CHAPTER - 1

INTRODUCTION

The **"GRAPHICAL IMPLEMENTATION OF 3D ROUTE MAP OF BMSIT&M USING OPENGL"** software interfaces and develops the route map. This project uses the simple techniques such as line-loops, classes etc.

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. A particular graphics software system called OpenGL, which has become a widely accepted standard for developing graphics applications.

The applications of computer graphics in some of the major areas are as follows

1. Display of information
2. Design
3. Simulation and Animation
4. User Interfaces.

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that you use to specify the objects and operations needed to produce interactive three-dimensional applications.

1.1 PROBLEM STATEMENT

Before GPS, before radar, and even before sonar, a traveler's best chance at finding the way around the big land was charts and maps. The National Oceanic and Atmospheric Administration provide nautical charts and maps, as well as many chart and mapmakers. Navigation Charts combine aspects of topographic, general reference and thematic maps and are produced as navigation aids for ships, boats, travellers and aircraft. Specialist knowledge is usually required to read charts.

Maps are very useful & important to us, how:

- To understand roads and subways at new places.
- To calculate distance between two places.
- To know whether there are two or more paths to the same place and which is the shortest.

- We can get information about mountains, rivers, valleys or any other thing, which may come on the way, and we can prepare for that.
- We can get the information like height of the place or ups and downs on the road.
- Boundaries of the land to define ownership.
- Places like houses, farmhouses, mines can be shown on the map.
- We can also mark crops, weather reports, direction of wind, and rainfall on the maps.
- Government needs the map to keep the record of the owners.

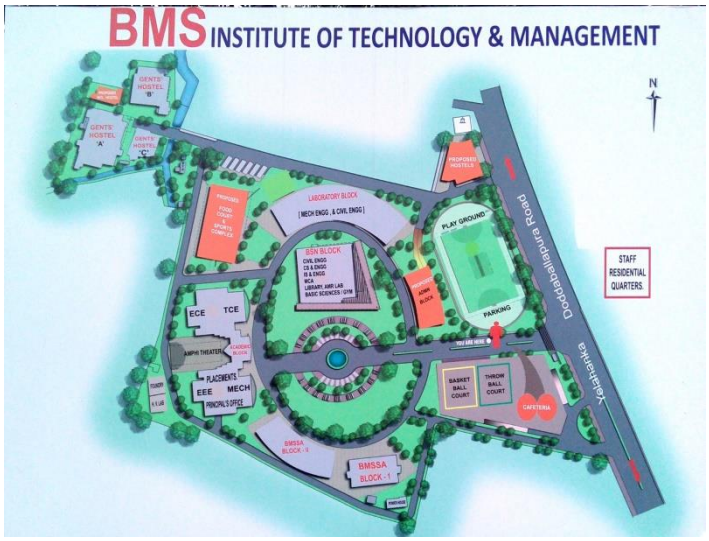

Figure 1.1: Route Map

1.2 OBJECTIVES OF THIS PROJECT

1. To show the navigation path through BMSIT&M using openGL.
2. Effective usage of c language.
3. To provide a variety of options like menus, commands etc.
4. To provide multiple interfaces by either using keyboard or mouse.
5. To make user friendly application supporting with all information required.
6. To show navigation in top, front and 3d views.

1.3 SCOPE OF THE PROJECT

The application program developed can be used in various fields as follows:

- It can be implemented for educative purpose.

- Sound effects can be added.
- Can be used as a base for developing more attractive and fascinating models.

1.4 SUMMARY

The introduction is mainly about the project. It has information about how the project is going to function the objective and scope of the project is also included in it. It gives the reader a brief idea of the project.

CHAPTER - 2

SYSTEM REQUIREMENT SPECIFICATION

A software requirement definition is an abstract description of the services which the system should provide, and the constraints under which the system must operate. It should only specify the external behaviour of the system. The requirements are specified below.

2.1 FUNCTIONAL REQUIREMENTS

In software engineering, a functional requirement defines a function of software system or its component. A function is described as a set of inputs, the behaviour, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system supposed to accomplish. Behavioural requirements describing all the cases where the system uses the functional requirements are captured in use cases. The various methods used in this project are as follows:-

Display() : The module draws the output on the screen and functions in it.

Init() : This module is used to initialize the structure variables.

2.2 NON-FUNCTIONAL REQUIREMENTS

These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process and standards. Non-functional requirements often apply to the system as a whole.

2.2.1 Dependability

The dependability of a computer system is a property of the system that equates to its trustworthiness. Trustworthiness essentially means the degree of user confidence that the system will operate as they expect and that the system will not fail in normal use.

2.2.2 Availability

The ability of the system to deliver services when requested. There is no error in the program while executing the program.

2.2.3 Reliability

The ability of the system to deliver services as specified. The program is compatible with all types of operating system without any failure.

2.2.4 Safety

The ability of the system to operate without catastrophic failure. This program is user friendly and it will never affect the system.

2.2.5 Security

The ability of the system to protect itself against accidental or deliberate intrusion.

2.3 DETAILS OF THE SOFTWARE

Here, the coding of our project is done in Microsoft Visual C++ which is commercial integrated development environment (IDE) with OpenGL (Open Graphics Library) which is standard specification to produce 2D and 3D computer graphics. We use, the OpenGL Utility Toolkit called GLUT which is a library of utilities for OpenGL programs.

2.3.1 Computer Graphics

Computer Graphics is concerned with all aspects of producing pictures or images. The term computer graphics includes almost everything on computers that is not text or sound. Today nearly all computers use some graphics and users expect to control their computer through icons and pictures rather than just by The term Computer Graphics has several meanings:

- the representation and manipulation of pictorial data by a computer
- the various technologies used to create and manipulate pictorial data
- the images so produced, and
- the sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content, see study of computer graphics.

Today computers and computer-generated images touch many aspects of our daily life. Computer imagery is found on television, in newspapers, in weather reports, and during surgical procedures. A well-constructed graph can present complex statistics in a form that is easier to understand and interpret. Such graphs are used to illustrate

papers, reports, theses, and other presentation material. A range of tools and facilities are available to enable users to visualize their data, and computer graphics are used in many disciplines.

2.3.2 OpenGL Overview

OpenGL (***Open G**raphics **L**ibrary*) is a standard specification for writing applications that produce 2D and 3D computer graphics. That means you provide the data in an OpenGL-useable form and OpenGL will show this data on the screen (render it).

It is developed by many companies and it is free to use. You can develop OpenGL-applications without licensing. OpenGL is a hardware and system-independent interface. An OpenGL-application will work on every platform, as long as there is an installed implementation, because it is system independent, there are no functions to create windows etc. A very useful thing is GLUT. OpenGL is a state machine.

2.4 SOFTWARE REQUIREMENTS

- Operating System : Windows XP/ 7/ Linux
- Compiler Used : Visual Studio 2005
- Language Used : C++ language
- Editor : Visual C++

2.5 HARDWARE REQUIREMENTS

- Main processor : PENTIUM III
- Processor Speed : 800 MHz
- RAM Size : 128 MB DDR
- Keyboard : Standard qwerty serial or PS/2 keyboard
- Mouse : Standard serial or PS/2 mouse
- Compatibility : AT/T Compatible
- Cache memory : 256 KB
- Diskette drive A : 1.44 MB, 3.5 inches

2.6 SUMMARY

The requirements for the project are listed in the requirement specification. It gives the user a fair idea about how to execute the project and what are all the requirements needed. With the help of this information the user can check the availability of the resources.

CHAPTER - 3

SYSTEM ANALYSIS AND DESIGN

Design a project with specific goals, task and outcomes. The more specific, the better, the more closely aligned with traditional instructional objectives, the better.

System analysis is a process of collecting factual data, understand the processes involved, identifying problems and recommending feasible solution for improving the system functioning.

3.1 ANALYSIS:

In Implementation of 3D route map, we have developed a 3D model of BMSIT&M using OPEN GL and glut libraries, we have used advanced OPEN GL functionalities to make the model as realistic as possible.

3.2 ALGORITHM:

Algorithm is an effective method expressed as a finite list of well-defined instructions for calculation, data processing and automated reasoning. It makes us to understand the steps in the program easily and also helps in debugging in case of any logical errors.

The algorithm for our project is as follows:-

Step 1: Start

Step 2: Initialization

Step 3: Set the Position and Size of the Window to be created and then create window.

Step 4: Draw the output which is described above.

Step 5: Provide options for displaying pages associated.

Step 6: Call the callback function glutMainLoop () to attend the event list continuously.

Step 7: Stop.

