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

Report on Mini Project

"Rollercoaster Ride"

Course Code: 16CS703

Course Name: Computer Graphics and Multimedia

Semester: VII Section: B

Submitted To,

Mr. Pradeep Kanchan
Asst Prof Gd III
Dept. Of CSE, NMAMIT

Submitted By:

NAME:	USN:
KEERTHESH S	**4NM17CS087**
KIRAN MAHADEV GIRADDI	**4NM17CS088**
KIRTHI PUTRAN	**4NM17CS089**
KISHAN	**4NM17CS090**

Date of submission: 22th December 2020

Signature of Course Instructor

ABSTRACT

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television.

A **rollercoaster** is a type of amusement ride that employs a form of elevated railroad track designed with tight turns, steep slopes, and sometimes inversions.[1] People ride along the track in open cars, and the rides are often found in amusement parks and theme parks around the world. LaMarcus Adna Thompson obtained one of the first known patents for a roller coaster design in 1885, related to the Switchback Railway that opened a year earlier at Coney Island. The track in a coaster design does not necessarily have to be a complete circuit, as shuttle roller coasters demonstrate. Most roller coasters have multiple cars in which passengers sit and are restrained. Two or more cars hooked together are called a train. Some roller coasters, notably Wild Mouse roller coasters, run with single cars.

The aim of this project is to simulate a roller-coaster ride using OpenGL. We make use of basic OpenGL primitives to construct various polygons and components that form the big picture. The project showcases both 2D and 3D graphic elements.

CONTENTS

INTRODUCTION

OpenGL (Open Graphics Library) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering. This report contains implementation of ‘ROLLERCOASTER RIDE’ using a set of OpenGL functions. The project consists of different views simulating the typical environment surrounding a roller coaster. The objects are drawn by using GLUT functions.

The project implements basic 2D animation of roller-coaster entrance and a 3D animation of a roller-coaster ride that can be interactively controlled by user using keyboard.

IMPLEMENTATION DETAILS

This Project is on “ROLLERCOASTER RIDE” using OpenGL Functions. It is a User interactive program where in the User can view the required display by making use of the input devices like Keyboard and Mouse. The roller-coaster can be viewed in any direction using Keyboard.

Our Project mainly consists of three important screens:

1. **Initial welcome screen:** This screen displays the name of the group members as well as the topic of the project is displayed. This is implemented using the function *display()* as shown below.

```
void display()
{
  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(0.0,1.0,0.0);
  glRasterPos2i(15,40);
  glRasterPos3f(19.5,40,0);
  Display_on_screen("ROLLERCOASTER RIDE");

  glRasterPos3f(7,26,0);
  Display_on_screen("Keerthesh S");
  glRasterPos3f(34,26,0);
  Display_on_screen("4NM17CS087");

  glRasterPos3f(7,22,0);
  Display_on_screen("Kiran Mahadev Giraddi");
  glRasterPos3f(34,22,0);
  Display_on_screen("4NM17CS088");

  glRasterPos3f(7,18,0);
  Display_on_screen("Kirthi Puthran");
  glRasterPos3f(34,18,0);
  Display_on_screen("4NM17CS089");

   glRasterPos3f(7,14,0);
  Display_on_screen("Kishan");
  glRasterPos3f(34,14,0);
  Display_on_screen("4NM17CS090");

  glEnd();
  glFlush();
}
```

Initial window also has a next button which upon mouse click leads to next screen. The <u>*mymouse()*</u> function implements this feature. A left mouse click will create a new window *Rollercoaster Entrance* i.e., next screen.

```
void mymouse(int button,int state,int x,int y)
{
        if(button==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
        {
        glutCreateWindow("Rollercoaster Entrance");
        init();
        }
}
```

2.Roller-Coaster Entrance Screen: This screen displays the entrance to the roller-coaster. We can see a person queued outside the main entrance gate for a ride. We have animated him buying a ticket from ticket counter which is followed by his entry to the main gate. Below is the list of all the components used to construct the screen

- *sky():* This function is used to implement the background sky and constructed using GL_QUADS and its almost blue in colour.
- *floor() & compoundwall():* display the floor and compound of the building respectively and are both constructed using quadrilateral primitives.
- *gate()*: displays the entrance to the roller-coaster and is made up of several quadrilaterals. When a person buys a ticket, the gate opens and he is ready to enter.
- *board()*: is used to display the billboard which has the mini display of the roller-coaster as an advertisement.
- *ticketcounter() & ticketcounterman()*: are used to display the ticket counter.
- *standingman()*: draws a person who is animated to buy a ticket.

