VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590014, Karnataka, INDIA

A mini project on

"ACTS OF PUBLIC APATHY"

Submitted in partial fulfillment of the requirement of 6th semester

Bachelor of Engineering

In

Computer Science & Engineering

Of

Visvesvaraya Technological University, Belagavi

Submitted by

A N Raviteja [1DS14CS001]
Antra Guha [1DS08CS010]
Anushree G [1DS14CS013]

Under the guidance of

Prof. Sasidhar B
Assistant Professor, Dept of CSE
DSCE, Bengaluru

Prof.Sahana Damale
Assistant Professor, Dept of CSE
DSCE, Bengaluru

2016-2017

Department of computer science & engineering

Dayananda Sagar college of Engineering, Bengaluru-560078

DAYANANDA SAGAR COLLEGE OF ENGINEERING
Shavige Malleswara Hills, Kumarswamy Layout, Bengaluru -560078
Department of Computer Science & Engineering

Certificate

This is certified that the project work entitled **ACTS OF PUBLIC APATHY** for the Computer Graphics and Visualization Lab, is a bonafide work carried out by **DIVAKAR R** (1DS14CS030),**CHANDAN KUMAR B K**(1DS14CS024) and **DHEEMANTH G K** (1DS14CS029) in partial fulfillment of the 6th Semester of Bachelor of Engineering in Computer Science & Engineering of the Visvesvaraya Technological University, Belgavi during the year 2016-2017. It is certified that all corrections/ subjections indicated for the Internal Assessment have been incorporated on the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Mini Project Work prescribed for 6th Semester Computer Graphics and Visualization laboratory.

Signature of Internal guides **Signature of HOD**

1 :……………………………………

2:…………………………………….

External viva:

Name of examiners **Signature with date**

1:……………………………………

2:……………………………………

ABSTRACT

In the modern where the people are very busy with their day to day works, they should also be self-aware of their surroundings and be responsible enough while walking on road.

This project is a simulation in 2D developed using the basic function provided by OpenGL. The standard OpenGL libraries such as GLUT to demonstrate a person not being self-aware of his surroundings and meeting with an accident .And also we have stressed here about a law that has been passed by the government which states that no person shall be held upon and enquired for helping the accident victim .

Acknowledgement

The satisfaction that accompanies the successful completion of our project would be incomplete without mentioning the people who made it possible, whose constant guidance and encouragement crowns all the efforts with success.

We would greatly mention the enthusiastic influence provided by Prof.Shashidhar B, Prof. Swetha M D and Prof. Sahana Damale for their ideas and co-operation showed on us during our venture and making this project a great success.

We are thankful to Dr. Ramesh Babu, HOD, Department of CSE for his co-operation and encouragement at all moments of our approach.

We are also thankful to Dr.C P S Prakash, Principal, DSCE, Bangalore for being kind enough to provide us an opportunity to work on a project in this institution.

Finally, it’s pleasure and happiness to the friendly co-operation showed by all the staff members of Computer Science Department, DSCE.

DIVAKAR R [1DS14CS030]
CHANDAN KUMAR B K [1DS14CS024]
DHEEMANTH G K [1DS14CS029]

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Computer Graphics

The phrase “Computer Graphics” was coined in 1960 by William Fetter, a graphic designer for Boeing. Computer graphics, or simply referred to as CG, is one of the most powerful and interesting facets of computers. There is a lot more to Computer Graphics than just drawing figures of various shapes. It’s the graphics created using computers and, more generally, the representation and manipulation of image data by a computer. The development of computer graphics has made computers easier to interact with, and better for understanding and interpreting many types of data. Developments in computer graphics have had a profound impact on many types of media and have revolutionized animation, movies and the video game industry. It has been used in a broad sense to describe "almost everything on computers that is not text or sound".

All gaming consoles, animations, multimedia make excessive use of graphical capabilities of Computers ,which today, is one of the most widely researched and rapidly developing fields of Computer Science. Activities as diverse as film making, advertising, banking and education continue to undergo revolutionary changes.

Many powerful tools have been developed to visualize data. Computer generated imagery can be categorized into several different types: 2D, 3D, and animated graphics. As technology has improved, 3D computer graphics have become more common, but 2D computer graphics are still widely used. Computer graphics has emerged as a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content. Over the past decade, other specialized fields have been developed like information visualization, and scientific visualization more concerned with "the visualization of three dimensional phenomena (architectural, meteorological, medical, biological, etc.), where the emphasis is on realistic renderings of volumes, surfaces, illumination sources, and so forth, perhaps with a dynamic (time) component".

1.2 Application of Computer Graphics

It is always difficult to try to categorize endeavors in any field, but some of the major fields that computer graphics has an impact on can be summarized as follows:

• Modeling deals with the mathematical specification of shape and appearance properties in a way that can be stored on the computer. For example, a coffee mug might be described as a set of ordered 3D points along with some interpolation rule to connect the points and a reflection model that describes how light interacts with the mug.

• Rendering is a term inherited from art and deals with the creation of shaded images from 3D computer models.

• Animation is a technique to create an illusion of motion through sequences of images. Here, modeling and rendering are used, with the handling of time as a key issue not usually dealt with in basic modeling and rendering.

There are many other areas that involve computer graphics, and whether they are core graphics areas is a matter of opinion. These will all be at least touched on in the text. Such related areas include the following:

• User interaction deals with the interface between input devices such as mice and tablets, the application, and feedback to the user in imagery and other sensory feedback. Historically, this area is associated with graphics largely because graphics researchers had some of the earliest access to the input/output devices that are now ubiquitous.

• Virtual reality attempts to immerse the user into a 3D virtual world. This typically requires at least stereo graphics and response to head motion. For true virtual reality, sound and force feedback should be provided as well. Because this area requires advanced 3D graphics and advanced display technology, it is often closely associated with graphics.

• Visualization attempts to give users insight via visual display. Often there are graphic issues to be addressed in a visualization problem.

• Image processing deals with the manipulation of 2D images and is used in both the fields of graphics and vision.

• 3D scanning uses range-finding technology to create measured 3D models. Such models are useful for creating rich visual imagery, and the processing of such models often requires graphics algorithms.

Almost any endeavor can make some use of computer graphics, but the major consumers of computer graphics technology include the following industries:

• Video games increasingly use sophisticated 3D models and rendering algorithms.

• Cartoons are often rendered directly from 3D models. Many traditional 2D cartoons use backgrounds rendered from 3D models which allows a continuously moving viewpoint without huge amounts of artist time.

• Film special effects use almost all types of computer graphics technology. Almost every modern film uses digital compositing to superimpose backgrounds with separately filmed foregrounds. Many films use computer-generated foregrounds with 3D models.

• CAD/CAM stands for computer-aided design and computer-aided manufacturing. These fields use computer technology to design parts and products on the computer and then, using these virtual designs, to guide the manufacturing procedure. For example, many mechanical parts are now .designed in a 3D computer modeling package, and are then automatically produced on a computer-controlled milling device.

• Simulation can be thought of as accurate video gaming. For example, a flight simulator uses sophisticated 3D graphics to simulate the experience of flying an airplane. Such simulations can be extremely useful for initial training in safety-critical domains such as driving, and for
scenario training for experienced users such as specific fire-fighting situations that are too costly or dangerous to create physically.

• Medical imaging creates meaningful images of scanned patient data. For example, a magnetic resonance imaging (MRI) dataset is composed of a 3D rectangular array of density values. Computer graphics is used to create shaded images that help doctors digest the most salient information from such data.