3.3 FLOWCHART:

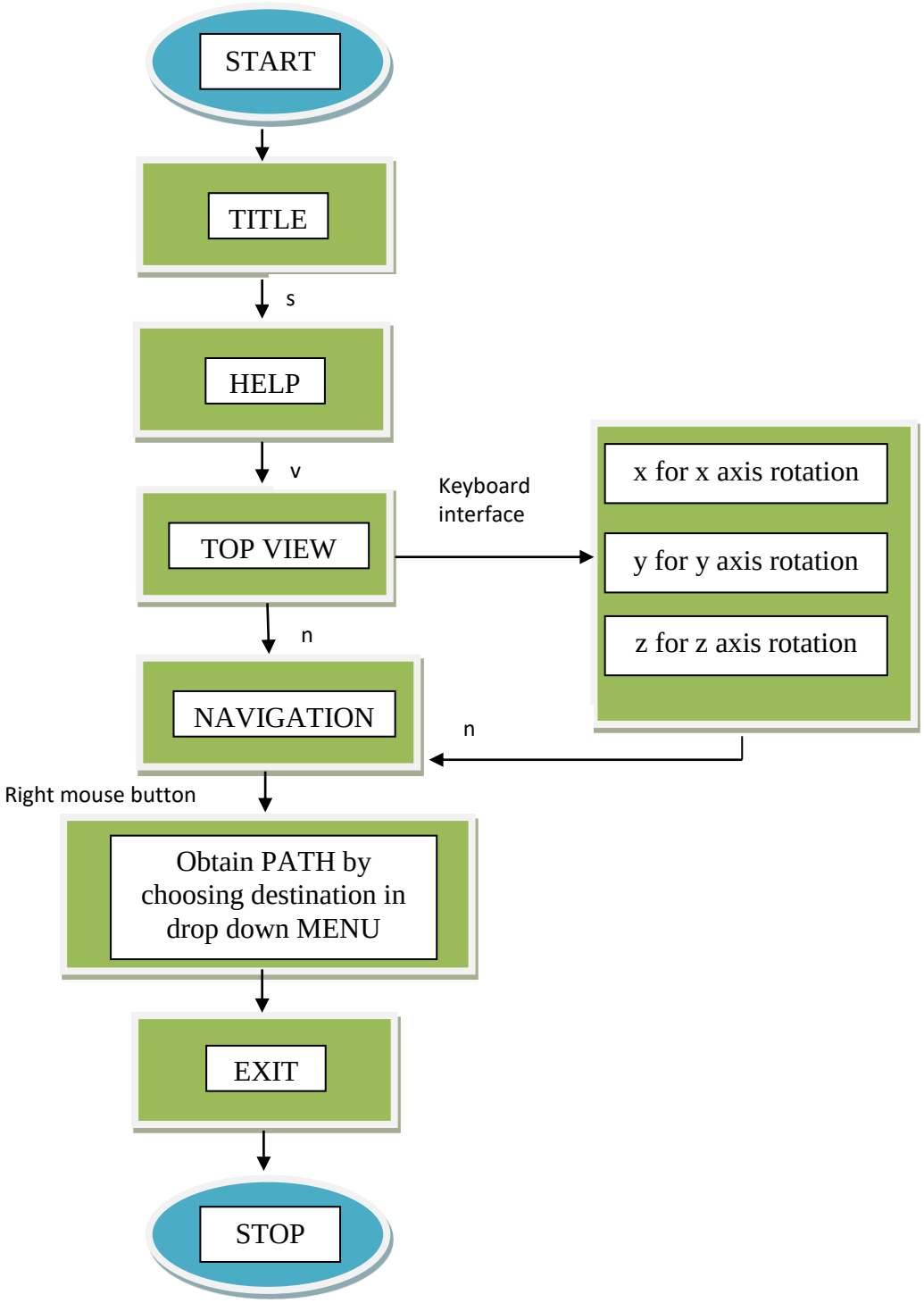

Figure 3.1: Flowchart of 3d Route Map

3.4 ARCHITECTURE:

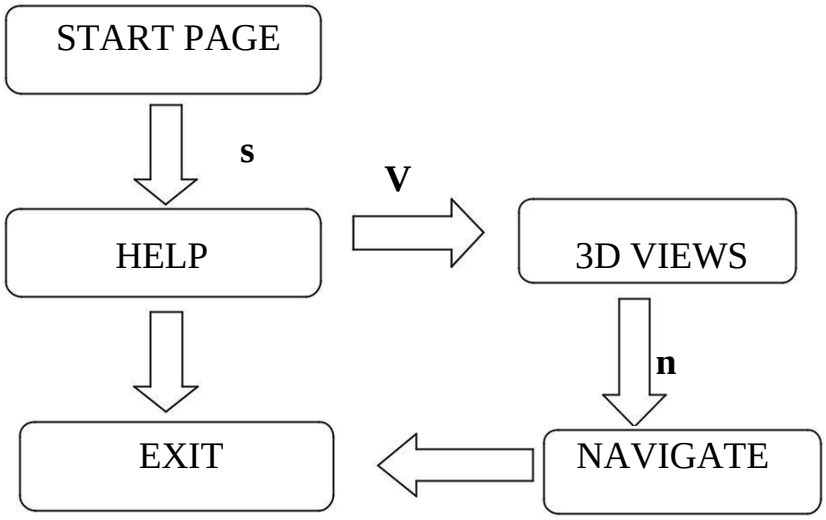

Figure 3.2: Architecture of 3d Route Map

Our project has 2 different stages of display in the above architecture:

Initially the user will be provided with a start page. The user should press‘s’ to go to ‘help’, and then in help page press v to get top view. Then you have following two choices:

1. To view 3D model of campus.
2. To know the path for different destinations.
3. Exit by choosing option in drop down menu.

3.5 DESIGN:

Design specification of 3D route map:

- Display the 3D model on screen.
- Design the corresponding menu options to implement the paths.
- Design the model such that the path will be shown correctly to the exact location.

3.6 SUMMARY:

By analysing the flow chart and algorithm the users will have an idea about the functionality of the project. Users will have knowledge about the how the project is going to function. The architecture is going to provide user information about the inner view of the system.

CHAPTER - 4

SYSTEM IMPLEMENTATION

In Computer Science, an Implementation is a realization of a technical specification or algorithm as program, software component, or other computer system through computer programming and deployment. Many implementations may exist for a given specification or standard. The following are the implementation done to our project:

4.1 OPENGL HEADER AND FUNCTIONS

Header Files

#include<glut.h>

It is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs for rendering multiple windows, callback driven event processing, support for bitmap and stroke fonts, idle routines and timers & may other features.

#include<stdio.h>

stdio.h stands for "Standard input/output header", is the header in the C standard library that contains macro definitions, constraints & declarations of function and types used for various standard input and output operations.

#include<stdlib.h>

stdlib.h is the header file of the general purpose standard library of C programming language which includes functions involving memory allocation, process control, conversions & others.

#include<math.h>

math.h is a header file in the standard library of the C programming language designed for basic mathematical operations.

#include<string.h>

This contains ANSI compatible string manipulation routines.

Window Manager Setup using GLUT

glutInit

- **Syntax:** glutInit(int argc, char **argv);
- **Purpose:** Initializes GLUT. The arguments from main are passed in and can be used by the application.

glutInitDisplayMode

- **Syntax:** glutInitDisplayMode(unsigned int mode);
- **Purpose:** It requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model (GLUT_RGB, GLUT_INDEX) and buffering (GLUT_SINGLE|GLUT_DOUBLE).

glutInitWindowSize

- **Syntax:** glutInitWindowSize(int width, int height);
- **Purpose:** sets size of screen window in pixels.

glutInitWindowPosition

- **Syntax:** glutInitWindowPosition(int x,int y);
- **Purpose:** sets initial window position on the screen for upper left corner.

glutCreateWindow

- **Syntax:** glutCreateWindow(char *title);
- **Purpose:** It creates a window on display. The string title can be used to label the window.

glutDisplayFunc

- **Syntax:** glutDisplayFunc(void (*func)(void));
- **Purpose:** sets the callback function, provided by your program, that GLUT will call when a window must be redisplayed (e.g. on resize). Simply pass the name of a void function that takes no parameters.

Color and Clearing the Screen

glClearColor

- **Syntax:** glClearColor(float red,float green,float blue,float alpha);
- **Purpose:** set the clear color to (red,green,blue) with value alpha.

glClear

- **Syntax:** glClear(GLbitfield mask);
- **Purpose:** clear the buffer indicated by mask using the current clear color. Values for mask are:

GL_COLOR_BUFFER_BIT – color buffer

GL_DEPTH_BUFFER_BIT – depth buffer

GL_STENCIL_BUFFER_BIT – stencil

buffer

glColor3f

- **Syntax:** glColor3f(float red,float green,float blue);
- **Purpose:** set the current drawing color to (rea,green,blue).

OpenGL Drawing

Primitives glBegin

- **Syntax:** glBegin(glEnum mode);
- **Purpose:** Initializes a new primitive of type mode and starts the collection of vertices. Values of mode GL_POINTS, GL_LINES and GL_POLYGON.

glVertex

- **Syntax:** glVertex{234}{sifd}{v}(…..);
- **Purpose:** specify a vertex in either 2,3 or 4-D coordinates, using either 16 or 32 bit intergers or floats or doubles, and either listing all coordinates or using an array parameter.

glEnd

- **Syntax:** glEnd(void);
- **Purpose:** end of vertex list.

glBitmapCharacter

- **Syntax:** glutBitmapCharacter(void * font, char c);
- **Purpose:** draws one bitmap character using the indicated font, fonts include: GLUT_TIMES_ROMAN_10,GLUT_TIMES_ROMAN_24,GLUT_HELVETICA_12 etc.

Window Manager Input Handling using GLUT

glutPostRedisplay

- **Syntax:** glutPostRedisplay();

- **Purpose:** It requests that the display callback be executed after the current callback returns.

glutReshapeFunc

- **Syntax:** glutReshapeFunc(void *f(int width, int height));
- **Purpose:** It registers the reshape callback function f. The callback function returns the height and width of the new window. The reshape callback invokes a display callback.

glutIdleFunc

- **Syntax:** glutIdleFunc(void(*f)(void));
- **Purpose:** It registers the display callback function f that is executed whenever there are no other events to be handled.

glFlush

- **Syntax:** glFlush();
- **Purpose:** flush the graphics output buffer.

Transformations

glTranslate

- **Syntax:** glTranslate[fd](TYPE x,TYPE y,TYPE z);
- **Purpose:** It alters the current matrix by a displacement of (x,y,z);

glPushMatrix and glPopMatrix

- **Syntax:** glPushMatrix() & glPopMatrix();
- **Purpose:** It pushes to and pops from the matrix stack corresponding to current matrix mode. In the robot arm it pushes the matrix for the base, upper arm and lower arm. And also it pops those matrices as well.

