```
#include <math.h>
#include <stdlib.h>
#if defined(__APPLE__)
#include <GLUT/glut.h>    /* Header File For The GLUT Library  */
#include <OpenGL/gl.h>  /* Header File For The OpenGL Library  */
#include <OpenGL/glu.h> /* Header File For The GLu Library  */
#else
#include <windows.h>
#include <C:\Users\Dylan\Documents\GitHub\ECE-251_new\project17\glut.h>
//connects to glut.h
#endif

#define MOVESPEED 40.0f   // Move 40 meters per second
#define TURNSPEED 2.0f    // 2 radians per second

// angle of rotation for the camera direction
float angle = 0.0f;

// actual vector representing the camera's direction
float lx=0.0f,lz=-1.0f;

// XZ position of the camera
float x=0.0f, z=5.0f;

// the key states. These variables will be zero
//when no key is being presses
float deltaAngle = 0.0f;
float deltaMove = 0;
int xOrigin = -1;

int etime = 0;
// Calculate elapsed time since last render (milliseconds)
int getEtime(void) {
  static int lasttime = 0;
  int time = glutGet(GLUT_ELAPSED_TIME);
  int etime = time - lasttime;
  lasttime = time; // store current time for next frame
  return (etime);
}

void changeSize(int w, int h) {

  // Prevent a divide by zero, when window is too short
  // (you cant make a window of zero width).
  if (h == 0)
    h = 1;

```

```
47    float ratio =  w * 1.0 / h;
48
49    // Use the Projection Matrix
50    glMatrixMode(GL_PROJECTION);
51
52    // Reset Matrix
53    glLoadIdentity();
54
55    // Set the viewport to be the entire window
56    glViewport(0, 0, w, h);
57
58    // Set the correct perspective.
59    gluPerspective(45.0f, ratio, 0.1f, 100.0f);
60
61    // Get Back to the Modelview
62    glMatrixMode(GL_MODELVIEW);
63  }
64
65  void drawWindows(){
66    glColor3f(0.992f, 0.992f, 0.588f); //yellow
67    glBegin(GL_QUADS);             // Draw a Quadrangle
68    glVertex3f(0.5f, 0.5f, 0.0f); // Bottom Left
69    glVertex3f(0.5f, 1.5f, 0.0f); // Top Left
70    glVertex3f(1.5f,1.5f, 0.0f); // Top Right
71    glVertex3f(1.5f, 0.5f, 0.0f); //Bottom Right
72    glEnd();               // Finished Square (Quadrangle)
73    glBegin(GL_QUADS);             // Draw a Quadrangle
74    glVertex3f(0.5f, 4.5f, 0.0f); // Bottom Left
75    glVertex3f(0.5f, 5.5f, 0.0f); // Top Left
76    glVertex3f(1.5f,5.5f, 0.0f); // Top Right
77    glVertex3f(1.5f, 4.5f, 0.0f); //Bottom Right
78    glEnd();               // Finished Square (Quadrangle)
79    glBegin(GL_QUADS);             // Draw a Quadrangle
80    glVertex3f(0.5f, 0.5f, 0.0f); // Bottom Left
81    glVertex3f(0.5f, 1.5f, 0.0f); // Top Left
82    glVertex3f(1.5f,1.5f, 0.0f); // Top Right
83    glVertex3f(1.5f, 0.5f, 0.0f); //Bottom Right
84    glEnd();               // Finished Square (Quadrangle)
85    glBegin(GL_QUADS);             // Draw a Quadrangle
86    glVertex3f(4.5f, 0.5f, 0.0f); // Bottom Left
87    glVertex3f(4.5f, 1.5f, 0.0f); // Top Left
88    glVertex3f(5.5f,1.5f, 0.0f); // Top Right
89    glVertex3f(5.5f, 0.5f, 0.0f); //Bottom Right
90    glEnd();               // Finished Square (Quadrangle)
91    glBegin(GL_QUADS);             // Draw a Quadrangle
92    glVertex3f(4.5f, 4.5f, 0.0f); // Bottom Left
93    glVertex3f(4.5f, 5.5f, 0.0f); // Top Left
```

```
  glVertex3f(5.5f,5.5f, 0.0f); // Top Right
  glVertex3f(5.5f, 4.5f, 0.0f); //Bottom Right
  glEnd();               // Finished Square (Quadrangle)
}
void drawDoor(){
  //begin handle
  glColor3f(0.75f, 0.75f, 0.75f); //blue
  glBegin(GL_QUADS);             // Draw a Quadrangle
  glVertex3f(3.5f, 1.2f, 0.0f); // Bottom Left
  glVertex3f(3.5f, 1.5f, 0.0f); // Top Left
  glVertex3f(3.75f,1.5f, 0.0f); // Top Right
  glVertex3f(3.75f, 1.2f, 0.0f); //Bottom Right
  glEnd();               // Finished Square (Quadrangle)
  //end handle
  //begin frame
  glColor3f(0.467f, 0.675f, 0.796f); //blue
  glBegin(GL_QUADS);             // Draw a Quadrangle
  glVertex3f(2.0f, 0.0f, 0.0f); // Bottom Left
  glVertex3f(2.0f, 3.0f, 0.0f); // Top Left
  glVertex3f(4.0f,3.0f, 0.0f); // Top Right
  glVertex3f(4.0f, 0.0f, 0.0f); //Bottom Right
  glEnd();               // Finished Square (Quadrangle)
  //end frame
}