• Information visualization creates images of data that do not necessarily have a "natural" visual depiction. For example, the temporal trend of the price of ten different stocks does not have an obvious visual depiction, but clever graphing techniques can help humans find patterns in such data.

1.3 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. 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.

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.

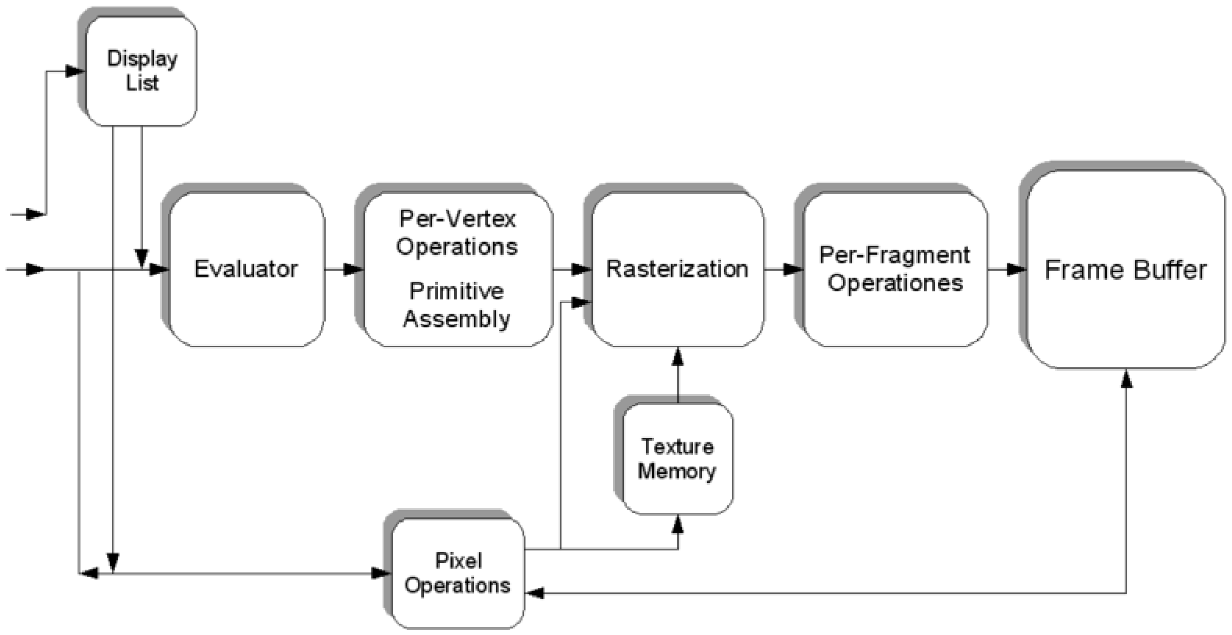

Figure 1.1 OpenGL Graphics Pipeline

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.

1.4 Statement of the Problem

To design and stimulate the idea of Universal Design for handicapped in buses, trains, agriculture and school.

1.5 Objective of the Project Motivation

One of the main objectives of the Project is to explore the features provided by the OpenGL API to design the graphics and to visually illustrate the Universal Design For Handicapped

CHAPTER 2
SYSTEM SPECIFICATION

System requirements are intended to communicate in precise way, the functions that the system must provide. To reduce ambiguity, they may be written in a structured form of natural language supplemented by tables and system models.

2.1 Hardware Requirements

The physical components required are:

- Processor – Intel core i5
- Memory - 128MB RAM
- 40GB Hard Disk Drive
- Mouse or other pointing device
- Keyboard

2.2 Software Requirements

The software used in building this program are as specified:-

- Operating system – Windows 10.
- Microsoft Visual C++ editor.
- Compiler – C++ Compiler.
- Graphics Library – glut.h
- OpenGL 2.0

CHAPTER 3
ANALYSIS

3.1 Analysis

The flow diagram represents the working of the project. The starts symbol indicates the beginning of execution. The display screen consists of an option to start the project, to know the details of control and quit the project which is done by keyboard. Once the project is started the animation will be displayed automatically and the keyboard event will provide the next slide as well as previous slides. The user manual which specifies all the options available is also displayed at the beginning.

CHAPTER 4
DESIGN

4.1 Flow diagram

Flow diagram is a collective term for a representing a or set of dynamic in a. The term flow diagram is also used as synonym of the flowchart and sometimes as counterpart of the flowchart. (UD-Universal Design)