Viewing

gluOrtho2D

- **Syntax:** gluOrtho2D(double left, double right, double bottom, double top);
- **Purpose:** sets the 2-dimensional clipping rectangle as (left,bottom) and (right,top).

glLoadIdentity

- **Syntax:** glLoadIdentity(void)
- **Purpose:** set the current matrix to the identiy.

glMatrixMode

- **Syntax:** glMatrixMode(Glenum mode);
- **Purpose:** It specifies which matrix will be affected by subsequent transformations. The modes are GL_MODEL_VIEW, GL_PROJECTION.

4.2 MODULES IMPLEMENTATION

The project has been implemented in modules so as to make it user friendly i.e. if a code viewer is finding trouble with a particular function it can go to a particular module and make corresponding changes in it instead of searching in the program. Also it is more advantageous for future modifications as when modifications are to be made instead of searching the whole program module search can be useful as it decreases the time consumption.

The various modules implanted in this project are:

- Module for drawing the scene – display()
- Module for keyboard interactions – keyboard()
- Module for mouse interactions – mouse()

4.3 CODES

To draw trees:-

```
void trees()
{
 //trunk1
        glColor3ub(95,6,5);
        glBegin(GL_POLYGON);
        glVertex3f(-105,0,0);
        glVertex3f(-100,0,0);
        glVertex3f(-100,0,15);
        glVertex3f(-105,0,15);
        glEnd();
//leaf1
        glColor3f(0.0,0.2,0.0);
        glBegin(GL_POLYGON);
        glVertex3f(-115,0,15);
        glVertex3f(-90,0,15);
        glVertex3f(-102.5,0,35);
```

```
        glEnd();
//leaf2
        glColor3f(0.0,0.2,0.0);
        glBegin(GL_POLYGON);
        glVertex3f(-113,0,25);
        glVertex3f(-92,0,25);
        glVertex3f(-102.5,0,45);
        glEnd();
//leaf3
        glColor3f(0.0,0.2,0.0);
        glBegin(GL_POLYGON);
        glVertex3f(-111,0,35);
        glVertex3f(-94,0,35);
        glVertex3f(-102.5,0,55);
        glEnd();
}
```

To draw man:-

```
void man1()
{
        //face
        glColor3ub(255,191,128);
        glBegin(GL_POLYGON);
        glVertex3f(0,0,16);
        glVertex3f(3,0,16);
        glVertex3f(5,0,18.5);
        glVertex3f(3,0,23);
        glVertex3f(0,0,23);
        glVertex3f(-2,0,18.5);
        glEnd();
//shirt
        glColor3ub(160,150,250);
        glBegin(GL_POLYGON);
        glVertex3f(6,0,6);
        glVertex3f(6,0,16);
        glVertex3f(-3,0,16);
        glVertex3f(-3,0,6);
        glEnd();
//left arm
        glColor3ub(255,191,128);
        glBegin(GL_POLYGON);
        glVertex3f(6,0,16);
        glVertex3f(10,0,6);
        glVertex3f(9,0,9);
        glVertex3f(6,0,11);
        glEnd();
//right arm
        glColor3ub(255,191,128);
```

```
        glBegin(GL_POLYGON);
        glVertex3f(-3,0,16);
        glVertex3f(-3,0,11);
        glVertex3f(-6,0,9);
        glVertex3f(-7,0,6);
        glEnd();
//right leg
        glColor3ub(80,80,230);
        glBegin(GL_POLYGON);
        glVertex3f(-5,0,0);
        glVertex3f(0.5,0,0);
        glVertex3f(1.5,0,6);
        glVertex3f(-3,0,6);
        glEnd();
//left leg
        glColor3ub(80,80,230);
        glBegin(GL_POLYGON);
        glVertex3f(2.5,0,0);
        glVertex3f(8,0,0);
        glVertex3f(6,0,6);
        glVertex3f(1.5,0,6);
        glEnd();
//hair right
        glColor3f(0,0,0);
        glBegin(GL_POLYGON);
        glVertex3f(0,-0.1,21);
        glVertex3f(0,-0.1,19);
        glVertex3f(1,-0.1,19);
        glVertex3f(1.5,-0.1,21);
        glEnd();
//hair left
        glColor3f(0,0,0);
        glBegin(GL_POLYGON);
        glVertex3f(1.5,-0.1,21);
        glVertex3f(2,-0.1,19);
        glVertex3f(3,-0.1,19);
        glVertex3f(3,-0.1,21);
        glEnd();
}
```

To draw woman:-

```
void woman1()
{
  //face
        glColor3ub(255,191,128);
        glBegin(GL_POLYGON);
        glVertex3f(100,-180,16);
```

```
        glVertex3f(103,-180,16);
        glVertex3f(105,-180,18.5);
        glVertex3f(103,-180,23);
        glVertex3f(100,-180,23);
        glVertex3f(98,-180,18.5);
        glEnd();
  //shirt
        glColor3ub(123,104,238);
        glBegin(GL_POLYGON);
        glVertex3f(106,-180,6);
        glVertex3f(106,-180,16);
        glVertex3f(97,-180,16);
        glVertex3f(97,-180,6);
        glEnd();
  //left arm
        glColor3ub(255,191,128);
        glBegin(GL_POLYGON);
        glVertex3f(106,-180,16);
        glVertex3f(110,-180,6);
        glVertex3f(109,-180,9);
        glVertex3f(106,-180,11);
        glEnd();
  //right arm
        glColor3ub(255,191,128);
        glBegin(GL_POLYGON);
        glVertex3f(97,-180,16);
        glVertex3f(97,-180,11);
        glVertex3f(94,-180,9);
        glVertex3f(93,-180,6);
        glEnd();
  //hair right
        glColor3f(0,0,0);
        glBegin(GL_POLYGON);
        glVertex3f(100,-180.1,21);
        glVertex3f(100,-180.1,19);
        glVertex3f(101,-180.1,19);
        glVertex3f(101.5,-180.1,21);
        glEnd();
  //hair left
        glColor3f(0,0,0);
        glBegin(GL_POLYGON);
        glVertex3f(101.5,-180.1,21);
        glVertex3f(102,-180.1,19);
        glVertex3f(103,-180.1,19);
        glVertex3f(103,-180.1,21);
        glEnd();
  //skirt
```

```
        glColor3f(1,0,0);
        glBegin(GL_POLYGON);
        glVertex3f(97,-180,6);
        glVertex3f(95,-180,0);
        glVertex3f(108,-180,0);
        glVertex3f(106,-180,6);
        glEnd();
}
```

To draw path:-

```
//path for academic block
void func1()
{
        glLineWidth(5);
        glBegin(GL_LINES);
        glColor3f(1,1,0);
        glVertex3f(17,-197.5,2);
        glVertex3f(25,-197.5,2);
        glEnd();
        glBegin(GL_LINES);
        glColor3f(1,1,0);
        glVertex3f(25,-197.5,2);
        glVertex3f(25,100,2);
        glEnd();
        glBegin(GL_LINES);
        glColor3f(1,1,0);
        glVertex3f(25,100,2);
        glVertex3f(15,100,2);
        glEnd();
        glBegin(GL_LINES);
        glColor3f(1,1,0);
        glVertex3f(15,100,2);
        glVertex3f(15,135,2);
        glEnd();
        glLineWidth(1.f);
}
```

To draw light pole:-

```
// light pole 1
        glColor3f(1,1,1);
        glBegin(GL_LINES);
        glVertex3f(15,70,0);
        glVertex3f(15,70,30);
        glEnd();
        glColor3f(1,1,1);
        glBegin(GL_LINES);
        glVertex3f(10,70,30);
        glVertex3f(20,70,30);
```

```
        glEnd();
        glColor3f(1,1,1);
        glBegin(GL_TRIANGLES);
        glVertex3f(10,70,30);
        glVertex3f(12,70,25);
        glVertex3f(8,70,25);
        glEnd();
        glColor3f(1,1,1);
        glBegin(GL_TRIANGLES);
        glVertex3f(20,70,30);
        glVertex3f(22,70,25);
        glVertex3f(18,70,25);
        glEnd();
```

To draw divider:-

```
//divider1
        glColor3ub(0,100,0);
        glBegin(GL_POLYGON);
        glVertex3f(12,30,1.5);
        glVertex3f(12,80,1.5);
        glVertex3f(17,80,1.5);
        glVertex3f(17,30,1.5);
        glEnd();
```

To draw lands and grounds:-

```
void land()
{
        trees();
        man1();
        man2();
        man3();
        man4();
        woman1();
        woman2();
        woman3();
        woman4();
  // light pole 1
  //divider1
  //ground for hostels
        glColor3f(1,0.9,0.9);
        glBegin(GL_POLYGON);
        glVertex3f(250,250,1);
        glVertex3f(250,170,1);
        glVertex3f(150,170,1);
        glVertex3f(150,250,1);
        glEnd();
//ground for foundry
        glColor3ub(230,230,250);
        glBegin(GL_POLYGON);
```

```
        glVertex3f(-100,240,2);
        glVertex3f(-100,190,2);
        glVertex3f(-70,190,2);
        glVertex3f(-70,240,2);
        glEnd();
 ///////cricket area color
        glColor3f(0.8627,0.0784,0.2352);
        glBegin(GL_POLYGON);
        glVertex3f(220,150,1.5);
        glVertex3f(220,80,1.5);
        glVertex3f(150,80,1.5);
        glVertex3f(150,150,1.5);
        glEnd();
///////playground area color
        glColor3f(0.8627,0.0784,0.2352);
        glBegin(GL_POLYGON);
        glVertex3f(-180,-170,0.1);
        glVertex3f(-180,-100,.1);
        glVertex3f(-100,-100,0.1);
        glVertex3f(-100,-170,0.1);
        glEnd();
///////buses parking area color
        glColor3f(0.6,0.6,0);
        glBegin(GL_POLYGON);
        glVertex3f(-50,-170,.1);
        glVertex3f(-50,-100,.1);
        glVertex3f(-5,-100,.1);
        glVertex3f(-5,-170,.1);
        glEnd();
///////car and bike parking area color
        glColor3f(0.8,0.6,0.4);
        glBegin(GL_POLYGON);
        glVertex3f(50,-80,.1);
        glVertex3f(50,-210,.1);
        glVertex3f(200,-210,.1);
        glVertex3f(200,-80,.1);
        glEnd();
///////canteen area color
        glColor3f(1,0.9,0.9);
        glBegin(GL_POLYGON);
        glVertex3f(-230,-220,1);
        glVertex3f(-230,-105,1);
        glVertex3f(-170,-105,1);
        glVertex3f(-170,-220,1);
        glEnd();
//green color land
        glColor3f(0,.6,0);
```

```
        glBegin(GL_POLYGON);
        glVertex3f(-250,-250,0);
        glVertex3f(250,-250,0);
        glVertex3f(250,250,0);
        glVertex3f(-250,250,0);
        glEnd();
}
```

