## VISVESVARAYA TECHNOLOGICAL UNIVERSITY

#### BELGAVI, KARNATAKA -590 018

**A Minor Project Report on**

**“village stimulation”**

***Submitted in partial fulfillment for the Computer Graphics and Visualization Laboratory course of Sixth Semester of Bachelor of Engineering in Computer Science & Engineering during the academic year 2019-20.***

#### By

**Oswin s rego 4MN17CS023 Harshith n 4MN17CS015**

**|| Under the Guidance of ||**

#### Prof. Bharath Bharadwaj B S

**Assistant Professor Dept. of CS&E MIT Thandavapura**

#### 2019-20

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURA**

#### NH 766, Nanjangud Taluk, Mysuru – 571302

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURA



# ~~ CERTIFICATE ~~

*Certified that the minor project work entitled “****Multiplex Theatre****” is a bonafide work carried out by* ***oswin s rego*** *(4MN17CS023) and Harshith N**(4MN17CS015) for the Computer Graphics and Visualization Laboratory with course code 15CSL68 of Sixth Semester in Computer Science & Engineering under Visvesvaraya Technological University, Belagavi during academic year 2019-20.*

*It is certified that all corrections/suggestions indicated for Internal Assignment have been incorporated in the report. The report has been approved as it satisfies the course requirements.*

|  |  |
| --- | --- |
| \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ Signature of Lab Staff In-Charge  **Prof. Bharath Bharadwaj B S**  Assistant Professor  Dept. of CS&E MIT Thandavapura | \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ Signature of the HOD  **Prof. Lakshmi Durga**  Assistant Professor & HOD Dept. of CS&E  MIT Thandavapura |

**External viva**

**Name of the Examiners Signature with date**

1)…………………………………………………………………………………….

2)…………………………………………………………………………………….

# ~~~~ ACKNOWLEDGEMENT ~~~~

It is the time to acknowledge all those who have extended their guidance, inspiration and their whole hearted co-operation all along our project work.

We are grateful to **Dr. Y T Krishnegowda**, principal, MIT Thandavapura, **Dr. H K Chethan,** Professor and Mentor, CS&E, MIT Thandavapura and also **Prof. Lakshmi Durga**, HOD, CS&E, MIT Thandavapura for having provided us academic environment which nurtured my practical skills contributing to the success of our project.

We wish to place a deep sense of gratitude to all Teaching and Non-Teaching staffs of Computer Science and Engineering Department for whole-hearted guidance and constant support without which Endeavour would not have been possible.

Our gratitude will not be complete without thanking our parents and also our friends, who have been a constant source of support and aspirations.

Oswin s rego

Harshith n

# ~~~ ABSTRACT ~~~

Computer Graphics is one of the ever emerging fields with the new inventions in the display technologies. Computer Graphics is an effective medium for communication between man and computer.

The main of this project is the pictorial view of village stimulation using openGL. The features of the village stimulation revolve around showing the view of scenery of the village including all its attributes or the features.

The project involves the following features:

1) Helicopter.

2) Bullock cart.

3) Houses and the railway station.

The above features are supported by an easy to use keyboard interface.

There is the night vision of the village, which involves the helicopter the railway station and the some common features of the village such as the houses bullock cart etc.

# ~~~~~ CONTENTS ~~~~~

|  |  |  |
| --- | --- | --- |
| **SL.No** | **Index** | **Page No.** |
| **1** | **INTRODUCTION** | **1** |
| 1.1 | Aim of the Project | 1 |
| 1.2 | Overview of the Project | 1 |
| 1.3 | Outcome of the Project | 2 |
| **2** | **DESIGN AND IMPLEMENTATION** | **3** |
| 2.1 | Algorithm | 3 |
| 2.2 | Flowchart | 7 |
| 2.3 | OpenGL API’s Used with Description | 13 |
| **3** | **RESULT ANALYSIS** | **16** |
| 3.1 | Snapshots | 16 |
| 3.2 | Discussion | 23 |
| **4** | **CONCLUSION AND FUTURE WORK** | **24** |
|  | **APPENDIX** | **25** |
|  | **REFERENCES** |  |

## Chapter 1

**INTRODUCTION**

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

Although early application in engineering & science had to rely on expensive & cumbersome equipment, advances in computer technology have made interactive computer graphics a practical tool. Computer Graphics is used in a diverse area such as science, engineering, medicine, business, industry, government, art, entertainment, education and training. Now it can be answered about computer graphics as generalized tool for drawing and creating pictures and simulates the real world situations within a small computer window.

### Aim of the project

The main aim of this project is to simulate a village using OpenGL. The features of the village simulation revolve around showing the common features with the use of the night effect.

### Overview of the project

The project involves the following features:

* + - Helicopter.
    - Houses and hills.
    - Bullock cart and railway station.

The above features are supported by an easy to use keyboard interface. All the movie screens are organized .

There is the night vision of the village, which involves the helicopter the railway station and the some common features of the village such as the houses bullock cart etc.

### Outcome of the project

To draw the computer version of the simple ‘village stimulation’ and its role in daily life. It is the graphical representation of village stimulation and it’s working. The village stimulation show the real life graphical implementation of what happens in a common village, how people life moves around in the village.

