**CHAPTER 1**

**INTRODUCTION**

* 1. **INTRODUCTION TO OPENGL**

Computer Graphics is a complex and diversified technology. To understand the technology it is necessary to subdivide it into manageable Parts. This can be accomplished by considering that the end product of computer graphics is a picture. The picture may, of course, be used for a large variety of purposes; e.g., it may be an engineering drawing, an exploded parts illustration for a service manual, a business graph, an architectural rendering for a proposed construction or design project, an advertising illustration, or a single frame from an animated movie. The picture is the fundamental cohesive concept in computer graphics.

Consider how: Pictures are represented in computer graphics.

* Pictures are prepared for presentation.
* Previously prepared pictures are presented.
* Interaction with the picture is accomplished.

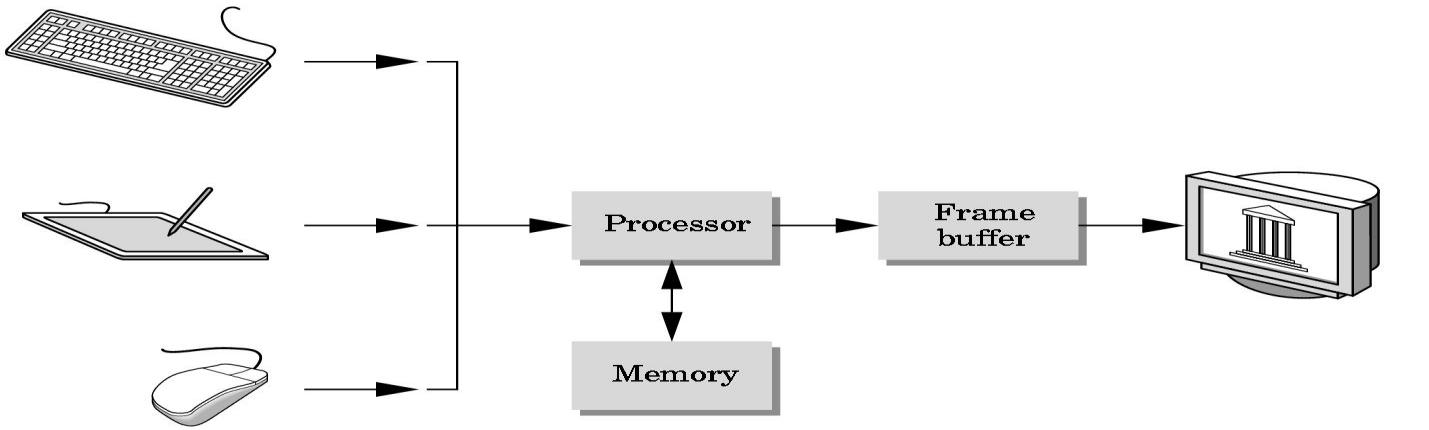
Here “picture” is used in its broadest sense to mean any collection of lines, points, text, etc. displayed on a graphics device.

* 1. **Computer Graphics**

The totality of computer graphics software encompasses the concepts from data structures, from data base design and management, from the psychology, ergonometric of the man-machine interface, from programming languages and operating system.

Numerous computer graphics standards can be grouped following categories.

* First is the graphics application interface, where ideas are translated into a form that is understandable by a computer system. Current representative standards are the GKS, GKS-3D, and the Programmer’s Hierarchical Interactive Graphics Standards (PHIGS).
* The Second is concerned with the storage and transmission of data between graphics systems and between graphics-based computer aided design and computer aided manufacturing systems. The current standard in this area is the Initial Graphics Exchange Specification (IGES).

A computer graphics system is a computer system with all the components of the general purpose computer system. There are five major elements in system: input devices, processor, memory, frame buffer, output devices.

**Fig. 1.1: A graphics system**

* 1. **A Brief History Of OpenGl**

OpenGL was developed by Silicon Graphics and is popular in the video games industry where it competes with Direct3D on Microsoft Windows IrisGL, a propietary graphics API, is the precursor of OpenGL. It is developed by Silicon Graphics Inc. (SGI). IrisGL was used as the starting point of an open standard for computer graphics that would save time porting applications by avoiding direct hardware access. After SGI cleaned up IrisGL and opened up the standard to other companies, OpenGL was born.

In 1992 the OpenGL Architectural Review Board (OpenGL ARB) was established. The OpenGL ARB is a group of companies that maintain and update the OpenGL standard.

In 2003 the first OpenGL (Exchange Specification) ES specification was released. OpenGL ES is a subset of OpenGL designed for mobile phones, embedded devices and video game systems.

In 2004 the OpenGL 2.0 specification was released, including the GLSL (OpenGL Shading Language) specification.

In August 2008, the OpenGL 3.0 specification was released.

1.2.1 What is OpenGL?

* OpenGL is a Software Interface to Graphics Hardware
* OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms
* It Consists of about 250 Distinct Commands.
* It is Hardware-independent Interface
* No command for windows or user input handling
* Does not include low-level I/O management
* It is developed primarily by SGI
* It consists of 2D/3D graphics, lower-level primitives (polygons)
* It is Basis for higher-level libraries/toolkits

What does it do?

* Main purpose is to render two and three dimensional objects into a frame buffer. These objects are described as sequences of vertices (which define geometric objects) or pixels (which define images).

**1.2.2 Programming using OpenGL: A first Introduction**

OpenGL is an API …

• Application programmers’ interface: link between

* low-level: graphics hardware
* high-level: application program you write

OpenGL is a …

Library for 2D and 3D graphics programming

• 200+ functions for building application programs

• Portable to many platforms (Win, Mac, Unix, Linux)

• Callable from many programming languages (C, Java,Perl, Python)

• Primarily concerned with modeling and rendering

Operations

* Specify geometric primitives (lines, pixels, polygons ...)
* Apply geometric transformations
* Specify camera, light, color, texture information, etc.

