Lighting a Scene

OpenGL

Using Light Sources

- Manipulating the ambient light has its uses, but for most applications attempting to model the real world, you must specify one or more specific sources of light.
- In addition to their intensities and colors, these sources have a location and/or a direction.
- The placement of these lights can dramatically affect the appearance of the scene.
- OpenGL supports at least eight independent light sources located anywhere in your scene or out of the viewing volume.
- You can locate a light source an infinite distance away and make its light rays parallel or make it a nearby light source radiating outward.
- You can also specify a spotlight with a specific cone of light radiating from it, as well as manipulate its characteristics.

Enabling the Light Source

- Instead of a general ambient light model from previous examples, we identify a specific light source - named GL_LIGHT0.

- Set the ambient and diffuse properties of this light, and enable it.

- Also enable colour tracking and vector normalization

```
float  ambientLightModerate[] = { 0.3f, 0.3f, 0.3f, 1.0f };
float  diffuseLightModerate[] = { 0.7f, 0.7f, 0.7f, 1.0f };

glEnable(GL_LIGHTING);

glLightfv(GL_LIGHT0,GL_AMBIENT,ambientLightModerate);
glLightfv(GL_LIGHT0,GL_DIFFUSE,diffuseLightModerate);

glEnable(GL_LIGHT0);

glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);

glClearColor(0.0f, 0.0f, 0.6f, 1.0f);
glEnable(GL_NORMALIZE);
```

Positioning the Light

```
void positionLight()
{
  float lightPos[] = { -50.f, 50.0f, 0.0f, 1.0f };
  glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
}
```

```
void World::render()
{
  Color::Blue.renderClear();
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);


  if (projectors.isPerspective())
  {
    glLoadIdentity();
    cameras.currentCamera->render();
  }
  positionLight();

  scene->render();

  glutSwapBuffers();
}
```

- The light is positioned in the World::render() function.
- The last value in the lightPos array is 1.0, which specifies that the designated coordinates are the position of the light source.
- If the last value in the array is 0.0, it indicates that the light is an infinite distance away along the vector specified by lightPos.

By placing the light's position when the viewing transformation is performed, we ensure that the light is in the proper location regardless of how we subsequently transform the geometry.