## Chapter 2

**DESIGN AND IMPLEMENTATION**

### Algorithm:

Step 1: Start

Step 2: print the keyboard options Step 3: Initialize the viewing window Step 4: Initialize the display mode Step 5: Initialize window size

Step 6: Initialize window position

Step 7: Display the theater goto step 13 Step 8: For keyboard interface goto step 14 Step 9: For mouse interface goto step 24

Step 10: For initializing the view goto step 11 Step 11: Clear color

Step 12: Matrix mode Step 13: Return to step 10

Step 14: For display1 function goto step 28 Step 15: If key==’e’||’E’ then goto step 33 Step 16: If key==’h’||’H’ then goto step 33 Step 17: If key==’k’||’K’ then goto step 33 Step 18: If key==’t’||’T’ then goto step 33 Step 19: If key==’n’||’N’ then goto step 36 Step 20: If key==’x’||’X’ then goto step 28 Step 21: If key==’m’||’M’ then goto step 41

Step 22: If key==’p’||’P’ then goto step 44 Step 23: Return to step 8

Step 24: If button == GLUT\_LEFT\_BUTTON &amp;&amp; state == GLUT\_DOWN

Step 25: If id2 != 1 OR id2 == 2 then goto step 41 return to step 9

Step 26: If button == GLUT\_RIGHT\_BUTTON &amp;&amp; State == GLUT\_DOWN then goto step 28

Step 27: Return to step 9

//display 1

Step 28: Create building goto step 48 Step 29: Create tree1 and tree2

Step 30: Create car goto step 50 Step 31: Create sun goto step 60 Step 32: Return to step 14

Step 33: Create theatre goto step 62 Step 34: Create a man goto step 65 Step 35: Return to step 15

//night

Step 36: Create building goto step 48 Step 37: Create trees

Step 38: Create car goto step 50 Step 39: Create moon goto step 67 Step 40: Return to step 19

//display2

Step 41: Clear color buffer bit

Step 42: Create theatre goto step 62 Step 43: Return to step 24

//popcorn

Step 44: Create theatre goto step 62 Step 45: Create salesman

Step 46: Create a man Step 47: Return to step 22

//building

Step 48: Create railings Step 49: Create side railings Step 50: Create board

Step 51: Create door

Step 52: Return to step 28

//car

Step 53: Create trees Step 54: Create front part

Step 55: Create head lights Step 56: Create mirrors

Step 57: Create window with rear Step 58: Create line over light Step 59: Return to step 30

//sun

Step 60: Create sun with circle Step 61: return to step 31

//theatre

Step 62: Create train,helicopter,cart

Step 63: Create sun goto step 60 Step 64: Return to step 44

//scaleman

Step 65: Create hill

Step 66: return to step 34

//moon

Step 67: Create moon with circle Step 68: Create star goto step 70 Step 69: Return to 39

