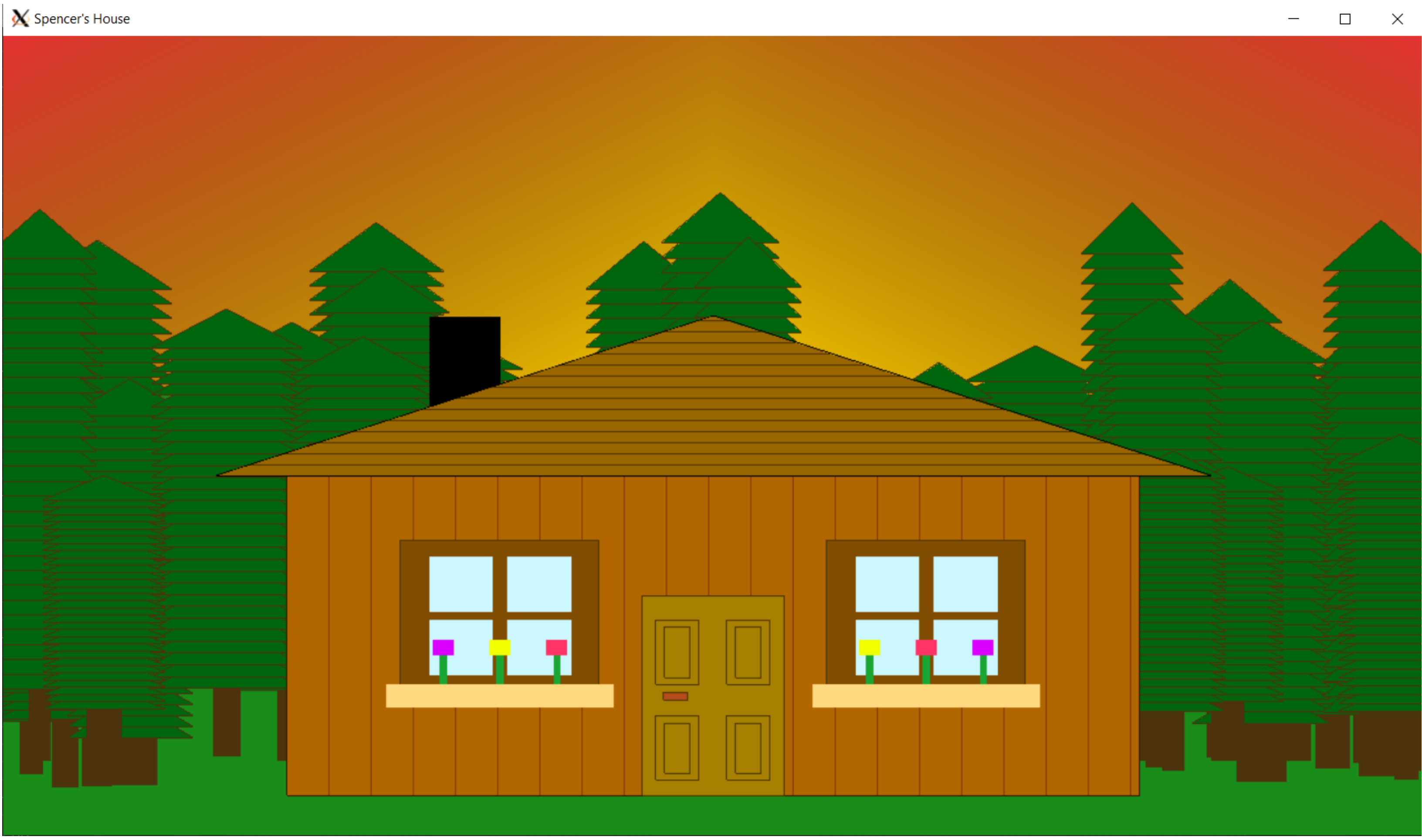
CSE 4200 Lab 1 – Spencer Wallace

***Analysis/Summary at the end**

Part 1) Draw

draw output



draw.cpp

```
#include <GL/glut.h>

#include "draw_util.h"

using namespace std;


//initialization

void init( void )

{
  glClearColor( 1.0, 0.3, 0.0, 0.0 );          //get white background color - changed to orange
  glPointSize( 8.0 );                          //specifies dot size
  glMatrixMode( GL_PROJECTION );
  glLoadIdentity();                            //replace current matrix with identity matrix
  gluOrtho2D( 0.0, SCREENWIDTH, 0.0, SCREENHEIGHT );
  glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
  glEnable( GL_BLEND );
}

void drawEmptyRect(float x, float y, float Width, float Height)
{
  glBegin( GL_LINE_STRIP );            //draw polyline
    glVertex2i( x, y);
    glVertex2i( x, y+Height);
    glVertex2i( x+Width, y+Height);
    glVertex2i( x+Width, y);
    glVertex2i( x, y);
  glEnd();
}

void drawEmptyTriangle(float X1, float Y1, float X2, float Y2, float X3, float Y3)
{
  glBegin( GL_LINE_STRIP );            //draw polyline
    glVertex2i( X1, Y1);
    glVertex2i( X2, Y2);
    glVertex2i( X3, Y3);
    glVertex2i( X1, Y1);
  glEnd();
}
```

```
void fillEmptyTriangle(float X1, float X2, float Y, float dX, float dY, float dXPS, GLfloat C1[], GLfloat C2[],
bool UseC2, int C2interval)
{
 for(int i = 1; i <= (int)dY; i++)
 {
     if(UseC2 && i%C2interval == 0)
         glColor3fv(C2);

     float yChange = dY*((float)i/dY);
     float yPos = Y + yChange;
     float xPos1 = X1 + (int)yChange*(dXPS/dY);
     float xPos2 = X2 - (int)yChange*(dXPS/dY);
     glBegin( GL_LINES );
      glVertex2i( xPos1, yPos );
      glVertex2i( xPos2, yPos );
     glEnd();

     if(UseC2 && i%C2interval == 0)
         glColor3fv(C1);
 }
}

void fillEmptyTriangle(float X1, float X2, float Y, float dX, float dY, float dXPS)
{
 for(int i = 1; i <= (int)dY; i++)
 {
     float yChange = dY*((float)i/dY);
     float yPos = Y + yChange;