In order to animate the person buying a ticket and opening of the gate, a special function *timer()* is used. This function uses an inbuilt GLUT function *glutTimerFunc();* to move vertices with respect to time and this gives us a view of movement of person and the gate. The implementation of this function can be seen below.

```
void timer(int)
{
   glutPostRedisplay();
   glutTimerFunc(30000/60,timer,0);
   switch(state)
   {
   case 1:
      if(x_position<0.71 && y_position<-0.09)
        {
         x_position+=0.075;
         y_position+=0.1;
        }
      else
         state=-1;
      break;
   case -1:
      if(x_position>0.49 && y_position<0)
         {x_position-=0.075;
         y_position+=0.05;
         }
      else
         if(x_pos>-0.04)
         x_pos-=0.1;
      break;
   }
}
```

3. 3D Rollercoaster Simulation: This window showcases a three-dimensional View of a rollercoaster ride. We can see the outside view, inside view and view from any angle using keyboard.

List of all the keys used in simulation and their respective functions:

- *LEFT Key:* Move the view left
- *RIGHT Key:* Move the view right
- *PLUS(+) Key*: Move the view upside
- *MINUS(-) Key*: Move the view downside
- *UP Key:* Roller-coaster moves forward
- *DOWN Key*: Roller-coaster moves backward
- *R Key*: Start the rollercoaster motion
- *W Key*: Switch view between third person perspective and first person perspective

Bezier curves are used to draw the rollercoaster model and define 3D paths as well as 2D curves for keyframe interpolation. This is implemented in the methods *bezier()* and *moveToBezier()*.

```
double bezier(double x0, double x1, double x2, double x3, double t)
{

        return  0.5*((2.0f*x1)+(-x0+x2)*t+(2.0f*x0-5.0f*x1+4*x2-x3)*t*t+(-x0+3.0f*x1-
3.0f*x2+x3)*t*t*t);
}

void moveToBezier( double t){

          int n=0.0;
          viewer[0]=1.0;
                  viewer[1]=0.0;
                  viewer[2]=0.0;

  if(camw==1)
   {


                getCurveAt(&movcord[0],&movcord[1], &movcord[2], ni, t);
                movcord[0]+=1.0;
                movcord[1]-=3.5;

                 camera[0]=bezier(bez[0+ni][0],bez[1+ni][0],bez[2+ni][0],bez[3+ni]
[0],t+0.1)-bezier(bez[0+ni][0],bez[1+ni][0],bez[2+ni][0],bez[3+ni][0],t);

        camera[1]=bezier(bez[0+ni][1],bez[1+ni][1],bez[2+ni][1],bez[3+ni]
[1],t+0.1)-bezier(bez[0+ni][1],bez[1+ni][1],bez[2+ni][1],bez[3+ni][1],t);

        camera[2]=bezier(bez[0+ni][2],bez[1+ni][2],bez[2+ni][2],bez[3+ni][2],t+0.1)
-bezier(bez[0+ni][2],bez[1+ni][2],bez[2+ni][2],bez[3+ni][2],t);

        if(gy<movcord[1]+2.5)
                  movcord[1]=gy-2.5;
                  display();
                }
}
```

