**Lab Practice-7**

Submission Guidelines-

* Rename the file to your id only. If your id is 18-XXXXX-1, then the file name must be 18-XXXXX-1.docx.

|  |
| --- |
| **Question-**  Rain effect |
| **Graph** |
| **Code-**  **#include<windows.h>**  **#include <iostream>**  **#include <GL/glut.h>**  **#include <math.h>**  **//////water**  **void rainyRiver();**  **int waterStatus = 0;**  **float waterX = 0;**  **float waterY = 0;**  **float boatx=0;**  **float boaty=0;**  **//////rain**  **int rainstatus=0;**  **int x=0;**  **int y=0;**  **float wx=0;**  **float wy=0;**  **void DrawCircle(float cx, float cy, float r, int num\_segments) {**  **glBegin(GL\_TRIANGLE\_FAN);**  **for (int i = 0; i < num\_segments; i++)**  **{**  **float theta = 2.0f \* 3.1415926f \* float(i) / float(num\_segments);//get current angle**  **float x = r \* cosf(theta);//calculate x**  **float y = r \* sinf(theta);//calculate y**  **glVertex2f(x + cx, y + cy);//output vertex**  **}**  **glEnd();**  **glFlush();**  **}**  **void river(){**  **glColor3ub (108, 210, 175);**  **glBegin(GL\_POLYGON);**  **glVertex2f(0, 4);**  **glVertex2f(20, 4);**  **glVertex2f(20, 10);**  **glVertex2f(0, 10);**  **glEnd();**  **}**  **void land()**  **{**  **glColor3ub (113, 99, 56);**  **glBegin(GL\_POLYGON);**  **glVertex2f(0, 4);**  **glVertex2f(20, 4);**  **glVertex2f(20, 0);**  **glVertex2f(0, 0);**  **glEnd();**  **}**  **void daywater(){**  **if(rainstatus==0){**  **//glColor3ub (169, 242, 242);**  **glColor4f(1.0f, 1.0f, 1.0f, 0.0f);//white**  **glBegin(GL\_LINES);**  **glVertex2f(0.5,9);**  **glVertex2f(1.5,9);**  **glVertex2f(2,8.5);**  **glVertex2f(1, 8.5);**  **glVertex2f(0.5,8);**  **glVertex2f(1,8);**  **glVertex2f(1,7);**  **glVertex2f(2.5,7);**  **glVertex2f(1.5,6.5);**  **glVertex2f(2,6.5);**  **glVertex2f(0.5,5.5);**  **glVertex2f(2,5.5);**  **glVertex2f(3.5,8);**  **glVertex2f(5,8);**  **glVertex2f(4, 6.5);**  **glVertex2f(5.5, 6.5);**  **glVertex2f(4.5,5.5);**  **glVertex2f(6, 5.5);**  **glVertex2f(6.85,8.5);**  **glVertex2f(7.5,8.5);**  **glVertex2f(6,7.5);**  **glVertex2f(7.5,7.5);**  **glVertex2f(6.5,6);**  **glVertex2f(7.5,6);**  **glVertex2f(9,9);**  **glVertex2f(11,9);**  **glVertex2f(9.5,8);**  **glVertex2f(10.5,8);**  **glVertex2f(9,7);**  **glVertex2f(11.5,7);**  **glVertex2f(9,5);**  **glVertex2f(10.5,5);**  **glVertex2f(12.5,8.5);**  **glVertex2f(14,8.5);**  **glVertex2f(13,6.5);**  **glVertex2f(15,6.5);**  **glVertex2f(12,5);**  **glVertex2f(13.5,5);**  **glVertex2f(15.5,9);**  **glVertex2f(17,9);**  **glVertex2f(16.5,8);**  **glVertex2f(18,8);**  **glVertex2f(18,7);**  **glVertex2f(19.5,7);**  **glVertex2f(16.5,6);**  **glVertex2f(18.5,6);**  **glEnd();**  **}**  **}**  **void moveWater(int x){**  **if (waterStatus ==1)**  **{**  **waterX += 0.01;//left or right side move water**  **}**  **if (waterX>2)**  **{**  **waterX = -5;**  **}**  **glPushMatrix();**  **glTranslatef(waterX, waterY, 0);**  **if(x==1)**  **{**  **rainyRiver();**  **daywater();**  **}**  **else{**  **rainyRiver();**  **}**  **glPopMatrix();**  **glFlush();**  **}**  **void rain()**  **{**  **glColor3d(0,1,0);**  **glBegin(GL\_POINTS);**  **for(int i=1;i<=10000;i++)**  **{**  **x=rand(),y=rand();**  **x%=1000; y%=30;**  **glBegin(GL\_LINES);**  **glColor3b(1,1,1);**  **glVertex2d(x,y);**  **glVertex2d(x+0.3,y+0.3);**  **glEnd();**  **}**  **for(int