• No windowing or (platform-specific) input/interaction

* functions— these are the jobs of GLUT

**1.3 OpenGL-Related Libraries**

OpenGL provides a powerful but primitive set of rendering commands, and all higher-level drawing must be done in terms of these commands. Also, OpenGL programs have to use the underlying mechanisms of the windowing system. A number of libraries exist to allow you to simplify your programming tasks, including the following:

* The 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, performing polygon tessellation, and rendering surfaces. GLU routines use the prefix glu.
* The OpenGL Utility Toolkit (GLUT) is a window system-independent toolkit, written by Mark Kilgard, to hide the complexities of differing window system APIs. GLUT is the subject of the next section, and it’s described in more detail in Mark Kilgard’s book OpenGL Programming for the X Window System (ISBN 0-201-48359-9). GLUT routines use the prefix glut. "How to Obtain the Sample Code" in the Preface describes how to obtain the source code for GLUT, using ftp.

CHAPTER 2

REQUIREMENTS SPECIFICATION

2.1 Hardware Requirements:

* Intel® Pentium 4 CPU and higher versions
* 128 MB or more RAM.
* A standard keyboard, and Microsoft compatible mouse
* VGA monitor.

2.2 Software requirements:

* The graphics package has been designed for OpenGL; hence the machine must

Have code blocks

* Software installed preferably 6.0 or later versions with mouse driver installed.
* GLUT libraries, Glut utility toolkit must be available.
* Operating System**:** Windows
* Version of Operating System**:** Windows XP, Windows NT and Higher
* Language**:** C++
* Code::Blocks: cross-platform Integrated Development Environment (IDE)

**2.3 MISCELLANEOUS REQUIREMENTS:**

All the required library and header files should be available in the include directories. The files associated with this editor should be placed in either the same folder or in a specified folder.

**2.4 LANGUAGE USED IN CODING:**

C++ and OpenGL as an API.

CHAPTER 3

SYSTEM DESIGN

Design of any software depends on the architecture of the machine on which that software runs, for which the designer needs to know the system architecture. Design process involves design of suitable algorithms, modules, subsystems, interfaces etc.

**3.1 System Architecture**

**CONTROL FLOW DIAGRAM**

START

main()

special()

TimeEvent()

Display()

menu

timer()

Speed increase

day

background()

Speed decrease

night

flower()

quit

bee()

mountain()

Sun ()

Moon()

Fig. 3.1: System Control flow diagram

CHAPTER 4

IMPLEMENTATION

The implementation stage of this model involves the following phases.

* Implementation of OpenGL built in functions.
* User defined function Implementation.

**4.1 Implementation of OpenGL Built In Functions used:**

**1.glutInit():**

glutInit is used to initialize the GLUT library.

**Usage:**void glutInit(int \*argcp, char \*\*argv);

**Description:**glutInit will initialize the GLUT library and negotiate a session with the window system.

**2.glutInitDisplayMode():**

glutInitDisplayMode sets the initial display mode.

**Usage:**void glutInitDisplayMode (unsigned int mode);

Mode-Display mode, normally the bitwise OR-ing of GLUT display mode bit masks.

**Description:**The initial display mode is used when creating top-level windows, sub windows, and overlays to determine the OpenGL display mode for the to-be-created window or overlay.

**3.glutCreateWindow():**

glutCreateWindow creates a top-level window.

**Usage:**int glutCreateWindow(char \*name);

Name - ASCII character string for use as window name.

**Description:**glutCreateWindow creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name. Implicitly, the current window is set to the newly created window. Each created window has a unique associated OpenGL context.

**4.glutDisplayFunc():**

glutDisplayFunc sets the display callback for the current window.

**Usage:**void glutDisplayFunc(void (\*func)(void));

Func*-* The new display callback function.

**Description:**glutDisplayFunc sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback for the window is called. Before the callback, the current window is set to the window needing to be redisplayed and the layer in use is set to the normal plane. The display callback is called with no parameters. The entire normal plane region should be redisplayed in response to the callback.

**5.glutMainLoop():**

glutMainLoop enters the GLUT event processing loop.

**Usage:**void glutMainLoop(void);

**Description:** glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

**6.glMatrixMode( ):**

The two most important matrices are the model-view and projection matrix. At any time, the state includes values for both of these matrices, which are initially set to identity matrices. There is only a single set of functions that can be applied to any type of matrix. Select the matrix to which the operations apply by first set in the matrix mode, a variable that is set to one type of matrix and is also part of the state.

## 7.glutTimerFunc();

 glutTimerFunc registers a timer callback to be triggered in a specified number of milliseconds.

**Usage**:voidglutTimerFunc(unsigned int msecs,void (\*func)(int value), value);

**Description**:glutTimerFunc registers the timer callback func to be triggered in at least msecs milliseconds. The value parameter to the timer callback will be the value of the value parameter to glutTimerFunc. Multiple timer callbacks at same or differing times may be registered simultaneously. The number of milliseconds is a lower bound on the time before the callback is generated. GLUT attempts to deliver the timer callback as soon as possible after the expiration of the callback's time interval. There is no support for canceling a registered callback. Instead, ignore a callback based on its value parameter when it is triggered.

**4.2 Implementation of User Defined Functions:**

**1.void drawCircleFilled (double centerX, double centerY, double radiusX, double radiusY ):**

This function is used to draw a circle at a given position as centerx and given radius radius x and radius y. Here points are calculated using sin and cos functions.

**2.void bee():**

This function is used to draw a bee on the screen. This function intern uses wing() and leg() user defined functions.

**3.void drawCloud(int g1,int h1):**

This function is used to draw clouds at position (g1,h1) . Here we use the drawCircleFilled() function to draw clouds.