//stars

Step 70: Create star with point size function Step 71: Return to step 39

//night

Step 72: If key==’d’||D goto step 28 Step 73: Return to step 28

Step 74: End

### Flowchart:

#### display function:

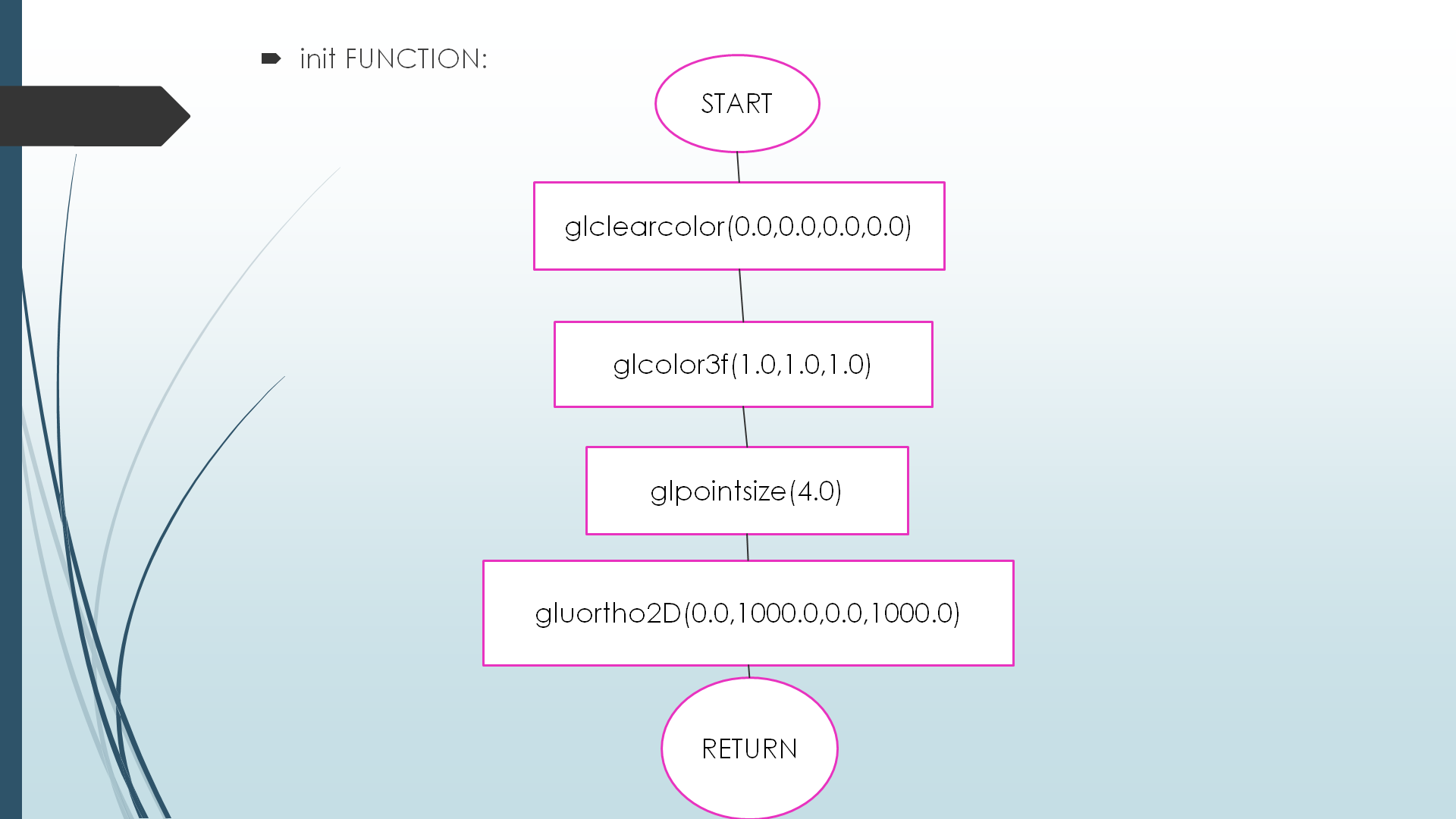


start

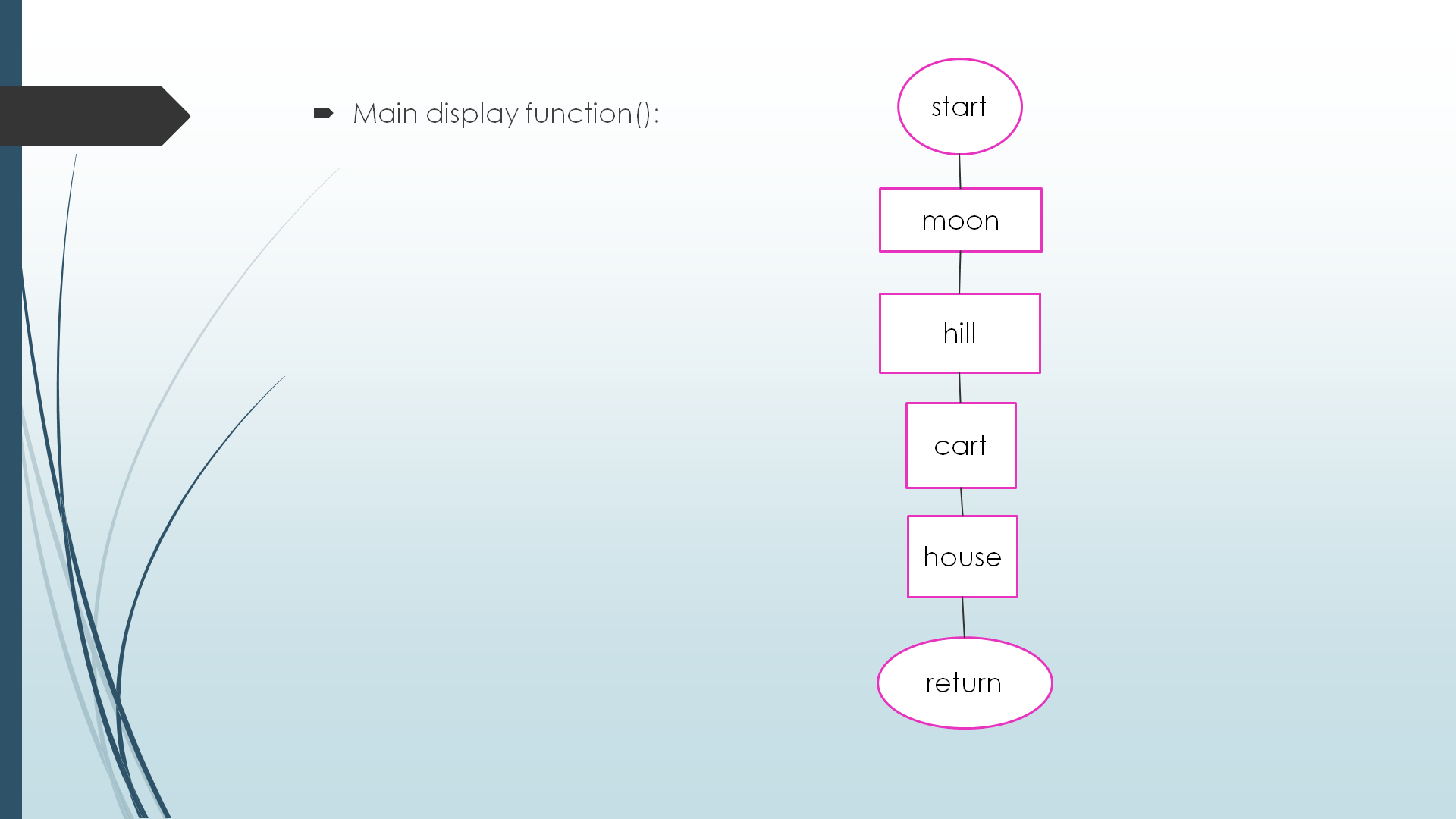
end

|  |  |  |
| --- | --- | --- |
|  | glClear(GL\_COLOR\_B UFFER\_BIT) |  |

|  |  |  |
| --- | --- | --- |
|  | display() |  |

****



****

**Fig:flow chart**

* 1. **openGL API’s used with description**

**OpenGL (Open Graphics Library):**

OpenGL has become a widely accepted standard for developing graphics application. OpenGL is easy to learn, and it possesses most of the characteristics of other popular graphic system. It is top-down approach. OpenGL 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 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 direct 3D on Microsoft Windows platforms.

OpenGL provides a powerful but primitive set of rendering command, and all higher- level drawing must be done in terms of these commands. There are several libraries that allow you to simplify your programming tasks, including the following:

* OpenGL Utility Library (GLU) contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections and rendering surfaces.
* OpenGL Utility Toolkit (GLUT) is a window-system-independent toolkit, used to hide the complexities of differing window APIs. Rather than using a different library for each system we use available library called openly utility toolkit. It is used as #include

<glut.h>.