```

```
    float xPos1 = X1 + (int)yChange*(dXPS/dY);

    float xPos2 = X2 - (int)yChange*(dXPS/dY);

    glBegin( GL_LINES );

      glVertex2i( xPos1, yPos );

      glVertex2i( xPos2, yPos );

    glEnd();

 }

}


void drawDoorDetails(float x, float y, float Width, float Height)

{

float innerX = x + (int)Width*0.2; float innerWidth = (int)Width*0.6;

float innerY = y + (int)Height*0.1; float innerHeight = (int)Height*0.8;

drawEmptyRect( x, y, Width, Height );

drawEmptyRect( innerX, innerY, innerWidth, innerHeight );

}


void drawSunset(float Xstart, float Ystart, float Xpixels, float Ypixels, GLfloat weakColor[], GLfloat
strongColor[])

{

 float Xstrong = Xstart + Xpixels/2.0;

 float Ystrong = Ypixels;

 for(int y = 0; y <= (int)Ypixels; y++)

 {


   for(int x = Xstart; x <= (int)Xpixels; x++)

   {

     float Xstrength;

     if(x <= Xstrong)
```

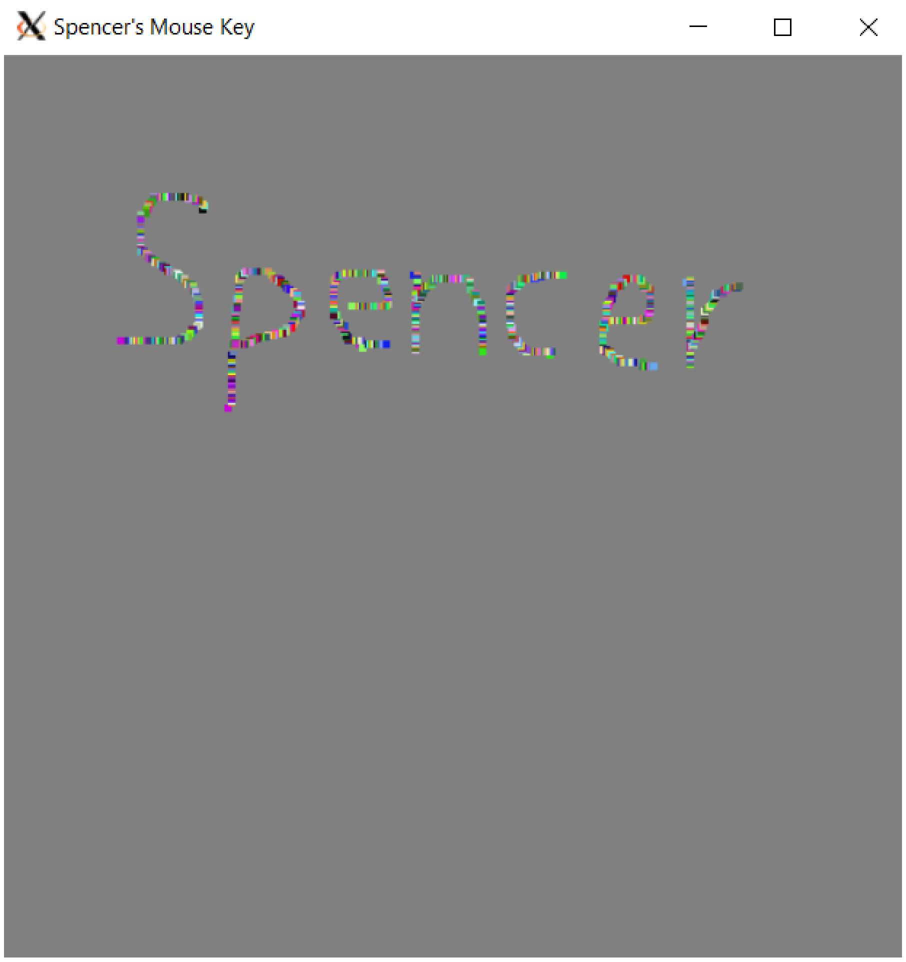
```
            Xstrength = x/Xstrong;
     else
            {
              float dif = x - Xstrong;
              Xstrength = (Xstrong - dif)/Xstrong;
            }
     float Ystrength = y/Ystrong;
     float weakCoeff = (1-Xstrength + 1-Ystrength)/2.0 * (1-Xstrength + 1-Ystrength)/2.0;
     float R = weakColor[0]*weakCoeff + (strongColor[0]*Xstrength + strongColor[0]*Ystrength)/2.0;
     float G = weakColor[1]*weakCoeff + (strongColor[1]*Xstrength + strongColor[1]*Ystrength)/2.0;
     float B = weakColor[2]*weakCoeff + (strongColor[2]*Xstrength + strongColor[2]*Ystrength)/2.0;

     GLfloat blendedColor [] = {R, G, B};
     glColor3fv(blendedColor);
     glPointSize(1.0);
     glBegin(GL_POINTS);
       glVertex2i( x, Ystart - y );
     glEnd();
   }
 }
}

void drawTrees(float X, float Y, float W, float H, float bX, float bY, float bW, float bH, float bHO, int
numBranches, GLfloat tC [], GLfloat bC [])
{
 for(int i = 0; i < numBranches; i++)
   {
     glColor3fv( tC );
     glRecti(X, Y, X + W, Y + H);
```

```
    // glColor3f( tC[0]/2.0, tC[1]/2.0, tC[2]/2.0 );