4.void background():

This function is used to set the background for the project. This function divides the screen into sky and ground and assigns the respective color .

5.void flower():

This function is used to draw the sunflower . Here we have used the drawCircleFilled() function to draw the petals and sepals of the flower . We have used glRotated() inbuilt function to draw all the petals. The GL\_LINES is used to draw the stem of the flower.

6.void moon():

This function draws the moon on the screen. We have overlapped two circles to get the curved part of the moon. This function uses the drawCircleFilled() function.

7.void star():

This function draws stars. Here we have used two overlapped triangles to draw stars.

8.void bird():

This function draws bird pictures on the screen at a given point. To draw the body we have used the drawCircleFilled() function and To draw tail we used GL\_LINES and To draw beak we use triangle.

9.void tree():

This function draws trees on the screen , This funcion intern calls the leaf() user defined

function. We used GL\_POLYGON to draw the leaves and branches of the tree.

**10.void PrintText(float x, float y, void\* font,int col, char \*string):**

This function prints the text on the screen. we have used glutBitmapCharacter() function to write the letter on the screen. String is printed at the position (x,y).

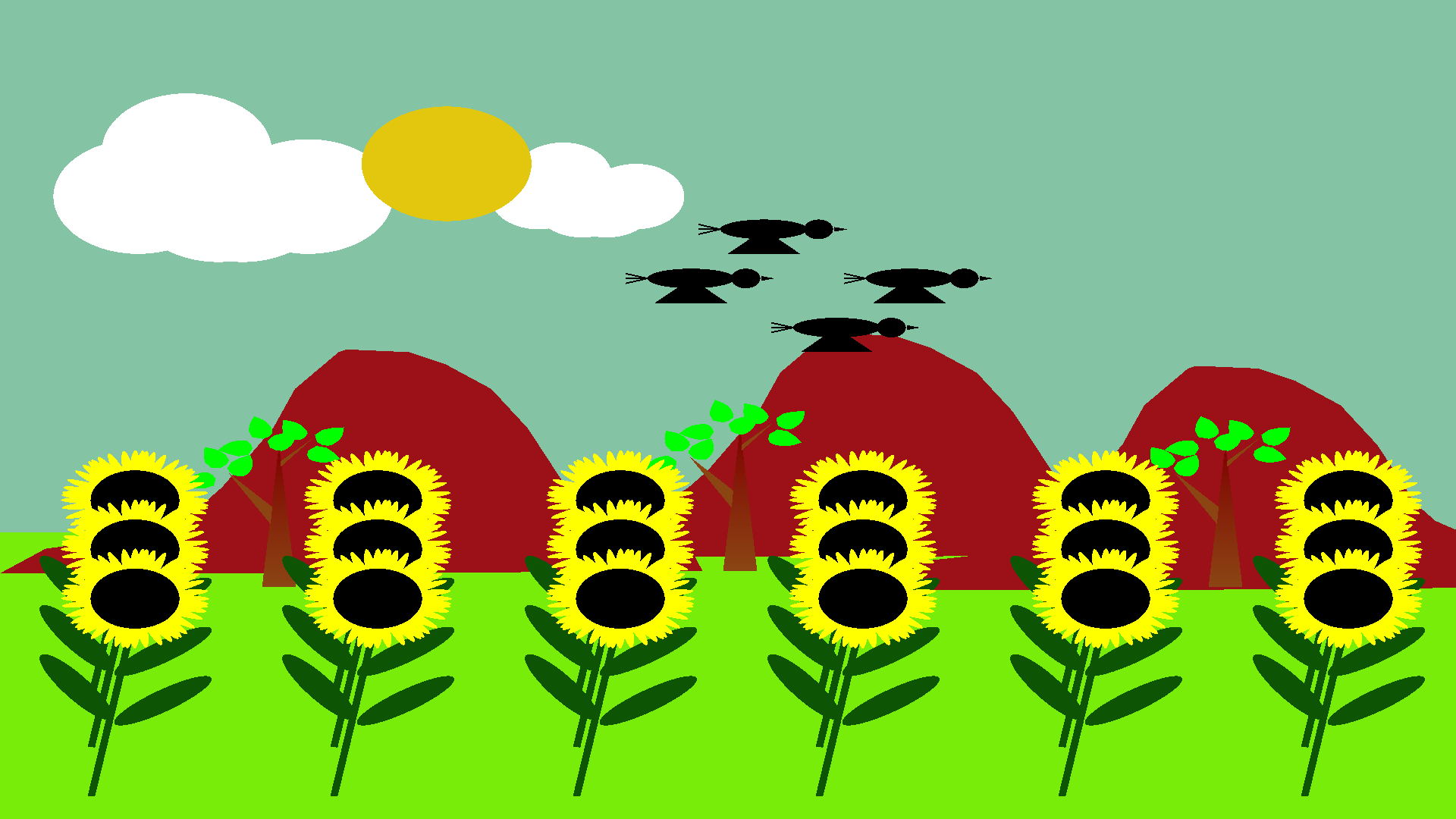
**11.void mountain():**

This function draws the mountain. We used GL\_POLYGON to draw the mountain.