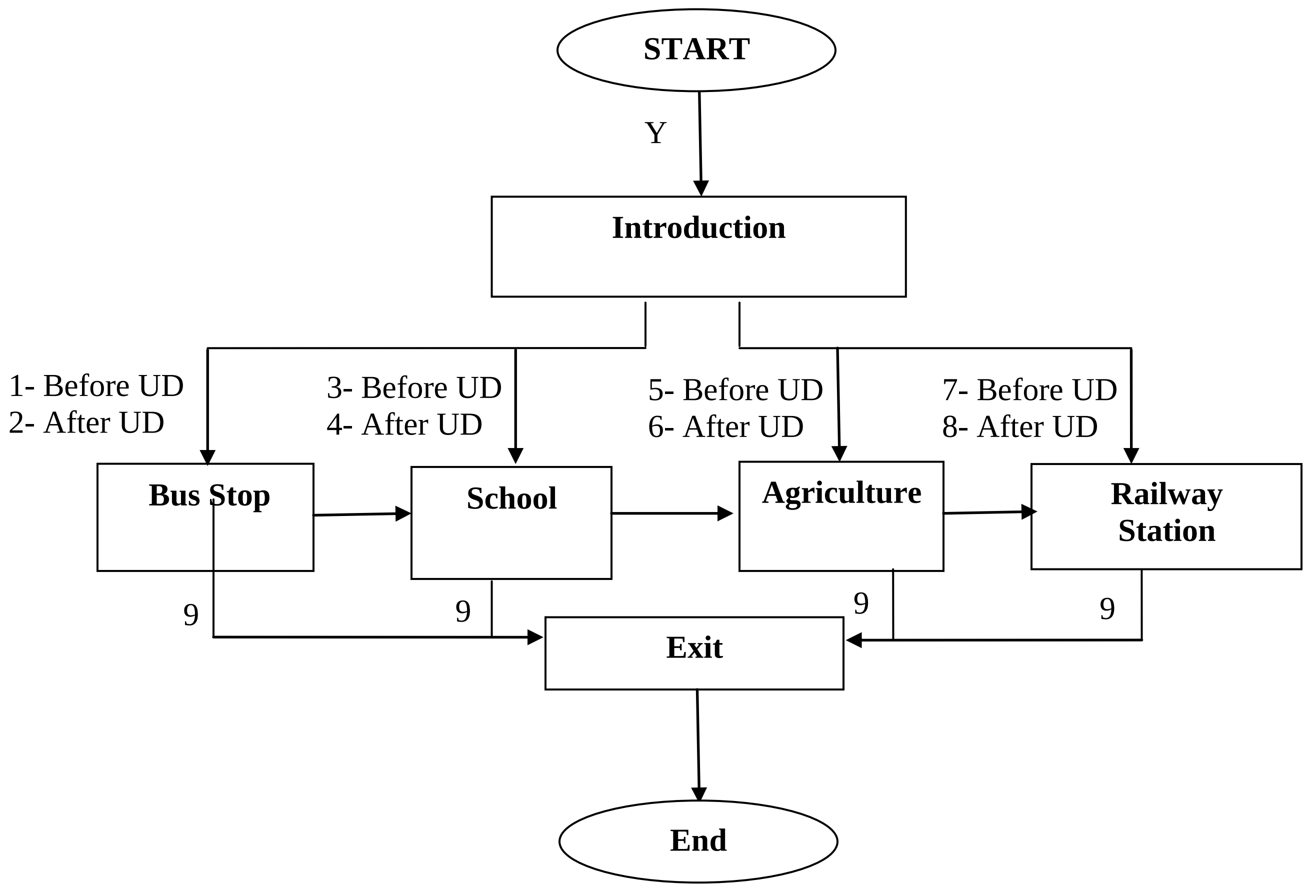

Fig:4.1

4.2 Description of Flow Diagram

- The flow of the program starts from the main function. The buffer memory is cleared, output screen size is fixed and the window is created.
- Then the display function calls all functions of all the scenes. Based on the user input from the keyboard the next slide to be displayed is decided.
- Function display() is used to call all the slides based on the '1','2','3'or'4'.
- Functions bus stop, school, agriculture and railway station are called to illustrate the respective scenes.
- In order to provide motion for the object, idle function was used.

Chapter-5
IMPLEMENTATION

The various built in functions and user defined function are used for the implementation of this projects. Few of the function used are:

5.1Built In Functions

- **glutInit( )**

It should be called before any other GLUT routine because it initializes the GLUT library. The parameters to glutInit() should be the same as those to main(), specifically main(int argc, char** argv) and glutInit(&argc, argv).

- **glutInitDisplayMode( )**

The next thing we need to do is call the glutInitDisplayMode() procedure to specify the display mode for a window. You must first decide whether you want to use an RGBA (GLUT_RGBA) or color-index (GLUT_INDEX) color model. The RGBA mode stores its color buffers as red, green, blue, and alpha color components. The forth color component, alpha, corresponds to the notion of opacity. Another decision you need to make when setting up the display mode is whether you want to use single buffering (GLUT_SINGLE) or double buffering (GLUT_DOUBLE). Applications that use both front and back color buffers are double-buffered.

- **glutInitWindowSize( )**

A call to glutInitWindowSize() will be used to specify the size, in pixels, of your initial window. The arguments indicate the height and width (in pixels) of the requested window.

- **glutCreateWindow( )**

To actually create a window, with the previously set characteristics (display mode, size, location, etc), the programmer uses the glutCreateWindow() command. It takes a string as a parameter, which may appear in the title bar if the window system you are using, supports it. The window is not actually displayed until the glutMainLoop() is entered.

glutDisplayFunc()

The glutDisplayFunc() procedure is an important event callback function. A callback function is one where a programmer-specified routine can be registered to be called in response to a specific type of event.

- **glutMainLoop()**

The very last thing you must do is call glutMainLoop(). All windows that have been created can now be shown, and rendering those windows is now effective. In addition, the registered display callback (from our glutDisplayFunc()) is triggered. Once this loop is entered, it is never exited.

- **glClearColor() and glClear()**

Drawing on a computer screen is different from drawing on paper in that the paper starts out white, and all you have to do is draw the pictureTo change the background color, we call and specify the color we have chosen for the background.. To clear the color buffer use the argument GL_COLOR_BUFFER_BIT.

- **glMatrixMode()**

Specifies whether the model view, projection or texture matrix will be modified using the arguments GL_MODEL_VIEW, GL_PROJECTION for mode. The subsequent transformation command affects the specified matrix.

- **glutKeyboardFunc()**

Sets the special keyboard callback for the current window. It is triggered when the key is pressed.

5.2 USER DEFINED FUNCTIONS (MODULES)

drawCircle: Used to draw circle for the given given radius**.**

```
void drawCircle(float x1,float y1, double radius)
{
        float x2,y2;
        float angle;
        glPointSize(1);
        glBegin(GL_TRIANGLE_FAN);
        glVertex2f(x1,y1);
        for (angle=0.0f;angle<=3*3.14159f;angle+=0.2)
        {
                x2 = x1+sin(angle)*radius;
                y2 = y1+cos(angle)*radius;
                glVertex2f(x2,y2);
        }
        glEnd();
}
```

drawText: Used for writing text at the point where coordinates are given.

```
void drawText( float x, float y, int r, int g, int b, const char *string )
{       int j = strlen( string );
        glColor3f( r, g, b );
        glRasterPos2f( x, y );
        for( int i = 0; i < j; i++ ) {
        glutBitmapCharacter( GLUT_BITMAP_TIMES_ROMAN_24, string[i] );
        }
}
```

Bus : includes the code to draw bus and the movement of the bus.

```
void bus(float p1,float p2,float p3,int s,int i)
{
   glColor3f(0.5,0.5,0.5);
         drawCircleC(167.0,217.0,15.0,-7,0);
   glColor3f(0,0,0);
         drawCircleC(166.0,212.0,5.0,-2,0);
         glPushMatrix();
         glTranslatef(166.0,212.0,0);
         glRotatef(i,0,0,1);
   drawCircleR(6);
         glPopMatrix();
         glColor3f(0.376,0.376,0.376);
         glBegin(GL_POLYGON);
         glVertex2f(150.0,300.0);
         glVertex2f(350.0,400.0);
         glVertex2f(380.0,350.0);
         glVertex2f(200.0,250.0);
         glEnd();
         glColor3f(p1,p2,p3);
         glBegin(GL_POLYGON);
         glVertex2f(150.0,300.0);
         glVertex2f(150.0,225.0);
         glVertex2f(200.0,175.0);
         glVertex2f(200.0,250.0);
         glEnd();
         glBegin(GL_POLYGON);
         glVertex2f(380.0,350.0);
         glVertex2f(380.0,280.0);
         glVertex2f(200.0,175.0);
         glVertex2f(200.0,250.0);
```

```
        glEnd();
        glColor3f(0.0,0.0,0.0);
        glBegin(GL_LINES);
        glVertex2f(150.0,300.0);
        glVertex2f(150.0,225.0);
        glVertex2f(200.0,175.0);
        glVertex2f(200.0,250.0);
        glEnd();
        glBegin(GL_LINES);
        glVertex2f(380.0,350.0);
        glVertex2f(380.0,280.0);
        glVertex2f(200.0,175.0);
        glVertex2f(200.0,250.0);
        glEnd();
        glBegin(GL_LINES);
        glVertex2f(150.0,300.0);
        glVertex2f(200.0,250.0);
        glEnd();
        glLineWidth(1.0);
        glColor3f(0.5,0.5,0.5);
        drawCircleC(220.0,187.0,15.0,-7,0);
        drawCircleC(350.0,265.0,15.0,-7,0);
  glColor3f(0.0,0.0,0.0);
  drawCircleC(220.0,187.0,5.0,-2,0);
        drawCircleC(350.0,265.0,5.0,-2,0);
        glPushMatrix();
        glTranslatef(220.0,187.0,0);
        glRotatef(i,0,0,1);
  drawCircleR(8);
        glPopMatrix();
        glPushMatrix();
        glTranslatef(350.0,265.0,0);
        glRotatef(i,0,0,1);
        drawCircleR(8);
        glPopMatrix();
        glColor3f(1.0,1.0,1.0);
glBegin(GL_POLYGON);
        glVertex2f(270.0,260.0);
        glVertex2f(270.0,240.0);
        glVertex2f(280.0,247.0);
        glVertex2f(280.0,267.0);
        glEnd();
glBegin(GL_POLYGON);
        glVertex2f(290.0,272.0);
        glVertex2f(290.0,252.0);
        glVertex2f(300.0,259.0);
        glVertex2f(300.0,279.0);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex2f(310.0,284.0);
        glVertex2f(310.0,264.0);
        glVertex2f(320.0,271.0);
        glVertex2f(320.0,291.0);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex2f(330.0,296.0);
        glVertex2f(330.0,276.0);
        glVertex2f(340.0,283.0);
        glVertex2f(340.0,303.0);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex2f(350.0,308.0);
        glVertex2f(350.0,288.0);
        glVertex2f(360.0,295.0);
        glVertex2f(360.0,315.0);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex2f(370.0,320.0);
        glVertex2f(370.0,300.0);
        glVertex2f(380.0,307.0);
        glVertex2f(380.0,327.0);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex2f(235.0,250.0);
        glVertex2f(235.0,195.0);
        glVertex2f(255.0,207.5);
        glVertex2f(255.0,262.0);
        glEnd();
        glColor3f(0.0,0.0,0.0);
        glBegin(GL_LINES);
        glVertex2f(235.0,250.0);
        glVertex2f(235.0,195.0);
        glVertex2f(235.0,195.0);
        glVertex2f(255.0,207.5);
        glVertex2f(255.0,207.5);
        glVertex2f(255.0,262.0);
        glVertex2f(255.0,262.0);
        glVertex2f(235.0,250.0);
```