i=1;i<=10000;i+=1)**  **{**  **x=rand(),y=rand();**  **x%=10000; y%=2000;**  **glBegin(GL\_LINES);**  **glColor3b(1,1,1);**  **glVertex2d(x,y);**  **glVertex2d(x+0.3,y+0.3);**  **glEnd();**  **}**  **}**  **void moverain(int x)**  **{**  **if(rainstatus ==1)**  **{**  **wx -= 0.01;**  **wy -= 0.01;**  **glPushMatrix();**  **glTranslatef(wx, wy, 0);**  **if (x==1)**  **{**  **rain();**  **}**  **glPopMatrix();**  **glFlush();**  **}**  **}**  **void rainyRiver()**  **{**  **if(rainstatus==1)**  **{**  **int xaxis=20;**  **int yaxis=10;**  **glColor3ub (160, 220, 220);**  **glBegin(GL\_LINES);**  **for(int i=0; i<6; i++)**  **{**  **for(int j=0; j<15; j++){**  **glVertex2f( xaxis-0.5, yaxis-0.1);**  **glVertex2f( xaxis-1.5, yaxis);**  **glVertex2f( xaxis, yaxis);**  **glVertex2f( xaxis-0.5, yaxis-0.1);**  **xaxis-=1.5;**  **}**  **yaxis-=0.5;**  **xaxis=20;**  **}**  **glEnd();**  **}**  **}**  **void myDisplay(void){**  **river();**  **land();**  **moveWater(1);**  **moverain(1);**  **glFlush();**  **glutPostRedisplay();**  **glutSwapBuffers();**  **}**  **void mykeyboard(unsigned char key, int x, int y){**  **if (key == 'C' || key == 'c'){ ////water strting**  **waterStatus = 1;**  **}**  **if (key == 'D' || key == 'd'){ //water stop**  **waterStatus = 0;**  **}**  **else if (key == '1' ){ //rain starting**  **rainstatus = 1;**  **}**  **else if (key == '2' ){ //rain stop**  **rainstatus = 0;**  **}**  **}**  **void myInit(void){**  **glClearColor(1.0f,1.0f,1.0f,1.0f);**  **glColor3f(.0f,.0f,.0f);**  **glLoadIdentity();**  **gluOrtho2D(0.0, 20.0, 0.0, 10.0);**  **}**  **int main(int argc, char\*\* argv)**  **{**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutInitWindowSize(1920,1080);**  **glutCreateWindow("Rain Effect");**  **glutPostRedisplay();**  **glutDisplayFunc(myDisplay);**  **glutKeyboardFunc(mykeyboard);**  **myInit();**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
| **Question-**  Create a scenerio where there will be ground, river and it will rain. The rain will ooverflow the  river thay will cause flood. |
| **Graph** |
| **Code-**  **#include<windows.h>**  **#include <iostream>**  **#include <GL/glut.h>**  **#include <math.h>**  **//////water**  **void rainyRiver();**  **int waterStatus = 0;**  **float waterX = 0;**  **float waterY = 0;**  **float boatx=0;**  **float boaty=0;**  **float \_move=0;**  **//////rain**  **int rainstatus=0;**  **int x=0;**  **int y=0;**  **float wx=0;**  **float wy=0;**  **void DrawCircle(float cx, float cy, float r, int num\_segments) {**  **glBegin(GL\_TRIANGLE\_FAN);**  **for (int i = 0; i < num\_segments; i++)**  **{**  **float theta = 2.0f \* 3.1415926f \* float(i) / float(num\_segments);//get current angle**  **float x = r \* cosf(theta);//calculate x**  **float y = r \* sinf(theta);//calculate y**  **glVertex2f(x + cx, y + cy);//output vertex**  **}**  **glEnd();**  **glFlush();**  **}**  **void river()**  **{**  **glMatrixMode(GL\_MODELVIEW);**  **glPushMatrix();**  **glColor3ub (108, 210, 175);**  **glBegin(GL\_POLYGON);**  **glVertex2f(0, 4);**  **glVertex2f(20, 4);**  **glVertex2f(20, 10);**  **glVertex2f(0, 10);**  **glEnd();**  **if(rainstatus==1)**  **{**  **glTranslatef(0,\_move,0);**  **glBegin(GL\_POLYGON);**  **glColor3ub (108, 210, 175);**  **glVertex2f(0, 4);**  **glVertex2f(20, 4);**  **glVertex2f(20, 10);**  **glVertex2f(0, 