JSPM'S
JAYAWANTRAO SAWANT COLLEGE OF ENGINEERING

Department of Computer Engineering
Second year Engineering (A.Y. 2020 – 2021)
Savitribai Phule Pune University

210248: OOP and Computer Graphics Laboratory

PART 2: Computer Graphics Laboratory

COMPUTER GRAPHICS MINI PROJECT

To depict Sunrise and Sunset with motion control in OpenGL

Program code with Output

JOEL SILAS
JSPM'S JSCOE, Hadapsar, Pune-28.
Comp. Engg. Dept.
SE Division B (A.Y. 2020-2021)
Roll no: 2259

PROGRAM CODE

```
// header files or C++ standard libraries ie. preprocessor derectives

#include<iostream>       //for standard input/output in C++
#include<stdlib.h>       //for exit(0) function

//if Mac is used
#ifdef __APPLE__
#include<openGL/openGL.h>
#include<GLUT/glut.h>

//if Windows is used         //Similarly, different for linux,etc...
#else
#include<GL/glut.h>
#endif

using namespace std;       // to use the namespace defined in C++ libraries and not any other or user defined namespace, to avoid any ambiguity

// Initial position or coordinates of the SUN when it is at bottom left side of display screen
float ballX = -0.8f;    //x coordinate of sun
float ballY = -0.3f;    //y coordinate of sun
float ballZ = -1.2f;    //z coordinate of sun

// SUN colour in RGB  colour model
float colR=3.0;       // use appropriate quantities of RGB for the initial sun colour float colG=1.5;
float colB=1.0;

// Initial background colour in RGB is black colour when SUN is at bottom left side of display screen
float bgColR=0.0;     //0.0 indicates black colour  &&   1.0 indicates white colour
float bgColG=0.0;
float bgColB=0.0;

static int flag=1;  /* only one variable of the name flag is available for the entire scope of this program which is explicitly (i.e. by user) initialised to 1 else garbage value or 0 is assigned to flag by compiler implicitly...depends on the compiler used
*/

//TO DRAW SUN / BALL
void drawBall(void)   // user defined function
{
      glColor3f(colR,colG,colB); //set ball colour
```

```
        glTranslatef(ballX,ballY,ballZ); /*motion or translation ie. change in position of the ball along the screen from L.H.S to R.H.S of screen */
        glutSolidSphere(0.05, 30, 30);  //create ball or sun    //explained by ppt
}

// TO DRAW MOUNTAINS
void drawAv(void)
{
   glBegin(GL_POLYGON);

        glColor3f(1.0,1.0,1.0);

        glVertex3f(-0.9,-0.7,-1.0);       //explained by ppt

        glVertex3f(-0.5,-0.1,-1.0);

        glVertex3f(-0.2,-1.0,-1.0);

        glVertex3f(0.5,0.0,-1.0);

        glVertex3f(0.6,-0.2,-1.0);

        glVertex3f(0.9,-0.7,-1.0);

   glEnd();
}


void keyPress(int key, int x, int y)  // to control horizontal and vertical motion of the sun
{
     if(key==GLUT_KEY_RIGHT)
       ballX += 0.05f;       //increment x to move towards the right
       if(key==GLUT_KEY_LEFT)
       ballX  -= 0.05f;      //decrement x to move towards the left
      if(key==GLUT_KEY_UP)
       ballY += 0.05f;       //increment y to move upwards
     if(key==GLUT_KEY_DOWN)
       ballY -= 0.05f;       //decrement y to move downwards

   glutPostRedisplay();
}

void handleKeypress(unsigned char k, int x, int y)     // for exit(0) from program // k is a non-negative(unsigned) character
{
if(k==27)
exit(0);
```

```
}

void initRendering()   // Removal of hidden surfaces and shading algorithms
{
   glEnable(GL_DEPTH_TEST);      //allows SUN to move behind the mountains and not in the
front of them, using depth / Z Buffer algo
   glEnable(GL_COLOR_MATERIAL);  //to shade the sun as per its position on screen

//Enable lighting
glEnable(GL_LIGHTING);
//following types of light will give a combination of white and black i.e. a gray shade to the
objects visible in front view
 glEnable(GL_LIGHT0); //Enable white light #0
glEnable(GL_LIGHT1); //Enable black light #1

//Optional shading effect
   glEnable(GL_NORMALIZE);  //Automatically normalize normals... for phong shading