CHAPTER 5

SNAPSHOTS

Fig. 5.1: Ortho view



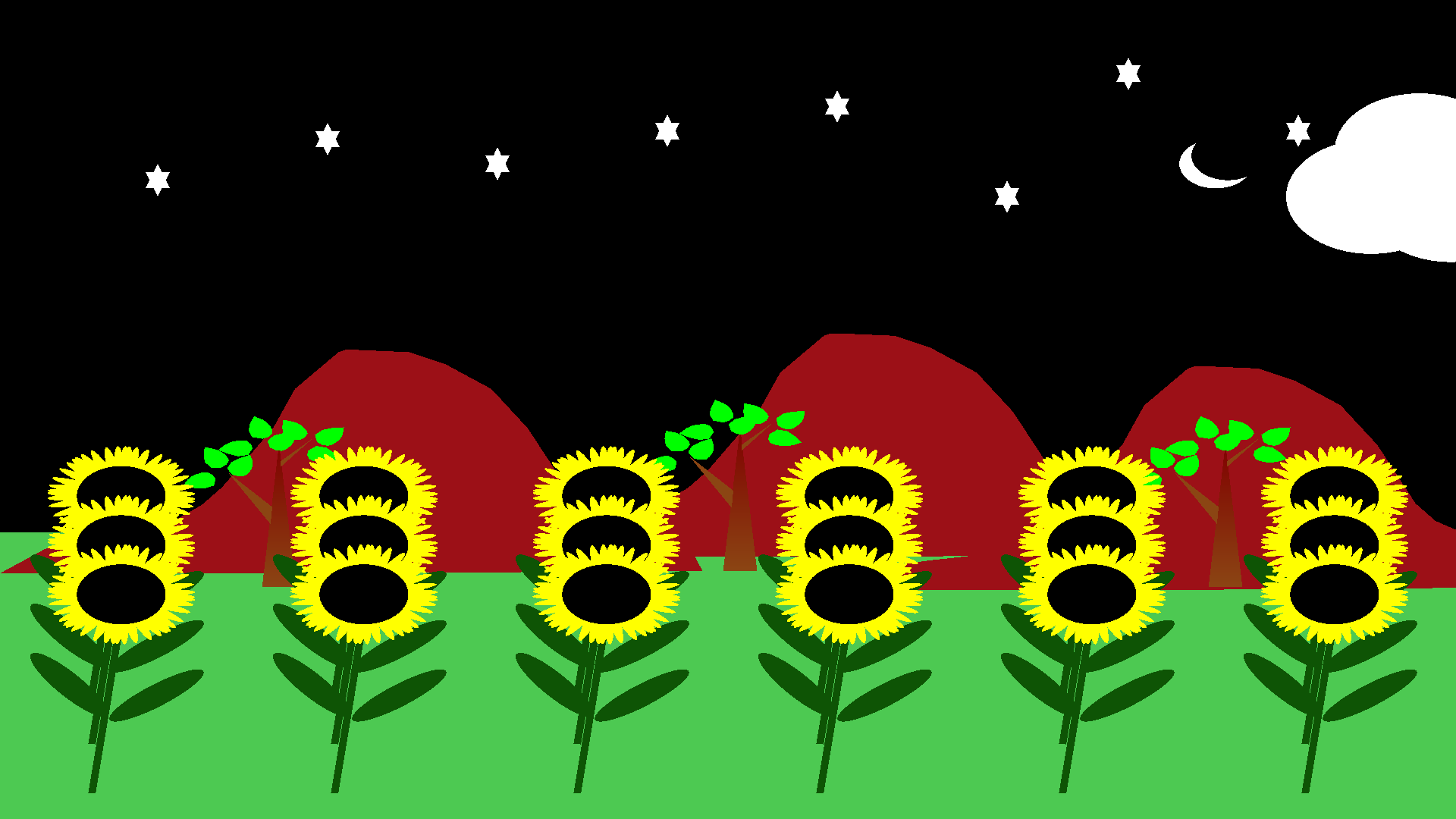
Explanation of the figure

At dawn, the sun begins to rise from east and make it’s way all over to the west while whole fields of sunflowers stand at attention, all facing east, and start to follow the rising sun. As that special star appears to move across the sky, the sunflowers follow its light, looking up, then over to westward, catching one final glance as the sun disappears over the horizon.

As the sun flowers get old, they await visits from insects, most likely honeybees that will

spread their pollen and make new sunflowers. Those flowers too, will follow the sun.

Fig. 5.2: Perspective view



Explanation of the figure

At night, in sun’s absence, the sunflowers face east again, anticipating the sun’s return.

They do this until they get old. As they get old they begin to grow seeds, which are then

utilized for various purposes.

the stems of sunflowers grow more at night — but only on their west side, which is what allows their heads to bend eastward. During the day, the stems’ east side grows, and they bend west with the sun.

**CHAPTER 6**

FUTURE ENHANCEMENT

We have implemented the essential features of the SunFlower: Interaction with the sun to our best knowledge. Even still, we would like to enhance the quality and appearance of the clock in the following ways:

* 3D View
* Enhanced appearance
* Lighting and shading
* Implementation of various objects.

Perhaps the most challenging task will be to implement the functionality of ‘Layers’ which

are used extensively by Professionals in graphics industry.

In future versions of this project, addition of some more built-in 2D and 3D models of geometric figures like cone, and pyramids, option to change line thickness, pick up color of line and various options to set light properties is a feasible idea.

**CHAPTER 7**

CONCLUSION

Thus, in this project we have acquired a lot of knowledge about various techniques in OpenGL programming. We have explored many new concepts on the World Wide Web, such as Texture mapping, randomizing, color swapping etc.

The project is fun and educational. It describes the interaction of sunflower with the sun.

After the completion of this project we came to know how to work with OpenGL and how we can implement a project using an open source OpenGL tool kit. By implementing a project using OpenGL, we came to know how to use the functions lie lighting, rotation, translation and scaling.

With the completion of this project we have achieved a sense of happiness and we want to thank all those who helped us directly or indirectly to make this idea come true.