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

This project makes extensive use of translations, rotations and scaling for creating provides the description of the following functions.

|  |  |  |
| --- | --- | --- |
| **Sl No** | **OpenGL API’s Used** | **Description** |
| 1 | **void glBegin(glEnum mode);** | Initiates a new primitive of type mode and starts the collection of vertices. Values of mode include GL\_POINTS, GL\_LINES and  GL\_POLYGON. |

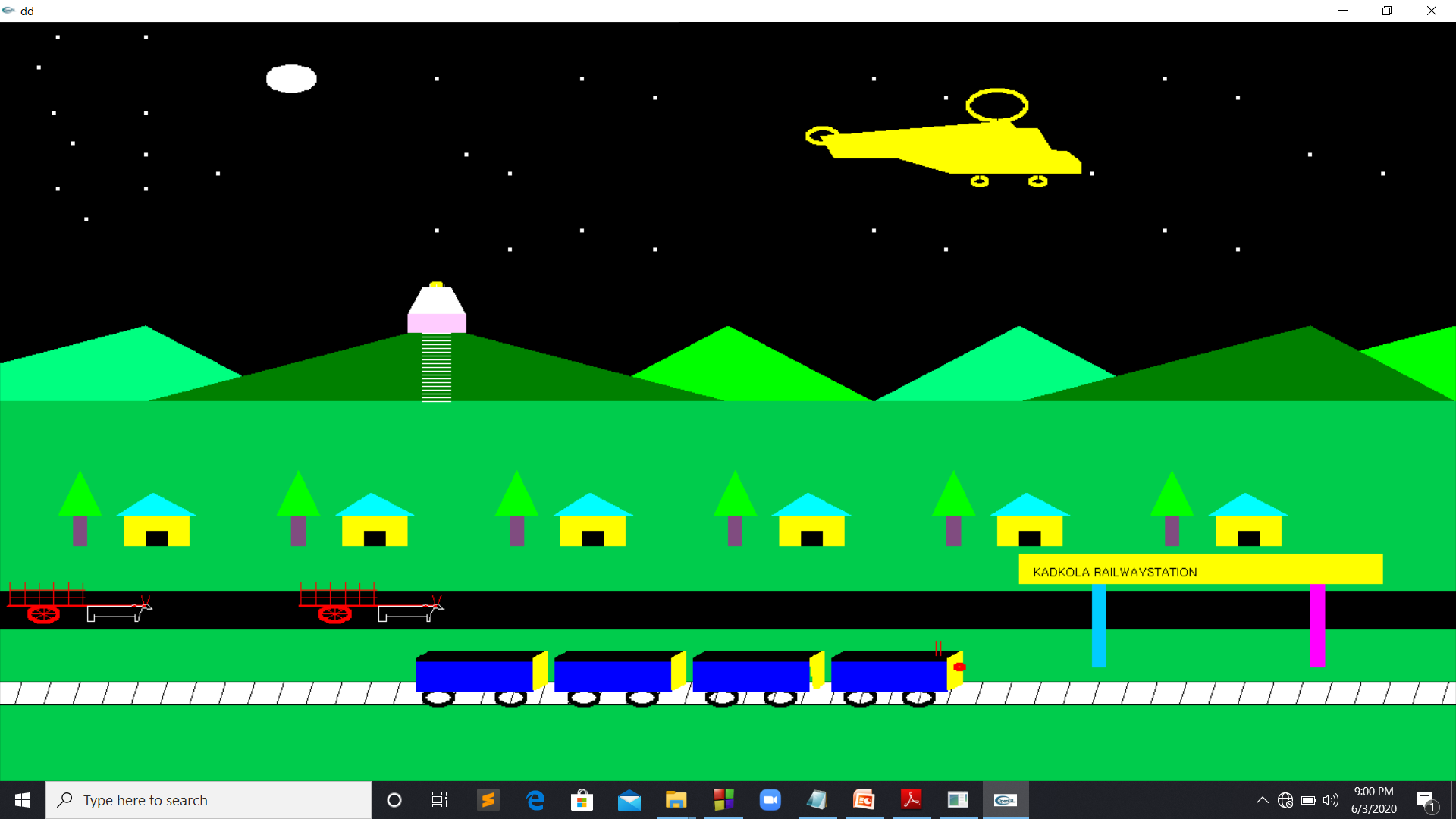
|  |  |  |
| --- | --- | --- |
| 2 | **void glEnd( );** | It terminates a list of vertices. |
| 3 | **void glColor3f[ i f d ] (TYPE r, TYPE g, TYPE b);** | Sets the present RGB colors. Valid types are int ( i ), float ( f ) and double ( d ). The maximum and minimum values of the floating-point types are 1.0 and 0.0,  respectively. |
| 4 | **void glClearColor(GLclampf r,GLclampf g,GLclampf b,GLclampf a);** | Sets the present RGBA clear color used when clearing the color buffer. Variables of GLclampf are floating-point numbers between  0.0 and 1.0. |
| 5 | **int glutCreateWindow(char**  **\*title);** | Creates a window on the display. The string title can be used to label the window. The return value provides a reference to the window that can be used where there are  multiple windows. |
| 6 | **void glutInitWindowSize(int**  **width, int height);** | Specifies the initial height and width of the  window in pixels. |
| 7 | **void glutInitWindowPosition(int**  **x, int y);** | Specifies the initial position of the top-left  corner of the window in pixels. |
| 8 | **void glutInitDisplayMode(unsigned int mode);** | Request a display with the properties in mode. The value of mode is determined by the logical OR of operation including the color model (GLUT\_RGB, GLUT\_INDEX) and buffering (GLUT\_SINGLE,  GLUT\_DOUBLE); |
| 9 | **void glFlush( );** | Forces any buffered any OpenGL commands  to execute. |
| 10 | **void glutInit (int argc, char**  **\*\*argv);** | Initializes GLUT. The arguments from main are passed in and can be used by the  application. |
| 11 | **void glutMainLoop( );** | Cause the program to enter an event  processing loop. It should be the last statement in main. |

