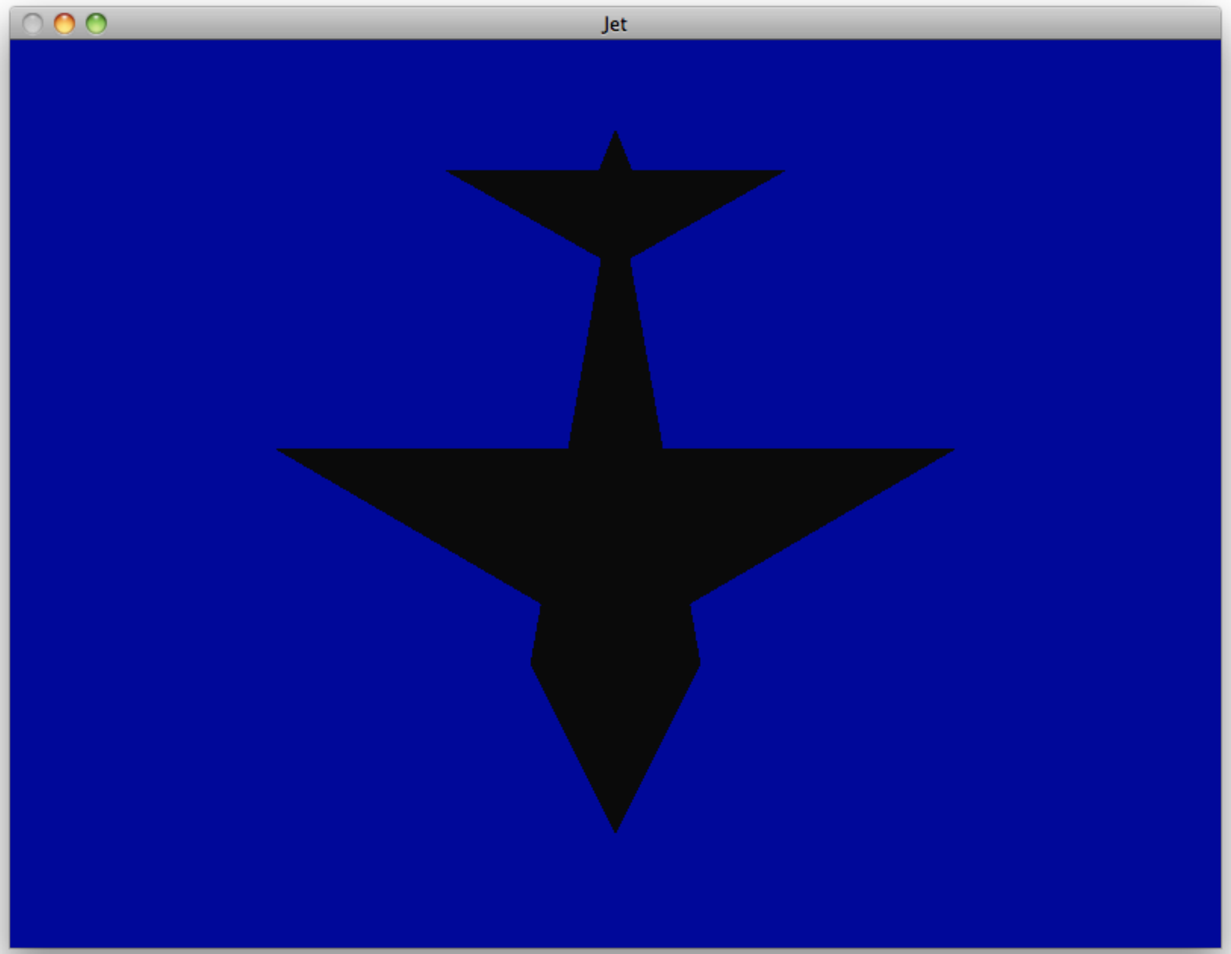
Lighting

OpenGL

Enable Lighting

```
glEnable(GL_LIGHTING);
```

- To tell OpenGL to use lighting calculations, call glEnable with the GL_LIGHTING parameter:
- This call tells OpenGL to use material properties and lighting parameters in deter-mining the color for each vertex in the scene.
- However, without any specified material properties or lighting parameters, the object remains dark and unlit