Man : Man code along with his movement along x-direction.

```
void man(void)
{
   glColor3f(0.917,0.752,0.525);
        drawCircle(250,87.5,7.5);//face
```

```
glBegin(GL_POLYGON);
glVertex2f(248,80);
glVertex2f(248,75);
glVertex2f(252,75);
glVertex2f(252,80);
glEnd();//neck
glColor3f(0.917,0.752,0.525);
        glBegin(GL_POLYGON);
glVertex2f(243,71);
glVertex2f(241,76);
glVertex2f(240,76);
glVertex2f(241,67);
glEnd();//hand
glBegin(GL_POLYGON);
glVertex2f(257,71);
glVertex2f(251,63);
glVertex2f(251,61);
glVertex2f(259,67);
glEnd();//hand
glColor3f(0.0,0.0,1.0);
glBegin(GL_POLYGON);
glVertex2f(248,75);
glVertex2f(243,71);
glVertex2f(241,67);
glVertex2f(248,73);
glEnd();//shoulder
glBegin(GL_POLYGON);
glVertex2f(252,75);
glVertex2f(257,71);
glVertex2f(259,67);
glVertex2f(252,73);
```

```
        glEnd();//shoulder
                glBegin(GL_POLYGON);
        glVertex2f(248,75);
        glVertex2f(245,60);
        glVertex2f(255,60);
        glVertex2f(252,75);
        glEnd();//body
glColor3f(1.0,1.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(245,60);
        glVertex2f(242,48);
        glVertex2f(247,48);
        glVertex2f(250,60);
        glEnd();//leg
        glBegin(GL_POLYGON);
        glVertex2f(255,60);
        glVertex2f(258,48);
        glVertex2f(253,48);
        glVertex2f(250,60);
        glEnd();//leg
        glColor3f(0.0,0.0,0.0);
        glBegin(GL_POLYGON);
        glVertex2f(242,48);
        glVertex2f(247,48);
        glVertex2f(247,45);
        glVertex2f(242,45);
        glEnd();//shoe
        glBegin(GL_POLYGON);
        glVertex2f(253,48);
        glVertex2f(258,48);
        glVertex2f(258,45);
        glVertex2f(253,45);
        glEnd();//shoe
  }
```

WheelChair: The code to draw wheel chair along the movement of the wheels(circle).

```
void wheelChair(int i)
{
//H-capped person
        glLineWidth(1.0);
         glColor3f(0.917,0.752,0.525);
        drawCircle(340,116,10);
        glBegin(GL_POLYGON);
        glVertex2f(342,107);
        glVertex2f(338,107);
        glVertex2f(338,102);
        glVertex2f(342,102);
        glEnd();//neck
   glColor3f(1.0,1,0);
                glBegin(GL_POLYGON);
        glVertex2f(345,102);
        glVertex2f(335,102);
        glVertex2f(327,80);
        glVertex2f(347,75);
        glEnd();//body
        glColor3f(0,0,1);
        drawCircleL(345,70,7);
        glBegin(GL_POLYGON);
        glVertex2f(347,75);
        glVertex2f(347,65);
        glVertex2f(325,65);
        glVertex2f(325,80);
        glEnd();//legs
```

```
glColor3f(0.917,0.752,0.525);
        glBegin(GL_POLYGON);
        glVertex2f(320,77);
        glVertex2f(315,50);
        glVertex2f(320,50);
        glVertex2f(323,65);
        glEnd();//legs
        drawCircleK(325,72.5,5);
        glBegin(GL_POLYGON);
        glVertex2f(330,67);
        glVertex2f(327,67);
        glVertex2f(327,50);
        glVertex2f(330,50);
        glEnd();//legs
                    glColor3f(0,0,1);
                    glBegin(GL_POLYGON);
        glVertex2f(340,95);
        glVertex2f(337,95);
        glVertex2f(334,85);
        glVertex2f(340,85);
        glEnd();//hand
        glColor3f(0.57,0.42,0.18);
        glBegin(GL_POLYGON);
        glVertex2f(340,85);
        glVertex2f(323,82);
        glVertex2f(323,80);
        glVertex2f(340,80);
        glEnd();//hand
      //chair
        glColor3f(0.7,0.13,0.13);
        glBegin(GL_POLYGON);
        glVertex2f(350,60);
        glVertex2f(350,120);
        glVertex2f(353,120);
        glVertex2f(353,60);
        glEnd();
glBegin(GL_POLYGON);
        glVertex2f(350,63);
        glVertex2f(350,58);
        glVertex2f(320,63);
        glVertex2f(320,68);
        glEnd();
        glColor3f(0.7,0.13,0.13);
        drawCircleW(332,42,18);
drawCircleW(342,42,18);
        glColor3f(0,0,0.56);
        glPushMatrix();
        glTranslatef(332.0,42.0,0);
        glRotatef(i,0,0,1);
        drawCircleR(18);
glPopMatrix();
        glPushMatrix();
        glTranslatef(342.0,42.0,0);
        glRotatef(i,0,0,1);
        drawCircleR(18);
        glPopMatrix();
        glColor3f(0,0,0.5);
        drawCircle(332,42,3);
        drawCircle(342,42,3);
                glColor3f(0.7,0.13,0.13);
        glLineWidth(2.0);
glBegin(GL_LINES);
```

```
        glVertex2f(346,64);
   glVertex2f(346,80);
        glVertex2f(346,80);
        glVertex2f(322,85);
        glVertex2f(322,85);
        glVertex2f(322,66.5);
        glEnd();
        glLineWidth(1.0);
        glBegin(GL_LINES);
        glVertex2f(350,115);
   glVertex2f(360,115);
   glEnd();
        glColor3f(0,0,0);
                glLineWidth(4.0);
                glBegin(GL_LINES);
        glVertex2f(356,115.5);
        glVertex2f(360,115.5);
        glVertex2f(360,117.5);
        glVertex2f(360,105.5);
        glEnd();
        }
```