|  |  |  |
| --- | --- | --- |
| 12 | **void glutDisplayFunc(void (\*func) (void));** | Registers the display function func that is  executed when the window needs to be redrawn. |
| 13 | **gluOrtho2D(GLdouble left, GLdouble right, GLdouble**  **bottom, GLdouble top);** | Defines a two-dimensional viewing rectangle in the plane Z=0; |
| 14 | **void glutBitmapCharacter(void**  **\*font, int char);** | Renders the character with ASCII code char at the current raster position using the raster font given by font. Fonts include GLUT\_BITMAP\_TIMES\_ROMAN\_10 and GLUT\_BITMAP\_TIMES\_ROMAN\_8\_Y\_1  3. The raster position is incremented by the  width of the character. |
| 15 | **void glClear(GL\_COLOR\_BUFFER**  **\_BIT);** | To make the screen solid and white. |
| 16 | **void MouseFunc(myMouse);** | It is used for the implementation of mouse interface. Passing the control to void myMouse(int button,int state,int x,int y); |
| 17 | **17 void KeyboardFunc(key);** | It is used for the implementation of keyboard interface.  Passing control to void key(unsigned char  key,int x,int y); |
| 18 | **void glLoadMatrix[fd](TYPE**  **\*m);** | Loads the 16 element array of TYPE GLfloat or GLdouble as a current matrix. |

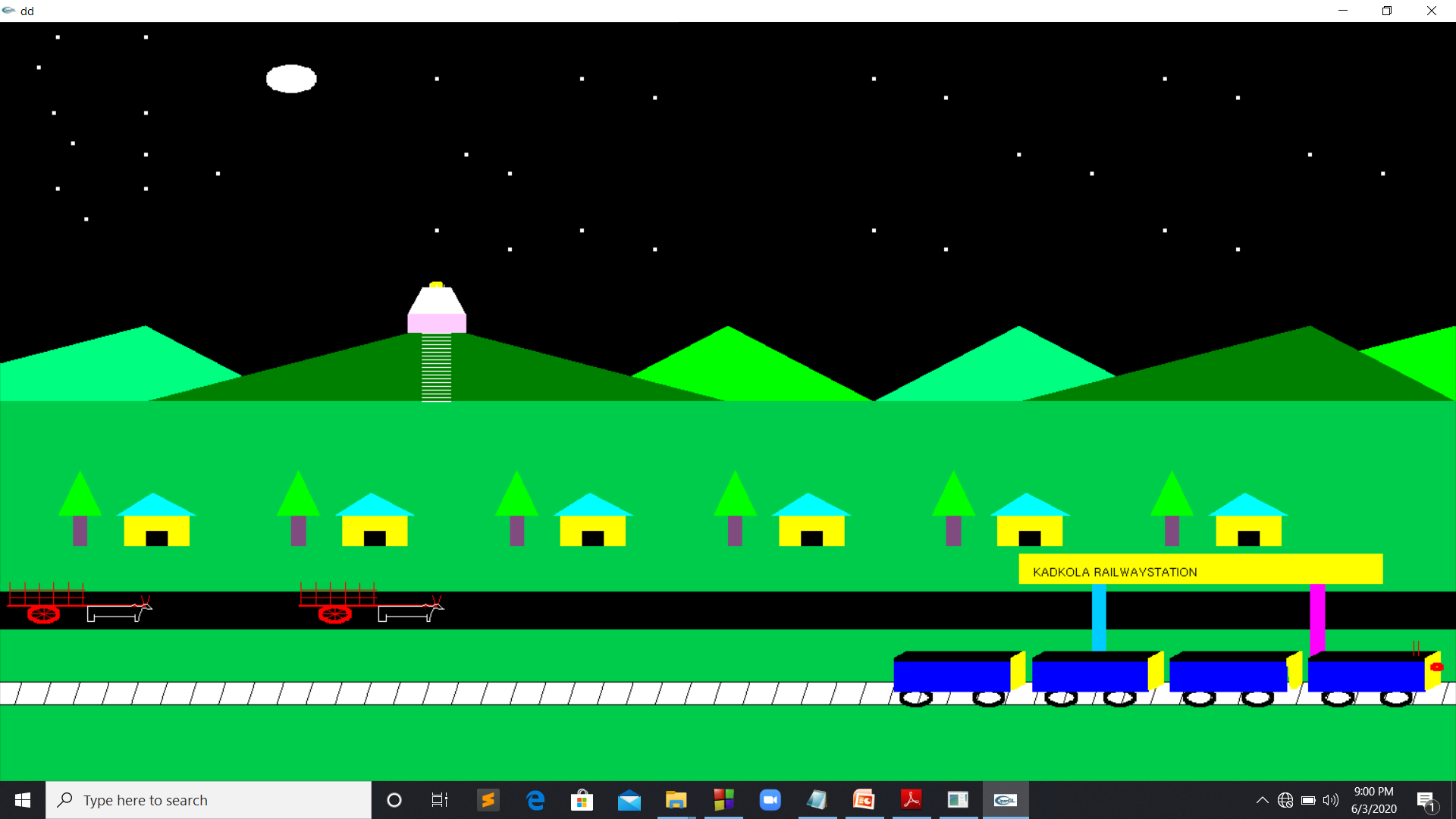
## Chapter 3

#### SNAPSHOTS:

**RESULT ANALYSIS**

****

**fig3.1 view of the village simulaton**

****

**f**

## Chapter 4

**CONCLUSION AND FUTURE ENHANCEMENT**

### CONCLUSION:

The village simulation has been drawn and displayed in 2D objects by using the computer graphics, while showing how it’s works in general. It can be easily understood by the user. The demo is made more interactive with a keyboard and mouse interaction module in the program.

In this project, we tried to implement all the ideas and expectations to bring this project into reality. Although, we were limited by time period and our abilities to perform these ideas and make it real.

### FUTURE ENHANCEMENT:

Even though demo designed is enriched with many options, it is a two dimensional demo, in future it can be redesigned with 3D animation and sound effects. In future we can make a movement of a man. And also we can add a video to the multiplex screen. Along with this music can also be added. Making the user interface of this program simpler and more user friendly. By using this demo further for no of screen from user’s choice can be implemented.