10);**  **glEnd();**  **}**  **glPopMatrix();**  **//glutSwapBuffers();**  **}**  **void land()**  **{**  **glColor3ub (113, 99, 56);**  **glBegin(GL\_POLYGON);**  **glVertex2f(0, 4);**  **glVertex2f(20, 4);**  **glVertex2f(20, 0);**  **glVertex2f(0, 0);**  **glEnd();**  **}**  **void daywater(){**  **if(rainstatus==0){**  **//glColor3ub (169, 242, 242);**  **glColor4f(1.0f, 1.0f, 1.0f, 0.0f);//white**  **glBegin(GL\_LINES);**  **glVertex2f(0.5,9);**  **glVertex2f(1.5,9);**  **glVertex2f(2,8.5);**  **glVertex2f(1, 8.5);**  **glVertex2f(0.5,8);**  **glVertex2f(1,8);**  **glVertex2f(1,7);**  **glVertex2f(2.5,7);**  **glVertex2f(1.5,6.5);**  **glVertex2f(2,6.5);**  **glVertex2f(0.5,5.5);**  **glVertex2f(2,5.5);**  **glVertex2f(3.5,8);**  **glVertex2f(5,8);**  **glVertex2f(4, 6.5);**  **glVertex2f(5.5, 6.5);**  **glVertex2f(4.5,5.5);**  **glVertex2f(6, 5.5);**  **glVertex2f(6.85,8.5);**  **glVertex2f(7.5,8.5);**  **glVertex2f(6,7.5);**  **glVertex2f(7.5,7.5);**  **glVertex2f(6.5,6);**  **glVertex2f(7.5,6);**  **glVertex2f(9,9);**  **glVertex2f(11,9);**  **glVertex2f(9.5,8);**  **glVertex2f(10.5,8);**  **glVertex2f(9,7);**  **glVertex2f(11.5,7);**  **glVertex2f(9,5);**  **glVertex2f(10.5,5);**  **glVertex2f(12.5,8.5);**  **glVertex2f(14,8.5);**  **glVertex2f(13,6.5);**  **glVertex2f(15,6.5);**  **glVertex2f(12,5);**  **glVertex2f(13.5,5);**  **glVertex2f(15.5,9);**  **glVertex2f(17,9);**  **glVertex2f(16.5,8);**  **glVertex2f(18,8);**  **glVertex2f(18,7);**  **glVertex2f(19.5,7);**  **glVertex2f(16.5,6);**  **glVertex2f(18.5,6);**  **glEnd();**  **}**  **}**  **void moveWater(int x){**  **if (waterStatus ==1)**  **{**  **waterX += 0.01;//left or right side move water**  **}**  **if (waterX>2)**  **{**  **waterX = -5;**  **}**  **glPushMatrix();**  **glTranslatef(waterX, waterY, 0);**  **if(x==1)**  **{**  **rainyRiver();**  **daywater();**  **}**  **else{**  **rainyRiver();**  **}**  **glPopMatrix();**  **glFlush();**  **}**  **void rain()**  **{**  **glColor3d(0,1,0);**  **glBegin(GL\_POINTS);**  **for(int i=1;i<=10000;i++)**  **{**  **x=rand(),y=rand();**  **x%=1000; y%=30;**  **glBegin(GL\_LINES);**  **glColor3b(1,1,1);**  **glVertex2d(x,y);**  **glVertex2d(x+0.3,y+0.3);**  **glEnd();**  **}**  **for(int i=1;i<=10000;i+=1)**  **{**  **x=rand(),y=rand();**  **x%=10000; y%=2000;**  **glBegin(GL\_LINES);**  **glColor3b(1,1,1);**  **glVertex2d(x,y);**  **glVertex2d(x+0.3,y+0.3);**  **glEnd();**  **}**  **}**  **void moverain(int x)**  **{**  **if(rainstatus ==1)**  **{**  **wx -= 0.01;**  **wy -= 0.01;**  **glPushMatrix();**  **glTranslatef(wx, wy, 0);**  **if (x==1)**  **{**  **rain();**  **}**  **glPopMatrix();**  **glFlush();**  **}**  **}**  **void rainyRiver()**  **{**  **if(rainstatus==1)**  **{**  **int xaxis=20;**  **int yaxis=10;**  **glColor3ub (160, 220, 220);**  **glBegin(GL\_LINES);**  **for(int i=0; i<6; i++)**  **{**  **for(int j=0; j<15; j++){**  **glVertex2f( xaxis-0.5, yaxis-0.1);**  **glVertex2f( xaxis-1.5, yaxis);**  **glVertex2f( xaxis, yaxis);**  **glVertex2f( xaxis-0.5, yaxis-0.1);**  **xaxis-=1.5;**  **}**  **yaxis-=0.5;**  **xaxis=20;**  **}**  **glEnd();**  **}**  **}**  **void update(int value)**  **{**  **\_move -= .01;**  **if(\_move > 4)**  **{**  **\_move = -2;**  **}**  **glutPostRedisplay();**  **glutTimerFunc(20, update, 0);**  **}**  **void myDisplay(void)**  **{**  **land();**  **river();**  **moveWater(1);**  **moverain(1);**  **glFlush();**  **glutPostRedisplay();**  **glutSwapBuffers();**  **}**  **void mykeyboard(unsigned char key, int x, int y){**  **if (key == 'C' || key == 'c'){ ////water strting**  **waterStatus = 1;**  **}**  **if (key == 'D' || key == 'd'){ //water stop**  **waterStatus = 0;**  **}**  **else if (key == '1' ){ //rain starting**  **rainstatus = 1;**  **}**  **else if (key == '2' ){ //rain stop**  **rainstatus = 0;**  **}**  **}**  **void myInit(void){**  **glClearColor(1.0f,1.0f,1.0f,1.0f);**  **glColor3f(.0f,.0f,.0f);**  **glLoadIdentity();**  **gluOrtho2D(0.0, 20.0, 0.0, 10.0);**  **}**  **int main(int argc, char\*\* argv)**  **{**  **glutInit(&argc, argv);**  **glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);**  **glutInitWindowSize(1920,1080);**  **glutCreateWindow("Rain & flood Effect ");**  **glutPostRedisplay();**  **glutDisplayFunc(myDisplay);**  **glutKeyboardFunc(mykeyboard);**  **myInit();**  **glutTimerFunc(20, update, 0);**  **glutMainLoop();**  **return 0;**  **}** |
| **Output Screenshot (Full Screen)-** |

|  |
| --- |
|  |
|  |
| #include <windows.h> // for MS Windows  #include <GL/glut.h> // GLUT, include glu.h and gl.h  #include <math.h>  float \_move = 0.0f;  float \_move1=0.0f;  bool isVolcano=false;  void mainBackGround()  {  //mainbackground(sky)  if(!isVolcano)  glColor3ub(169, 169, 169);  else  glColor3ub(20, 137, 139);  glBegin(GL\_POLYGON);  glVertex2f(20,12);  glVertex2f(20,15);  glVertex2f(0,15);  glVertex2f(0,12);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(3.33,10);  glVertex2f(5,12);  glVertex2f(0,12);  glVertex2f(0,10);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(14.88,12);  glVertex2f(5,12);  glVertex2f(6.66,10);  glVertex2f(13.28,10);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(14.88,12);  glVertex2f(16.56,10);  glVertex2f(20,10);  glVertex2f(20,15);  glEnd();  }  void riverbank()  {  //riverbank  glColor3ub(101, 67, 33); // Saddle brown  glBegin(GL\_POLYGON);  glVertex2f(0,6);  glVertex2f(0,5.5);  glVertex2f(20,5.6);  glVertex2f(20,5.8);  glEnd();  }  void river()  {  //river  glColor3ub(173, 216, 230); // Light blue  glBegin(GL\_POLYGON);  glVertex2f(0,5.5);  glVertex2f(20,5.6);  glVertex2f(20,0);  glVertex2f(0,0);  glEnd();  }  void lawn()  {  //lawn  glColor3ub(85, 107, 47); // Olive drab  glBegin(GL\_POLYGON);  glVertex2f(0,6);  glVertex2f(3.33,10);  glVertex2f(0,10);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(6.66,10);  glVertex2f(10,6);  glVertex2f(13.28,10);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(16.56,10);  glVertex2f(20,6);  glVertex2f(20,10);  glEnd();  }  void hill()  {  //hill  glColor3ub(54, 69, 79); // Dark slate gray  glBegin(GL\_POLYGON);  glVertex2f(-4,1.5);  glVertex2f(-3,2.5);  glVertex2f(-2.5,2.5);  glVertex2f(-1.5,1.5);  glVertex2f(0,1);  glVertex2f(-5,1);  glVertex2f(-4,1.5);  glEnd();  }  void volcano\_river()  {  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(0.0f, \_move1, 0.0f);  glColor3ub(255, 69, 0);  glBegin(GL\_POLYGON);  glVertex2f(0,12);  glVertex2f(20,12);  glVertex2f(20,24);  glVertex2f(0,24);  glEnd();  glPopMatrix();  }  void volcano1(float x)  {  //volcano1  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glTranslatef(x, 0, 0);  glColor3ub(255, 69, 0); // Dim gray  if(!isVolcano)  {  glBegin(GL\_POLYGON);  