grass: For agriculture scene, the field is done using this function.

```
void grass(int x,int y,int width,int height)
{
        int i=x;
        glColor3f(0,1,0);
        for(int l=0;l<height;l++)
        {
                for(int k=0;k<width;k++)
                {
                        glBegin(GL_TRIANGLES);
                        glVertex2f(x,y);
                        glVertex2f(x+10,y);
                        glVertex2f(x+5,y+15);
                        glVertex2f(x,y);
                        glVertex2f(x+12,y);
                        glVertex2f(x-5,y+10);
                        glVertex2f(x-3,y);
                        glVertex2f(x+10,y);
                        glVertex2f(x+15,y+10);
                        glEnd();
                        x+=20;
                }
                y+=15;
                x=i;
        }}
```

leaves: In agriculture scene, the coconut tree leaves is made through this function.

```
void leaves(float x,float y)
{
        glColor3f(0.2,1.0,0.1);
        glLineWidth(2.0);
        for(float x1=x,y1=y,i=0;i<20;i++,x1+=(250-x)/20,y1+=(350-y)/20)
        {
        glBegin(GL_LINES);
        glVertex2f(x1,y1);
        glVertex2f(x1-3,y1+10);
        glVertex2f(x1,y1);
        glVertex2f(x1-3,y1-10);
        glEnd();
        }
        glColor3f(0.69,0.13,0.13);
        glBegin(GL_LINES);
        glVertex2f(250,350);
        glVertex2f(x,y);
        glEnd();
}
```

kathi: The knife in agriculture scene is made from this function.

```
void kathi(void)
{
        glColor3f(0,0,0);
        glBegin(GL_POLYGON);
        glVertex2f(255,294);
        glVertex2f(258,294);
        glVertex2f(264,315);
        glVertex2f(259,315);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex2f(264,315);
        glVertex2f(259,315);
        glVertex2f(257,320);
        glEnd();
        glLineWidth(2.0);
        glBegin(GL_LINES);
        glVertex2f(257,294);
        glVertex2f(255,288);
        glEnd();
        glLineWidth(1.0);
}
```

Machine: The code for a machine used to climb the tree.

```
void machine(void)
{
        glColor3f(0,0,0);
```

```
 drawCircleC(245,150,5,0,2);
 glColor3f(0.5,0.5,0.5);
 glBegin(GL_POLYGON);
 glVertex2f(242,158);
 glVertex2f(225,70);
 glVertex2f(195,70);
 glVertex2f(178,158);
 glEnd();
 glLineWidth(2.0);
 glBegin(GL_LINES);
 glVertex2f(230,103);
 glVertex2f(260,103);
 glVertex2f(230,108);
 glVertex2f(250,108);
 glEnd();
glLineWidth(2.0);
 glBegin(GL_LINE_LOOP);
 for (int angle=270;angle<450;angle++)
 {
         float d=angle*(3.14/180);
         glVertex2f(cos(d)*(3)+260,sin(d)*3+105);
 }
 glEnd();
 glLineWidth(1.0);
 glColor3f(1,0.87,0.67);
 drawCircle(220,190,12);
 glBegin(GL_POLYGON);
 glVertex2f(222,180);
 glVertex2f(222,170);
 glVertex2f(218,170);
 glVertex2f(218,180);
```

```
        glEnd();//neck
        glColor3f(0.957,0.643,0.376);
        glBegin(GL_POLYGON);
        glVertex2f(223,170);
        glVertex2f(217,170);
        glVertex2f(215,158);
        glVertex2f(230,158);
        glEnd();//body
}
```

tracks: Railway tracks in railway station scene is made from this function.

```
void tracks(void)
{
                glColor3f(0,0,0);
        glBegin(GL_POLYGON);
        glVertex2f(0,153);
   glVertex2f(500,348.1);
        glVertex2f(500,448.1);
        glVertex2f(0,253);
        glEnd();
        glColor3f(0.827,0.827,0.827);
        glLineWidth(5.0);

        glBegin(GL_LINES);
        for(int i=0;i<=31;i++)
        {
        glVertex2f((15*i)+20.0,(6.1*i)+157.0);
        glVertex2f((15*i)+20.0,(6.1*i)+257.0);
        }
        glEnd();
        glBegin(GL_LINES);
        glVertex2f(0,153);
   glVertex2f(500,348.1);
        glVertex2f(0,253);
        glVertex2f(500,448.1);
        glEnd();
}
```

train: Train in railway station scene is made from this function.

```
void train(void)
{
                  glColor3f(1.0,0.0,0.0);
                  glLineWidth(1.0);
         glBegin(GL_POLYGON);
         glVertex2f(700.0,515.0);
         glVertex2f(200,370.0);
         glVertex2f(160.0,220.0);
         glVertex2f(700.0,425.0);
         glEnd();
         glColor3f(1,1,1);
         glBegin(GL_POLYGON);
   glVertex2f(160.0,220.0);
         glVertex2f(205.0,235.0);
         glVertex2f(205.0,255.0);
         glVertex2f(164.0,235.0);
         glEnd();
         //glColor3f(0.0,1.0,0.0);
         glBegin(GL_POLYGON);
         glVertex2f(190.0,315.0);
         glVertex2f(220,325);
         glVertex2f(228,359);
         glVertex2f(200,350);
   glEnd();
}
```

Movement function: The movement for the objects in our code is done through translate function as shown below. Here the object is bus(), you can use any object and give the direction in (x,y,z) for movement of the object in that direction. And u have to use translate function with for loop .

```
void movement(void)
{
int i;
        for(i=0;i<1200;i+=1)
        {
        glClear(GL_COLOR_BUFFER_BIT);
                glPushMatrix();
                glTranslatef(-(0.0+i)/10,-(0.0+i)/10,0.0);
                bus(0,0.5,1,0,i);
                glPopMatrix();
                glFlush();
                glutSwapBuffers();
        }
}
```

CHAPTER 6
TESTING

S.No	Module	Expected Output	Actual Output	Status
1.	Module1	Objects: Bus, clouds, man in wheelchair, background. Action: Moving bus without plank ,floating clouds, People in the bus stop.	Objects: Bus, clouds, man in wheelchair, background. Action: Moving bus without plank ,floating clouds, People in the bus stop.	Pass
2.	Module2	Objects: Bus, buildings, stars ,man in wheelchair, background. Action: Moving bus with plank ,twinkling stars, People in the bus stop and man in the wheel chair getting into the bus.	Objects: Bus, buildings, stars ,man in wheelchair, background. Action: Moving bus with plank ,twinkling stars, People in the bus stop and man in the wheel chair getting into the bus	Pass
3.	Module3	Object: Stairs, ground, Wheelchair, students Action: Students going to class through steps	Object: Stairs, ground, Wheelchair, students Action: Students going to class through steps	Pass
4.	Module4	Object: Stairs, ground, Wheelchair, students Action: Students going to class through steps and the man in wheelchair going to the class using slope.	Object: Stairs, ground, Wheelchair, students Action: Students going to class through steps and the man in wheelchair going to the class using slope.	Pass
5.	Module5	Objects: Coconut tree, knife,	Objects: Coconut tree, knife,	Pass

		grass, farmer, mountains Action: A man climbing coconut tree and falling down.	grass, farmer, mountains Action: A man climbing coconut tree and falling down.	
6	Module 6	Objects: Coconut tree, knife, grass, farmer, mountains Action: A man climbing tree with help of a machine.	Objects: Coconut tree, knife, grass, farmer, mountains Action: A man climbing tree with help of a machine.	Pass
7	Module 7	Objects: Train, man in the wheelchair, other passengers, clouds, Sun, tree Action: A moving train, people in the railway station, floating clouds.	Objects: Train, man in the wheelchair, other passengers, clouds, Sun, tree Action: A moving train, people in the railway station, floating clouds.	
8.	Module8	Objects: Train, man in the wheelchair, Signal control office, other passengers, clouds, Action: A moving train, people in the railway station, floating clouds	Objects: Train, man in the wheelchair, Signal control office, other passengers, clouds, Action: A moving train, people in the railway station, floating clouds	Pass