3D simulation of a rollercoaster ride is drawn by combining basic OpenGL elements and Bezier curves and it is mainly implemented in the method *ride3d()*

```
void ride3d(){

	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	//glColor4f(1.0,1.0,1.0,1.0);
	glColor4f(0.0,0.0,0.0,0.0);
	glLoadIdentity();
	gluLookAt(viewer[0], viewer[1], viewer[2],camera[0], camera[1], camera[2],0, 1, 0);
	glRotatef(x_r, 0, 1, 0);

Draw_Skybox(viewer[0]+(0.05*movcord[0]),viewer[1]+(0.05*movcord[1]),viewer[2]
	+(0.05*movcord[2]),250,250,250);

	glTranslatef(movcord[0],movcord[1],movcord[2]);
	draw_ground();
	glPushMatrix();
	glTranslatef(80,0,165);
	draw_tank();
	glPopMatrix();

	double tx,ty,tz,nx,ny,nz;
	getCurveAt(&tx,&ty,&tz,ni,bez_prog+0.058);
	getCurveAt(&nx,&ny,&nz,ni,bez_prog+0.070);
	gy=ny;

	float bz1=bezier(bez[0+ni][2],bez[1+ni][2],bez[2+ni][2],bez[3+ni][2],bez_prog+0.02)-
				1*fabs(cos(angle*3.14/180.0));
	float bx1=bezier(bez[0+ni][0],bez[1+ni][0],bez[2+ni][0],bez[3+ni][0],bez_prog+0.02)-
				1*fabs(sin(angle*3.14/180.0));
	float bz2=bezier(bez[0+ni][2],bez[1+ni][2],bez[2+ni][2],bez[3+ni]
	   [2],bez_prog+0.02)+1*fabs(cos(angle*3.14/180.0));
	float bx2=bezier(bez[0+ni][0],bez[1+ni][0],bez[2+ni][0],bez[3+ni]
[0],bez_prog+0.02)+1*fabs(sin(angle*3.14/180.0));
	double degreer = atan2(1,bx2-bx1)*fabs(sin(angle*3.14/180.0))* 180 /
3.14+fabs(cos(angle*3.14/180.0))*atan2(1,bz2-bz1)* 180 / 3.14;
	double angler = degreer ;
	double degree= atan2(nz-tz, nx-tx);
	angle = degree * 180 / 3.14;
	double degreey= atan2(ny-ty,1);
	double angley = degreey * 180 / 3.14;
	glPushMatrix();
	glTranslatef(-nx,-ny,-nz);
	glRotatef(-angle, 0.0, 1.0, 0.0);
	glRotatef(angley-90, 0, 0, 1.0);
	glRotatef(angler-45, 0.0, 1.0, 0.0);
	if(camw==0) angle=90.0;
	glTranslatef(-2.5, 3.0, 0.0);
	draw_loco();
	glPopMatrix();

	glutSwapBuffers();
}
```

CONCLUSIONS

We have successfully implemented a rollercoaster simulation using OpenGL. This project displays the various components of a rollercoaster environment and is also interactive and makes use of various methods and components of OpenGL. The rollercoaster simulation was carried out in third-person perspective as well as first-person perspective. The camera alignment was designed to support both these modes, which overall helped the user assess the true experience of a rollercoaster from the perspective of a rider, as well as of a person watching it from the ground.

We found designing and developing this project as an interesting and insightful learning experience. It helped us learn about computer graphics, designing of different components, interface to the user, user interaction handling and screen management.

REFERENCES

- https://www.khronos.org/opengl/wiki/Main_Page
- https://www.youtube.com/playlist?list=PLWzp0Bbyy_3jy34HlDrEWlcG3rF99gkvk