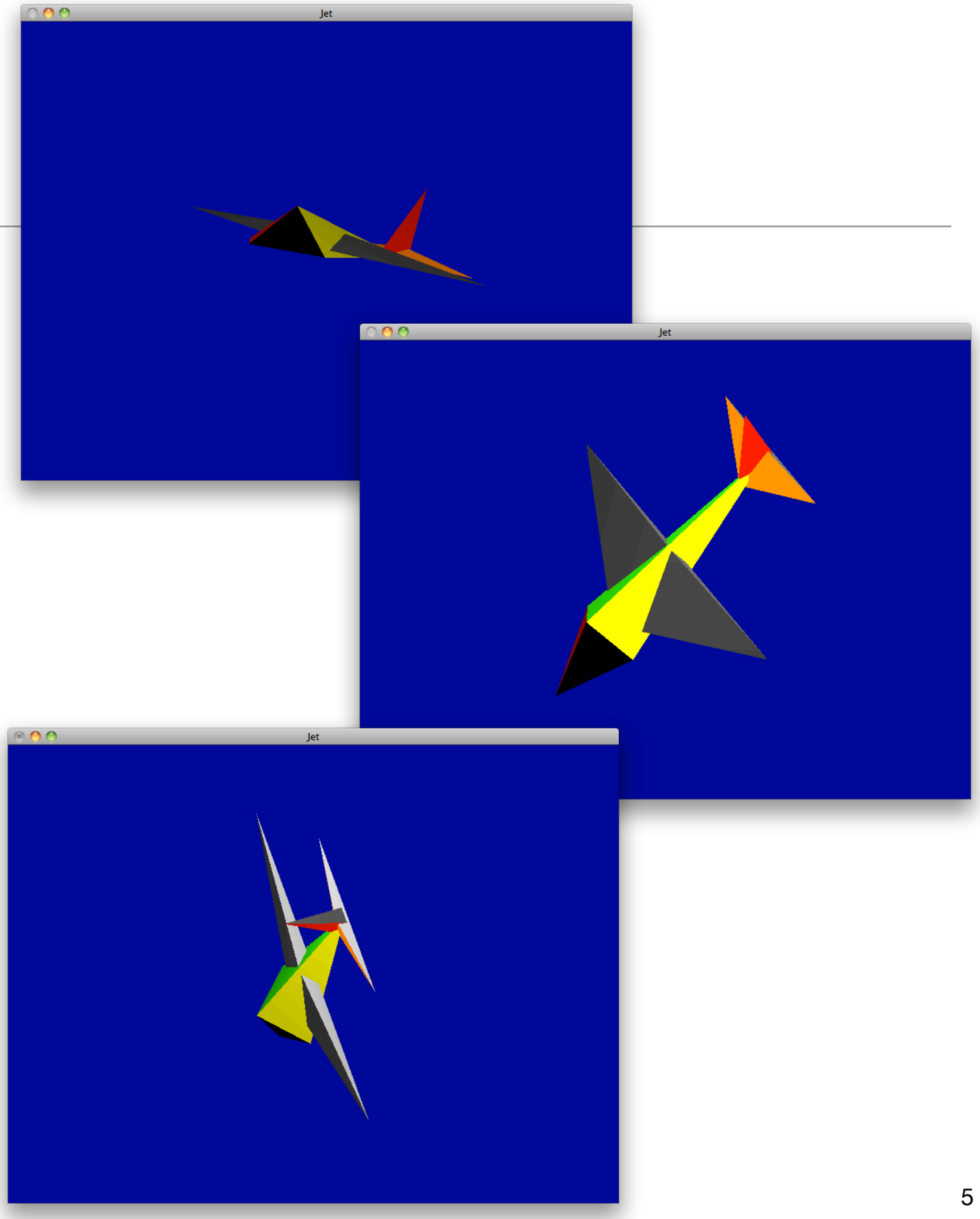
Behaviour

- Light behind left shoulder of viewer

- Scene 1 - Yellow clearly not reflecting light to full intensity

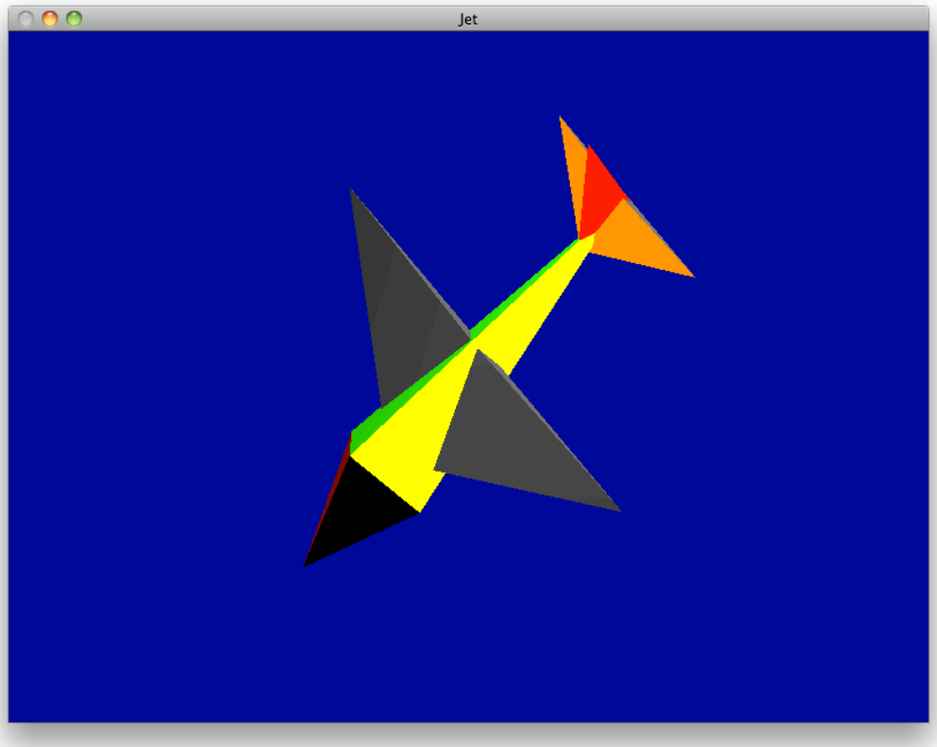
- Scene 2 - Yellow at its brightest as it is reflecting light directly at viewer

- Scene 3 - Yellow dimmer again as plane turns away



Using Normals

- The Scene requires normals to be passed into the pipeline for the correct lighting calculations to be performed.

