3D Objects

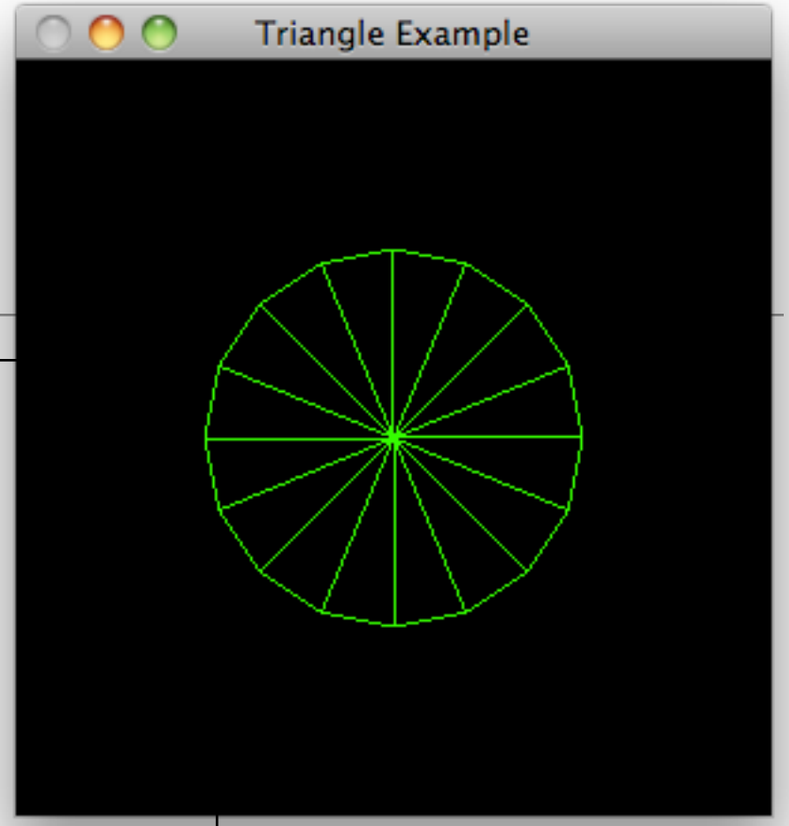
OpenGL

Learning Outcomes

- Be able to use glutSpecialFuction() and glutPostRedisplay()
- Be able to draw a 3 Dimensional Object
- Be able to Rotate this object and view it from different angles
- Understand the difference between glPolygonMode G_FILL and GL_LINE
- Have seen a simple shading model

Circular Triangle Fan

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


  glBegin(GL_TRIANGLE_FAN);

  glVertex3f(0.0f, 0.0f, 0.0f);

   float x, y, angle;
   for(angle = 0.0f; angle < (2.0f*GL_PI); angle += (GL_PI/8.0f))
   {
     x = 50.0f*sin(angle);
     y = 50.0f*cos(angle);

     glVertex2f(x, y);
   }

  glEnd();

  glutSwapBuffers();
}
```



lab04a_01

- This version is on the same (z=0) plane

drawCone()

```
void drawCone(float x, float y, float z, float radius)
{
  glBegin(GL_TRIANGLE_FAN);
    glVertex3f(x, y, z);
    float angle;
    for(angle = 0.0f; angle < (2.0f*GL_PI); angle += (GL_PI/8.0f))
    {
      x = radius*sin(angle);
      y = radius*cos(angle);
      glVertex2f(x, y);
    }
  glEnd();
}
```

- Parameterise the creation of the fan

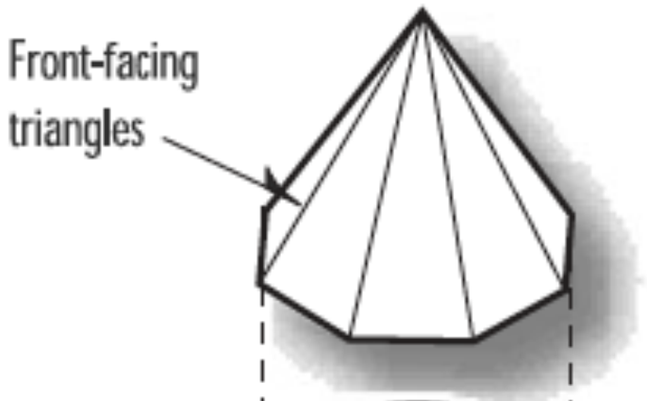
```
void renderScene(void)
{
  glClear(GL_COLOR_BUFFER_BIT);