    // drawEmptyRect(X, Y, W, H);



     glColor3fv( bC );
     fillEmptyTriangle(bX, bX + bW, bY + i*bHO, bW, bH, bW/2.0);

     glColor3fv( tC );
     drawEmptyTriangle(bX, bY + i*bHO, bX + bW/2.0, bY + i*bHO + bH, bX + bW, bY + i*bHO);
   }
}

void drawRandTree(float xMin, float xMax, float yMin, float yMax, float hMin, float hMax, float wMin,
float wMax, int bMinW, int bMaxW, int bMinH, int bMaxH, int num)
{
  //trees
  for(int i = 0; i < num; i++)
   {
     float xRange = xMax - xMin;
     float X = xMin + xRange*(rand()%1000)/1000.0;

     float hRange = hMax - hMin;
     float H = hMin + hRange*(rand()%1000)/1000.0;

     float yRange = yMax - yMin;
     float Y = yMin + yRange*(rand()%1000)/1000.0;

     float wRange = wMax - wMin;
     float W = wMin + wRange*(rand()%1000)/1000.0;
     //float bRange = bMax - bMin;
     //float B = bMin + bRange*(rand()%1000)/1000.0;

     float bRangeW = bMaxW - bMinW;
     float BW = bMinW + bRangeW*(rand()%1000)/1000.0;

     float bRangeH = bMaxH - bMinH;
     float BH = bMinH + bRangeH*(rand()%1000)/1000.0;

     float treeBaseX = X; float treeBaseWidth = W;
     float treeBaseY = Y; float treeBaseHeight = H;

     float treeLeavesX1 = X + W/2.0 - BW/2.0; float treeLeavesWidth = BW;
     float treeLeavesX2 = treeLeavesX1 + treeLeavesWidth; float treeLeavesHeight = BH;
     float treeLeavesY = treeBaseY + treeBaseHeight;

     float treeIterationHeightOffset = treeLeavesHeight*0.3;
     GLfloat tree [] = {0.3, 0.2, 0.05};
     GLfloat leaf [] = {0.0, 0.4, 0.05};
     drawTrees(treeBaseX, treeBaseY, treeBaseWidth, treeBaseHeight, treeLeavesX1, treeLeavesY,
treeLeavesWidth, treeLeavesHeight,
                    treeIterationHeightOffset, 30, tree, leaf);
   }
}

void drawHouse()
{
  //draw sunset
```

```
 GLfloat weakSunsetColor[] = {0.9, 0.2, 0.2};

 GLfloat strongSunsetColor[] = {1.0, 0.8, 0.0};

 drawSunset(0, SCREENHEIGHT, SCREENWIDTH, SCREENHEIGHT*0.45,  weakSunsetColor,
strongSunsetColor);

 //draw grass