To draw roads:-

```
//to draw the roads color effect
void road1(float x,float y,float z,float h,float w)
{
        glColor3f(0,0,0);
        glBegin(GL_POLYGON);
        glVertex3f(x,y,z);
        glVertex3f(x,y-h,z);
        glVertex3f(x+w,y-h,z);
        glVertex3f(x+w,y,z);
        glEnd();
        glColor3f(0.3,0.3,0.3);
        glBegin(GL_POLYGON);
        glVertex3f(x,y-1,z+1);
        glVertex3f(x,y-h+1,z+1);
        glVertex3f(x+w,y-h+1,z+1);
        glVertex3f(x+w,y-1,z+1);
        glEnd();
        glColor3f(0,0,0);
        glBegin(GL_POLYGON);
        glVertex3f(x,y,z);
        glVertex3f(x,y-1,z+1);
        glVertex3f(x+w,y-1,z+1);
        glVertex3f(x+w,y,z);
        glEnd();
        glColor3f(0,0,0);
        glBegin(GL_POLYGON);
        glVertex3f(x,y-h,z);
        glVertex3f(x,y-h+1,z+1);
        glVertex3f(x+w,y-h+1,z+1);
        glVertex3f(x+w,y-h,z);
        glEnd();
}
////main block road
        glPushMatrix();
        glTranslated(20,-30,0);
        glRotated(90,0,0,1);
        road1(-200,25,0.1,40,400);
        glPopMatrix();
//////////////////////////
```

```
//hostel road from main block
        glPushMatrix();
        glTranslated(20,5,0);
        glRotated(180,0,0,1);
        road1(-200,-100,0.1,20,190);
        glPopMatrix();
//////////////////////
//road to bmssa block 2 from main block
        glPushMatrix();
        glTranslated(20,5,0);
        glRotated(180,0,0,1);
        road1(20,-100,0.1,20,230);
        glPopMatrix();
//road  yelahanka
        glPushMatrix();
        glTranslated(20,5,0);
        glRotated(180,0,0,1);
        road1(20,255,0.1,30,250);
        glPopMatrix();
//road  dodaballapur
        glPushMatrix();
        glTranslated(20,5,0);
        glRotated(180,0,0,1);
        road1(-230,255,0.1,30,250);
        glPopMatrix();


//road from bsn to main block
        glPushMatrix();
        glTranslated(20,-30,0);
        glRotated(270,0,0,1);
        road1(-135,80,0.1,9,30);
        glPopMatrix();
//road from bsn block to lab block
        glPushMatrix();
        glTranslated(20,5,0);
        glRotated(180,0,0,1);
        road1(-200,70,0.1,25,190);
        glPopMatrix();
//road from bmssa block 1 to bsn bllock
        glPushMatrix();
        glTranslated(20,5,0);
        glRotated(180,0,0,1);
        road1(20,70,0.1,25,230);
        glPopMatrix();
//road from bsn block to garden
        glPushMatrix();
```

```
        glTranslated(20,-30,0);
        glRotated(270,0,0,1);
        road1(-20,90,0.1,9,30);
        glPopMatrix();
//road from canteen to bmssa block1
        glPushMatrix();
        glTranslated(20,-30,0);
        glRotated(270,0,0,1);
        road1(35,-196,0.1,9,40);
        glPopMatrix();
//right road complete
        glPushMatrix();
        glTranslated(20,-30,0);
        glRotated(90,0,0,1);
        road1(-200,-200,0.1,30,400);
        glPopMatrix();
//left road complete
        glPushMatrix();
        glTranslated(20,-30,0);
        glRotated(90,0,0,1);
        road1(-200,270,0.1,30,480);
        glPopMatrix();
//road to cafe from  gate
        glPushMatrix();
        glTranslated(20,5,0);
        glRotated(180,0,0,1);
        road1(0,200,0.1,10,190);
        glPopMatrix();
//road to bank from  gate
        glPushMatrix();
        glTranslated(20,5,0);
        glRotated(180,0,0,1);
        road1(0,110,0.1,10,200);
        glPopMatrix();
//road car bike paring
        glPushMatrix();
        glTranslated(20,5,0);
        glRotated(180,0,0,1);
        road1(-200,180,.1,10,200);
        glPopMatrix();
//vertical road to hostel
        glPushMatrix();
        glTranslated(20,-30,0);
        glRotated(270,0,0,1);
        road1(-225,100,0.1,9,70);
        glPopMatrix();
//horizontal road to hostel
```

```
        glPushMatrix();
        glTranslated(20,5,0);
        glRotated(180,0,0,1);
        road1(-130,-180,.1,15,39);
        glPopMatrix();
//road to foundary
        glPushMatrix();
        glTranslated(20,-30,0);
        glRotated(270,0,0,1);
        road1(-225,-100,0.1,9,70);
        glPopMatrix();
//road from bmssa block to foundary
        glPushMatrix();
        glTranslated(20,-30,0);
        glRotated(270,0,0,1);
        road1(10,-196,0.1,9,-200);
        glPopMatrix();
```

To draw gate:-

```
//for gate1
        glPushMatrix();
        glColor3f(0,0,1);
        glBegin(GL_LINES);
        glVertex3f(-3,-220,0);
        glVertex3f(-3,-220,30);
        glEnd();
        glBegin(GL_LINES);
        glVertex3f(0,-220,0);
        glVertex3f(0,-220,30);
        glEnd();
        glBegin(GL_LINES);
        glVertex3f(3,-220,0);
        glVertex3f(3,-220,30);
        glEnd();
        glBegin(GL_LINES);
        glVertex3f(6,-220,0);
        glVertex3f(6,-220,30);
        glEnd();
        glBegin(GL_LINES);
        glVertex3f(9,-220,0);
        glVertex3f(9,-220,30);
        glEnd();
        glBegin(GL_LINES);
        glVertex3f(12,-220,0);
        glVertex3f(12,-220,30);
        glEnd();
        glBegin(GL_LINES);
        glVertex3f(15,-220,0);
```

```
glVertex3f(15,-220,30);
glEnd();
glBegin(GL_LINES);
glVertex3f(18,-220,0);
glVertex3f(18,-220,30);
glEnd();
glBegin(GL_LINES);
glVertex3f(21,-220,0);
glVertex3f(21,-220,30);
glEnd();
glBegin(GL_LINES);
glVertex3f(24,-220,0);
glVertex3f(24,-220,30);
glEnd();
glBegin(GL_LINES);
glVertex3f(27,-220,0);
glVertex3f(27,-220,30);
glEnd();
glBegin(GL_LINES);
glVertex3f(30,-220,0);
glVertex3f(30,-220,30);
glEnd();
glBegin(GL_LINES);
glVertex3f(34,-220,0);
glVertex3f(34,-220,30);
glEnd();
glBegin(GL_LINES);
glVertex3f(-3,-220,30);
glVertex3f(34,-220,30);
glEnd();
glBegin(GL_LINES);
glVertex3f(-3,-220,2);
glVertex3f(34,-220,2);
glEnd();
glBegin(GL_LINES);
glVertex3f(-3,-220,15);
glVertex3f(34,-220,15);
glEnd();
glBegin(GL_LINES);
glVertex3f(-3,-220,30);
glVertex3f(34,-220,0);
glEnd();
glBegin(GL_LINES);
glVertex3f(-3,-220,0);
glVertex3f(34,-220,30);
glEnd();
```

To draw cafetaria:-

```
//for cafeteria
void house(float x,float y,float z)
{
	glColor3f(0.01,0.05,0.3);
	glBegin(GL_POLYGON);
	glVertex3f(x,y,z);
	glVertex3f(x,y,z+30);
	glVertex3f(x,y+15,z+30);
	glVertex3f(x,y+15,z);
	glEnd();
	glColor3f(0.01,0.05,0.3);
	glBegin(GL_POLYGON);
	glVertex3f(x-15,y,z);
	glVertex3f(x-15,y+15,z);
	glVertex3f(x-15,y+15,z+30);
	glVertex3f(x-15,y,z+30);
	glEnd();
	glColor3f(0.01,0.05,0.3);
	glBegin(GL_POLYGON);
	glVertex3f(x,y,z+30);
	glVertex3f(x-15,y,z+30);
	glVertex3f(x-15,y+15,z+30);
	glVertex3f(x,y+15,z+30);
	glEnd();
	glColor3f(0.01,0.05,0.3);
	glBegin(GL_POLYGON);
	glVertex3f(x-15,y,z);
	glVertex3f(x-15,y+15,z);
	glVertex3f(x,y+15,z);
	glVertex3f(x,y,z);
	glEnd();
	glColor3f(0.01,0.05,1.5);
	glBegin(GL_POLYGON);
	glVertex3f(x-15,y,z);
	glVertex3f(x,y,z);
	glVertex3f(x,y,z+30);
	glVertex3f(x-15,y,z+30);
	glEnd();
	glColor3f(0.51,0.015,0.008);
	glBegin(GL_POLYGON);
	glVertex3f(x+1.69*3,y+3.5*3,z);
	glVertex3f(x-2.5*3,y+7.5*3,z);
	glVertex3f(x-2.5*3,y+7.5*3,z+30);
	glVertex3f(x+1.69*3,y+3.5*3,z+30);
	glEnd();
	glColor3f(0.51,0.015,0.008);
```

```
        glBegin(GL_POLYGON);
        glVertex3f(x-6.69*3,y+3.5*3,z);
        glVertex3f(x-6.69*3,y+3.5*3,z+30);
        glVertex3f(x-2.5*3,y+7.5*3,z+30);
        glVertex3f(x-2.5*3,y+7.5*3,z);
        glEnd();
        glColor3f(0.1,0.015,0.13);
        glBegin(GL_POLYGON);
        glVertex3f(x,y+15,z);
        glVertex3f(x-15,y+5,z);
        glVertex3f(x-2.5*3,y+7.5*3,z);
        glEnd();
        glColor3f(0.1,0.015,0.13);
        glBegin(GL_POLYGON);
        glVertex3f(x,y+15,z+30);
        glVertex3f(x-15,y+15,z+30);
        glVertex3f(x-2.5*3,y+7.5*3,z+30);
        glEnd();
}
```