glVertex2f(4.85,12.02);  glVertex2f(4.49,12.16);  glVertex2f(4.81,12.51);  glVertex2f(5.61,12.63);  glVertex2f(5.52,12.07);  glVertex2f(5.77,11.59);  glVertex2f(4.97,11.27);  glVertex2f(4.49,11.6);  glEnd();  }  glPopMatrix();  }  void update(int value)  {  \_move +=0.2;  if(\_move > 2)  {  isVolcano=true;  \_move1 -= 0.2;  if(\_move1<-10)  {  \_move1=0.5;  }  }  glutPostRedisplay();  glutTimerFunc(200, update, 0);  }  void display()  {  glClearColor(0.0f, 0.0f, 0.0f, 0.0f); // Set background color to black and opaque  glClear(GL\_COLOR\_BUFFER\_BIT);  riverbank();  river();  hill();  volcano\_river();  mainBackGround();  lawn();  volcano1(0);  volcano1(9.98);  glFlush(); // Render now  }  /\* Main function: GLUT runs as a console application starting at main() \*/  int main(int argc, char\*\* argv) {  glutInit(&argc, argv); // Initialize GLUT  glutCreateWindow("OpenGL Setup Test");  //gluOrtho2D(-0.1,0.7,-0.1,0.3); // Create a window with the given title  glutInitWindowSize(320, 320);// Set the window's initial width & height  glutDisplayFunc(display);// Register display callback handler for window re-paint  gluOrtho2D(0,20,0,15);  glutTimerFunc(4, update, 0);  glutMainLoop(); // Enter the event-processing loop  return 0;  } |
|  |

|  |
| --- |
| Create a zoom in and out animation using scaling. |
| Graph- |
| Code-  #include <windows.h> // for MS Windows  #include <GL/glut.h> // GLUT, include glu.h and gl.h  #include <math.h>  float \_move=1;  int zoomStatus=2;  void building()  {  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glScalef(\_move,\_move,0);  glColor3ub(142, 53, 206);  glBegin(GL\_POLYGON);  glVertex2f(80,200);  glVertex2f(200,200);  glVertex2f(200,0);  glVertex2f(80,0);  glEnd();  glPopMatrix();  }  void window()  {  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glScalef(\_move,\_move,0);  glColor3ub(202, 218, 218);  glBegin(GL\_POLYGON);  glVertex2f(100,190);  glVertex2f(120,190);  glVertex2f(120,170);  glVertex2f(100,170);  glEnd();  //glScalef(\_move,\_move,0);  glBegin(GL\_POLYGON);  glVertex2f(160,190);  glVertex2f(180,190);  glVertex2f(180,170);  glVertex2f(160,170);  glEnd();  //glScalef(\_move,\_move,0);  glBegin(GL\_POLYGON);  glVertex2f(100,150);  glVertex2f(120,150);  glVertex2f(120,130);  glVertex2f(100,130);  glEnd();  //glScalef(\_move,\_move,0);  glBegin(GL\_POLYGON);  glVertex2f(160,150);  glVertex2f(180,150);  glVertex2f(180,130);  glVertex2f(160,130);  glEnd();  // glScalef(\_move,\_move,0);  glBegin(GL\_POLYGON);  glVertex2f(160,110);  glVertex2f(180,110);  glVertex2f(180,90);  glVertex2f(160,90);  glEnd();  //glScalef(\_move,\_move,0);  glBegin(GL\_POLYGON);  glVertex2f(100,110);  glVertex2f(120,110);  glVertex2f(120,90);  glVertex2f(100,90);  glEnd();  // glScalef(\_move,\_move,0);  glBegin(GL\_POLYGON);  glVertex2f(160,70);  glVertex2f(180,70);  glVertex2f(180,50);  glVertex2f(160,50);  glEnd();  //glScalef(\_move,\_move,0);  glBegin(GL\_POLYGON);  glVertex2f(100,70);  glVertex2f(120,70);  glVertex2f(120,50);  glVertex2f(100,50);  glEnd();  //glScalef(\_move,\_move,0);  glBegin(GL\_POLYGON);  glVertex2f(160,30);  glVertex2f(180,30);  glVertex2f(180,10);  glVertex2f(160,10);  glEnd();  //glScalef(\_move,\_move,0);  glBegin(GL\_POLYGON);  glVertex2f(100,30);  glVertex2f(120,30);  glVertex2f(120,10);  glVertex2f(100,10);  glEnd();  glPopMatrix();  }  void door()  {  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glScalef(\_move,\_move,0);  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<200;i++)  {  glColor3ub(65, 81, 7);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r=10;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x+140,y+30 );  }  glEnd();  //glScalef(\_move,\_move,0);  glColor3ub(202, 218, 218);  glBegin(GL\_POLYGON);  glVertex2f(130,30);  glVertex2f(150,30);  glVertex2f(150,0);  glVertex2f(130,0);  glEnd();  //glScalef(\_move,\_move,0);  glColor3ub(241, 160, 7);  glBegin(GL\_POLYGON);  glVertex2f(130,30);  glVertex2f(138,26);  glVertex2f(138,0);  glVertex2f(130,0);  glEnd();  //glScalef(\_move,\_move,0);  glColor3ub(241, 160, 7);  glBegin(GL\_POLYGON);  glVertex2f(150,30);  glVertex2f(142,26);  glVertex2f(142,0);  glVertex2f(150,0);  glEnd();  glPopMatrix();  }  void border()  {  glMatrixMode(GL\_MODELVIEW);  glPushMatrix();  glScalef(\_move,\_move,0);  glColor3ub(246, 220, 4);  glLineWidth(8);  glBegin(GL\_LINES);  glVertex2f(80,160);  glVertex2f(200,160);  glVertex2f(200,120);  glVertex2f(80,120);  glVertex2f(80,80);  glVertex2f(200,80);  glVertex2f(80,40);  glVertex2f(200,40);  glVertex2f(140,40);  glVertex2f(140,200);  glEnd();  glPopMatrix();  }  void update(int value)  {  if(zoomStatus==1)  {  \_move += .005;  if(\_move > 1.5)  {  \_move=1.5;  }  }  if(zoomStatus==0)  {  \_move -= .005;  if(\_move < 0.5)  {  \_move=0.5;  }  }  glutPostRedisplay();  glutTimerFunc(20, update, 0);  }  void mykeyboard(unsigned char key, int x, int y)  {  if (key == 'Z' || key == 'z'){ //zoom in  zoomStatus = 1;  }  if (key == 'O' || key == 'o'){ //zoom out  zoomStatus = 0;  }  }  void display() {  glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Set background color to black and opaque  glClear(GL\_COLOR\_BUFFER\_BIT);  building();  window();  door();  border();  glFlush(); // Render now  }  /\* Main function: GLUT runs as a console application starting at main() \*/  int main(int argc, char\*\* argv) {  glutInit(&argc, argv); // Initialize GLUT  glutCreateWindow("OpenGL Setup Test");  glutInitWindowSize(320, 320);// Set the window's initial width & height  glutDisplayFunc(display);// Register display callback handler for window re-paint  glutKeyboardFunc(mykeyboard);  gluOrtho2D(0,500,-120,500);  glutTimerFunc(20, update, 0);  glutMainLoop(); // Enter the event-processing loop  return 0;  } |
| Output- |

|  |
| --- |
| Create an animation that will show a story in a texual form based on a still image. Your animation will start with a still image and texts will animated on the screen to describe the image. |
| Graph- |
| Code- #include <windows.h> // for MS Windows  #include <GL/glut.h> // GLUT, includes glu.h and gl.h  #include <string.h>  #include<math.h>  float \_move = 1;  char story[3][50] = {"In this scenario have a building","A bench with a lamppost"," and have a tree"};  int storyIndex = 0;  int charIndex = 0;  char displayText[50] = "";  void building()  {  glColor3ub(142, 53, 206);  glBegin(GL\_POLYGON);  glVertex2f(-50,500);  glVertex2f(200,500);  glVertex2f(200,0);  glVertex2f(-50,0);  glEnd();  //window  glColor3ub(202, 218, 218);  glBegin(GL\_POLYGON);  glVertex2f(-20,480);  glVertex2f(40,480);  glVertex2f(40,440);  glVertex2f(-20,440);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(100,480);  