  drawCone(0,0,0, 50);

 glutSwapBuffers();
}
```

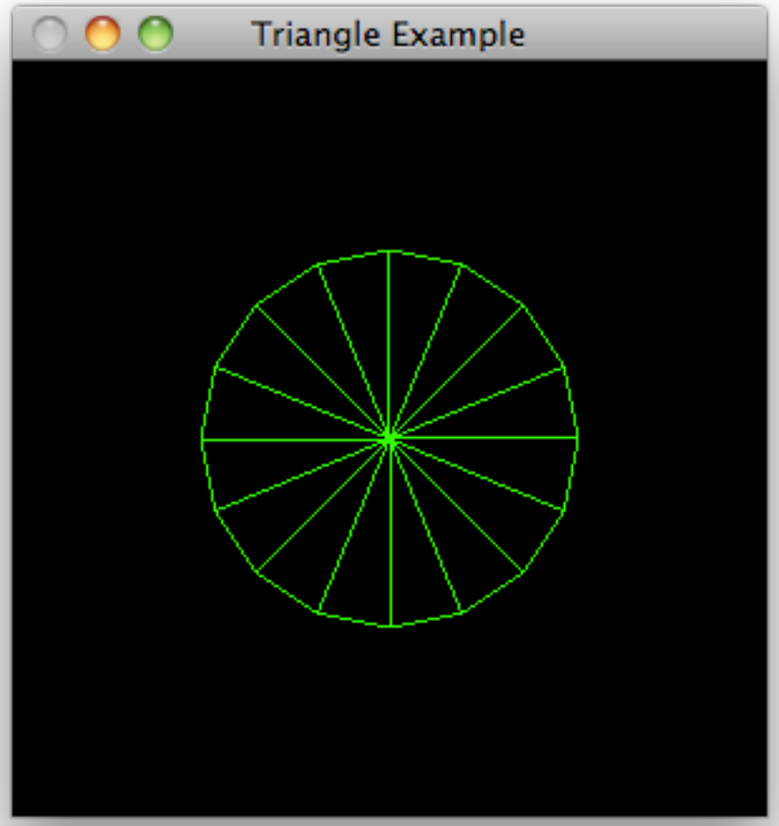
Cone

- Looking directly down the z-axis and can see only a circle composed of a fan of triangles.

- Produce a cone shape, using the first vertex as the point of the cone and there remaining vertices as points along a circle farther down the z-axis

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

  drawCone(0,0,75, 50);

 glutSwapBuffers();
}
```





lab04a_02a

Special Keys

glutSpecialFunc
glutSpecialFunc sets the special keyboard callback for the *current window*.
Usage
void glutSpecialFunc(void (*func)(int key, int x, int y));
func The new special callback function.
Description
glutSpecialFunc sets the special keyboard callback for the *current window*. The special keyboard callback is triggered when keyboard function or directional keys are pressed. The key callback parameter is a GLUT KEY * constant for the special key pressed. The x and y callback parameters indicate the mouse in window relative coordinates when the key was pressed. When a new window is created, no special callback is initially registered and special key strokes in the window are ignored. Passing NULL to glutSpecialFunc disables the generation of special callbacks. During a special callback, glutGetModifiers may be called to determine the state of modifier keys when the keystroke generating the callback occurred. An implementation should do its best to provide ways to generate all the GLUT KEY * special keys

GLUT KEY F1 F1 function key.
GLUT KEY F2 F2 function key.
GLUT KEY F3 F3 function key.
GLUT KEY F4 F4 function key.
GLUT KEY F5 F5 function key.
GLUT KEY F6 F6 function key.
GLUT KEY F7 F7 function key.
GLUT KEY F8 F8 function key.
GLUT KEY F9 F9 function key.
GLUT KEY F10 F10 function key.
GLUT KEY F11 F11 function key.
GLUT KEY F12 F12 function key.
GLUT KEY LEFT Left directional key.
GLUT KEY UP Up directional key.
GLUT KEY RIGHT Right directional key.
GLUT KEY DOWN Down directional key.
GLUT KEY PAGE UP Page up directional key.
GLUT KEY PAGE DOWN Page down directional key.
GLUT KEY HOME Home directional key.
GLUT KEY END End directional key.
GLUT KEY INSERT Inset directional key.