 glColor3f ( 0.1, 0.55, 0.1 );

 glRecti(0, 0, SCREENWIDTH, SCREENHEIGHT*0.55);

 //draw trees

 glColor3f ( 0.0, 0.0, 0.0 );

 drawRandTree(0, SCREENWIDTH, SCREENHEIGHT*0.06, SCREENHEIGHT*0.1, SCREENHEIGHT*0.06,
SCREENHEIGHT*0.1, SCREENWIDTH*0.015, SCREENWIDTH*0.04, SCREENWIDTH*0.07,
SCREENWIDTH*0.11, SCREENHEIGHT*0.03, SCREENHEIGHT*0.065, 50);


 //draw border of house and wall panels

 float WallX = SCREENWIDTH*0.2; float WallWidth = SCREENWIDTH*0.6; float WallStartY =
SCREENHEIGHT*0.05; float WallHeight = SCREENHEIGHT*0.4;

 glColor3f(0.7, 0.4, 0.0);

 for(int i = 1; i <= (int)WallWidth; i++)

 {

     if(i%((int)WallWidth/20) == 0)

          glColor3f(0.5, 0.23, 0.0);

     float xPos = WallX + WallWidth*((float)i/WallWidth);

     glBegin( GL_LINES );

       glVertex2i( xPos, WallStartY + WallHeight );

       glVertex2i( xPos, WallStartY );

     glEnd();

     if(i%((int)WallWidth/20) == 0)

          glColor3f(0.7, 0.4, 0.0);

 }

 glColor3f(0.2, 0.1, 0.0);
 drawEmptyRect(WallX, WallStartY, WallWidth, WallHeight);



 //draw roof

 float RoofX1 = SCREENWIDTH*0.15; float RoofX2 = SCREENWIDTH*0.85; float RoofStartY =
SCREENHEIGHT*0.45;

 float deltaX = SCREENWIDTH*0.7; float deltaY = SCREENHEIGHT*0.2; float deltaXPerSide = deltaX/2.0;

 float ChimOffsetX = SCREENWIDTH*0.05; float ChimX = RoofX1 + (deltaXPerSide/2.0) -
(ChimOffsetX/2.0);

 float ChimOffsetY = SCREENWIDTH*0.07; float ChimY = RoofStartY + deltaY*(deltaY/ChimX);

 //chimney

 glColor3f ( 0.0, 0.0, 0.0 );

 glRecti(ChimX+ChimOffsetX, ChimY+ChimOffsetY, ChimX, ChimY);

 //roof

 glColor3f ( 0.0, 0.0, 0.0 );

 drawEmptyTriangle(RoofX1, RoofStartY, RoofX1 + deltaXPerSide, RoofStartY + deltaY, RoofX2,
RoofStartY);

 glColor3f ( 0.6, 0.4, 0.0 );

 GLfloat RoofC1[] = {0.6, 0.4, 0.0};

 GLfloat RoofC2[] = {0.3, 0.2, 0.0};

 fillEmptyTriangle(RoofX1, RoofX2, RoofStartY, deltaX, deltaY, deltaXPerSide, RoofC1, RoofC2, true, 10);


 //draw door

 //door

 glColor3f ( 0.65, 0.5, 0.0 );