REFERENCES

**Books:**

Interactive Computer Graphics, 5th Edition, Edward Angel

Computer Graphics and Multimedia, Udit Sharma

**Websites:**

* <http://www.amazon.in/Computer-Graphics-Multimedia-Udit-Agarwal/dp/935014316X?tag=googinhydr18418-21>
* <http://en.wikipedia.org/wiki/Computer_graphics>
* <http://stackoverflow.com/questionsquickly>
* <http://www.opengl-tutorial.org/intermediate-tutorials/>
* <http://www.opengl-tutorial.org/>
* <https://open.gl/>
* <http://www.cs.uccs.edu/~ssemwal/indexGLTutorial.html>
* <http://www.videotutorialsrock.com/>
* <http://ogldev.atspace.co.uk/>
* <https://www.opengl.org/sdk/docs/tutorials/>
* <http://learnopengl.com/>
* <http://lazyfoo.net/tutorials/OpenGL/>
* http://en.wikibooks.org/wiki/OpenGL\_Programming

APPENDIX-A

**A.1 Source code**

#include<windows.h>

#include<iostream>

#include<GL/glut.h>

#include<math.h>

float x=.70, y=.40,z=.09;

int i=0,j=0;

float x\_pos=0;

float c1=0,f1=0;

int height=500;

int width=600;

float sx=227, sy=106, sz=14;

int state=1,state1=0,c11=0;

float fx=0,fy=0;

int speed=4;

int stars[][2]={{10,-15},{80,10},{150,-5},{220,15},{290,30},{360,-25},{410,50}, {480,15}};

int bi=0;

int stateb=0;

int bird1[][2]={{10,300},{40,330},{70,270},{100,300}};

int mountposition[][2]={ {0,150},{200,160},{350,140} };

int xb=650;

int beeposition[][2]={{0,0},{60,60},{90,90},{0,90}};

float x2 = 142, y2 = 237, z2 = 9;

int day=0,night=0;

float yt=0;

int days=0;

int angle1=0;

int pd=1;

int petaldropposition[][4]={{90,180,250,0},{45,160,150,0}};

GLdouble mountainpoints[][2]={ {1,0},{0,0},{0.125,0.1},{0.225,0.12},{0.35,0.14}, {0.45,0.19},{0.556,0.28},{.610,.35},{.750,.59},

{.81,.75},{.89,.85},{.93,.90},{.95,.91},{.97,.91},{1.125,0.90},{1.225,.85},{1.35,.75},{1.45,.59},{1.556,.35},{1.610,.28},

{1.750,.19},{1.81,.14},{1.89,.12},{1.93,.01},{2,0} };

int cloudposition[][3]={{0,0,0},{-400,0,0},{-800,0,0}};

void drawCircleFilled (double centerX, double centerY, double radiusX, double radiusY)

{

const float DEG2RAD = 3.14159 / 180;

glBegin(GL\_TRIANGLE\_FAN);

glVertex2f (centerX, centerY);

for (int i = -1; i < 360; i++) {

float degInRad = i \* DEG2RAD;

glVertex2f(centerX + cos (degInRad) \* radiusX, centerY + sin (degInRad) \* radiusY);

}

glEnd();

}

void wing()

{

glBegin(GL\_POLYGON);

glVertex3f(1.0, 0.0, 0.0);

glVertex3f(1.167, 0.083, 0.0);

glVertex3f(1.225, 0.208, 0.0);

glVertex3f(1.25, 0.42, 0.0);

glVertex3f(1.23, 0.583, 0.0);

glVertex3f(1.167, 0.75, 0.0);

glVertex3f(1.0, 1.0, 0.0);

glVertex3d(.75,.75,0);

glVertex3d(.583,.5,0);

glVertex3d(.56,.2,0);

glVertex3d(.8,0,0);

glEnd();

}

void leg()

{

glBegin(GL\_LINE\_STRIP);

glVertex3d(255,225,-2);

glVertex3d(245,195,-2);

glVertex3d(245,160,-2);

glEnd();

}

void bee()

{

glScaled(.3,.3,.3);

glTranslated(xb,0,0);

glRotated(180,0,1,0);

//lower body

glPushMatrix();

glColor3f(224/255.0, 232/255.0, 65/255.0);

glTranslated(200,200,0);

glRotated(30,0,0,1);

drawCircleFilled(0,0,50,20);

glColor3f(0/255.0, 0/255.0, 0/255.0);

drawCircleFilled(8,0,40,18);

glColor3f(224/255.0, 232/255.0, 65/255.0);

drawCircleFilled(20,0,30,15);

glColor3f(0/255.0, 0/255.0, 0/255.0);

drawCircleFilled(30,0,20,12);

glColor3f(224/255.0, 232/255.0, 65/255.0);

drawCircleFilled(40,0,10,10);

glPopMatrix();

//upper body

glPushMatrix();

glColor3f(217/255.0, 196/255.0, 63/255.0);

glTranslated(265,230,0);

glRotated(15,0,0,1);

drawCircleFilled(0,0,25,15);

glPopMatrix();

glColor3d(0,0,0);

//nose

glBegin(GL\_TRIANGLES);

glVertex2d(275,230);

glVertex2d(280,235);

glVertex2d(300,220);

glEnd();

glLineWidth(3);

glBegin(GL\_LINE\_STRIP);

glVertex2d(280,235);

glVertex2d(290,255);

glVertex2d(310,265);

glEnd();

glBegin(GL\_LINE\_STRIP);

glVertex2d(285,235);

glVertex2d(300,245);

glVertex2d(310,245);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex2d(160,185);

glVertex2d(170,185);

glVertex2d(158,155);

glEnd();

glPushMatrix();

leg();

glPopMatrix();

glPushMatrix();

glScaled(.6,.6,1);

glTranslated(180,140,0);

leg();

glPopMatrix();

glPushMatrix();

glScaled(.3,.3,1);

glTranslated(650,520,0);

leg();

glPopMatrix();

if(stateb==0)

{

glPushMatrix();

glColor3f(239/255.0, 245/255.0, 157/255.0);

glTranslated(300,300,0);

glScaled(50,70,0);

glRotated(180,0,0,1);

wing();

glPopMatrix();