To draw building:-

```
void apart(float x,float y,float z)
{
        int i;

//////////////back side of apartment
        glColor3f(0.6,0.6,0.6);
        glBegin(GL_POLYGON);
        glVertex3f(x,y,z+0.5);
        glVertex3f(x+45,y,z+0.5);
        glVertex3f(x+45,y+100,z+0.5);
        glVertex3f(x,y+100,z+0.5);
        glEnd();
        glColor3f(0.8,0.8,0.8);
        for(j=0;j<8;j++)
        {
                glPushMatrix();
                glTranslatef(0,-12*j,0);
                for(i=0;i<4;i++)
                {
                        glPushMatrix();
                        glTranslatef(11*i,0,0);
                        glBegin(GL_POLYGON);
                        glVertex3f(x+2,y+98,z);
                        glVertex3f(x+10,y+98,z);
                        glVertex3f(x+10,y+88,z);
                        glVertex3f(x+2,y+88,z);
                        glEnd();
```

```
                glPopMatrix();
        }
        glPopMatrix();
    }
    glColor3f(0,0,0);
    for(j=0;j<8;j++)
    {
        glPushMatrix();
        glTranslatef(0,-12*j,0);
        for(i=0;i<4;i++)
        {
            glPushMatrix();
            glTranslatef(11*i,0,0);
            glBegin(GL_LINE_LOOP);
            glVertex3f(x+2,y+98,z);
            glVertex3f(x+10,y+98,z);
            glVertex3f(x+10,y+88,z);
            glVertex3f(x+2,y+88,z);
            glEnd();
            glPopMatrix();
        }
        glPopMatrix();
    }


////////////////////////// front side of apartment
    glColor3ub(152,245,238);
    glBegin(GL_POLYGON);
    glVertex3f(x,y,z+45-0.5);
    glVertex3f(x+45,y,z+45-0.5);
    glVertex3f(x+45,y+100,z+45-0.5);
    glVertex3f(x,y+100,z+45-0.5);
    glEnd();
    glColor3f(0.8,0.8,0.8);
    for(j=0;j<8;j++)
    {

        glPushMatrix();
        glTranslatef(0,-12*j,0);
        for(i=0;i<4;i++)
        {
        glPushMatrix();
            glTranslatef(11*i,0,0);
            glBegin(GL_POLYGON);
            glVertex3f(x+2,y+98,z+45);
            glVertex3f(x+10,y+98,z+45);
            glVertex3f(x+10,y+88,z+45);
```

```
                glVertex3f(x+2,y+88,z+45);
                glEnd();
                glPopMatrix();
            }
            glPopMatrix();
        }
        glColor3f(0,0,0);
        for(j=0;j<8;j++)
        {
            glPushMatrix();
            glTranslatef(0,-12*j,0);
            for(i=0;i<4;i++)
            {
                glPushMatrix();
                glTranslatef(11*i,0,0);
                glBegin(GL_LINE_LOOP);
                glVertex3f(x+2,y+98,z+45);
                glVertex3f(x+10,y+98,z+45);
                glVertex3f(x+10,y+88,z+45);
                glVertex3f(x+2,y+88,z+45);
                glEnd();
                glPopMatrix();
            }
            glPopMatrix();
        }
////////////////////////// left side of apartment
        glColor3f(0.6,0.6,0.6);
        glBegin(GL_POLYGON);
        glVertex3f(x+0.5,y,z);
        glVertex3f(x+0.5,y,z+45);
        glVertex3f(x+0.5,y+100,z+45);
        glVertex3f(x+0.5,y+100,z);
        glEnd();
        glColor3f(0.8,0.8,0.8);
        for(j=0;j<8;j++)
        {
            glPushMatrix();
            glTranslatef(0,-12*j,0);
            for(i=0;i<4;i++)
            {
                glPushMatrix();
                glTranslatef(0,0,11*i);
                glBegin(GL_POLYGON);
                glVertex3f(x,y+98,z+2);
                glVertex3f(x,y+98,z+10);
                glVertex3f(x,y+88,z+10);
                glVertex3f(x,y+88,z+2);
```

```
                glEnd();
                glPopMatrix();
            }
            glPopMatrix();
        }
        glColor3f(0,0,0);
        for(j=0;j<8;j++)
        {
            glPushMatrix();
            glTranslatef(0,-12*j,0);
            for(i=0;i<4;i++)
            {
                glPushMatrix();
                glTranslatef(0,0,11*i);
                glBegin(GL_LINE_LOOP);
                glVertex3f(x,y+98,z+2);
                glVertex3f(x,y+98,z+10);
                glVertex3f(x,y+88,z+10);
                glVertex3f(x,y+88,z+2);
                glEnd();
                glPopMatrix();
            }
            glPopMatrix();
        }
//////////////////////////right side of appartment
        glColor3f(0.6,0.6,0.6);
        glBegin(GL_POLYGON);
        glVertex3f(x+45-0.5,y,z-0.5);
        glVertex3f(x+45-0.5,y,z+45-0.5);
        glVertex3f(x+45-0.5,y+100,z+45-0.5);
        glVertex3f(x+45-0.5,y+100,z-0.5);
        glEnd();
        //white color glasses
        glColor3f(0.8,0.8,0.8);
        for(j=0;j<8;j++)
        {
            glPushMatrix();
            glTranslatef(0,-12*j,0);
            for(i=0;i<4;i++)
            {
                glPushMatrix();
                glTranslatef(0,0,11*i);
                glBegin(GL_POLYGON);
                glVertex3f(x+45,y+98,z+2);
                glVertex3f(x+45,y+98,z+10);
                glVertex3f(x+45,y+88,z+10);
                glVertex3f(x+45,y+88,z+2);
```

```
                glEnd();
                glPopMatrix();
            }
            glPopMatrix();
        }
//border for white glass
        glColor3f(0,0,0);
        for(j=0;j<8;j++)
        {
            glPushMatrix();
            glTranslatef(0,-12*j,0);
            for(i=0;i<4;i++)
            {
                glPushMatrix();
                glTranslatef(0,0,11*i);
                glBegin(GL_LINE_LOOP);
                glVertex3f(x+45,y+98,z+2);
                glVertex3f(x+45,y+98,z+10);
                glVertex3f(x+45,y+88,z+10);
                glVertex3f(x+45,y+88,z+2);
                glEnd();
                glPopMatrix();
            }
            glPopMatrix();
        }
        glColor3f(0.5,0.5,0.5);
        glBegin(GL_POLYGON);
        glVertex3f(x,y,z-0.5);
        glVertex3f(x+45,y,z-0.5);
        glVertex3f(x+45,y,z+45-0.5);
        glVertex3f(x,y,z+45-0.5);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex3f(x,y+100,z);
        glVertex3f(x+45,y+100,z);
        glVertex3f(x+45,y+100,z+45);
        glVertex3f(x,y+100,z+45);
        glEnd();
}
```

To call all buildings:-

```
void house1()
{
        //canteen hut 1
        glPushMatrix();
        glRotated(90,1,0,0);
        house(-200,0.1,160);
        glPopMatrix();
```

```
//canteen hut 2
glPushMatrix();
glRotated(90,1,0,0);
house(-180,0.1,140);
glPopMatrix();
//main block start
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(-8,0,-180);
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(-50,0,-190);
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(35,0,-190);
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(-60,0,-230);
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(45,0,-230);//main block end
//BSNblock start
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(70,0,-75);
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(95,0,-75);
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(70,0,-30);
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(95,0,-30);//bsn block end
//bmssa block 1 start
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
```

```
apart(-230,0,-60);
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(-230,0,-15);
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(-230,0,25);//bmssa block 1 end
//bmssa block 2 start
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(-230,0,-250);
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(-230,0,-210);
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(-230,0,-170);
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(-190,0,-250);
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(-190,0,-210);
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(-190,0,-170);   //bmssa block 2 end
//hostel start
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(200,0,-250);
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
apart(150,0,-220);//end of hostel
//lab block start
glPopMatrix();
glPushMatrix();
glRotated(90,1,0,0);
```

```
        apart(160,0,-50);
        glPopMatrix();
        glPushMatrix();
        glRotated(90,1,0,0);
        apart(175,0,-10);
        glPopMatrix();
        glPushMatrix();
        glRotated(90,1,0,0);
        apart(175,0,30);
        glPopMatrix();
        glPushMatrix();
        glRotated(90,1,0,0);
        apart(175,0,60);
        glPopMatrix();
        glPushMatrix();
        glRotated(90,1,0,0);
        apart(160,0,90);//lab block end
        glPopMatrix();

}
```