CHAPTER 7

SNAPSHOTS

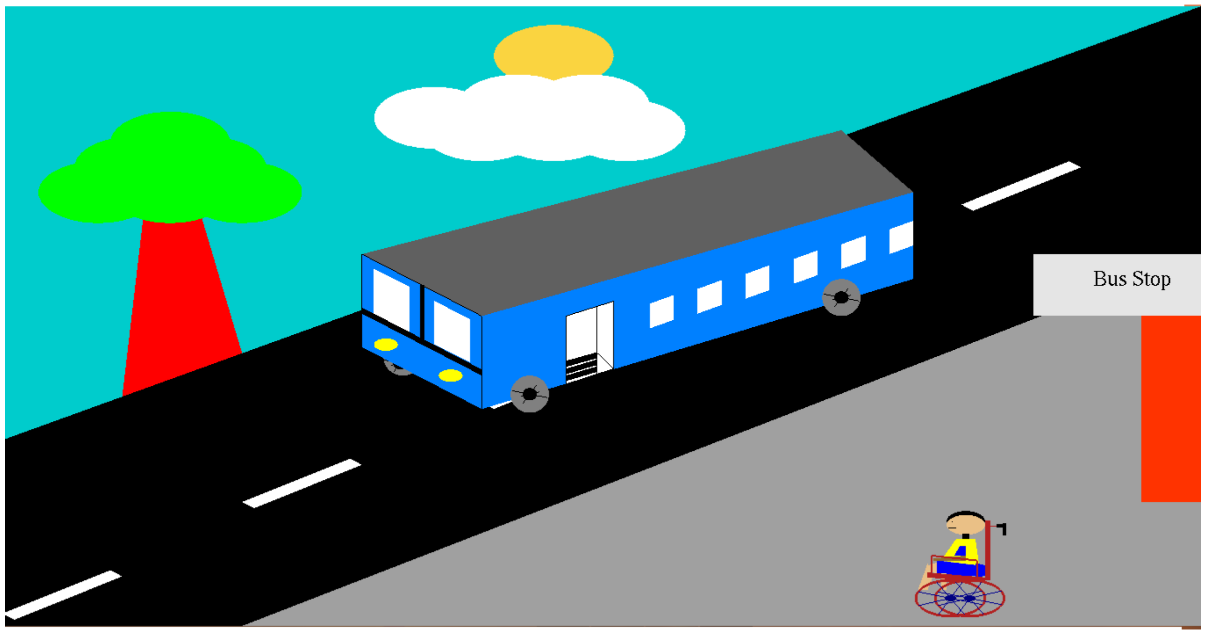

Fig 1: Bus scene(Before Universal design)

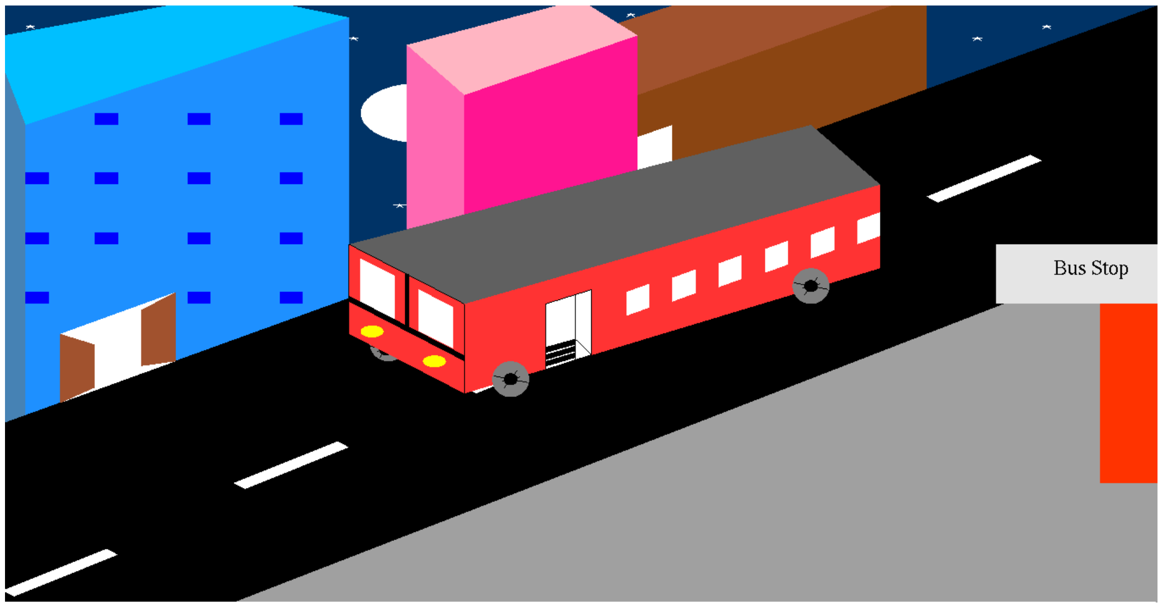

Fig 2: Bus scene(After Universal design)

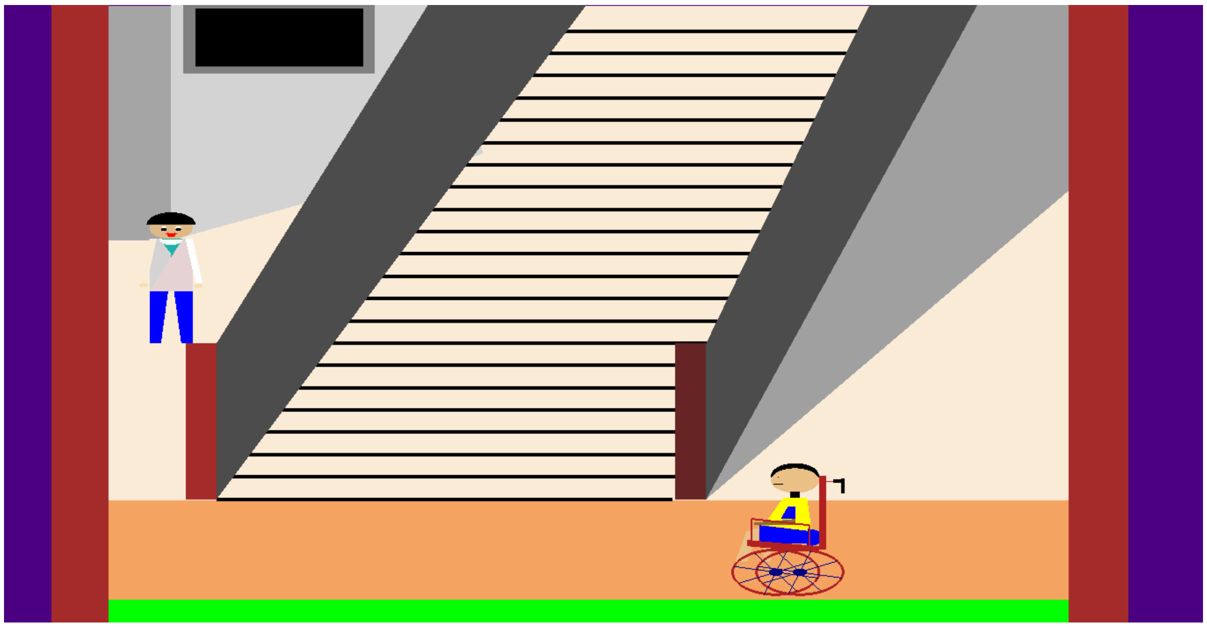
Fig 3: School scene(Before Universal design)