}

else

{

glPushMatrix();

glColor3f(239/255.0, 245/255.0, 157/255.0);

glTranslated(205,160,0);

glScaled(50,70,0);

glRotated(0,0,0,1);

wing();

glPopMatrix();

}

}

void drawCloud(int g1,int h1)

{

glTranslated(c1+c11/2,0,0);

glColor3f(1,1,1);

drawCircleFilled(g1, h1, 35, 35);

glColor3f(255.0, 255.0, 255.0);

drawCircleFilled(g1 + 35, h1 , 35, 35);

glColor3f(255.0, 255.0, 255.0);

drawCircleFilled(g1 + 8, h1 - 5, 35, 35);

glColor3f(255.0, 255.0, 255.0);

drawCircleFilled(g1 - 2 , h1 - 5, 35, 35);

glColor3f(255.0, 255.0, 255.0);

drawCircleFilled(g1 - 35, h1 , 35, 35);

glColor3f(255.0, 255.0, 255.0);

drawCircleFilled(g1 - 15, h1 + 28, 35, 35);

float m = g1 + 150;

glColor3f(1,1,1);

drawCircleFilled(m, h1, 20, 20);

glColor3f(255.0, 255.0, 255.0);

drawCircleFilled(m + 20, h1 , 20, 20);

glColor3f(255.0, 255.0, 255.0);

drawCircleFilled(m + 8, h1 - 5, 20, 20);

glColor3f(255.0, 255.0, 255.0);

drawCircleFilled(m - 2 , h1 - 5, 20, 20);

glColor3f(255.0, 255.0, 255.0);

drawCircleFilled(m - 20, h1 , 20, 20);

glColor3f(255.0, 255.0, 255.0);

drawCircleFilled(m - 10, h1 + 13, 20, 20);

}

void background()

{

glColor3f(x,y,z);

glBegin(GL\_QUADS);

glVertex2d(0, height\*35/100);

glVertex2d(width, height\*35/100);

glVertex2d(width, height);

glVertex2d(0, height);

glEnd();

glColor3f(x2/255.0,y2/255.0,z2/255.0);

glBegin(GL\_QUADS);

glVertex2d(0, 0);

glVertex2d(width, 0);

glVertex2d(width, height\*35/100);

glVertex2d(0, height\*35/100);

glEnd();

}

void flower()

{

glScaled(fx,fy,0);

glColor3f(14/255.0, 84/255.0, 5/255.0);

//stem

glBegin(GL\_POLYGON);

glVertex3d(60,0,0);

glVertex3d(60,200,0);

glVertex3d(65,200,0);

glVertex3d(65,0,0);

glEnd();

//leaf

glPushMatrix();

glTranslated(95,95,0);

glRotated(45,0,0,1);

glTranslated(-95,-95,0);

drawCircleFilled(100,100,40,10);

glPopMatrix();

glPushMatrix();

glTranslated(35,85,0);

glRotated(-45,0,0,1);

glTranslated(-35,-85,0);

drawCircleFilled(20,100,40,10);

glPopMatrix();

//petals

glColor3f(1,1,0);

for(int angle=0;angle<360;angle=angle+3+angle1)

{

glTranslated(60,200,0);

glRotated(angle,0,0,1);

glTranslated(-60,-200,0);

drawCircleFilled(60,200,50,5);

}

//center

glColor3f(0,0,0);

drawCircleFilled(60,200,30,30);

}

void moon(){

glColor3f(1,1,1);

glTranslated(c1-f1,0,0);

drawCircleFilled(c1-f1-35,400,15,15);

glColor3f(x,y,z);

drawCircleFilled(c1-f1-30,405,15,15);

//clouds

glPushMatrix();

drawCloud(-200,380);

glPopMatrix();

}

void star(){

glBegin(GL\_TRIANGLES);

glVertex2d(50,400);

glVertex2d(60,400);

glVertex2d(55,415);

glEnd();

glBegin(GL\_TRIANGLES);

glVertex2d(50,410);

glVertex2d(60,410);

glVertex2d(55,395);

glEnd();

}

//bird

void bird()

{

glScaled(.3,.3,0);

glTranslated(bi,0,0);

glPushMatrix();

//body color

glColor3f(0,0,0);

//body

drawCircleFilled(100,100,60,20);

//face

drawCircleFilled(175,100,20,20);

//mouth

glBegin(GL\_TRIANGLES);

glVertex2d(196,105);

glVertex2d(196,95);

glVertex2d(215,100);

glEnd();

//tail

int a[][2]={{10,110},{10,100},{10,90}};

for (int i=0;i<3;i++)

{

glLineWidth(2);

glBegin(GL\_LINES);

glVertex2d(40,100);

glVertex2d(a[i][0],a[i][1]);

glEnd();

}

//up wing

if (stateb==0){

glBegin(GL\_TRIANGLES);

glVertex2d(100,100);

glVertex2d(150,150);

glVertex2d(50,150);

glEnd();

}

//down wing

else{

glBegin(GL\_TRIANGLES);

glVertex2d(100,100);

glVertex2d(150,50);

glVertex2d(50,50);

glEnd();

}

//eye

//eye color

glColor3b(1,0,0);

glPointSize(5);

glBegin(GL\_POINTS);

glVertex2d(180,105);

glEnd();

glPopMatrix();

}

//mountain

void mountain()

{

glScaled(150,150,0);

glColor3d(156/255.0, 16/255.0, 23/255.0);

glBegin(GL\_POLYGON);

for(int i=0;i<24;i++)

{

glVertex2dv(mountainpoints[i]);

}

glEnd();

}

//tree

void Strees()