APPENDIX

Figure 1: Name USN front page

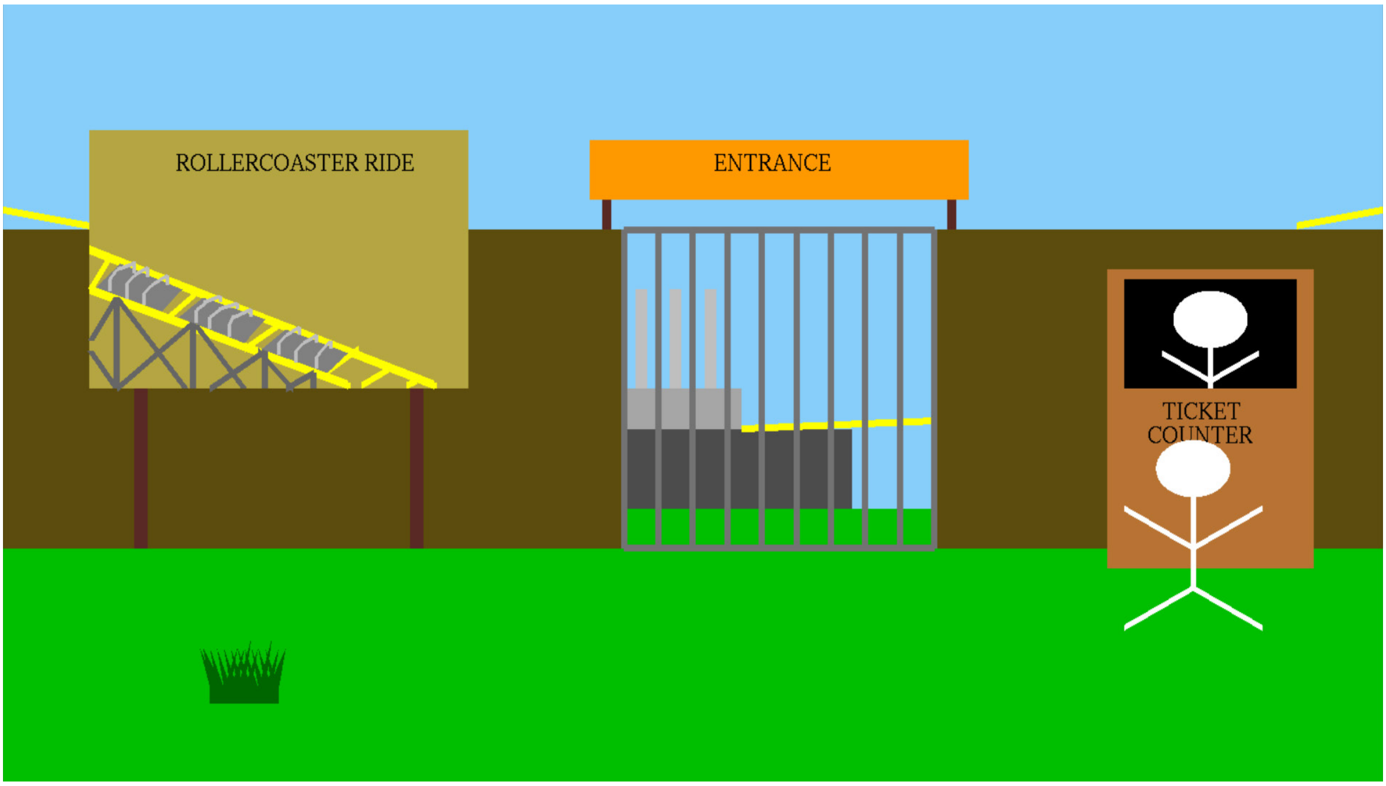

Figure 2: Roller-coaster Entrance – Initial view