 glRecti(SCREENWIDTH*0.55, SCREENHEIGHT*0.3, SCREENWIDTH*0.45, SCREENHEIGHT*0.05);

 //handle

 glColor3f( 0.7, 0.3, 0.1);

 glRecti(SCREENWIDTH*0.4825, SCREENHEIGHT*0.18, SCREENWIDTH*0.465, SCREENHEIGHT*0.17);

 //details
 glColor3f( 0.3, 0.2, 0.0 );

 drawEmptyRect(SCREENWIDTH*0.45, SCREENHEIGHT*0.05, SCREENWIDTH*0.1,
SCREENHEIGHT*0.25);//door border

 drawEmptyRect(SCREENWIDTH*0.465, SCREENHEIGHT*0.17, SCREENWIDTH*0.0175,
SCREENHEIGHT*0.01);//handle border

 drawDoorDetails(SCREENWIDTH*0.46, SCREENHEIGHT* 0.07, SCREENWIDTH*0.03,
SCREENHEIGHT*0.08);

 drawDoorDetails(SCREENWIDTH*0.46, SCREENHEIGHT* 0.19, SCREENWIDTH*0.03,
SCREENHEIGHT*0.08);

 drawDoorDetails(SCREENWIDTH*0.51, SCREENHEIGHT* 0.07, SCREENWIDTH*0.03,
SCREENHEIGHT*0.08);

 drawDoorDetails(SCREENWIDTH*0.51, SCREENHEIGHT* 0.19, SCREENWIDTH*0.03,
SCREENHEIGHT*0.08);


 //draw window(s)

 //background

 glColor3f( 0.5, 0.3, 0.0 );

 glRecti(SCREENWIDTH*0.42, SCREENHEIGHT*0.37, SCREENWIDTH*0.28, SCREENHEIGHT*0.18);

 glRecti(SCREENWIDTH*0.72, SCREENHEIGHT*0.37, SCREENWIDTH*0.58, SCREENHEIGHT*0.18);

 glColor3f( 0.3, 0.2, 0.0 );

 drawEmptyRect(SCREENWIDTH*0.28, SCREENHEIGHT*0.18, SCREENWIDTH*0.14,
SCREENHEIGHT*0.19);

 drawEmptyRect(SCREENWIDTH*0.58, SCREENHEIGHT*0.18, SCREENWIDTH*0.14,
SCREENHEIGHT*0.19);


 //glass

 glColor3f( 1.0, 1.0, 1.0 );

 glRecti(SCREENWIDTH*0.4, SCREENHEIGHT*0.35, SCREENWIDTH*0.3, SCREENHEIGHT*0.2);

 glRecti(SCREENWIDTH*0.7, SCREENHEIGHT*0.35, SCREENWIDTH*0.6, SCREENHEIGHT*0.2);

 //transperancy

 glColor4f ( 0.7, 0.95, 1.0, 0.65);

 glRecti(SCREENWIDTH*0.4, SCREENHEIGHT*0.35, SCREENWIDTH*0.3, SCREENHEIGHT*0.2);
 glRecti(SCREENWIDTH*0.7, SCREENHEIGHT*0.35, SCREENWIDTH*0.6, SCREENHEIGHT*0.2);
 //window pane
 glColor3f( 0.5, 0.3, 0.0 );
 glRecti(SCREENWIDTH*0.4, SCREENHEIGHT*0.28, SCREENWIDTH*0.3, SCREENHEIGHT*0.27);
 glRecti(SCREENWIDTH*0.355, SCREENHEIGHT*0.35, SCREENWIDTH*0.345, SCREENHEIGHT*0.2);
 glRecti(SCREENWIDTH*0.7, SCREENHEIGHT*0.28, SCREENWIDTH*0.6, SCREENHEIGHT*0.27);