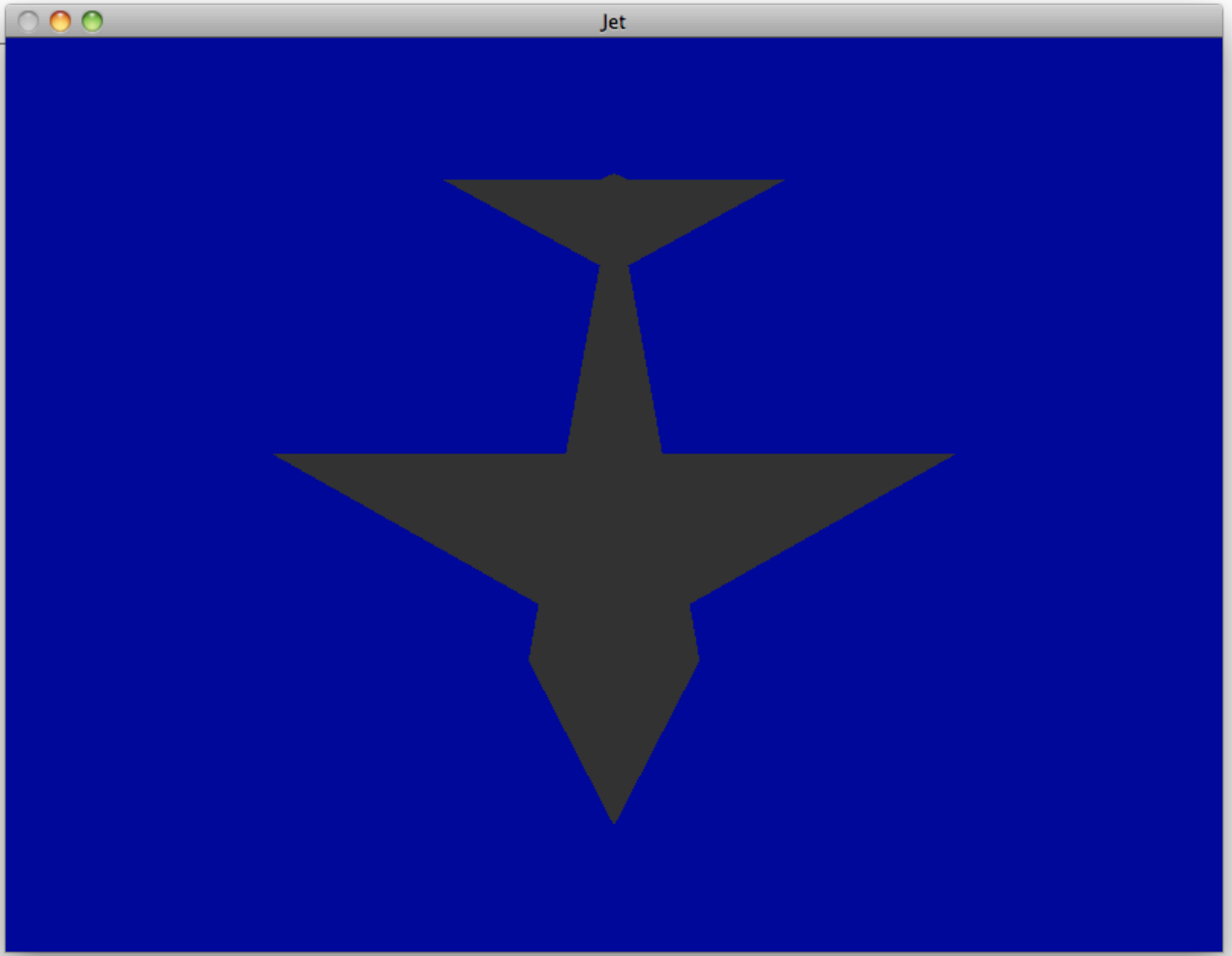
Ambient Lighting

- A zero-cost way to add a simple offset to the results of OpenGL lighting calculations.
- Can be useful to illuminate the back sides of objects that are not being illuminated directly by a light source.
- This global ambient light can be set with the glLightModel function.
- The default RGBA values of this global ambient light are (0.2, 0.2, 0.2,1.0)
- Other lighting model parameters allow you to determine whether the front, back, or both sides of polygons are illuminated and how the calculation of specular lighting angles is performed

```
GLfloat  ambientLightFull[]
 = { 1.0f, 1.0f, 1.0f, 1.0f };

glEnable(GL_LIGHTING);

glLightModelfv(GL_LIGHT_MODEL_AMBIENT,
                                ambientLightFull);
```