   glShadeModel(GL_SMOOTH); //Enable smooth shading (like analog signal)
}


/* Instructs OpenGL to convert from x,y,z real world window coordinates to pixel values of
viewport  since pixels and coordinates are different phenomena  to adjust the scene/image
position as per size of window
*/
void handleResize(int w, int h)   /* w=current width, h=current height i.e. current size of window
                                when the output console opens*/
{
glViewport(0,0,w,h);   //viewport is a quadrilateral (polygon having 4 vertices)
//bottom left vertex is (0,0) and upper right vertex is (w,h)

   glMatrixMode(GL_PROJECTION);     // for setting the camera perspective i.e front view
                                // helps resolve the overlapping images of objects
                                //sets a matrix of polygon vertices
   //Set the camera perspective
   glLoadIdentity();       //Reset the camera ie. reset the matrix of polygon vertices to identity matrix
                                 //Resetting the matrix generated above since x,y,z coordinates change with
        the    motion of sun
                        //This function takes no arguments, returns no value

   gluPerspective(45.0,                //The camera angle, it decides the viewing direction for
user/viewer

                (double)w / (double)h, //The width-to-height ratio (aspect ratio) of window
                                    //type casting done ,also increases precision from float to double
                1.0,       //The near z (hither) clipping coordinate i.e. starting point for starting 3D
clipping
                200.0);    //The far z (yon) clipping coordinate i.e. end point to stop 3D clipping
```

```
                    //clipping the depths(z values) of objects outside the window
                    //  .....studied in CG: Unit 03: 3D clipping }

// TO ADD COLOUR / SHADING TO THE OBJECTS OF THE SCENE
void drawScene()
{
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);   //for clear screen,
similar to cleardevice();
glClearColor(bgColR,bgColG,bgColB,0.0);              // clears the colour buffers
glMatrixMode(GL_MODELVIEW);                          // future matrix manipulations should affect
                                                      the modelview matrix

   glLoadIdentity();      //reset matrix to identity due to redisplaying the scene repeatedly


  //1.0 purest maximum form of the colour, 0.0 is absence of colour

   //Add ambient / scattered light
   GLfloat ambientColor[] = {0.2f, 0.2f, 0.2f, 1.0f}; //{ red , green , blue , transperancy/thickness of
colour
}
   glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientColor);    // selects ambient light for
the object

// directed light ie. as per direction of light, shadow of the oject will be formed
   //Add directed light 1 ie. light source 1 is at a particular position
   GLfloat lightColor0[] = {0.5f, 0.5f, 0.5f, 1.0f};
GLfloat lightPos0[] = {4.0f, 0.0f, 8.0f, 1.0f};
glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor0);
glLightfv(GL_LIGHT0, GL_POSITION, lightPos0);

   //Add directed light 2 ie.light source 2 is at a particular position
   GLfloat lightColor1[] = {0.5f, 0.2f, 0.2f, 1.0f}; //Color (0.5, 0.2, 0.2)
   //Coming from the direction (-1, 0.5, 0.5)
   GLfloat lightPos1[] = {-1.0f, 0.5f, 0.5f, 0.0f};
glLightfv(GL_LIGHT1, GL_DIFFUSE, lightColor1);
glLightfv(GL_LIGHT1, GL_POSITION, lightPos1);

   //drawing the SUN
   glPushMatrix();    // push matrix of coordinates of circle on the stack
drawBall();    // circle is drawn as per matrix entries
 glPopMatrix();     // pop this matrix

   //drawing the Mountain
glPushMatrix();
drawAv();
   glPopMatrix();
```

```
   glutSwapBuffers();     // implement double buffering
}


void update(int value)
{
   if(ballX>0.9f)  // x coordinate of the sun becomes positive when sun crosses more than half of the horizontal distance