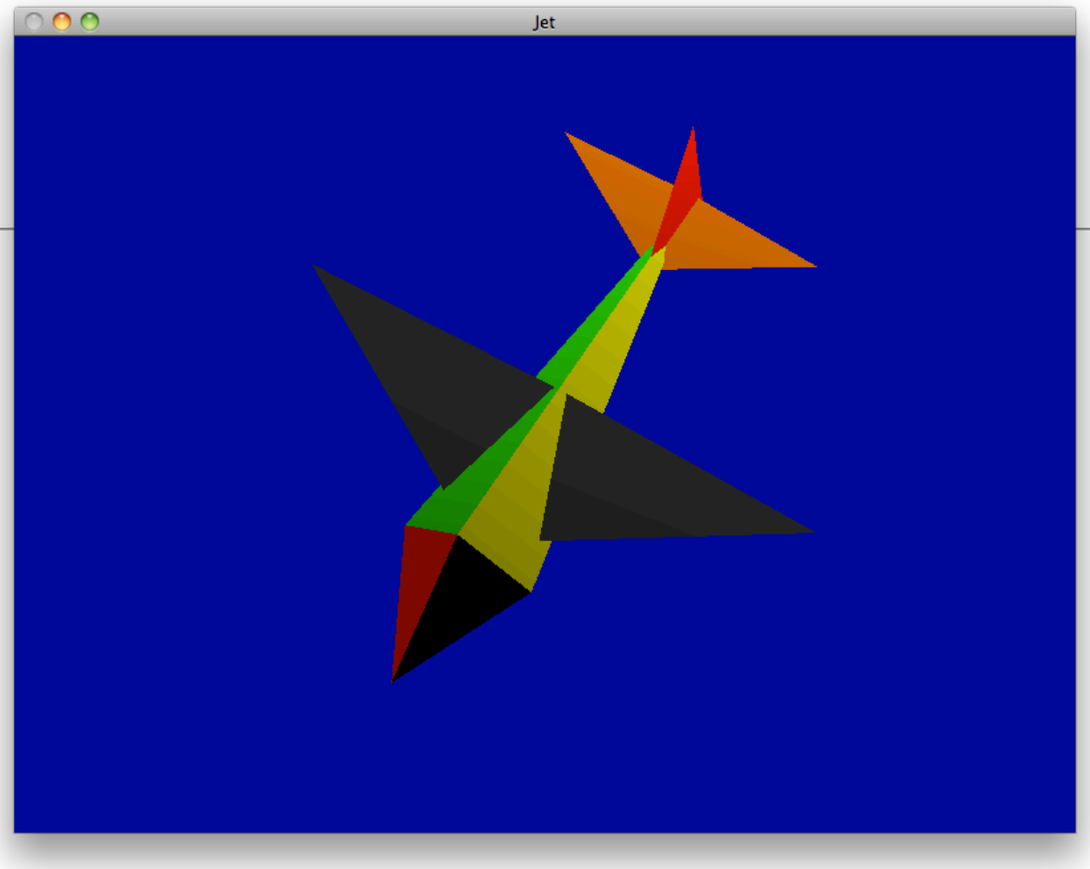



```
void render (Vector3 vectors[][3], int size)
{
  for (int i=0; i<size; i++)
  {
    glBegin(GL_TRIANGLES);
      Vector3 normal = findNormal(vectors[i][0],
                                  vectors[i][1],
                                  vectors[i][2]);
      glNormal3f(normal.X, normal.Y, normal.Z);
      vectors[i][0].render();
      vectors[i][1].render();
      vectors[i][2].render();
    glEnd();
  }
}

JetPlane::JetPlane()
{}

void JetPlane::render()
{
  glShadeModel(GL_SMOOTH);
  glPolygonMode(GL_FRONT,GL_FILL);

  Color::Yellow.render();
  ::render(noseCone, 3);
  Color::Red.render();
  ::render(body, 3);
  Color::Green.render();
  ::render(wings, 4);
  Color::Cyan.render();
  ::render(tail, 7);

  Color::White.render();
  glPolygonMode(GL_FRONT,GL_LINE);
}
```

```
glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
```