Ambient Default Behavior

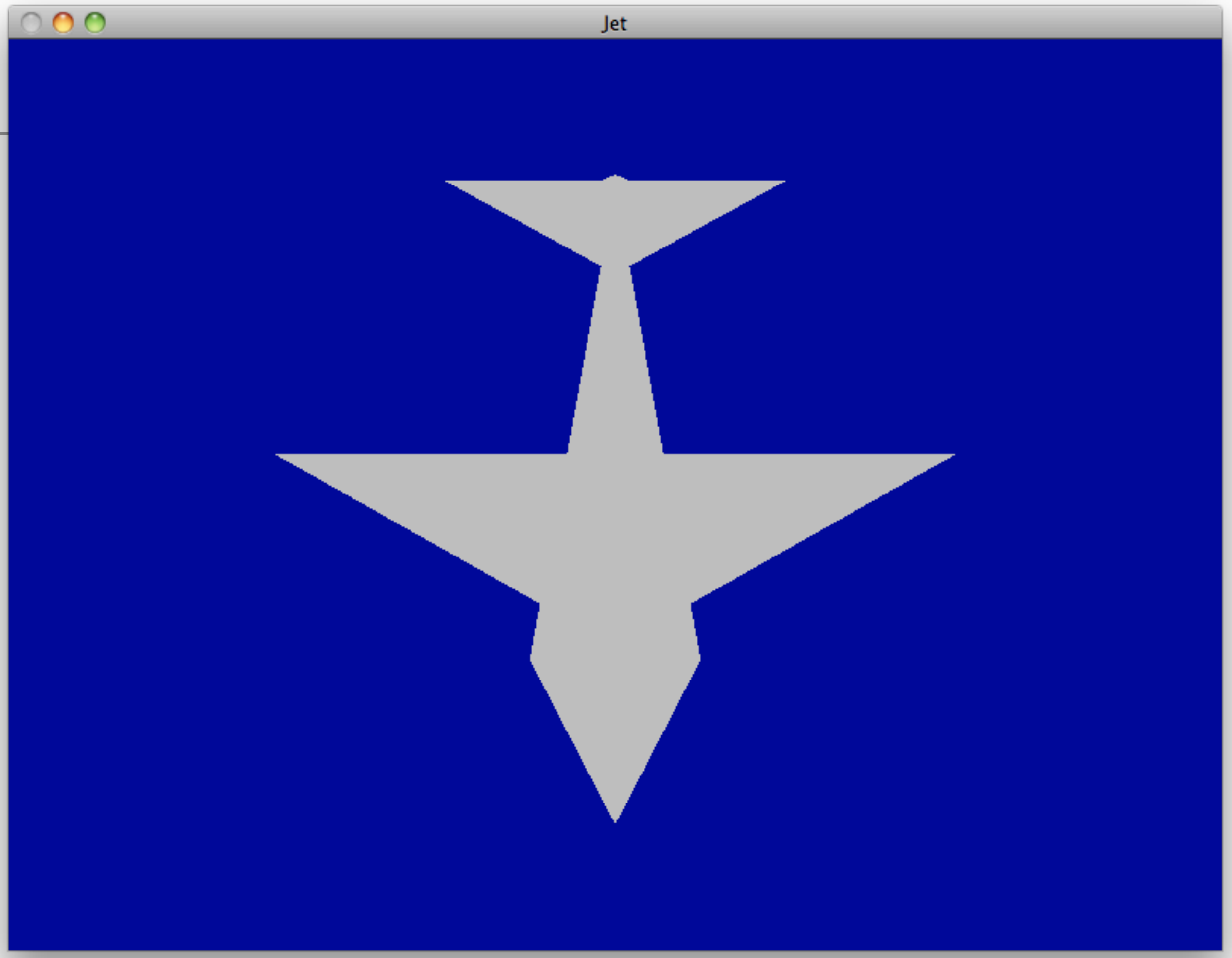
```
GLfloat  ambientLightFull[]
 = { 1.0f, 1.0f, 1.0f, 1.0f };

glEnable(GL_LIGHTING);

glLightModelfv(GL_LIGHT_MODEL_AMBIENT,
                              ambientLightFull);
```



- Full bright white ambient light
- Surfaces are still not illuminated
- By default, the lighting model expects the surfaces have material properties, upon the light will act.