{ //body

glColor3f(0.545, 0.271, 0.075);

glBegin(GL\_POLYGON);

glVertex2f(140,45);

glColor3f(0.502, 0.0, 0.0);

glVertex2f(150,170);

glColor3f(0.545, 0.271, 0.075);

glVertex2f(160,45);

glEnd();

//branches 1

glColor3f(0.545, 0.271, 0.075);

glBegin(GL\_POLYGON);

glVertex2f(145,100);

glVertex2f(146,115);

glVertex2f(120,145);

glVertex2f(145,100);

glEnd();

glBegin(GL\_POLYGON);

glVertex2f(151,155);

glVertex2f(151,150);

glVertex2f(170,175);

glVertex2f(153,155);

glEnd();

}

void leaf()

{

glPushMatrix();

glScaled(20,20,0);

glColor3d(0,1,0);

glBegin(GL\_POLYGON);

glVertex3f(1.0, 0.0, 0.0);

glVertex3f(1.167, 0.083, 0.0);

glVertex3f(1.225, 0.208, 0.0);

glVertex3f(1.25, 0.42, 0.0);

glVertex3f(1.23, 0.583, 0.0);

glVertex3f(1.167, 0.75, 0.0);

glVertex3f(1.0, 1.0, 0.0);

glVertex2d(.75,.75);

glVertex2d(.583,.5);

glVertex2d(.56,.2);

glVertex2d(.8,0);

glEnd();

glPopMatrix();

}

void leaves()

{

glPushMatrix();

glTranslated(160,182,0);

glRotated(-40,0,0,1);

leaf();

glPopMatrix();

glPushMatrix();

glTranslated(150,163,0);

glRotated(40,0,0,1);

leaf();

glPopMatrix();

glPushMatrix();

glTranslated(170,180,0);

glRotated(-100,0,0,1);

leaf();

glPopMatrix();

}

void tree()

{

glColor3d(1,0,0);

glScaled(.7,.7+yt,0);

Strees();

glPushMatrix();

leaves();

glPopMatrix();

glPushMatrix();

glRotated(80,0,0,1);

glTranslated(-5,-260,0);

leaves();

glPopMatrix();

glPushMatrix();

glRotated(65,0,0,1);

glTranslated(45,-225,0);

leaves();

glPopMatrix();

}

void PrintText(float x, float y, void\* font,int col, char \*string)

{

int len;

glColor3d(0,1,0);

glPointSize(5.0);

glRasterPos2f(x, y);

len = (int)strlen(string);

for (int i = 0; i < len; i++)

{

glutBitmapCharacter(font, string[i]);

}

}

void home()

{

glClearColor(0.0,0.0,0,1);

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

PrintText(120,410,GLUT\_BITMAP\_TIMES\_ROMAN\_24,38,"DSATM");

PrintText(120,390,GLUT\_BITMAP\_TIMES\_ROMAN\_24,18,"DEPARTMENT : CSE");

PrintText(120,340,GLUT\_BITMAP\_TIMES\_ROMAN\_24,18,"\"SUNFLOWER :: INTERACTION WITH SUN\"");

PrintText(235,270,GLUT\_BITMAP\_TIMES\_ROMAN\_24,18,"SUBMITTED BY :");

PrintText(120,250,GLUT\_BITMAP\_TIMES\_ROMAN\_24,18,"SANDEEP KAVALUR");

PrintText(120,230,GLUT\_BITMAP\_TIMES\_ROMAN\_24,18,"1DT17CS081");

PrintText(320,250,GLUT\_BITMAP\_TIMES\_ROMAN\_24,18,"SANGAMESH V");

PrintText(320,230,GLUT\_BITMAP\_TIMES\_ROMAN\_24,18,"1DT17CS082");

glutPostRedisplay();

}

int h=0;

int zz=0;

int ip=0;

void petaldrop(int degree,int xp,int yp , int zp)

{

glColor3f(1,1,0);

glPushMatrix();

glScaled(.3,.3,0);

glTranslated(xp,yp,zp);

glRotated(degree,0,0,1);

glTranslated(-xp,-yp,zp);

drawCircleFilled(xp,yp,50,5);

glPopMatrix();

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();

if(h==0){

home();

}

else{

background();

//mountain

for (int i=0;i<3;i++)

{

glPushMatrix();

glTranslated(mountposition[i][0],mountposition[i][1],0);

mountain();

glPopMatrix();

}

//tree

glPushMatrix();

glTranslated(10,110,0);

tree();

glPopMatrix();

glPushMatrix();

glTranslated(200,120,0);

tree();

glPopMatrix();

glPushMatrix();

glTranslated(400,110,0);

tree();

glPopMatrix();

//flower position

for (int i=0;i<18;i++){

int i1=i%6;

if (i<6){

j=8;

}

else if (i>=6 && i<12){

j=5;

}

else{

j=2;

}

glPushMatrix();

glTranslated(100\*i1,10\*j,0);

glRotated(-f1/10,0,0,1);

flower();

j=0;

glPopMatrix();

}

//moon

glPushMatrix();

moon();

glPopMatrix();

//sun

glPushMatrix();

glColor3f(sx/255.0,sy/255.0,sz/255.0);

glTranslated(c1,0,0);

drawCircleFilled(c1,400,35,35);

glPopMatrix();

//bird

for(int i=0;i<4;i++)

{

glPushMatrix();

glTranslated(bird1[i][0],bird1[i][1],0);

bird();

glPopMatrix();

}

//stars position

if (state1==1){

for (int i=0;i<8;i++){

glPushMatrix();

glColor3f(1,1,1);

glTranslated(stars[i][0],stars[i][1],0);

star();

glPopMatrix();

}

}

if(zz==1)

{

//bee position

for(int i=0;i<4;i++)

{

glPushMatrix();

glTranslated(beeposition[i][0]+500,beeposition[i][1],0);

bee();

glPopMatrix();