Display function:-

```
void display()
{
        glClearColor(1.0, 1, 1.0, 1.0);
        glClear( GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);
        if(then==0)
        { intro();texti();}
        if(then==1)
        {
                glClearColor(1.0, 1, 1.0, 1.0);
                glClear( GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);
                intro();
                helptxt();
        }
        if(then==1 && help==1)
        {
                glClearColor(0.1176, 0.5647, 1.0, 1.0);
                glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
                glPointSize(10);
                house1();
                glPushMatrix();
                glRotated(180,0,1,0);
                glScaled(0.5,5,0.1);
                glTranslated(-1,20,-17);
                glPopMatrix();
                glPushMatrix();
                land();
```

```
        glPopMatrix();
        aa=0;bb=0;cc=0;
        float a[3]={-30,30,25},b[3]={30,30,25},c[3]={30,-30,25},d[3]={-30,-30,25};
        glPushMatrix();
        glPopMatrix();
}
if(then==1 && help==1 && navi==1)
{
        glClearColor(0.1176, 0.5647, 1.0, 1.0);
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        glPointSize(10);
        house1();
        glPushMatrix();
        glRotated(180,0,1,0);
        glScaled(0.5,5,0.1);
        glTranslated(-1,20,-17);
        glPopMatrix();
        glPushMatrix();
        land();
        glPopMatrix();
        aa=0;bb=0;cc=0;
        float a[3]={-30,30,25},b[3]={30,30,25},c[3]={30,-30,25},d[3]={-30,-30,25};
        glPushMatrix();
        glPopMatrix();
        glPushMatrix();
        path(13,-200,17,-195);
        glPopMatrix();
        if(academic==1) {func1();}
        if(account==1)  {func1();}
        if(admin==1)    {func1();}
        if(bank==1)     {func8();}
        if(amphi==1)    {func1();}
        if(science==1)  {func6();}
        if(bmssa1==1)  {func3();}
        if(bmssa2==1)  {func4();}
        if(bsn==1)      {func6();}
        if(cafe==1)     {func5();}
        if(civil==1)    {func6();}
        if(cse==1)      {func6();}
        if(ece==1)      {func1();}
        if(found==1)    {func2();}
        if(hostel==1)   {func9();}
        if(gym==1)      {func6();}
        if(ise==1)      {func6();}
        if(lab==1)      {func10();}
        if(lib==1)      {func6();}
        if(mca==1)      {func6();}
```

```
            if(park==1)     {func7();}
            if(plac==1)     {func1();}
            if(princi==1)   {func1();}
        }
        glFlush();
        glutSwapBuffers();
}
```

Menu interface:-

```
void myMenu(int id)
{
        switch(id)
        {
        case 1:
        academic=1,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmssa2=0,bs
        n=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,mca=0,p
        ark=0,plac=0,princi=0;
        glutPostRedisplay();
        break;
        case2:
        academic=0,account=1,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmssa2=0,bs
        n=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,mca=0,p
        ark=0,plac=0,princi=0;
        glutPostRedisplay();
        break;
        case3:academic=0,account=0,admin=1,bank=0,amphi=0,science=0,bmssa1=0,bmssa
        2=0,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,m
        ca=0,park=0,plac=0,princi=0;
        glutPostRedisplay();
        break;
        case4:academic=0,account=0,admin=0,bank=1,amphi=0,science=0,bmssa1=0,bmssa
        2=0,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,m
        ca=0,park=0,plac=0,princi=0;
        glutPostRedisplay();
        break;
        case5:academic=0,account=0,admin=0,bank=0,amphi=1,science=0,bmssa1=0,bmssa
        2=0,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,m
        ca=0,park=0,plac=0,princi=0;
        glutPostRedisplay();
        break;
        case6:academic=0,account=0,admin=0,bank=0,amphi=0,science=1,bmssa1=0,bmssa
        2=0,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,m
        ca=0,park=0,plac=0,princi=0;
        glutPostRedisplay();
        break;
```

case7:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=1,bmssa2=0,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,mca=0,park=0,plac=0,princi=0;
glutPostRedisplay();
break;
case8:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmssa2=1,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,mca=0,park=0,plac=0,princi=0;
glutPostRedisplay();
break;
case9:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmssa2=0,bsn=1,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,mca=0,park=0,plac=0,princi=0;
glutPostRedisplay();
break;
case10:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmssa2=0,bsn=0,cafe=1,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,mca=0,park=0,plac=0,princi=0;
glutPostRedisplay();
break;
case11:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmssa2=0,bsn=0,cafe=0,civil=1,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,mca=0,park=0,plac=0,princi=0;
glutPostRedisplay();
break;
case12:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmssa2=0,bsn=0,cafe=0,civil=0,cse=1,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,mca=0,park=0,plac=0,princi=0;
glutPostRedisplay();
break;
case13:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmssa2=0,bsn=0,cafe=0,civil=0,cse=0,ece=1,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,mca=0,park=0,plac=0,princi=0;
glutPostRedisplay();
break;
case14:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmssa2=0,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=1,hostel=0,gym=0,ise=0,lab=0,lib=0,mca=0,park=0,plac=0,princi=0;
glutPostRedisplay();
break;
case15:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmssa2=0,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=1,gym=0,ise=0,lab=0,lib=0,mca=0,park=0,plac=0,princi=0;
glutPostRedisplay();
break;

```
        case16:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmss
        a2=0,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=1,ise=0,lab=0,lib=0,
        mca=0,park=0,plac=0,princi=0;
        glutPostRedisplay();
        break;
        case17:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmss
        a2=0,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=1,lab=0,lib=0,
        mca=0,park=0,plac=0,princi=0;
        glutPostRedisplay();
        break;
        case18:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmss
        a2=0,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=1,lib=0,
        mca=0,park=0,plac=0,princi=0;
        glutPostRedisplay();
        break;
        case19:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmss
        a2=0,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=1,
        mca=0,park=0,plac=0,princi=0;
        glutPostRedisplay();
        break;
        case20:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmss
        a2=0,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,
        mca=1,park=0,plac=0,princi=0;
        glutPostRedisplay();
        break;
        case21:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmss
        a2=0,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,
        mca=0,park=1,plac=0,princi=0;
        glutPostRedisplay();
        break;
        case22:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmss
        a2=0,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,
        mca=0,park=0,plac=1,princi=0;
        glutPostRedisplay();
        break;
        case23:academic=0,account=0,admin=0,bank=0,amphi=0,science=0,bmssa1=0,bmss
        a2=0,bsn=0,cafe=0,civil=0,cse=0,ece=0,found=0,hostel=0,gym=0,ise=0,lab=0,lib=0,
        mca=0,park=0,plac=0,princi=1;
        glutPostRedisplay();
        break;
        case 24:exit(0);
        }
}
```

Keyboard interface:-

```
void keyboard(unsigned char k,int x,int y)
{
```

```
        if(k=='s'){then=1; glutPostRedisplay();}
        if(k=='v'){help=1; glutPostRedisplay();}
        if(k == 'x'){glRotatef(10,1,0,0);glutPostRedisplay();}
        if(k == 'y'){glRotatef(10,0,1,0);glutPostRedisplay();}
        if(k == 'z'){glRotatef(10,0,0,1);glutPostRedisplay();}
         if(k=='n'){navi=1; glutPostRedisplay();}
        display();
        if(k == 'X'){glRotatef(-10,1,0,0);glutPostRedisplay();}
        if(k == 'Y'){glRotatef(-10,0,1,0);glutPostRedisplay();}
        if(k == 'Z'){glRotatef(-10,0,0,1);glutPostRedisplay();}

}
```

4.4 SUMMARY

The implementation of the project is about the source code of the project. It will give the user an idea about how the program is going to function. By having this knowledge about the program the user can implement and understand it a better way.

CHAPTER - 5

CONCLUSION AND FUTURE ENHANCEMENT

The development of a project is not an easy process as it involves a lot of challenges in different stages of software analysis, design, coding and testing. Thus, developing & implementation of this project has come in handy in better understanding of the concepts of Computer Graphics programming using OpenGL.

A modular approach is adopted in the development of the project. This project proves to be a simple and robust, providing the features offered by all standard graphical applications. The functions so described are successfully implementing a graphics demonstration. Finally we provided basic 3D model of BMSIT&M campus.

We have used simple inbuilt glut library functions in our project which provides a clear understanding of the model. Based on the different inputs chosen, different paths will be shown.

Future enhancements of this project may include:

- Animations can be done in detail.
- More graphical functions can be implemented.
- Detailed graphical implementation can be done.
- Collision detection algorithm can be added.
- Perspective views can be added and camera viewing options can be given.
- More than one path can be shown.