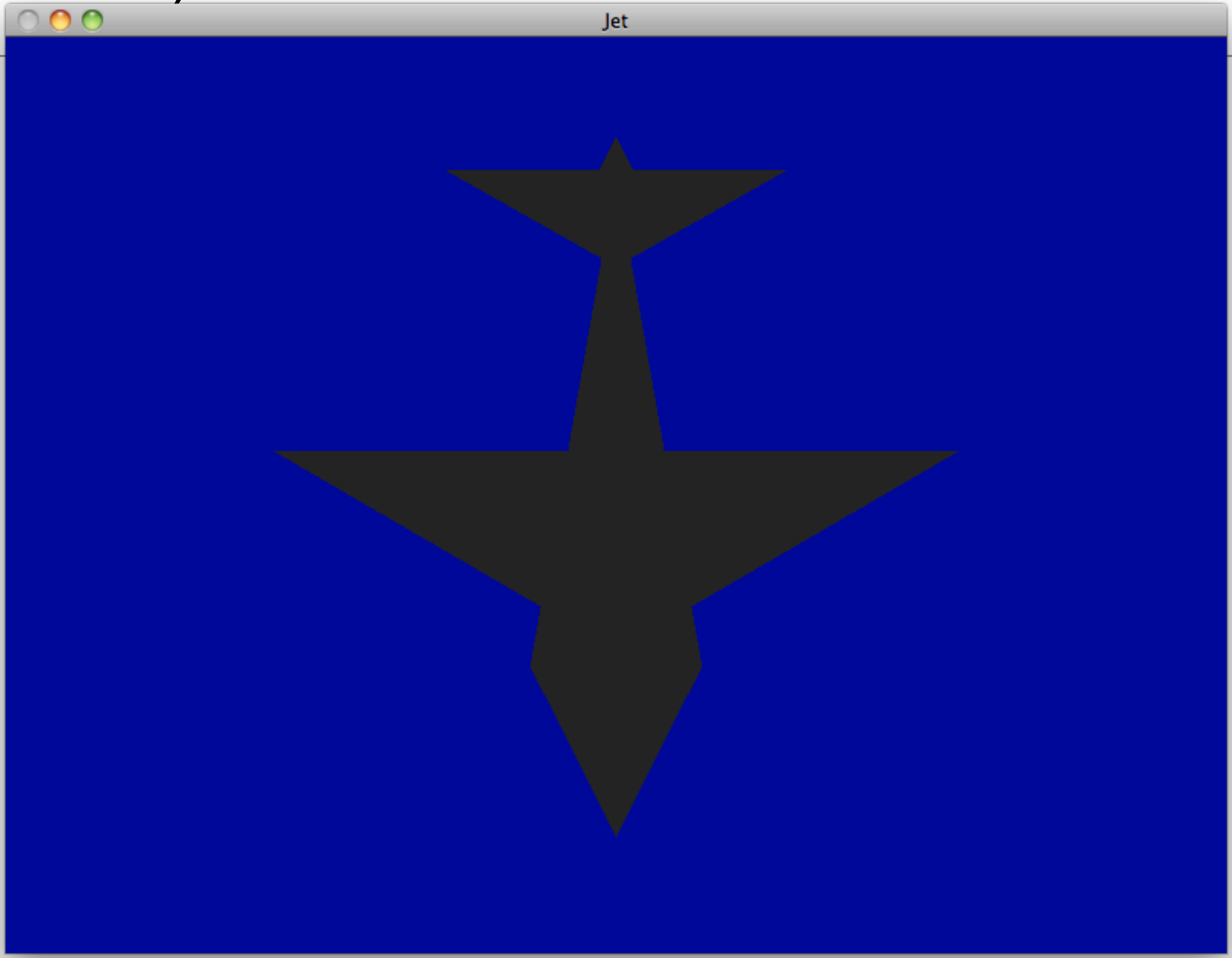
Simple Uniform Material

- Need to set material properties so the polygons reflect light
- Parameter 1 to glMaterialfv specifies whether the front, back, or both (GL_FRONT, GL_BACK, or GL_FRONT_AND_BACK) take on the material properties specified.
- Parameter 2 tells which properties are being set; in this instance, both the ambient and diffuse reflectances are set to the same values.
- Parameter 3 is an array containing the RGBA values that make up these properties.
- All primitives specified after the glMaterial call are affected by the last values set, until another call to glMaterial is made.



```
float gray[] =
          { 0.75f, 0.75f, 0.75f, 1.0f };

glMaterialfv(GL_FRONT,
             GL_AMBIENT_AND_DIFFUSE, gray);
```