 glRecti(SCREENWIDTH*0.655, SCREENHEIGHT*0.35, SCREENWIDTH*0.645, SCREENHEIGHT*0.2);
 //flower boxes
 glColor3f( 1.0, 0.85, 0.5 );
 glRecti(SCREENWIDTH*0.43, SCREENHEIGHT*0.16, SCREENWIDTH*0.27, SCREENHEIGHT*0.19);
 glRecti(SCREENWIDTH*0.73, SCREENHEIGHT*0.16, SCREENWIDTH*0.57, SCREENHEIGHT*0.19);
 //flowers
 //stems
 glColor3f( 0.1, 0.65, 0.2 );
 glRecti(SCREENWIDTH*0.3925, SCREENHEIGHT*0.19, SCREENWIDTH*0.3875, SCREENHEIGHT*0.225);
 glRecti(SCREENWIDTH*0.3525, SCREENHEIGHT*0.19, SCREENWIDTH*0.3475, SCREENHEIGHT*0.225);
 glRecti(SCREENWIDTH*0.3125, SCREENHEIGHT*0.19, SCREENWIDTH*0.3075, SCREENHEIGHT*0.225);
 glRecti(SCREENWIDTH*0.6925, SCREENHEIGHT*0.19, SCREENWIDTH*0.6875, SCREENHEIGHT*0.225);
 glRecti(SCREENWIDTH*0.6525, SCREENHEIGHT*0.19, SCREENWIDTH*0.6475, SCREENHEIGHT*0.225);
 glRecti(SCREENWIDTH*0.6125, SCREENHEIGHT*0.19, SCREENWIDTH*0.6075, SCREENHEIGHT*0.225);
 //petals
 glColor3f( 1.0, 0.2, 0.4 ); //red
 glRecti(SCREENWIDTH*0.3975, SCREENHEIGHT*0.225, SCREENWIDTH*0.3825, SCREENHEIGHT*0.245);
 glRecti(SCREENWIDTH*0.6575, SCREENHEIGHT*0.225, SCREENWIDTH*0.6425, SCREENHEIGHT*0.245);

 glColor3f( 0.95, 1.0, 0.0 ); //yellow
 glRecti(SCREENWIDTH*0.3575, SCREENHEIGHT*0.225, SCREENWIDTH*0.3425, SCREENHEIGHT*0.245);
 glRecti(SCREENWIDTH*0.6175, SCREENHEIGHT*0.225, SCREENWIDTH*0.6025, SCREENHEIGHT*0.245);
 glColor3f( 0.85, 0.0, 1.0 ); //purple

 glRecti(SCREENWIDTH*0.6975, SCREENHEIGHT*0.225, SCREENWIDTH*0.6825, SCREENHEIGHT*0.245);

 glRecti(SCREENWIDTH*0.3175, SCREENHEIGHT*0.225, SCREENWIDTH*0.3025, SCREENHEIGHT*0.245);


 glFlush();                          //send all output to screen

}


void display( void )

{

 glClear( GL_COLOR_BUFFER_BIT );  //clear screen

 drawHouse();

}
```

Part 2) Mouse Key

Output



mouse_key.cpp

```
//mouse_key.cpp
#include <GL/glut.h>
#include <stdlib.h>
#include <iostream>
#define screenHeight 500

using namespace std;
//initialization
```

```
void init( void )

{

  glClearColor( 0.5, 0.5, 0.5, 1.0 );            //get white background color