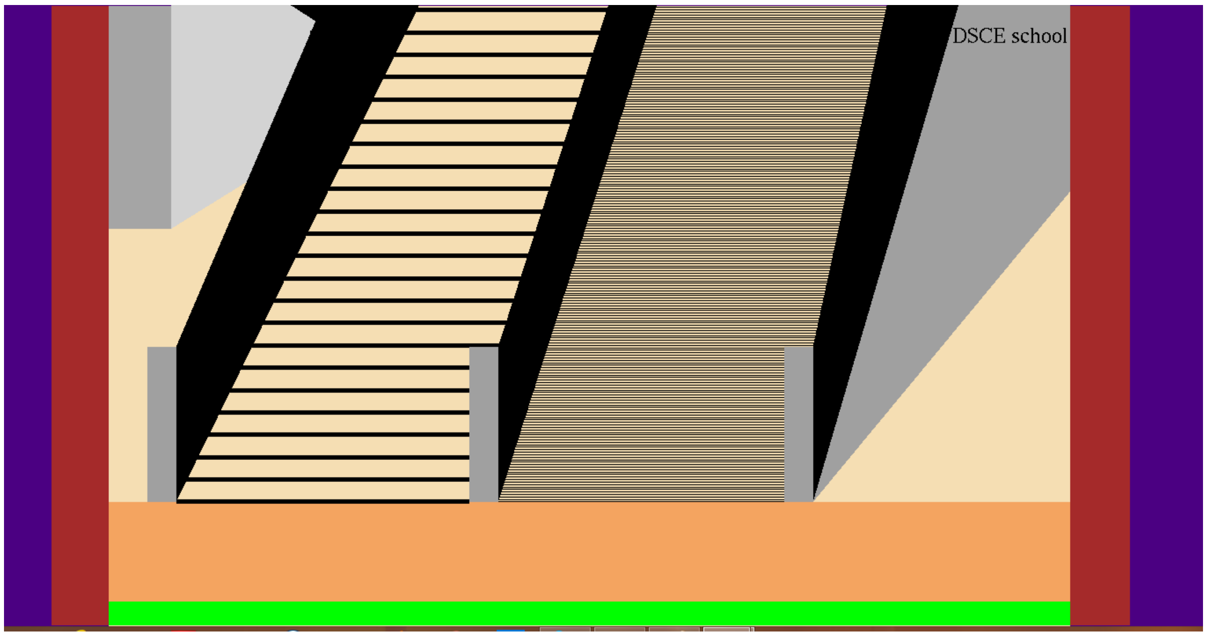

Fig 4: School Scene (After Universal design)

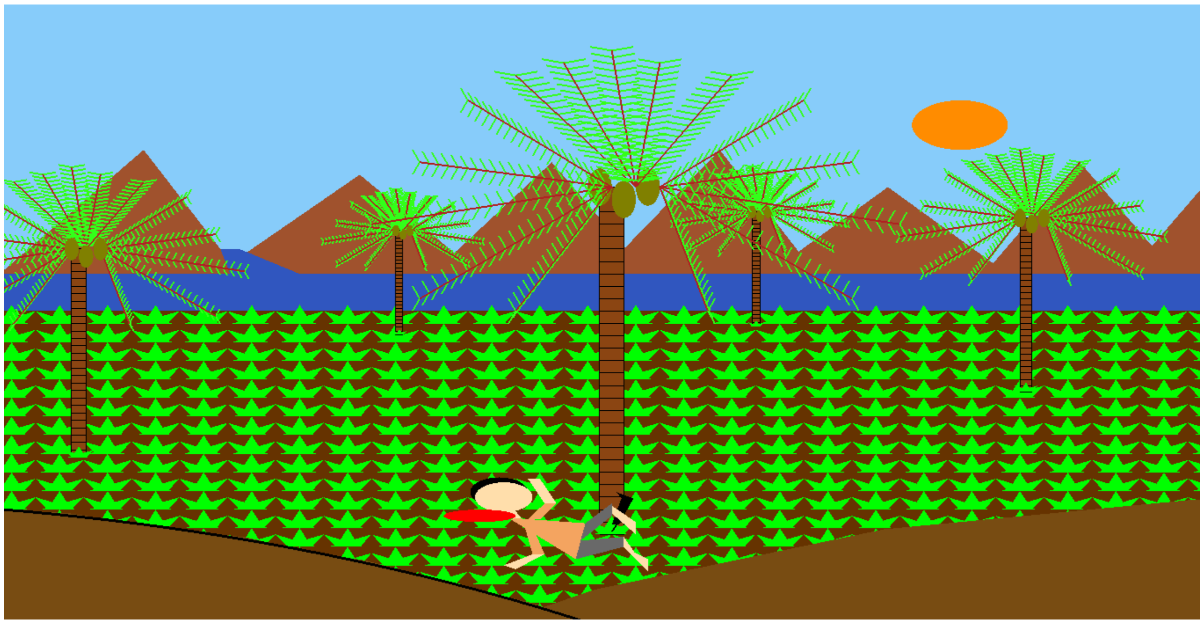

Fig 5: Agriculture Scene(Before Universal design)