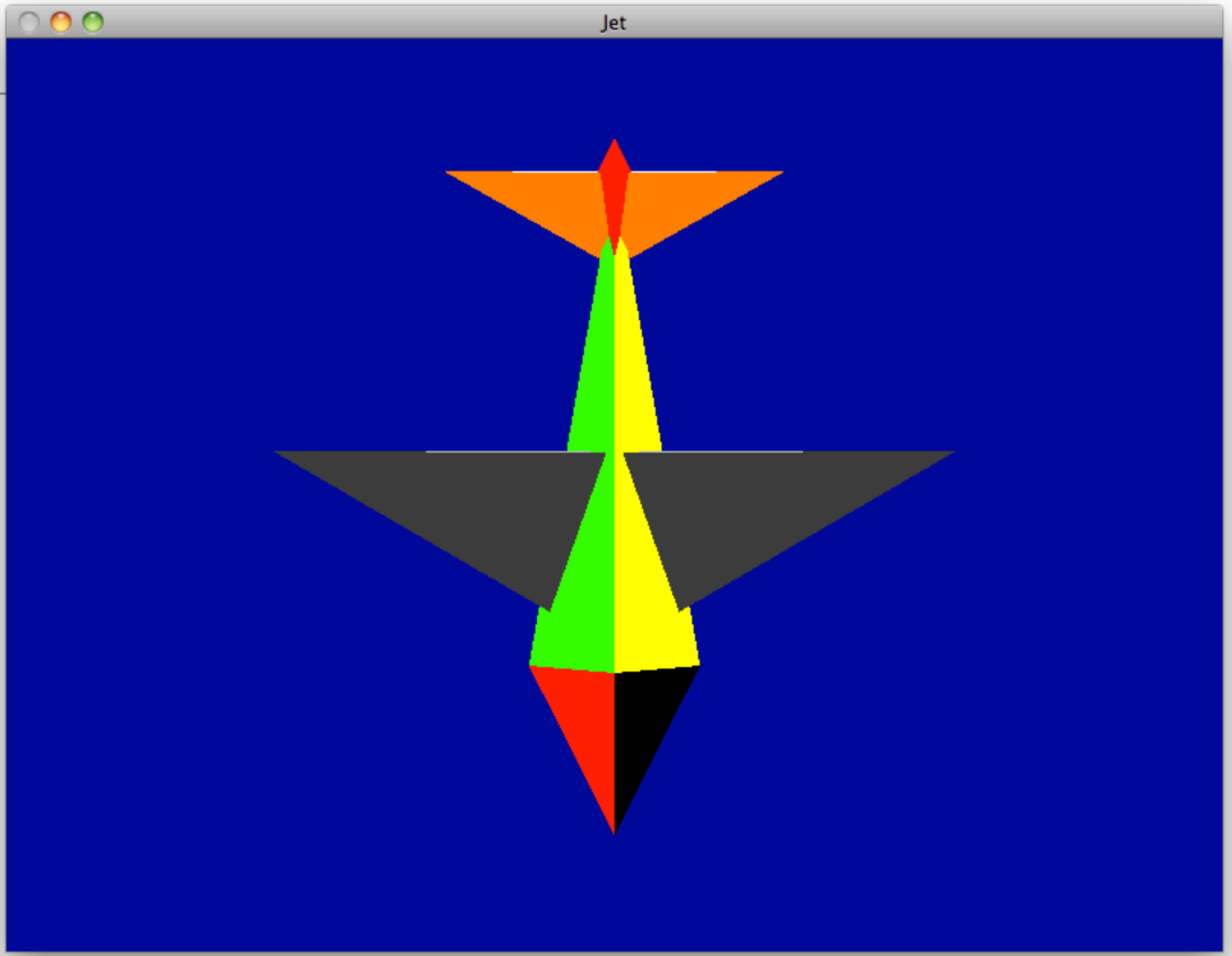
Darker (Default) Ambient



```
glEnable(GL_LIGHTING);
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLightDefault);

float gray[] = { 0.75f, 0.75f, 0.75f, 1.0f };
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, gray);
```