# APPENDIX

### PROJECT CODE:

#include<windows.h>

#include<GL/glut.h>

#include<stdlib.h>

GLfloat x1,x2,y1,y2,x3,x4,y3,y4,x5,y5;

void edgedetect(GLfloat x1,GLfloat y1,GLfloat x2,GLfloat y2,int \*le,int \*re)

{

float mx,x,temp;

int i;

if((y2-y1)<0)

{

temp=x1; x1=x2; x2=temp;

temp=y1; y1=y2; y2=temp;

}

if((y2-y1)!=0)

mx=(x2-x1)/(y2-y1);

else

mx=x2-x1;

x=x1;

for(i=y1;i<y2;i++)

{

if(x<le[i])

le[i]=x;

if(x>re[i])

re[i]=x;

x+=mx;

}

}

void dp(GLint cx,GLint cy)

{

glBegin(GL\_POINTS);

glVertex2i(cx,cy);

glEnd();

}

void scanfill(float x1,float y1, float x2, float y2, float x3, float y3,float x4,float y4)

{

int le[1000],re[1000];

int i,y;

for(i=0;i<1000;i++)

{

le[i]=1000;

re[i]=0;

}

edgedetect(x1,y1,x2,y2,le,re);

edgedetect(x2,y2,x3,y3,le,re);

edgedetect(x3,y3,x4,y4,le,re);

edgedetect(x4,y4,x1,y1,le,re);

for(y=0;y<1000;y++)

{

if(le[y]<=re[y])

for(i=le[y]+1;i<re[y];i++)

dp(i,y);

}

}

void pix(GLint h,GLint k,GLint x,GLint y)

{

dp(x+h,y+k);

dp(x+h,-y+k);

dp(-x+h,-y+k);

dp(-x+h,y+k);

dp(y+h,x+k);

dp(y+h,-x+k);

dp(-y+h,-x+k);

dp(-y+h,x+k);

}

void cd(GLint h,GLint k,GLint r)

{

GLint d=1-r,x=0,y=r;

while(y>x)

{

pix(h,k,x,y);

if(d<0)

d+=2\*x+3;

else

{

d+=2\*(x-y)+5;

--y;

}

++x;

}

pix(h,k,x,y);

}

void init()

{

glClearColor(0.0,0.0,0.0,0.0);

glColor3f(1.0,1.0,1.0);

glPointSize(4.0);

gluOrtho2D(0.0,1000.0,0.0,1000.0);

}

void display()

{

float i,j;

for(i=0;i<600;i+=0.5)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,1.0,1.0);

//sun

cd(200,925,0);

cd(200,925,1);

cd(200,925,2);

cd(200,925,3);

cd(200,925,4);

cd(200,925,5);

cd(200,925,6);

cd(200,925,7);

cd(200,925,8);

cd(200,925,9);

cd(200,925,10);

cd(200,925,11);

cd(200,925,12);

cd(200,925,13);

cd(200,925,14);

cd(200,925,15);

cd(200,925,16);

//glColor3f(0.9,0.3,0.0);

glColor3f(1.0,1.0,0.0);

cd(300,653,3);

glColor3f(1.0,1.0,1.0);

cd(300,925,0);

cd(900,825,0);

cd(800,925,0);

cd(700,825,0);

cd(400,925,0);

cd(320,825,0);

cd(600,925,0);

cd(100,825,0);

cd(300,725,0);

cd(900,825,0);

cd(800,725,0);

cd(700,825,0);

cd(400,725,0);

cd(320,825,0);

cd(600,725,0);

cd(100,825,0);

cd(300,925,0);

cd(950,800,0);

cd(850,900,0);

cd(750,800,0);

cd(450,900,0);

cd(350,800,0);

cd(650,900,0);

cd(150,800,0);

cd(350,700,0);

cd(950,800,0);

cd(850,700,0);

cd(750,800,0);

cd(450,700,0);

cd(350,800,0);

cd(650,700,0);

cd(150,800,0);

cd(100,980,0);

cd(100,880,0);

cd(100,780,0);

cd(40,980,0);

cd(37,880,0);

cd(40,780,0);

cd(27,940,0);

cd(50,840,0);

cd(59,740,0);

glColor3f(0.0,0.8,0.3);

glBegin(GL\_QUADS);

glVertex2i(0,500);

glVertex2i(1000,500);

glVertex2i(1000,0);

glVertex2i(0,0);

glEnd();

glColor3f(0.0,0.0,0.0);

glBegin(GL\_QUADS);

glVertex2i(0,250);

glVertex2i(1000,250);

glVertex2i(1000,200);

glVertex2i(0,200);

glEnd();

glColor3f(1.0,1.0,1.0);

glBegin(GL\_QUADS);

glVertex2i(0,100);

glVertex2i(0,130);

glVertex2i(1000,130);

glVertex2i(1000,100);

glEnd();

glColor3f(1.0,0.0,0.0);

cd(30,220,10);

glColor3f(1.0,0.0,0.0);

glBegin(GL\_LINES);

glVertex2i(20,220);

glVertex2i(40,220);

glVertex2i(30,230);

glVertex2i(30,210);

glVertex2i(37,227);

glVertex2i(23,213);

glVertex2i(37,213);

glVertex2i(23,227);

glVertex2i(5,230);

glVertex2i(100,230);

glVertex2i(100,230);

glVertex2i(100,232);

glVertex2i(100,232);

glVertex2i(5,232);

glVertex2i(7,232);

glVertex2i(7,262);

glVertex2i(17,232);

glVertex2i(17,262);

glVertex2i(27,232);

glVertex2i(27,262);

glVertex2i(37,232);

glVertex2i(37,262);

glVertex2i(47,232);

glVertex2i(47,262);

glVertex2i(57,232);

glVertex2i(57,262);

glVertex2i(7,242);

glVertex2i(59,242);

glVertex2i(7,252);

glVertex2i(59,252);

glVertex2i(99,233);

glVertex2i(97,245);

glVertex2i(101,233);

glVertex2i(103,245);

glEnd();

glColor3f(1.0,1.0,1.0);

glBegin(GL\_LINE\_LOOP);

glVertex2i(60,210);

glVertex2i(60,230);

glVertex2i(90,230);

glVertex2i(92,233);

glVertex2i(95,230);

glVertex2i(97,230);

glVertex2i(99,233);

glVertex2i(101,233);

glVertex2i(105,226);

glVertex2i(102,226);

glVertex2i(100,228);

glVertex2i(98,228);

glVertex2i(96,220);

glVertex2i(95,210);

glVertex2i(93,210);

glVertex2i(93,220);

glVertex2i(93,216);

glVertex2i(65,216);

glVertex2i(65,220);

glVertex2i(65,210);

glEnd();