Register Special Key Processing

```
int main(int argc, char* argv[])
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
  glutCreateWindow("Triangle Example");
  glutInitWindowSize(800,600);
  glutSpecialFunc(specialKeys);
  glutDisplayFunc(renderScene);
  retupRC();
  glutMainLoop();

  return 0;
}
```

Rotation

glRotate
Multiply the current matrix by a rotation matrix
C Specification
void **glRotatef**(GLfloat *angle*, **GLfloat *x*, GLfloat *y*, GLfloat *z*);
Parameters
angle Specifies the angle of rotation, in degrees.
x, *y*, *z* Specify the x, y, and z coordinates of a vector, respectively.
Description
glRotate produces a rotation of *angle* degrees around the vector (*x*,*y*,*z*). The current matrix (see glMatrixMode) is multiplied by a rotation matrix with the product replacing the current matrix.If the matrix mode is either GL_MODELVIEW or GL_PROJECTION, all objects drawn after glRotate is called are rotated. Use glPushMatrix and glPopMatrix to save and restore the unrotated coordinate system.
Notes
This rotation follows the right-hand rule, so if the vector (*x*,*y*,*z*) points toward the user, the rotation will be counterclockwise.
Errors
GL_INVALID_OPERATION is generated if glRotate is executed between the execution of
glBegin and the corresponding execution of glEnd.
Associated Gets
glGet with argument GL_MATRIX_MODE
glGet with argument GL_COLOR_MATRIX
glGet with argument GL_MODELVIEW_MATRIX
glGet with argument GL_PROJECTION_MATRIX
glGet with argument GL_TEXTURE_MATRIX
See Also
glMatrixMode, glMultMatrix, glPushMatrix, glScale, glTranslate