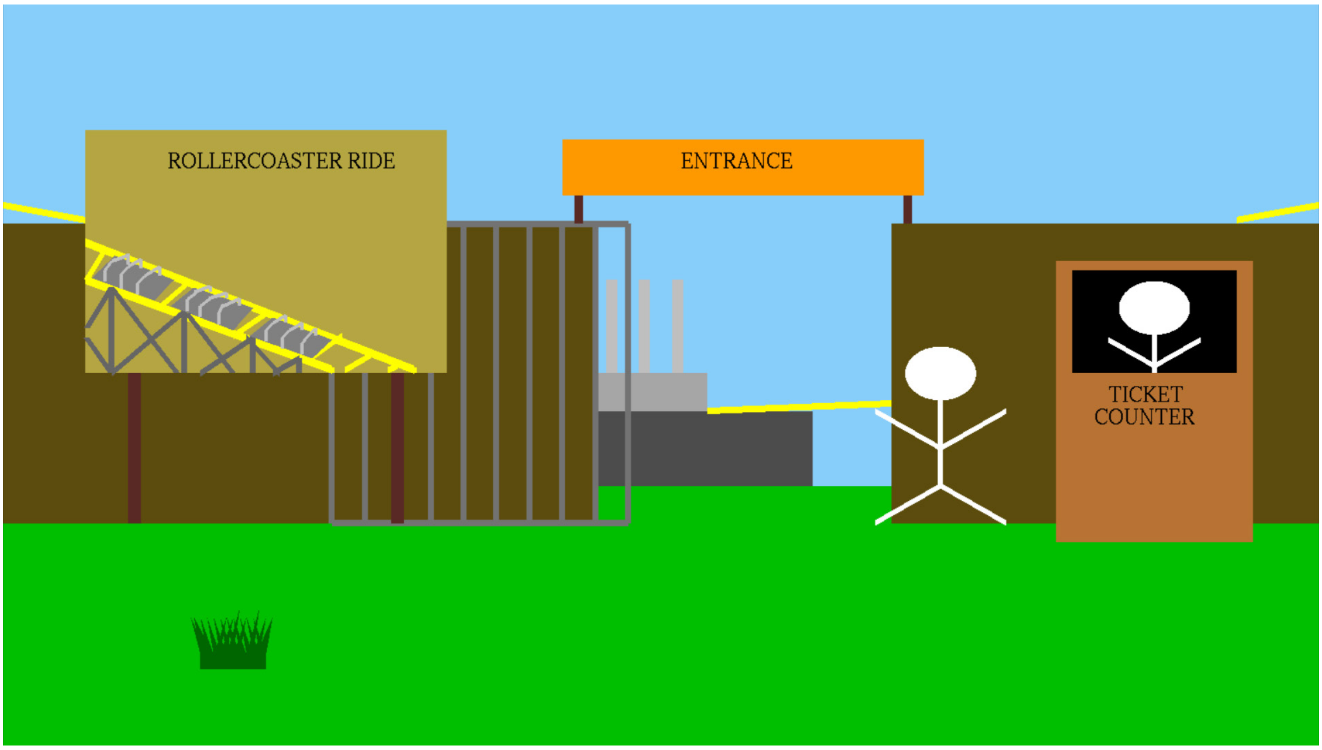

Figure 3: Roller-coaster Entrance – Final View

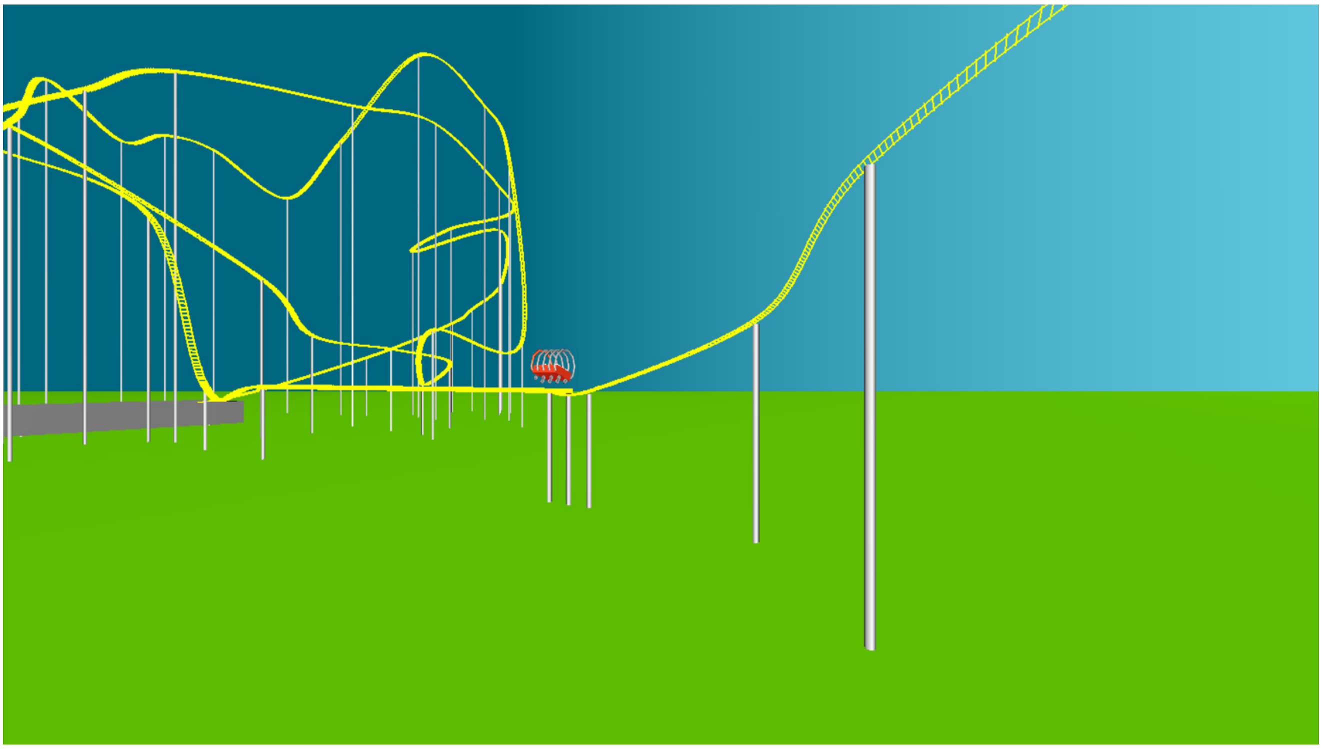