void drawFoundation(){
  glColor3f(0.796f, 0.255f, 0.329f); //brick red
  glBegin(GL_QUADS);             // Draw a Quadrangle
  glVertex3f(0.0f, 0.0f, 0.0f); // Bottom Left
  glVertex3f(0.0f, 6.0f, 0.0f); // Top Left
  glVertex3f( 6.0f,6.0f, 0.0f); // Top Right
  glVertex3f(6.0f, 0.0f, 0.0f); //Bottom Right
  glEnd();               // Finished Square (Quadrangle)
}
void drawRoof(){
  glColor3f(0.55f, 0.55f, 0.48f);     //pastel gray
  glBegin(GL_TRIANGLES);         // Draw a Triangle
  glVertex3f(3.0f, 12.0f, 0.0f); // Top
  glVertex3f(-2.0f,6.0f, 0.0f); // Bottom Left
  glVertex3f( 8.0f,6.0f, 0.0f); // Bottom Right
  glEnd();               // Finished Triangle
  glColor3f(0.796f, 0.255f, 0.329f); //brick red
  glBegin(GL_QUADS);             // Draw a Quadrangle
  glVertex3f(4.0f, 7.0f, 0.0f); // Bottom Left
  glVertex3f(4.0f, 12.0f, 0.0f); // Top Left
  glVertex3f(6.0f,12.0f, 0.0f); // Top Right
  glVertex3f(6.0f, 7.0f, 0.0f); //Bottom Right
  glEnd();               // Finished Square (Quadrangle)
```

```
140     glEnd();                // Finished Square (Quadrangle)
141   }
142   void drawHouse() {
143     //begin windows
144     drawWindows();
145     //end windows
146     //begin door
147     drawDoor();
148     //end door
149     //begin house
150     drawFoundation();
151     //end house
152     //begin roof
153     drawRoof();
154     //end roof
155
156   }
157
158   void drawGround() {
159
160     glColor3f(0.376f, 0.502f, 0.22f);
161     glBegin(GL_QUADS);
162     glVertex3f(-100.0f, 0.0f, -100.0f);
163     glVertex3f(-100.0f, 0.0f,  100.0f);
164     glVertex3f( 100.0f, 0.0f,  100.0f);
165     glVertex3f( 100.0f, 0.0f, -100.0f);
166     glEnd();
167
168   }
169
170   void computePos(float deltaMove) {
171     angle += deltaAngle * etime / 1000.0f;
172     lx = sin(angle);
173     lz = -cos(angle);
174     x += (deltaMove * etime / 1000.0f) * lx;
175     z += (deltaMove * etime / 1000.0f) * lz;
176   }
177
178   void renderScene(void) {
179
180     etime = getEtime();   // Get the elasped ms since last render
181
182     if (deltaMove||deltaAngle)
183       {
184         computePos(deltaMove);
185       }
186
```

```
187     // Clear Color and Depth Buffers
188     glClearColor(0.529f, 0.808f, 0.980f, 0.0f);
189     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
190
191     // Reset transformations
192     glLoadIdentity();
193     // Set the camera
194     gluLookAt(  x, 1.0f, z,
195       x+lx, 1.0f,  z+lz,
196       0.0f, 1.0f,  0.0f);
197
198     //Draw Ground
199     drawGround();
200
201     // Draw House
202     drawHouse();
203     glutSwapBuffers();
204   }
205
206   void processNormalKeys(unsigned char key, int xx, int yy) {
207
208     if (key == 27)
209       exit(0);
210   }
211
212   void pressKey(int key, int xx, int yy) {
213
214     switch (key) {
215     case GLUT_KEY_UP    : deltaMove  =  MOVESPEED; break;
216     case GLUT_KEY_DOWN  : deltaMove  = -MOVESPEED; break;
217     case GLUT_KEY_LEFT  : deltaAngle = -TURNSPEED; break;
218     case GLUT_KEY_RIGHT : deltaAngle =  TURNSPEED; break;
219     }
220   }
221
222   void releaseKey(int key, int x, int y) {
223
224     switch (key) {
225     case GLUT_KEY_UP :
226     case GLUT_KEY_DOWN : deltaMove = 0;break;
227     case GLUT_KEY_LEFT :
228     case GLUT_KEY_RIGHT : deltaAngle = 0; break;
229
230     }
231   }
232
233   int main(int argc, char **argv) {
```

```
234
235     // init GLUT and create window
236     glutInit(&argc, argv);
237     glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
238     glutInitWindowPosition(100,100);
239     glutInitWindowSize(320,320);
240     glutCreateWindow("House");
241
242     // register callbacks
243     glutDisplayFunc(renderScene);
244     glutReshapeFunc(changeSize);
245     glutIdleFunc(renderScene);
246
247     glutKeyboardFunc(processNormalKeys);
248     glutSpecialFunc(pressKey);
249     glutSpecialUpFunc(releaseKey);
250
251     // OpenGL init
252     glEnable(GL_DEPTH_TEST);
253
254     // enter GLUT event processing cycle
255     glutMainLoop();
256
257     return 1;
258 }
259
```