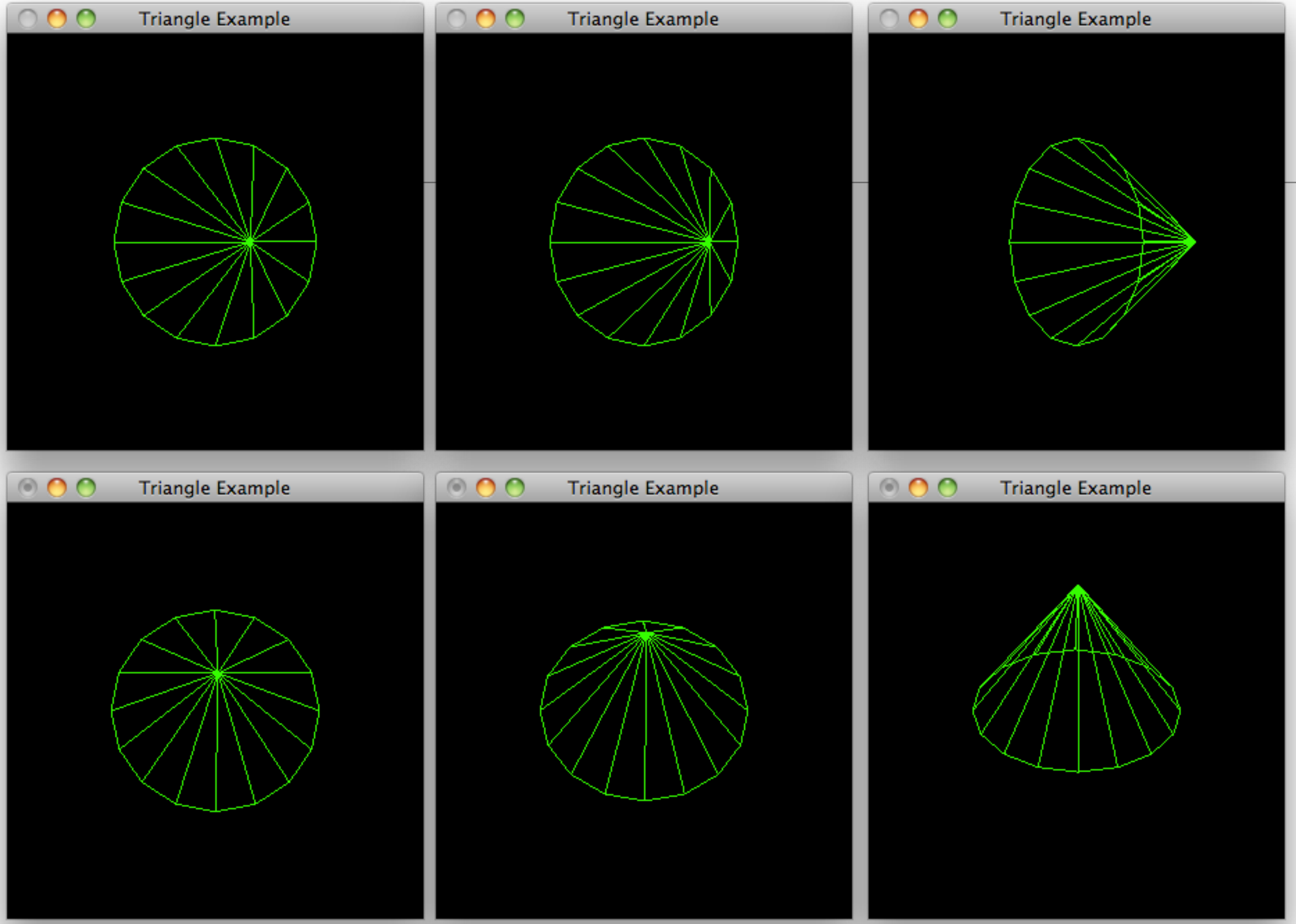
Refresh/Redisplay

glutPostRedisplay
glutPostRedisplay marks the *current window* as needing to be redisplayed.
Usage
void glutPostRedisplay(void);
Description
Mark the normal plane of *current window* as needing to be redisplayed. The next iteration through glutMainLoop, the window's display callback will be called to redisplay the window's normal plane. Multiple calls to glutPostRedisplay before the next display callback opportunity generates only a single redisplay callback. glutPostRedisplay may be called within a window's display or overlay display callback to re-mark that window for redisplay. Logically, normal plane damage notification for a window is treated as a glutPostRedisplay on the damaged window. Unlike damage reported by the window system, glutPostRedisplay will *not* set to true the normal plane's damaged status (returned by glutLayerGet(GLUT_NORMAL_DAMAGED).
Also, see glutPostOverlayRedisplay.

SpecialKeys Driven Rotation

- Rotate the current display by 1 degree
- around the X axis (UP and DOWN arrow keys)
- or around the Y axis (LEFT or RIGHT arrow keys)

```
void specialKeys(int key, int x, int y)
{
  int xRot=0,yRot=0;

  xRot = (key == GLUT_KEY_UP)?    -1 : xRot;
  xRot = (key == GLUT_KEY_DOWN)?   1 : xRot;
  yRot = (key == GLUT_KEY_LEFT)?  -1 : yRot;
  yRot = (key == GLUT_KEY_RIGHT)?  1 : yRot;

  glRotatef(xRot, 1.0f, 0.0f, 0.0f);
  glRotatef(yRot, 0.0f, 1.0f, 0.0f);

  glutPostRedisplay();
}
```