Disable Normals

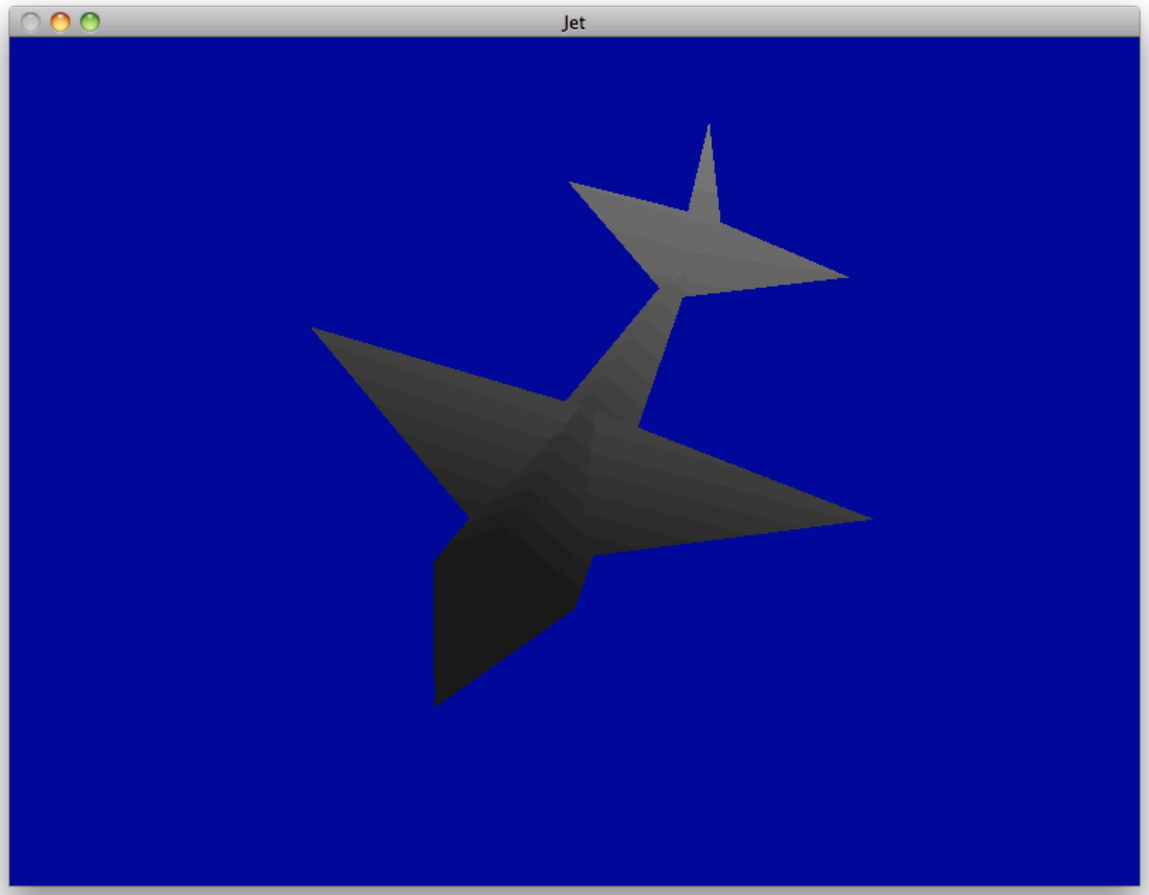
- However, if we disable normals, the colour tracking we have enabled will still render the scene reasonably well, although the lighting will be anomalous




```
void render (Vector3 vectors[][3], int size)
{
  for (int i=0; i<size; i++)
  {
    glBegin(GL_TRIANGLES);
      //Vector3 normal = findNormal(vectors[i][0],
      //                            vectors[i][1],
      //                            vectors[i][2]);
      //glNormal3f(normal.X, normal.Y, normal.Z);
      vectors[i][0].render();
      vectors[i][1].render();
      vectors[i][2].render();
    glEnd();
  }
}
```

Disable Colour Tracking AND Normals

- Although some lighting effects are apparent, the detail is completely lost.




```
glEnable(GL_LIGHTING);
glLightfv(GL_LIGHT0,GL_AMBIENT,ambientLightModerate);
glLightfv(GL_LIGHT0,GL_DIFFUSE,diffuseLightModerate);
glEnable(GL_LIGHT0);

//glEnable(GL_COLOR_MATERIAL);
//glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);

glClearColor(0.0f, 0.0f, 0.6f, 1.0f);
glEnable(GL_NORMALIZE);
```

```
void render (Vector3 vectors[][3], int size)
{
  for (int i=0; i<size; i++)
  {
    glBegin(GL_TRIANGLES);
      //Vector3 normal = findNormal(vectors[i][0],
      //                            vectors[i][1],
      //                            vectors[i][2]);
      //glNormal3f(normal.X, normal.Y, normal.Z);
      vectors[i][0].render();
      vectors[i][1].render();
      vectors[i][2].render();
    glEnd();
  }
}
```

Normals WITHOUT Colour Tracking

- The full benefit of computing normals is visible in the scene on the right

```
//glEnable(GL_COLOR_MATERIAL);
//glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
```

```
void render (Vector3 vectors[][3], int size)
{
  for (int i=0; i<size; i++)
  {
    glBegin(GL_TRIANGLES);
      Vector3 normal = findNormal(vectors[i][0],
                                  vectors[i][1],
                                  vectors[i][2]);
      glNormal3f(normal.X, normal.Y, normal.Z);
      vectors[i][0].render();
      vectors[i][1].render();
      vectors[i][2].render();
    glEnd();
  }
}
```