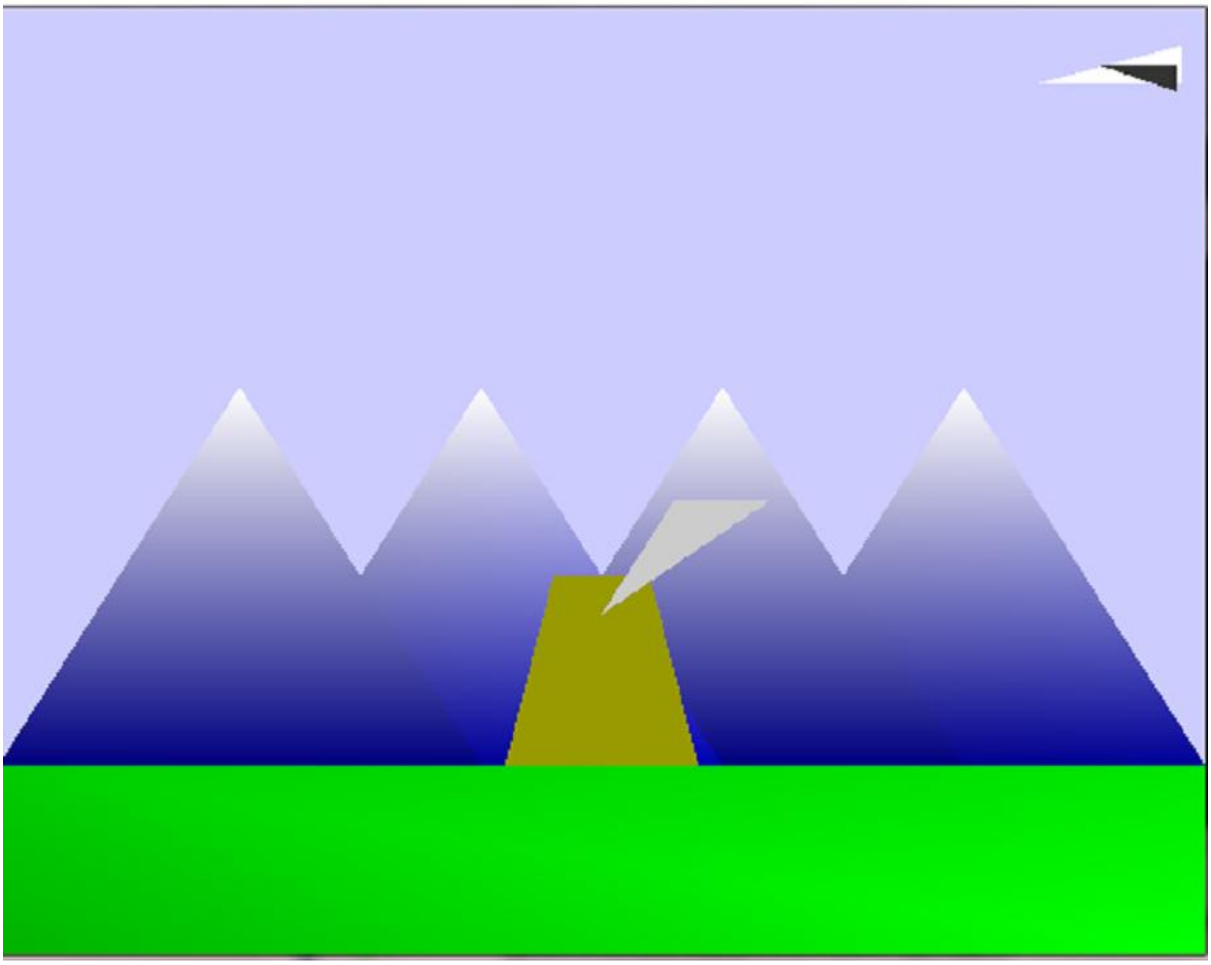
Lab: 2D Transformations

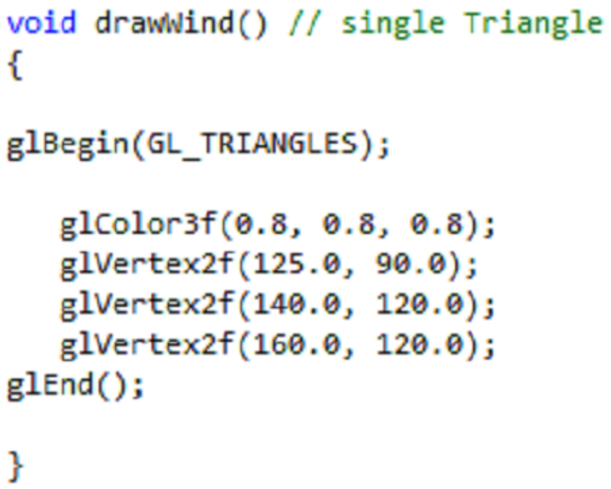
Task 1: Open new Project and add given w.cpp file into it.