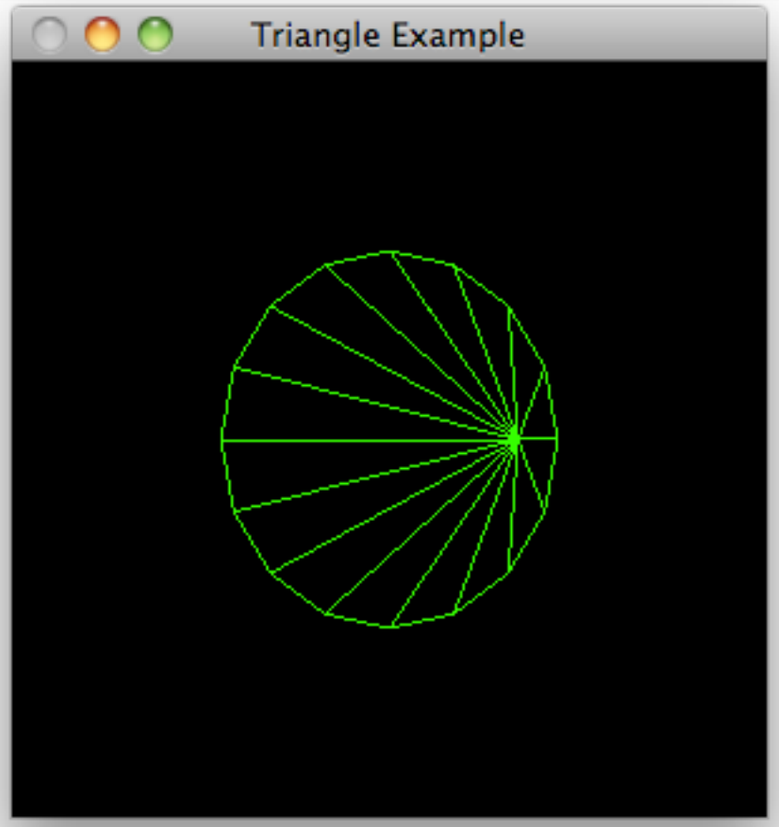

Triangle Example
Triangle Example
Triangle Example
Triangle Example
Triangle Example
Triangle Example

lab04a_02b

Polygon Modes - Front/Back reversed

- By default, polygons are drawn solid.
- The function glPolygonMode allows polygons to be rendered as filled solids, as outlines, or as points only.
- Can be applied to both sides of the polygons or only to the front or back

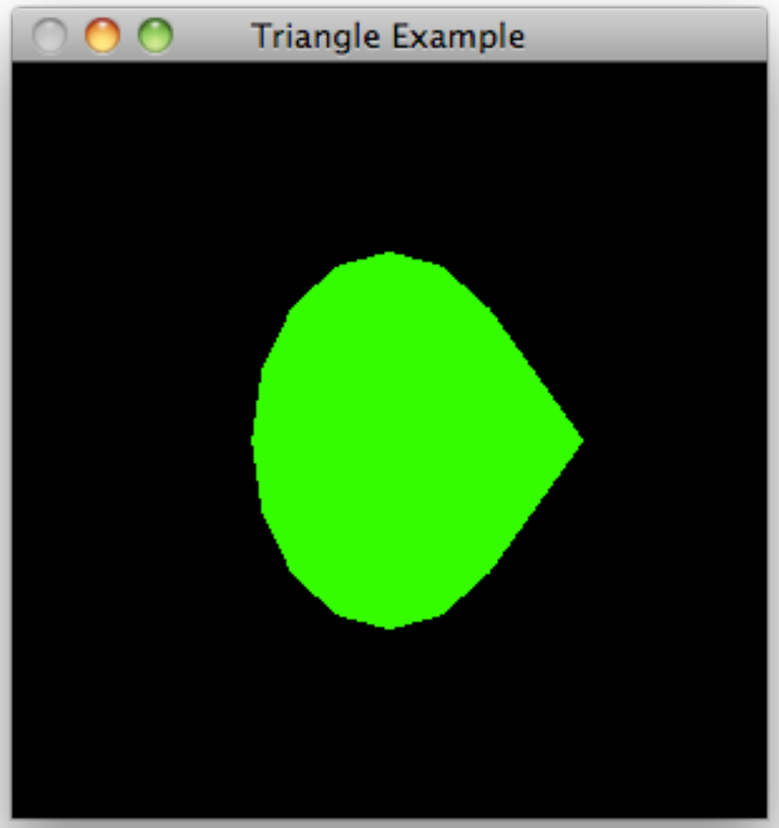
```
void retupRC()
{
  glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

  glColor3f(0.0f, 1.0f, 0.0f);

  glOrtho (-100.0f, 100.0f, -100.0f, 100.0f, -100.0f, 100.0f);

  glPolygonMode(GL_FRONT,GL_FILL);
  glPolygonMode(GL_BACK, GL_LINE);
}
```