Figure 4: 3D Roller-coaster ride outside view

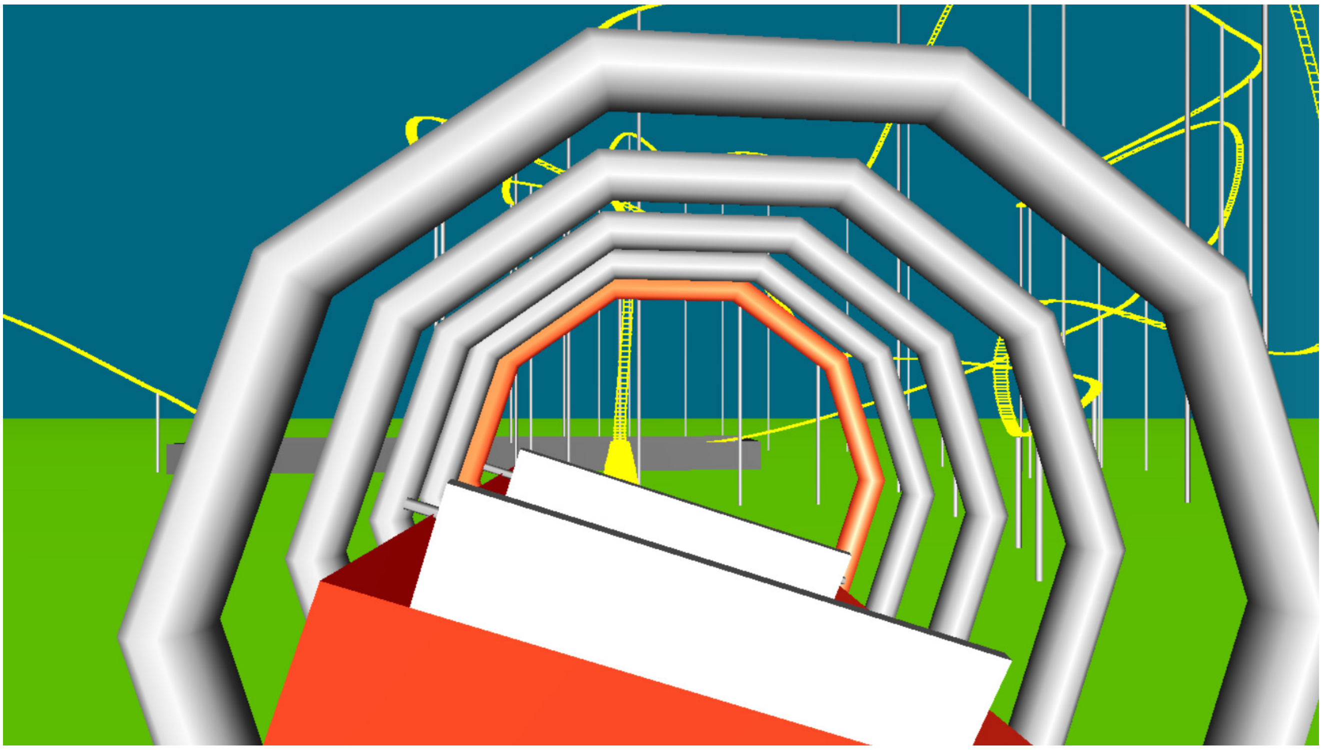

Figure 5: 3D Roller-coaster ride inside view