This will give you following scene:



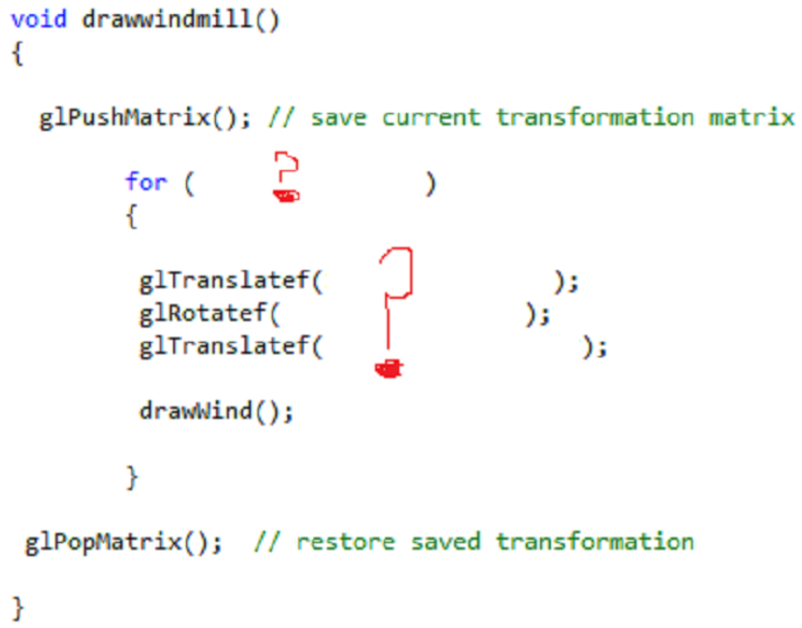
Task 2: Completing windmill via transformations

1. Read and Observe end points from drawWind()



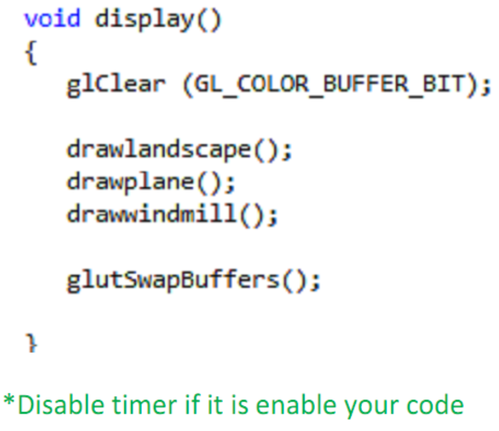
```
void drawWind() // single Triangle
{

glBegin(GL_TRIANGLES);

   glColor3f(0.8, 0.8, 0.8);
   glVertex2f(125.0, 90.0);
   glVertex2f(140.0, 120.0);
   glVertex2f(160.0, 120.0);
glEnd();

}
```

2. Now use a for loop and transformation commands in proper sequence to complete draw Windmill(..) function.



```
void drawwindmill()
{

  glPushMatrix(); // save current transformation matrix

        for (     ?         )
        {

         glTranslatef(    ?          );
         glRotatef(                 );
         glTranslatef(                 );

         drawWind();

        }

 glPopMatrix();  // restore saved transformation

}
```

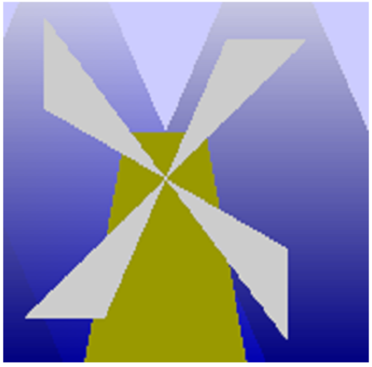
3. Check display function, it should look like this and run your code.



```
void display()
{
   glClear (GL_COLOR_BUFFER_BIT);

   drawlandscape();
   drawplane();
   drawwindmill();

   glutSwapBuffers();

}
```

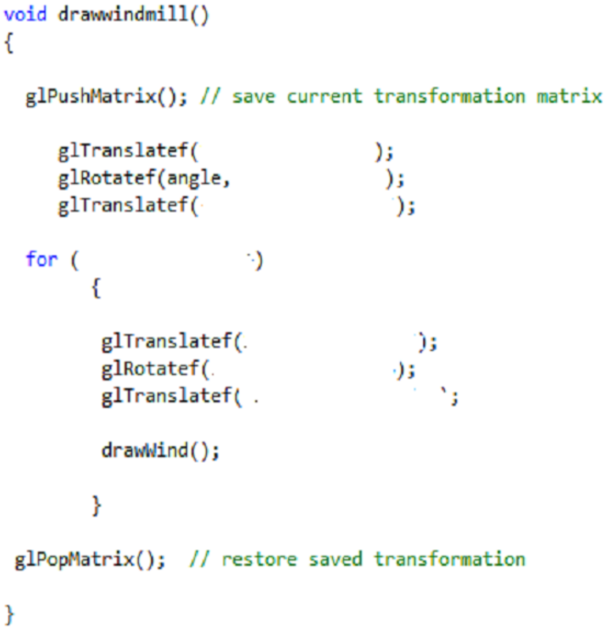
*Disable timer if it is enable your code

You windmill should now have all four blades but it is not animating.