SNAPSHOTS

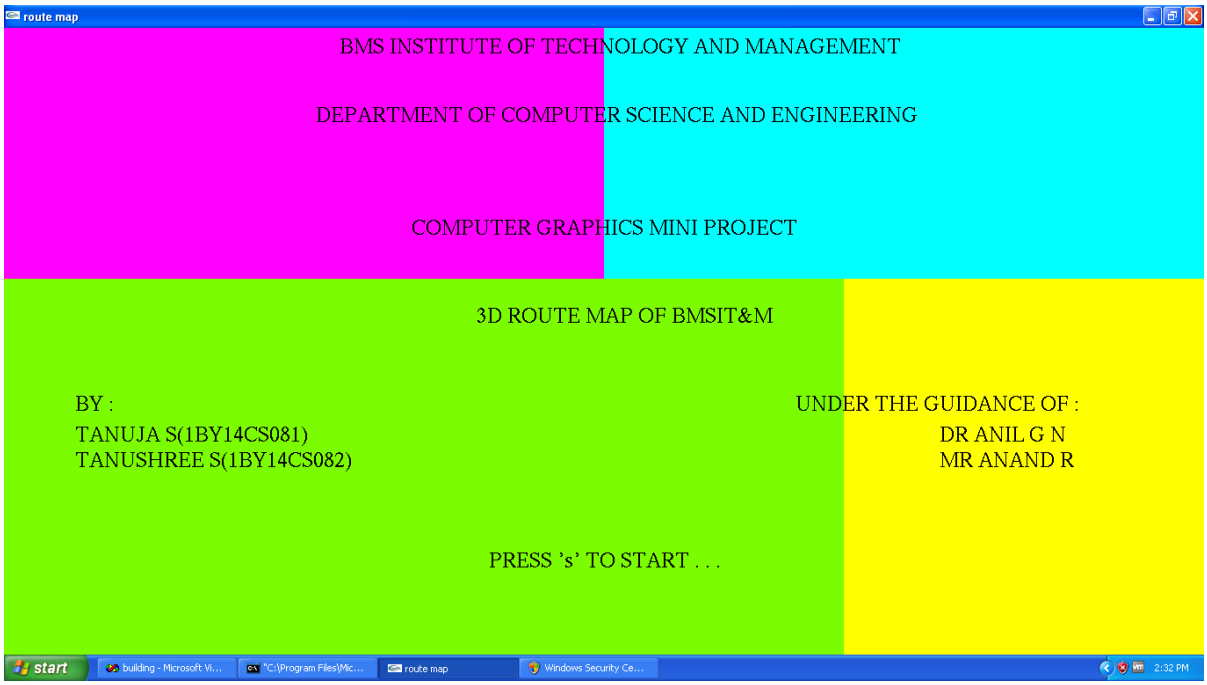

Fig 6. 1: Startup window.

Fig 6. 2: Help window.

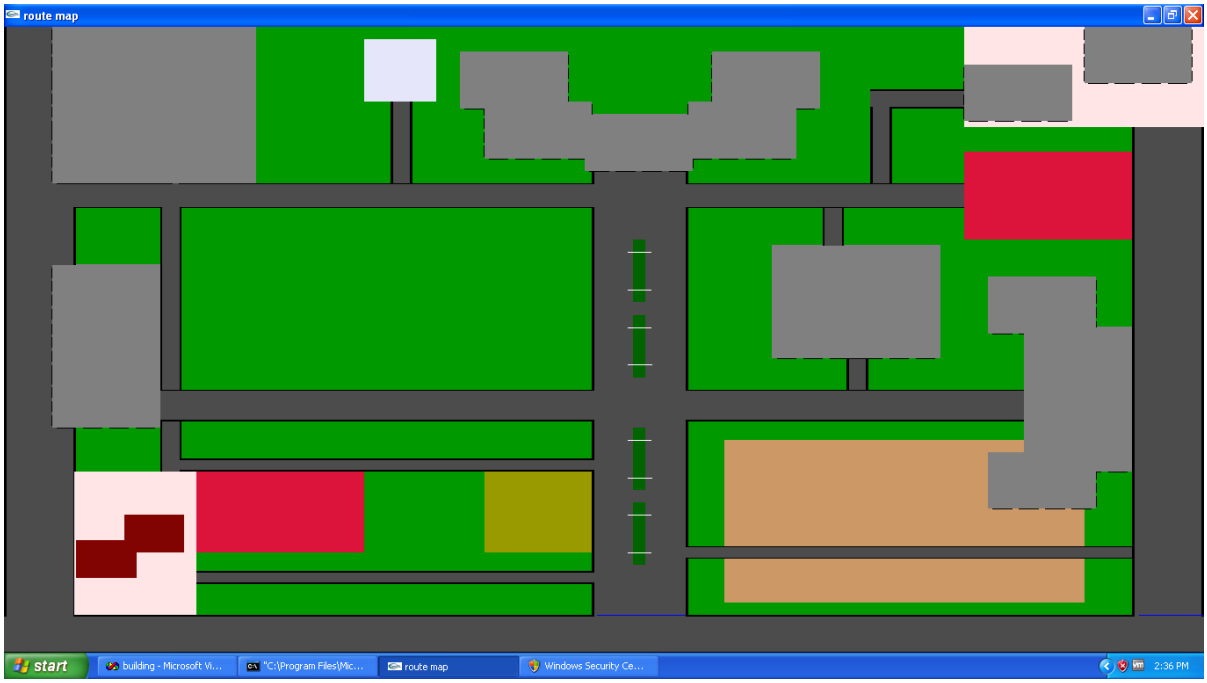

Fig 6.3: Top view of the campus.

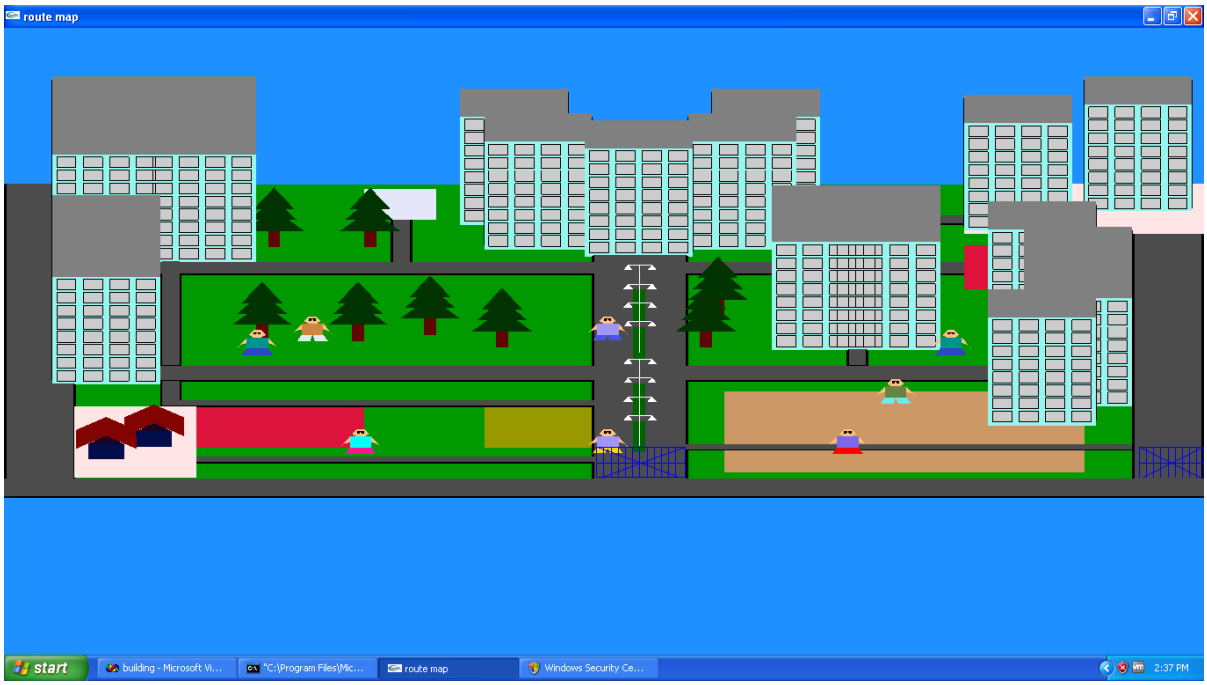

Fig 6. 4: Front view of the campus.

Fig 6.5: Side view of the campus.

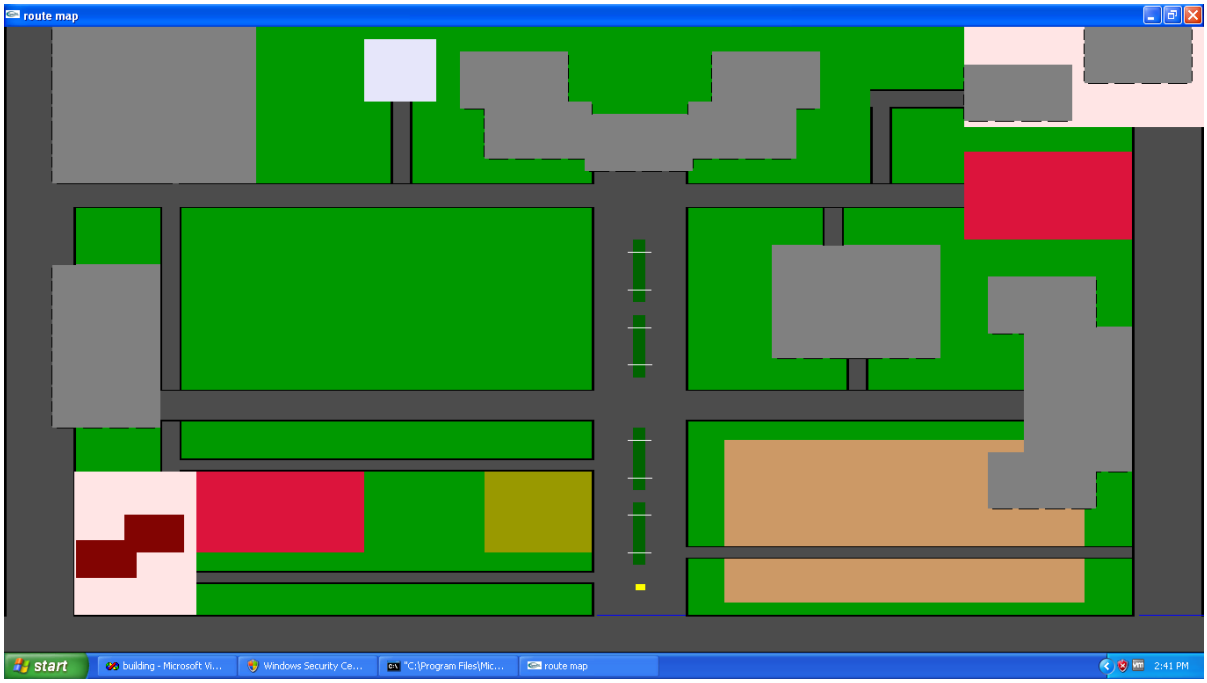

Fig 6.6: Top view showing the starting point of navigation.