  glColor3f( 0.0f, 0.0f, 0.0f );        //set drawing color

  glPointSize( 4.0 );                            //a dot is 4x4

  glMatrixMode( GL_PROJECTION );

  glLoadIdentity();

  gluOrtho2D( 0.0, 500.0, 0.0, 500.0 );

} //init


void display()

{

  glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

  glFlush();

}


void drawDot( int x, int y )

{

  glBegin( GL_POINTS );

    glVertex2i( x, y );                 //draw a points

  glEnd();

} //drawDot


void myMouse( int button, int state, int x, int y )

{

  if ( button == GLUT_LEFT_BUTTON && state == GLUT_DOWN )

    drawDot( x, screenHeight - y );

  glFlush();                                     //send all output to screen

}
```

```
void myMovedMouse(  int mouseX, int mouseY )
{
  GLint x = mouseX;
  GLint y = screenHeight - mouseY;
  GLint brushsize = 4;
  float R = (float)(rand() % 1000)/1000.0;
  float G = (float)(rand() % 1000)/1000.0;
  float B = (float)(rand() % 1000)/1000.0;
  glColor3f( R, G, B );
  glRecti ( x, y, x + brushsize, y + brushsize );
  glFlush();
} //myMovedMouse

void myKeyboard ( unsigned char key, int mouseX, int mouseY )
{
  GLint x = mouseX;
  GLint y = screenHeight - mouseY;
  switch( key )
  {
    case 'p':
            drawDot ( x, y );
            glFlush();
            break;
    case 'r':
        glRecti ( x, y, x + 20, y + 30 );
            glFlush();
            break;
    case 'e':
```

```
  case 27:
          exit ( -1 );
      break;

  default :
          break;
 }
}
```

Summary

I believe that both parts were completed successfully, thus I would give myself 20/20 points. For part 1 I added functions for drawing an empty rectangle or triangle (not filled with color), as well as a function for filling them with up to two colors. I also added functions for drawing the trees and the sunset. For part 2 I modified the color to be drawn with a mouse click to use random values for red, green, and blue. I also changed it so that all dots (initial click and held) were 4x4 pixels.