//second cart

glColor3f(1.0,0.0,0.0);

cd(230,220,10);

glColor3f(1.0,0.0,0.0);

glBegin(GL\_LINES);

glVertex2i(220,220);

glVertex2i(240,220);

glVertex2i(230,230);

glVertex2i(230,210);

glVertex2i(237,227);

glVertex2i(223,213);

glVertex2i(237,213);

glVertex2i(223,227);

glVertex2i(205,230);

glVertex2i(300,230);

glVertex2i(300,230);

//glVertex2i(105,224);

//glVertex2i(105,224);

glVertex2i(300,232);

glVertex2i(300,232);

glVertex2i(205,232);

glVertex2i(207,232);

glVertex2i(207,262);

glVertex2i(217,232);

glVertex2i(217,262);

glVertex2i(227,232);

glVertex2i(227,262);

glVertex2i(237,232);

glVertex2i(237,262);

glVertex2i(247,232);

glVertex2i(247,262);

glVertex2i(257,232);

glVertex2i(257,262);

glVertex2i(207,242);

glVertex2i(259,242);

glVertex2i(207,252);

glVertex2i(259,252);

glVertex2i(299,233);

glVertex2i(297,245);

glVertex2i(301,233);

glVertex2i(303,245);

glEnd();

glColor3f(1.0,1.0,1.0);

glBegin(GL\_LINE\_LOOP);

glVertex2i(260,210);

glVertex2i(260,230);

glVertex2i(290,230);

glVertex2i(292,233);

glVertex2i(295,230);

glVertex2i(297,230);

glVertex2i(299,233);

glVertex2i(301,233);

glVertex2i(305,226);

glVertex2i(302,226);

glVertex2i(300,228);

glVertex2i(298,228);

glVertex2i(296,220);

glVertex2i(295,210);

glVertex2i(293,210);

glVertex2i(293,220);

glVertex2i(293,216);

glVertex2i(265,216);

glVertex2i(265,220);

glVertex2i(265,210);

glEnd();

glColor3f(1.0,1.0,0.5);

glBegin(GL\_POLYGON);

glColor3f(1.0,1.0,0.0);

glVertex2i(20+i+i,850);

glVertex2i(30+i+i,820);

glVertex2i(100+i+i,820);

glVertex2i(110+i+i,800);

glVertex2i(200+i+i,800);

glVertex2i(200+i+i,815);

glVertex2i(190+i+i,830);

glVertex2i(180+i+i,830);

glVertex2i(170+i+i,860);

glVertex2i(155+i+i,860);

glVertex2i(150+i+i,870);

glVertex2i(135+i+i,860);

glVertex2i(115+i+i,860);

glVertex2i(100+i+i,825);

glVertex2i(30+i+i,850);

glEnd();

//glClear(GL\_COLOR\_BUFFER\_BIT);

cd(142+i+i,890,20);

cd(170+i+i,790,5);

cd(130+i+i,790,5);

cd(22+i+i,850,10);

glColor3f(0.0,0.0,0.0);

cd(30+i,110,10);

cd(80+i,110,10);

cd(130+i,110,10);

cd(170+i,110,10);

cd(225+i,110,10);

cd(265+i,110,10);

cd(320+i,110,10);

cd(360+i,110,10);

glBegin(GL\_TRIANGLES);

glVertex2i(200+i+i+i,800);

glVertex2i(200+i+i+i,815);

glVertex2i(215+i+i+i,808);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex2i(200+i+i+i+i,800);

glVertex2i(200+i+i+i+i,815);

glVertex2i(215+i+i+i+i,808);

//glEnd();

//glBegin(GL\_TRIANGLES);

glVertex2i(200+i+i+i+i+i,800);

glVertex2i(200+i+i+i+i+i,815);

glVertex2i(215+i+i+i+i+i,808);

glEnd();

glBegin(GL\_LINES);

glVertex2i(130+i+i,790);

glVertex2i(130+i+i,800);

glVertex2i(170+i+i,790);

glVertex2i(170+i+i,800);

glVertex2i(142+i+i,870);

glVertex2i(142+i+i,880);

glVertex2i(0,100);

glVertex2i(1000,100);

glVertex2i(0,130);

glVertex2i(1000,130);

for(j=0;j<=1000;j+=20)

{

glVertex2i(10+j,100);

glVertex2i(15+j,130);

}

glEnd();

glColor3f(0.0,0.8,1.2);

glBegin(GL\_POLYGON);

glVertex2i(750,150);

glVertex2i(750,260);

glVertex2i(760,260);

glVertex2i(760,150);

glEnd();

glColor3f(1.0,0.0,1.5);

glBegin(GL\_POLYGON);

glVertex2i(900,150);

glVertex2i(900,260);

glVertex2i(910,260);

glVertex2i(910,150);

glEnd();

glColor3f(1.0,1.0,0.0);

glBegin(GL\_POLYGON);

glVertex2i(700,260);

glVertex2i(950,260);

glVertex2i(950,300);

glVertex2i(700,300);

glEnd();

glColor3f(0.0,0.0,0.0);

glRasterPos2i(710,270);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'K');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'A');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'D');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'K');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'O');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'L');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'A');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,' ');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'R');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'A');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'I');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'L');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'W');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'A');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'Y');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'S');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'T');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'A');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'T');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'I');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'O');