}

}

//cloud movement and sun movement

if (c1<330 && state1==0){

c1+=speed;

f1=c1;

c11+=(30+speed);

//flower size

if(fx<=.6)

{

fx+=.01;

fy+=.01;

}

if(pd==1 && days%1==0 && zz==1)

{

petaldrop(petaldropposition[ip][0],petaldropposition[ip][1],petaldropposition[ip][2],petaldropposition[ip][3]);

petaldrop(petaldropposition[ip][0],petaldropposition[ip][1]+300,petaldropposition[ip][2],petaldropposition[ip][3]);

petaldrop(petaldropposition[ip][0],petaldropposition[ip][1]+600,petaldropposition[ip][2],petaldropposition[ip][3]);

petaldrop(petaldropposition[ip][0],petaldropposition[ip][1]+900,petaldropposition[ip][2],petaldropposition[ip][3]);

petaldrop(petaldropposition[ip][0],petaldropposition[ip][1]+1200,petaldropposition[ip][2],petaldropposition[ip][3]);

petaldrop(petaldropposition[ip][0],petaldropposition[ip][1]+1500,petaldropposition[ip][2],petaldropposition[ip][3]);

petaldrop(petaldropposition[ip][0],petaldropposition[ip][1]+1800,petaldropposition[ip][2],petaldropposition[ip][3]);

petaldrop(petaldropposition[ip][0],petaldropposition[ip][1]+500,petaldropposition[ip][2],petaldropposition[ip][3]);

petaldrop(petaldropposition[ip][0],petaldropposition[ip][1]+700,petaldropposition[ip][2],petaldropposition[ip][3]);

petaldrop(petaldropposition[ip][0],petaldropposition[ip][1]+800,petaldropposition[ip][2],petaldropposition[ip][3]);

petaldrop(petaldropposition[ip][0],petaldropposition[ip][1]+1000,petaldropposition[ip][2],petaldropposition[ip][3]);

petaldrop(petaldropposition[ip][0],petaldropposition[ip][1]+1100,petaldropposition[ip][2],petaldropposition[ip][3]);

petaldrop(petaldropposition[ip][0],petaldropposition[ip][1]+1300,petaldropposition[ip][2],petaldropposition[ip][3]);

ip+=1;

}

}

else{

state1=1;

f1-=speed;

if (f1<=0){

state1=0;

c1=0;

c11=0;

zz=1;

yt+=.01;

//reducing the petals

days+=1;

if(days%1==0)

{

angle1+=2;

pd=1;

}

}

}

//sky color

if (c1<120){

x-=.01\*speed/5;

y+=2\*.01\*speed/5;

z+=3\*.01\*speed/5;

}

else{

x-=.01\*speed/7;

y-=2\*.01\*speed/7;

z-=3\*.01\*speed/7;

}

if (c1<=0){

x=.70;

y=.40;

z=.09;

bi=0;

}

//ground

if (c1<120){

x2-=1;

}

else{

x2-=.3;

y2-=0.3;

z2+=.6;

}

if (c1==0){

x2=142;

y2=237;

z2=9;

}

//sun color

if (sy<224 && state==1)

sy+=2+speed;

else{

state=0;

sy-=1-speed/10;

if (sy<106){

sy=106;

state=1;

}

}

//bird speed

bi+=speed+30;

if (stateb==0){

stateb=1;

}

else{

stateb=0;

}

//bee speed

xb-=10\*speed;

if(xb<=-1800&& state1==0){

xb=650;

}

}

glFlush();

}

void reshape(int w,int h){

glViewport(0,0,w,h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,600,0,500);

glMatrixMode(GL\_MODELVIEW);

}

void timer(int){

glutPostRedisplay();

glutTimerFunc(1000/60,timer,0);

Sleep(100);

}

void menu(int val)

{

switch(val)

{

case 1:

day=1;

h=1;

state1=0;

c1=0;

f1=0;

c11=0;

x=.70, y=.40,z=.09;

xb=650;

break;

case 2:

night=1;

h=1;

state1=1;

c1=330;

f1=c1;

x=0, y=0,z=0;

xb=-1800;

break;

case 3:

exit(0);

break;

}

}

void special(int key,int x,int y)

{

switch(key)

{

case GLUT\_KEY\_LEFT:

speed-=1;

c11-=3;

glutPostRedisplay();

break;

case GLUT\_KEY\_RIGHT:

speed+=1;

c11+=3;

glutPostRedisplay();

break;

case GLUT\_KEY\_END:

exit(0);

break;

}

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_RGB |GLUT\_SINGLE);

glutInitWindowSize(width,height);

glutInitWindowPosition(30,30);

glutCreateWindow("SUNFLOWER");

int val=glutCreateMenu(menu);

glutAddMenuEntry("DAY",1);

glutAddMenuEntry("NIGHT",2);

glutAddMenuEntry("EXIT",3);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutTimerFunc(0,timer,0);

glutSpecialFunc(special);

glutFullScreen();

glutMainLoop();

}

**A.2User Manual**

Right click on the visualizer window, you will be asked for three options

1. Day time
2. Night Time
3. Quit

Depending on the selection the corresponding output will be displayed.

You can even speed up or slow down the speed of the time.

**Make sure that the folder ‘SunFlower: Interaction with the sun’ is copied to the C:\ drive ONLY**

**A.3Personal Details**

NAME: SANDEEP KAVALUR

USN: 1DT17CS081

SEMESTER AND SECTION: 6TH SEM, B SEC

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

EMAIL ID: sandeepskavalursk@gmail.com

NAME: SANGAMESH V

USN: 1DT17CS082

SEMESTER AND SECTION: 6TH SEM, B SEC

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

EMAIL ID: sangameshv05@gmail.com