glVertex2f(160,480);  glVertex2f(160,440);  glVertex2f(100,440);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(100,380);  glVertex2f(160,380);  glVertex2f(160,340);  glVertex2f(100,340);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-20,380);  glVertex2f(40,380);  glVertex2f(40,340);  glVertex2f(-20,340);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(100,280);  glVertex2f(160,280);  glVertex2f(160,240);  glVertex2f(100,240);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-20,280);  glVertex2f(40,280);  glVertex2f(40,240);  glVertex2f(-20,240);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(100,180);  glVertex2f(160,180);  glVertex2f(160,140);  glVertex2f(100,140);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-20,180);  glVertex2f(40,180);  glVertex2f(40,140);  glVertex2f(-20,140);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(110,80);  glVertex2f(160,80);  glVertex2f(160,40);  glVertex2f(110,40);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(-30,80);  glVertex2f(20,80);  glVertex2f(20,40);  glVertex2f(-30,40);  glEnd();  //door  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<200;i++)  {  glColor3ub(65, 81, 7);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r=25;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x+65,y+75);  }  glEnd();  glColor3ub(202, 218, 218);  glBegin(GL\_POLYGON);  glVertex2f(40,80);  glVertex2f(90,80);  glVertex2f(90,0);  glVertex2f(40,0);  glEnd();  glColor3ub(241, 160, 7);  glBegin(GL\_POLYGON);  glVertex2f(40,80);  glVertex2f(60,70);  glVertex2f(60,0);  glVertex2f(40,0);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(90,80);  glVertex2f(90,0);  glVertex2f(70,0);  glVertex2f(70,70);  glEnd();  //floor  glColor3ub(246, 220, 4);  glLineWidth(8);  glBegin(GL\_LINES);  glVertex2f(-50,400);  glVertex2f(200,400);  glVertex2f(200,300);  glVertex2f(-50,300);  glVertex2f(200,200);  glVertex2f(-50,200);  glVertex2f(200,100);  glVertex2f(-50,100);  glVertex2f(65,500);  glVertex2f(65,100);  glEnd();  }  void tree()  {  glColor3ub(176, 144, 9);  glBegin(GL\_POLYGON);  glVertex2f(340,10);  glVertex2f(340,180);  glVertex2f(390,180);  glVertex2f(390,10);  glVertex2f(400,0);  glVertex2f(330,0);  glEnd();  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<200;i++)  {  glColor3ub(20, 122, 51);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r=60;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x+310,y+100 );  }  glEnd();  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<200;i++)  {  glColor3ub(20, 122, 51);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r=60;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x+360,y+200 );  }  glEnd();  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<200;i++)  {  glColor3ub(20, 122, 51);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r=40;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x+300,y+160 );  }  glEnd();  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<200;i++)  {  glColor3ub(20, 122, 51);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r=40;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x+440,y+200 );  }  glEnd();  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<200;i++)  {  glColor3ub(20, 122, 51);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r=60;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x+460,y+140 );  }  