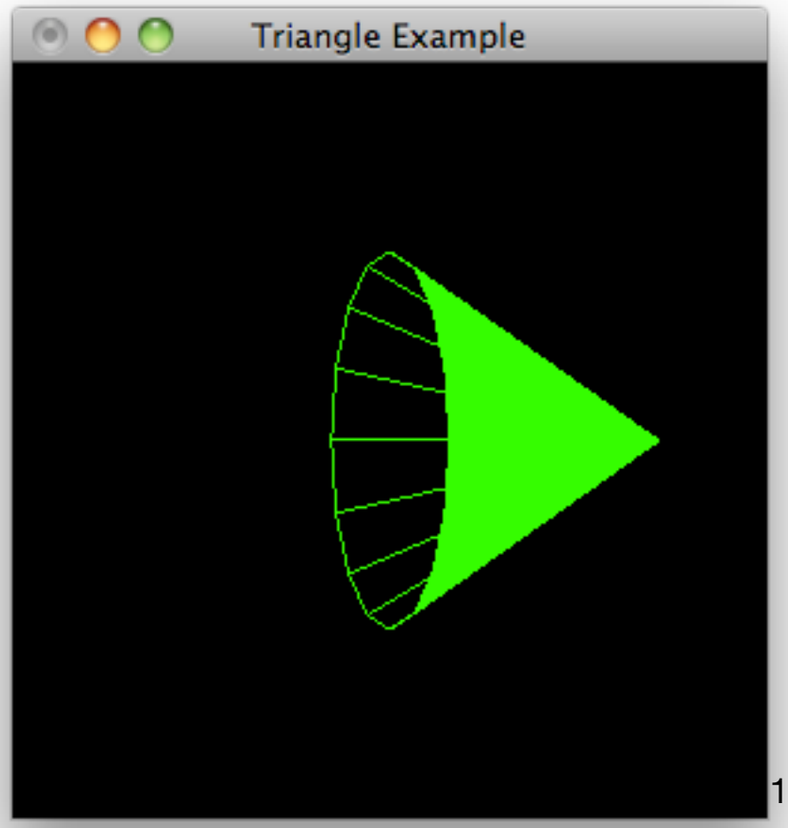


?

Polygon Modes

```
void retupRC()
{
  //...

  glFrontFace(GL_CW);
  glPolygonMode(GL_FRONT,GL_FILL);
  glPolygonMode(GL_BACK, GL_LINE);
}
```





lab04a_02c

Setting Polygon Colors

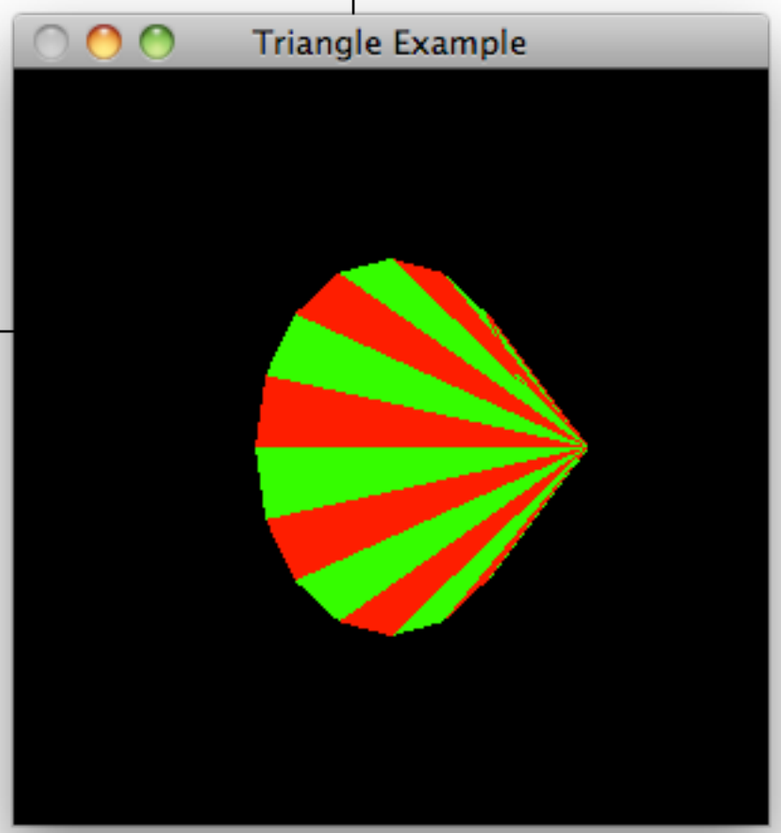
- Colors are specified per vertex, not per polygon.