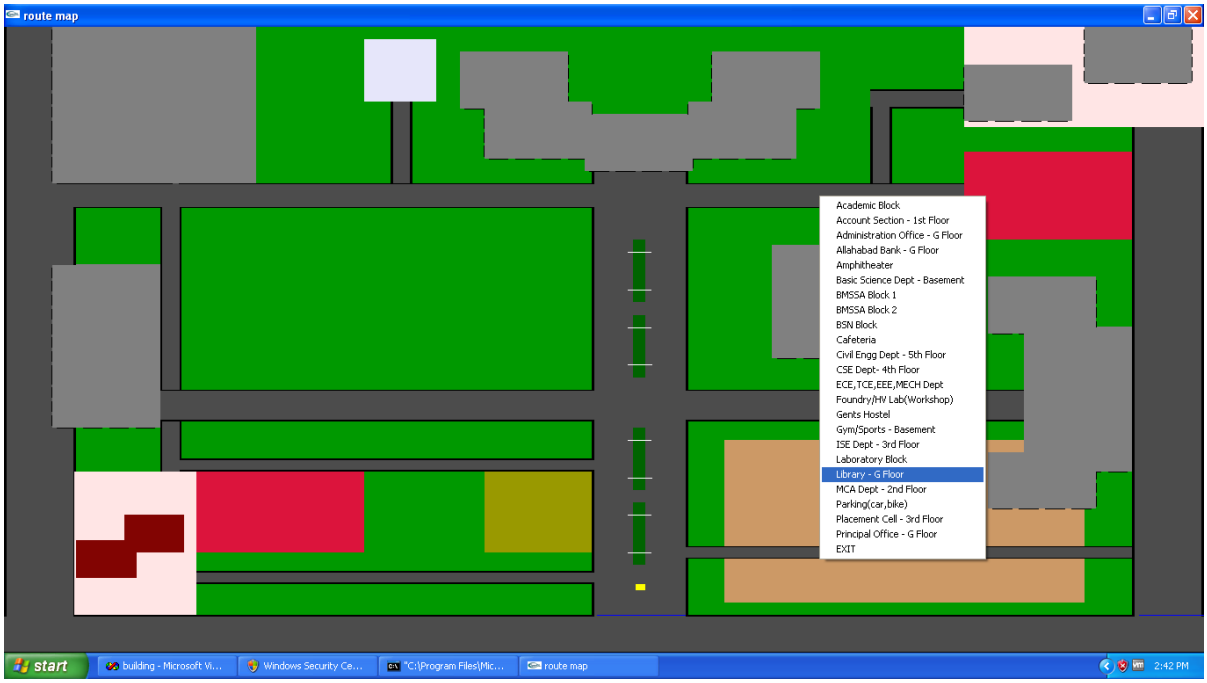

Fig 6.7: Selecting the destination (Library) using right mouse click.

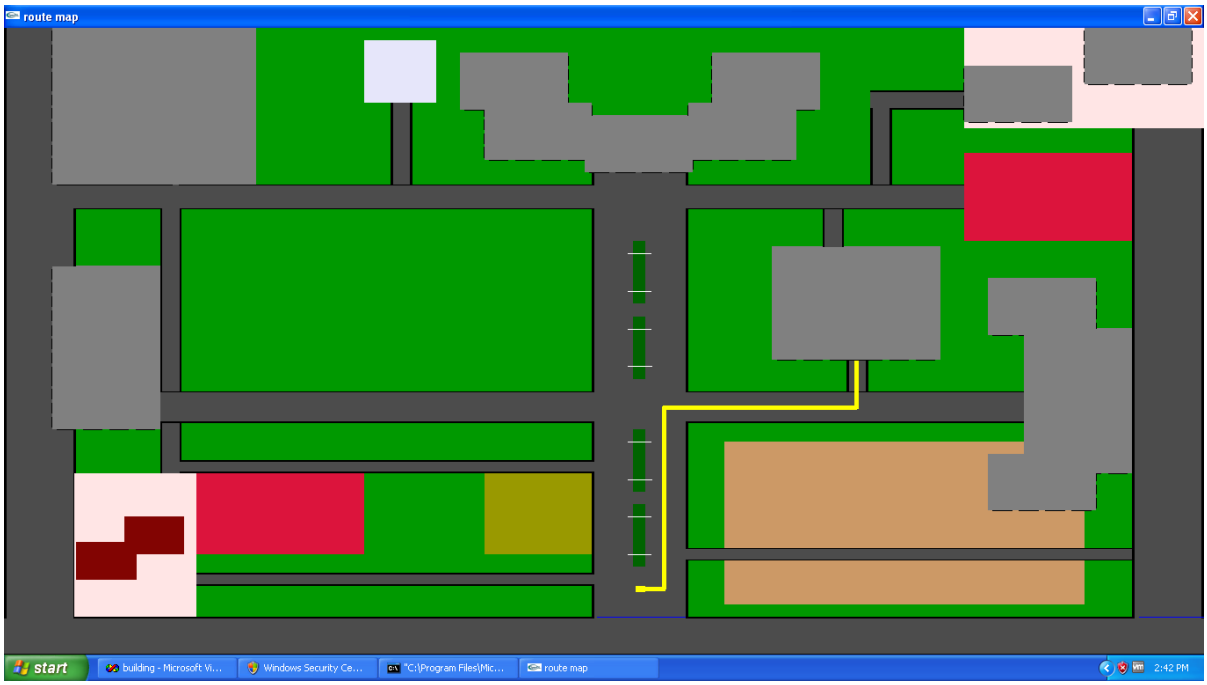

Fig 6.8: Output path shown for selected destination (Library)-TOP VIEW.

Fig 6.9: Output path shown for selected destination (Library)-FRONT VIEW.

Fig 6.10: Output path shown for selected destination (Library)-SIDE VIEW.

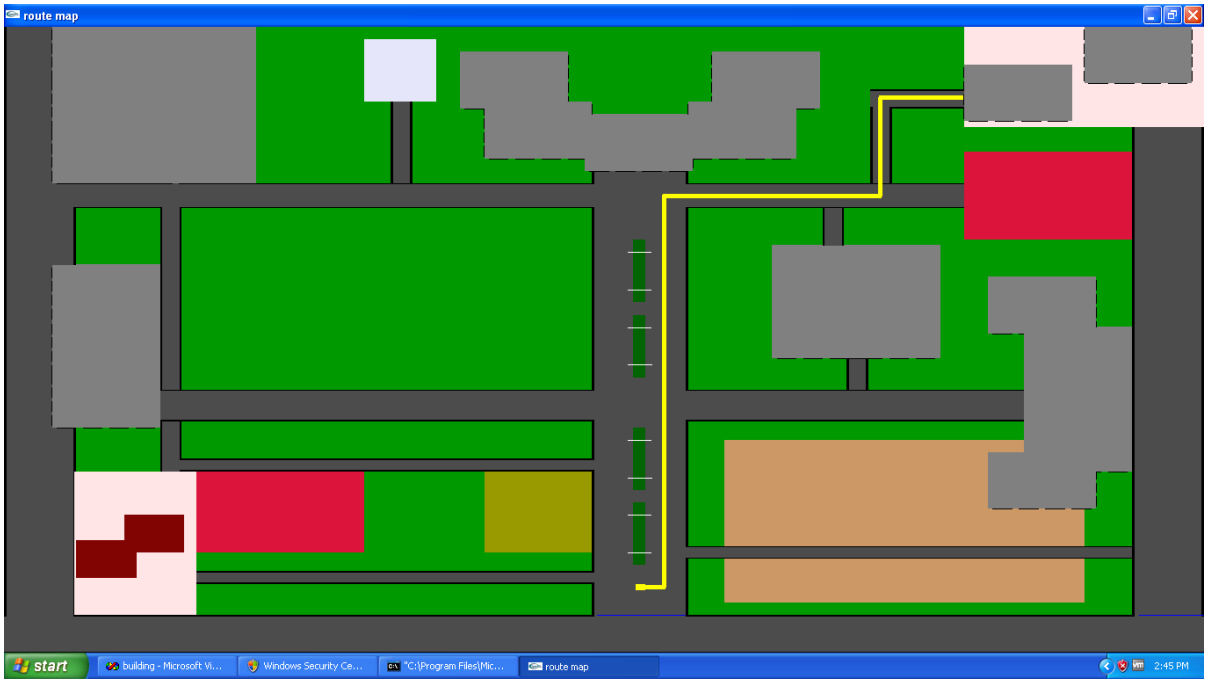

Fig 6.11: Output path shown for selected destination (Gents hostel)-TOP VIEW.

Fig 6.12: Output path shown for selected destination (Gents hostel)-FRONT VIEW.

Fig 6.13: Output path shown for selected destination (Gents hostel)-SIDE VIEW.

Fig 6.14: Output path shown for selected destination (Gents hostel)-BACK VIEW.

Fig 6.15: Choosing EXIT option from the Menu .

REFERENCES

Reference Books:

- **Edward Angel**, "Interactive Computer Graphics",5th edition, Pearson Education, 2009.
- **Donald D Hearn** and **M. Pauline Baker,** " Computer Graphics with OpenGL",3rd Edition.
- **Kunii T.L.** and **Springer Verlag , "**Computer Graphics", Berlin 2007.
- **Hearn** and **Bakar P.M., "**Computer Graphics"Prentice Hall , New Delhi-2007.
- **Kinetkar Y., "**Graphics under C" BPB Publications, New Delhi-2002.

Web site:

- www.opengl.org.
- www.cecs.pdx.edu.