glEnd();  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<200;i++)  {  glColor3ub(20, 122, 51);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r=60;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x+420,y+120 );  }  glEnd();  }  void bench()  {  glColor3ub(137, 130, 99);  glBegin(GL\_POLYGON);  glVertex2f(620,115);  glVertex2f(780,115);  glVertex2f(780,80);  glVertex2f(620,80);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(780,80);  glVertex2f(620,80);  glVertex2f(600,50);  glVertex2f(760,50);  glEnd();  glColor3ub(0,0,0);  glLineWidth(4);  glBegin(GL\_LINES);  glVertex2f(780,80);  glVertex2f(620,80);  glEnd();  glColor3ub(74, 63, 8);  glBegin(GL\_POLYGON);  glVertex2f(600,6);  glVertex2f(600,50);  glVertex2f(615,50);  glVertex2f(615,5);  glVertex2f(620,0);  glVertex2f(595,0);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(620,25);  glVertex2f(620,50);  glVertex2f(635,50);  glVertex2f(635,25);  glVertex2f(640,20);  glVertex2f(615,20);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(745,5);  glVertex2f(745,50);  glVertex2f(760,50);  glVertex2f(760,5);  glVertex2f(765,0);  glVertex2f(740,0);  glEnd();  glBegin(GL\_POLYGON);  glVertex2f(766,25);  glVertex2f(765.921969,58.8829539);  glVertex2f(780,80);  glVertex2f(780,25);  glVertex2f(784,20);  glVertex2f(762,20);  glEnd();  }  void lamppost()  {  glBegin(GL\_POLYGON);// Draw a Red 1x1 Square centered at origin  for(int i=0;i<200;i++)  {  glColor3ub(226, 220, 17);  float pi=3.1416;  float A=(i\*2\*pi)/200;  float r=17;  float x = r \* cos(A);  float y = r \* sin(A);  glVertex2f(x+780,y+223 );  }  glEnd();  glColor3ub(195, 216, 214);  glBegin(GL\_POLYGON);  glVertex2f(860,10);  glVertex2f(860,200);  glVertex2f(870,260);  glVertex2f(910,260);  glVertex2f(900,200);  glVertex2f(900,10);  glVertex2f(910,0);  glVertex2f(850,0);  glEnd();  glColor3ub(195, 216, 214);  glBegin(GL\_POLYGON);  glVertex2f(800,220);  glVertex2f(760,220);  glVertex2f(760,240);  glVertex2f(800,240);  glVertex2f(865,230);  glVertex2f(860,200);  glEnd();  }  void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  building();  tree();  bench();  lamppost();  // Display animated text  glColor3ub(255, 255 ,255); // Black text  glRasterPos2f(50, 40); // Position for text  for (int i = 0; i < strlen(displayText); i++) {  glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, displayText[i]);  }  glFlush();  }  void updateText(int value) {  if (storyIndex < 3) {  if (charIndex < strlen(story[storyIndex])) {  displayText[charIndex] = story[storyIndex][charIndex];  charIndex++;  displayText[charIndex] = '\0'; // Ensure string is properly terminated  } else {  charIndex = 0;  storyIndex++;  memset(displayText, 0, sizeof(displayText)); // Clear the display text  }  }  glutPostRedisplay();  if (storyIndex < 3) {  glutTimerFunc(100, updateText, 0); // Update text every 100 milliseconds  }  }  int main(int argc, char\*\* argv) {  glutInit(&argc, argv); // Initialize GLUT  glutInitWindowSize(500, 500); // Set the window's initial width & height  glutCreateWindow("OpenGL Simple Box Story"); // Window Title  glutDisplayFunc(display); // Register display callback handler for window re-paint  gluOrtho2D(-150,950,-100,600);  glutTimerFunc(500, updateText, 0); // Start text animation after 1 second  glutMainLoop(); // Enter the event-processing loop  return 0;  } |
| Output- |