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12,'N');

glColor3f(0.0,0.0,1.0);

scanfill(15+i,120,15+i,160,95+i,160,95+i,120);

glColor3f(0.0,0.0,0.0);

scanfill(15+i,160,25+i,170,105+i,170,95+i,160);

glColor3f(1.0,1.0,0.0);

scanfill(95+i,160,105+i,170,105+i,130,95+i,120);

glColor3f(0.0,0.0,1.0);

scanfill(110+i,120,110+i,160,190+i,160,190+i,120);

glColor3f(0.0,0.0,0.0);

scanfill(110+i,160,120+i,170,200+i,170,190+i,160);

glColor3f(1.0,1.0,0.0);

scanfill(190+i,160,200+i,170,200+i,130,190+i,120);

glColor3f(0.0,0.0,1.0);

scanfill(205+i,120,205+i,160,285+i,160,285+i,120);

glColor3f(0.0,0.0,0.0);

scanfill(205+i,160,215+i,170,295+i,170,285+i,160);

glColor3f(1.0,1.0,0.0);

scanfill(285+i,160,295+i,170,295+i,130,288+i,120);

glColor3f(0.0,0.0,1.0);

scanfill(300+i,120,300+i,160,380+i,160,380+i,120);

glColor3f(0.0,0.0,0.0);

scanfill(300+i,160,310+i,170,390+i,170,380+i,160);

glColor3f(1.0,1.0,0.0);

scanfill(380+i,160,390+i,170,390+i,130,380+i,120);

glColor3f(1.0,0.0,0.0);

cd(388+i,150,3);

glBegin(GL\_LINES);

glVertex2i(375+i,165);

glVertex2i(375+i,185);

glVertex2i(372+i,165);

glVertex2i(372+i,185);

glEnd();

glColor3f(0.0,1.0,0.5);

glBegin(GL\_QUADS);

glVertex2i(0,500);

glVertex2i(0,550);

glVertex2i(100,600);

glVertex2i(200,500);

glEnd();

glColor3f(0.0,1.0,0.0);

glBegin(GL\_TRIANGLES);

glVertex2i(400,500);

glVertex2i(500,600);

glVertex2i(600,500);

glEnd();

glColor3f(0.0,0.5,0.0);

glBegin(GL\_TRIANGLES);

glVertex2i(100,500);

glVertex2i(300,600);

glVertex2i(500,500);

glEnd();

//second hill set

glColor3f(0.0,1.0,0.5);

glBegin(GL\_TRIANGLES);

glVertex2i(600,500);

//glVertex2i(600,550);

glVertex2i(700,600);

glVertex2i(800,500);

glEnd();

glColor3f(0.0,1.0,0.0);

glBegin(GL\_TRIANGLES);

glVertex2i(800,500);

glVertex2i(1000,600);

glVertex2i(1000,500);

glEnd();

glColor3f(0.0,0.5,0.0);

glBegin(GL\_TRIANGLES);

glVertex2i(700,500);

glVertex2i(900,600);

glVertex2i(1000,500);

glEnd();

for(j=0;j<=90;j+=5)

{

glColor3f(1.0,1.0,1.0);

glBegin(GL\_LINES);

glVertex2i(290,500+j);

glVertex2i(310,500+j);

glEnd();

}

glColor3f(1.5,0.8,1.2);

glBegin(GL\_POLYGON);

glVertex2i(280,590);

glVertex2i(280,615);

glVertex2i(320,615);

glVertex2i(320,590);

glEnd();

glColor3f(1.0,1.0,1.0);

glBegin(GL\_POLYGON);

glVertex2i(280,615);

glVertex2i(320,615);

glVertex2i(310,650);

glVertex2i(290,650);

glEnd();

glColor3f(1.0,1.0,1.0);

glBegin(GL\_LINES);

glVertex2i(295,650);

glVertex2i(295,655);

glVertex2i(300,650);

glVertex2i(300,655);

glVertex2i(305,650);

glVertex2i(305,655);

glEnd();

//tree

int k;

for(k=0;k<800;k+=150)

{

glColor3f(0.5,0.3,0.5);

glBegin(GL\_POLYGON);

glVertex2i(50+k,310);

glVertex2i(50+k,350);

glVertex2i(60+k,350);

glVertex2i(60+k,310);

glEnd();

glColor3f(0.0,1.0,0.0);

glBegin(GL\_TRIANGLES);

glVertex2i(40+k,350);

glVertex2i(55+k,410);

glVertex2i(70+k,350);

glEnd();

//house

glColor3f(1.0,1.2,0.0);

glBegin(GL\_POLYGON);

glVertex2i(85+k,310);

glVertex2i(85+k,350);

glVertex2i(130+k,350);

glVertex2i(130+k,310);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0.0,0.0,0.0);

glVertex2i(100+k,310);

glVertex2i(100+k,330);

glVertex2i(115+k,330);

glVertex2i(115+k,310);

glEnd();

glColor3f(0.0,1.0,1.0);

glBegin(GL\_TRIANGLES);

glVertex2i(80+k,350);

glVertex2i(105+k,380);

glVertex2i(135+k,350);

glEnd();

}

glFlush();

}

}

int main(int argc,char \*\*argv)

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(700,700);

glutCreateWindow("dd");

glutInitWindowPosition(0,0);

glutDisplayFunc(display);

init();

glutMainLoop();

return 0;

}

***REFERENCES***

1. *Edward Angel -* ***Interactive Computer Graphics: A Top-Down Approach using OpenGL*** *Fifth Edition, Published by Pearson Education, 2009*
2. *Computer Graphics Using OpenGL - F.S. Hill Jr.*
3. *The OpenGL Programming Guide, 5th Edition. The Official guide to learning OpenGL Version 2.1 by OpenGL Architecture Review Board.*