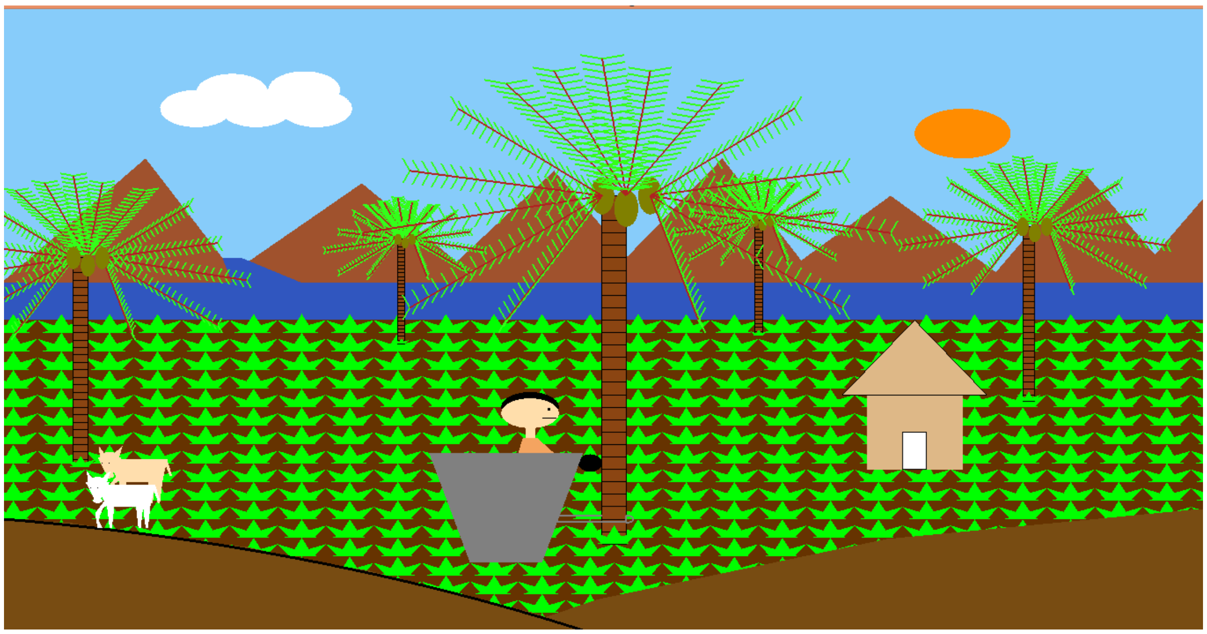

Fig 6: After Universal design

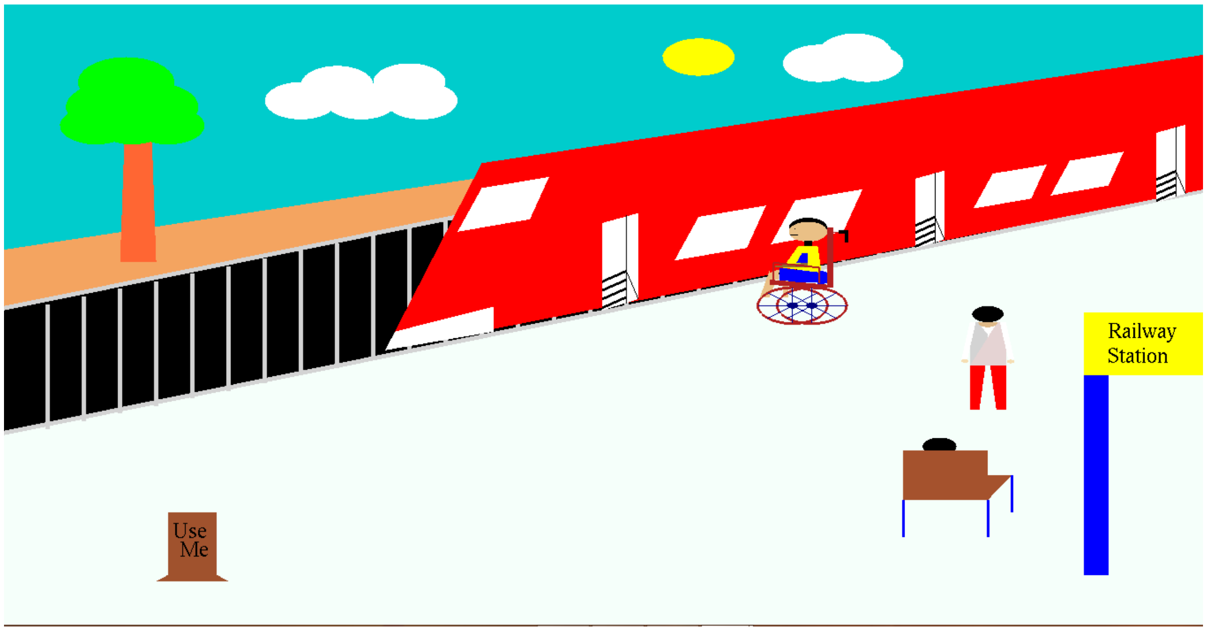

Fig 7: Train Scene(Before Universal design)

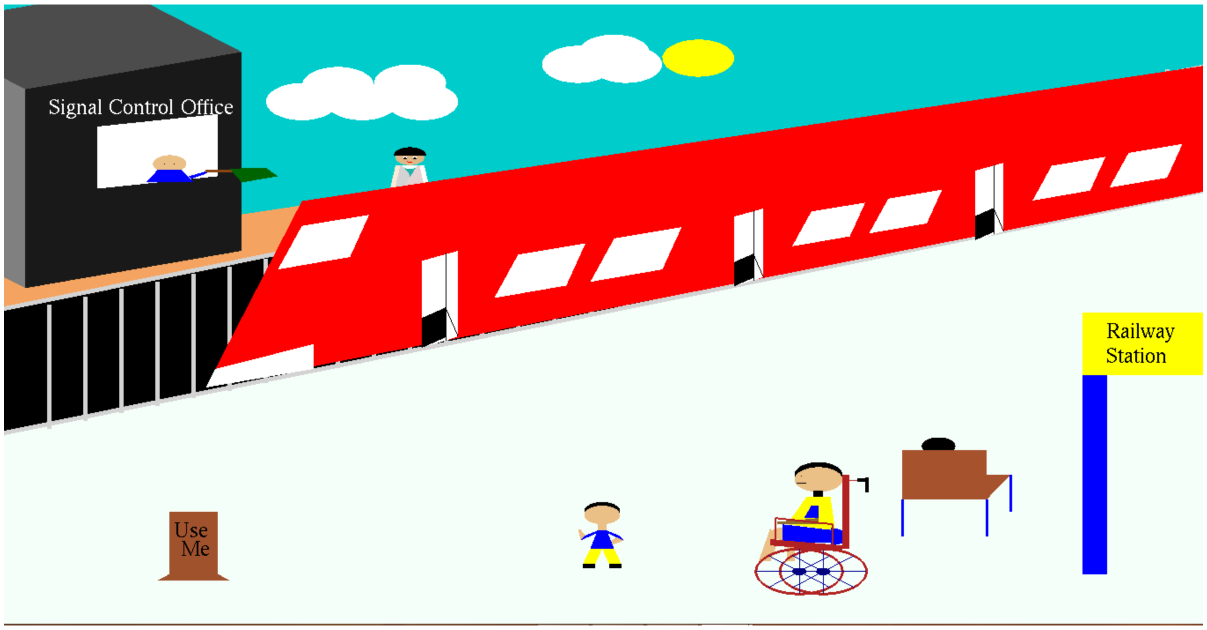

Fig 8: After Universal design

CHAPTER 8
CONCLUSION

As proposed in this project the Universal Design for Handicapped Scenes demonstration meets the requirements and works as specified in the design. The project can be enhanced as specified in the future enhancements by using more complex OpenGL functions. The concept can be well explained through computer graphics.

APPENDIX - 1

- Initially user needs to install the Microsoft visual C++ 6.0 version.
- Then he has to create the a new project which is of win 32 console application format so the creation must be done as shown below :

  File→New → In projects menu select win32 console application → Give appropriate name to the project → Finish. This creates the new project.

- Then insert a new class as follows

  Insert → New class →Give appropriate name →0k
- Select the file view and the click on the source name file →Source files → source filename.cpp then paste the file into the workspace.
- Compile the code by Build→Compile
- Link the object file by selecting Build→ Build filename.exe
- Execute the filename,exe file by Build→ execute filename.exe

APPENDIX - 2

9.1 Manual

- Press 'Y' for Introduction scene.
- Press ‘1’ to go to Bus Stop scene1.
- Press ‘2’ to go to Bus Stop scene2.
- Press ‘3’ to go to School scene1.
- Press ‘4’ to go to School scene2.
- Press ‘5’ to go to Agriculture scene1.
- Press ‘6’ to go to Agriculture scene2.
- Press ‘7’ to go to Railway station scene1.
- Press ‘8’ to go to Railway station scene2
- Press ‘9’ to Exit

BIBLIOGRAPHY

Books referred

1. Edward Angel – Interactiv``e Computer Graphics A Top-Down Approach with OpenGL, 5th Edition, Addison-Wesley, 2008.

2. F. S. Hill - Computer Graphics Using OpenGL – 2nd Edition, Pearson Education, 2001.