- The shading model affects whether the polygon is solidly colored (using the current color selected when the last vertex was specified) or smoothly shaded between the colors specified for each vertex.

- glShadeModel(GL_FLAT) tells OpenGL to fill the polygons with the solid color that was current when the polygon's last vertex was specified. This is why we can simply change the current color to red or green before specifying the next vertex in our triangle fan.

- glShadeModel(GL_SMOOTH) would tell OpenGL to shade the triangles smoothly from each vertex, attempting to interpolate the colors between those specified for each vertex

```
void drawCone(float x, float y, float z, float radius)
{
  int step = 0;

  glBegin(GL_TRIANGLE_FAN);
    glVertex3f(x, y, z);
    float angle;
    for(angle = 0.0f; angle < (2.0f*GL_PI); angle += (GL_PI/8.0f))
    {
      x = radius*sin(angle);
      y = radius*cos(angle);

      glColor3f(step % 2 == 0, step % 2, 0.0f);
      step++;

      glVertex2f(x, y);
    }
  glEnd();
}
```

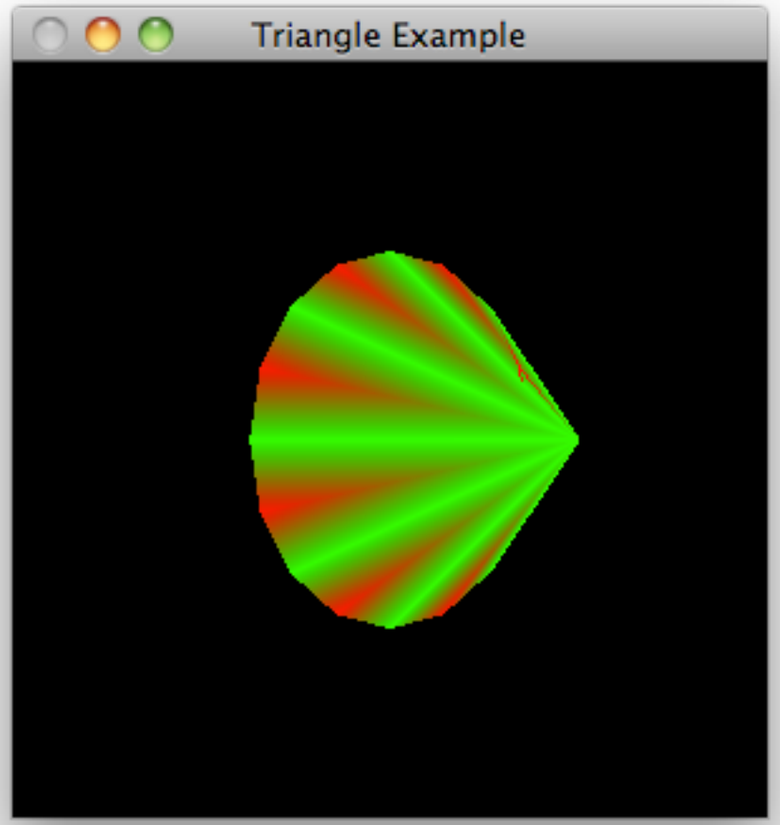


- Change from green to red for every polygon

lab04a_03a

Smooth Shading

```
void retupRC()
{
  //...

  glShadeModel(GL_SMOOTH);
}
```



lab04a_03b