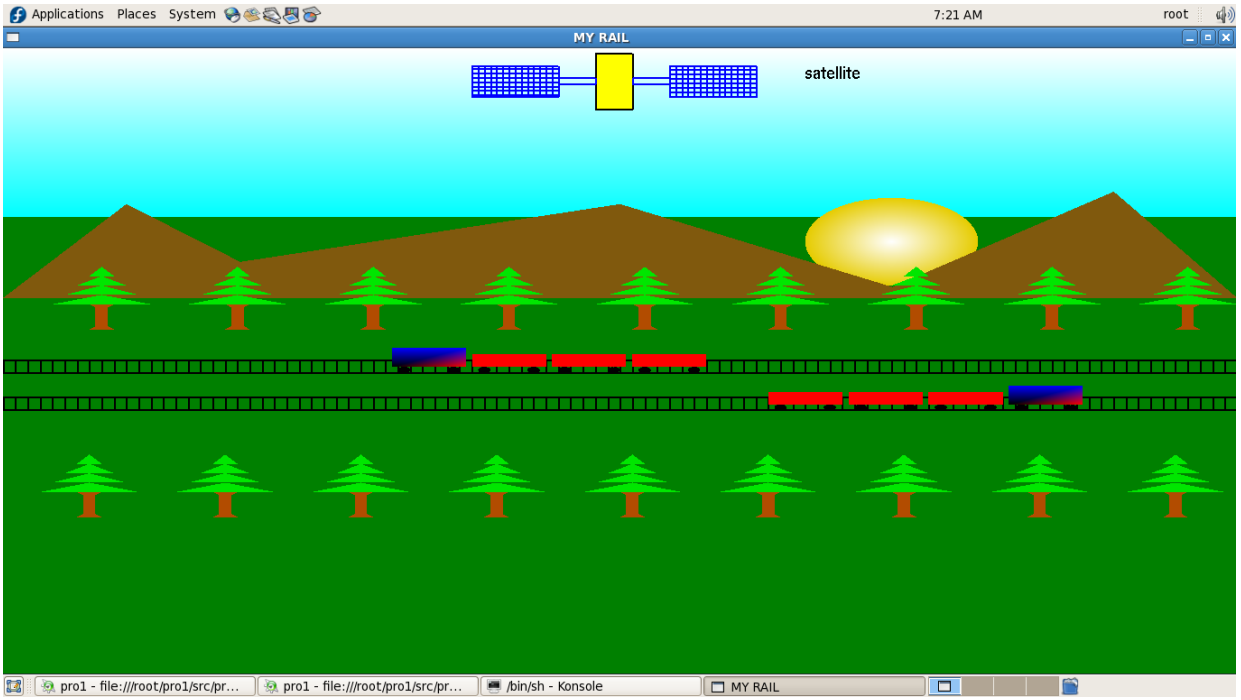
1.Main Page



2. Keyboard Input Phase



3. When 1 is Pressed , it shows how the collision is prevented through Acknowledgement when 2 trains are on same track.



4. When 2 is pressed it shows the condition when the trains are running on two different tracks and as a result no collision occurs.

6. CONCLUSION AND FUTURE SCOPE

CONCLUSION

The main intension of the project is to prevent train collisions. By using this project many human lives can be saved. This project can work in any atmospheric conditions. Without any human involvement the trains will automatically stops, with the help of RFID and GPS signal.

FUTURE SCOPE

The developed system can be enhanced further by adding automatic emergency help line where the movements of the trains and their speeds can be stored on an SD card in real-time for future analysis, just like the Black box of an aircraft. Finally, the system can be linked to the train brake system to halt the train should there be risk of collision in a pre-defined future time or distance.

REFERENCES

1. Edward Angel: Interactive Computer Graphics A Top-Down Approach with OpenGL, 5th Edition, Pearson Education, 2008.

2. F.S. Hill Jr.: Computer Graphics Using OpenGL, 3rd Edition, PHI, 2009.

3. Donald Hearn and Pauline Baker: Computer Graphics-OpenGL Version, 3rd Edition, Pearson Education, 2004.

4. http://www.tutorialspoint.com/computer_graphics/

5. https://www.courses.psu.edu/art/art201_jxm22/tutorials.html

6. http://lodev.org/cgtutor/

7. http://www.graphics.cornell.edu/online/tutorial/

8. http://www.opengl-tutorial.org/beginners-tutorials/

9. https://opengl/

10. www.ijcaonline.org/archives/volume152/number2/ibrahim-2016-ijca-911771

11. https://www.researchgate.net/publication/308827726_Sensor_based_identification_system_for_train_collision_avoidance