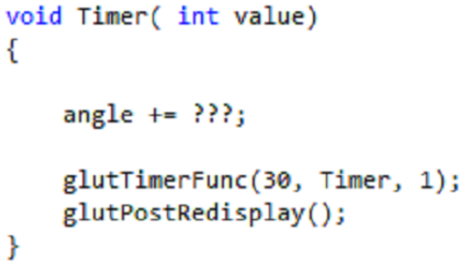


Task 3: Animating windmill via transformations

This time you need another set of transformations to continuously rotate about arbitrary point. Complete the drawWindmill(…) and enable timer as shown in below hint:

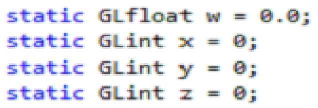


```
void drawwindmill()
{

  glPushMatrix(); // save current transformation matrix

     glTranslatef(                 );
     glRotatef(angle,               );
     glTranslatef(                   );

  for (                  )
        {

          glTranslatef(                 );
          glRotatef(                  );
          glTranslatef(                    ;

          drawWind();

        }

 glPopMatrix();  // restore saved transformation

}
```



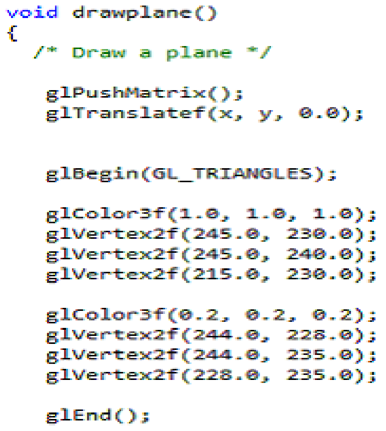
```
void Timer( int value)
{

    angle += ???;

    glutTimerFunc(30, Timer, 1);
    glutPostRedisplay();
}
```

Task 4: Animating Plane to drop it

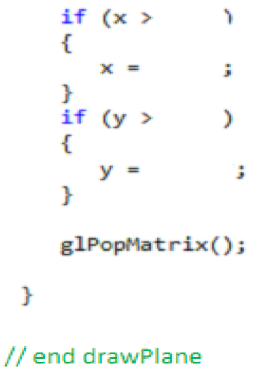


```
static GLfloat w = 0.0;
static GLint x = 0;
static GLint y = 0;
static GLint z = 0;
```

Make changes to your drawPlane function as shown below and right:

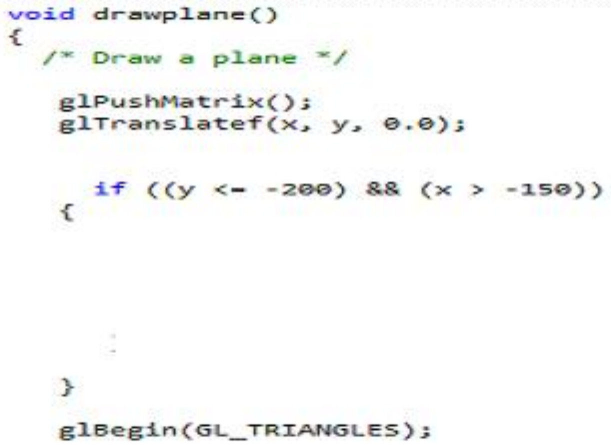


```
void drawplane()
{
  /* Draw a plane */

    glPushMatrix();
    glTranslatef(x, y, 0.0);


    glBegin(GL_TRIANGLES);

    glColor3f(1.0, 1.0, 1.0);
    glVertex2f(245.0, 230.0);
    glVertex2f(245.0, 240.0);
    glVertex2f(215.0, 230.0);

    glColor3f(0.2, 0.2, 0.2);
    glVertex2f(244.0, 228.0);
    glVertex2f(244.0, 235.0);
    glVertex2f(228.0, 235.0);

    glEnd();
```



```
        if (x >      )
        {
           x =       ;
        }
        if (y >      )
        {
           y =        ;
        }

        glPopMatrix();

    }

// end drawPlane
```

Task5: After collision with land make it some jerk using glRotatef



```
void drawplane()
{
  /* Draw a plane */

    glPushMatrix();
    glTranslatef(x, y, 0.0);

      if ((y <= -200) && (x > -150))
    {




    }
    glBegin(GL_TRIANGLES);
```