   {
      ballX = -0.8f;       // for proper sunset
      ballY = -0.3f;
      flag=1;
      colR=2.0;      //When sun crosses more than half of the horizontal distance increase red light for sun set time
      colG=1.50;     // accordingly use appropriate quantity of green and blue to get orange coloured sun
      colB=1.0;

      bgColB=0.0;      //initially, sky is blue so start making background (sky) dark for sun set time
            //since 0.0 is black , 1.0 is white
   }

   if(flag)     // initially flag=1 && ballx < 0.9f
   {
   ballX += 0.001f;   //so move the ball to R.H.S
   ballY +=0.0007f;   //also move the ball upwards        .....for sunrise

   colR-=0.001;      // accordingly adjust sun colour       ....this is sunrise
  colB+=0.005;

   bgColB+=0.001;    //adjust background by making sky more blue      .....since sunrise

     if(ballX>0.01)       //if ball crosses half of the screen
           flag=0;
   }

   if (!flag)             // initially !1 == 0 == false i.e. if statement is not executed
// after ball crosses half of the screen flag=0,  !0 == 1 ==true i.e. if statement gets executed
   {
      ballX += 0.001f;    //to the R.H.S. sun sets ,increase x, decrese y ,sun is moving downward and right side
      ballY -=0.0007f;
      colR+=0.001;        //to get orange coloured sun, adjust colours
      colB-=0.01;

      bgColB-=0.001;      //make sky darker by decreasing blue colour
```

```
      if(ballX<-0.3)     // after one complete cycle of sunrise & sunset
            //to make flag=1 again and start sunrise again from L.H.S to set proper colour for rising
sun         flag=1;
   }

   glutPostRedisplay(); /* Tells GLUT that the display has changed ie. state has been updated
                (after each iteration ,due to translation of sun)  so the scene needs to be redrawn
and redisplayed to reflect the new state.
               */

   // Tell GLUT to call update again after 25 milliseconds, this acts as a loop
glutTimerFunc(25, update, 0);   // glutTimerFunc(time in ms, function_name, number of times this
function will be called);

}

// Program Execution starts from main() function
 int main(int argc,char** argv)  //command line arguments necessary to initialise GLUT
{
   glutInit(&argc,argv);    //initialize GLUT needed for OpenGL programs

   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB |
GLUT_DEPTH);       //indicate use of double buffering , RGB colour model, depth / z buffer algo
                                                  // bitwise OR |    // logical OR ||
  // To set/initialize the size of output console window
   glutInitWindowSize(1300,650);    // glutInitWindowSize(int width,int height);

  /* OpenGL output console window is created    the name of that window is passed as a
parameter to this built-in or predefined or system defined or ready made function
  user only needs to be call/invoke this fnction and user does not need to define it
  */
   glutCreateWindow("sun rise and sun set");

   initRendering();     //function call

   glutDisplayFunc(drawScene);   //displays the scene drawn at the output window

   //glutFullScreen();   // To make OpenGL output console window fullscreen, but then exit from the
program is inconvenient

   glutSpecialFunc(keyPress);
   glutReshapeFunc(handleResize);    //real world window / display screen viewport and objects
within them are spaced out
                        //as per current size of output window
glutKeyboardFunc(handleKeypress); // for exit(0) function

   glutTimerFunc(25, update, 0);
```

```
    glutMainLoop();    // acts as an infinte loop ie. sun will keep rising and setting continuously
                       // enables us to use keyboard keys to control motion of objects
                       // helps in resizing the window
return(0);
}
```

OUTPUT SCREENSHOTS