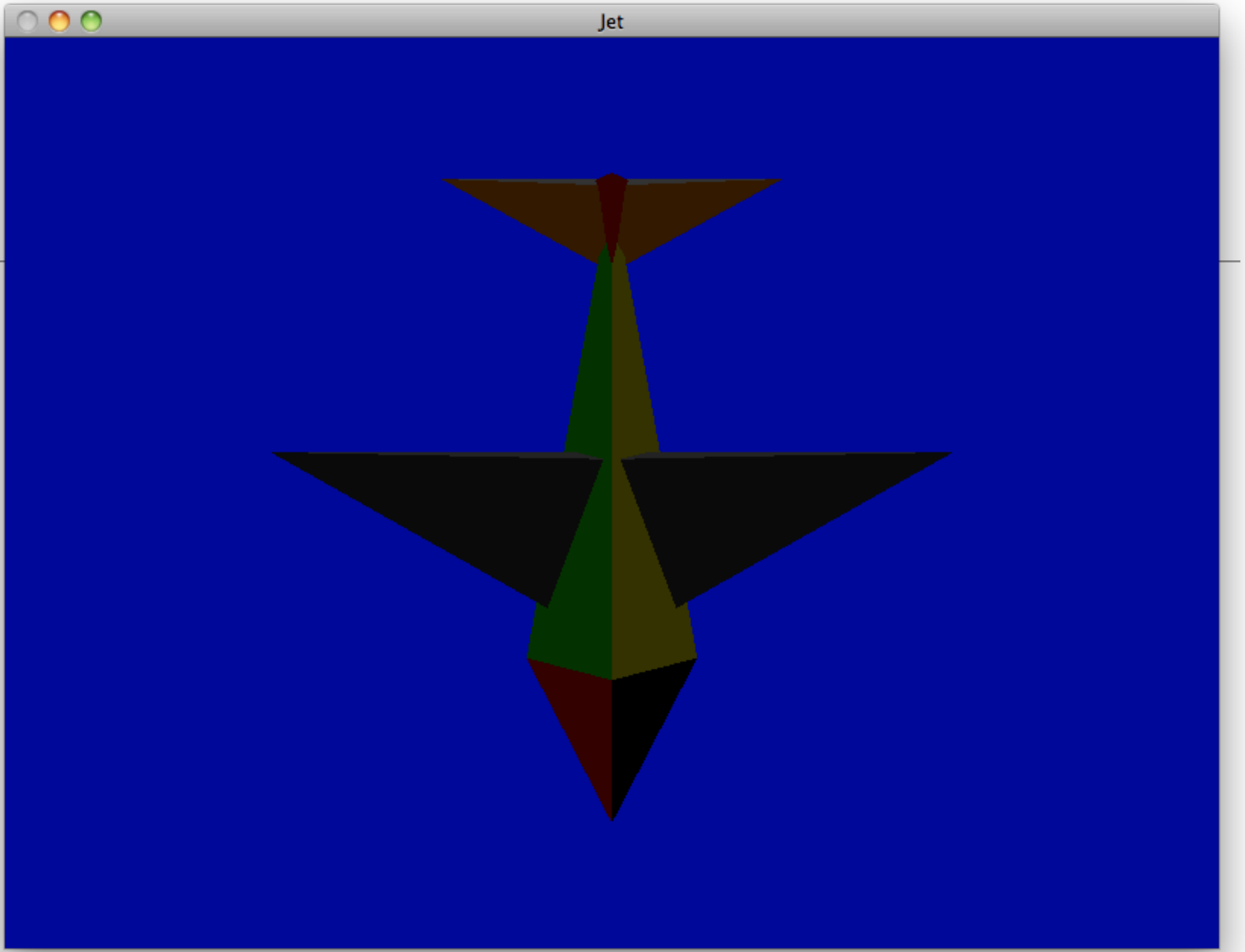
Colour Tracking

- Alternative to explicitly specifying materials.
- glEnable (GL_COLOR_MATERIAL) tells OpenGL to set material properties by via calls to glColor.
- glColorMaterial specifies the material parameters that are set by glColor.



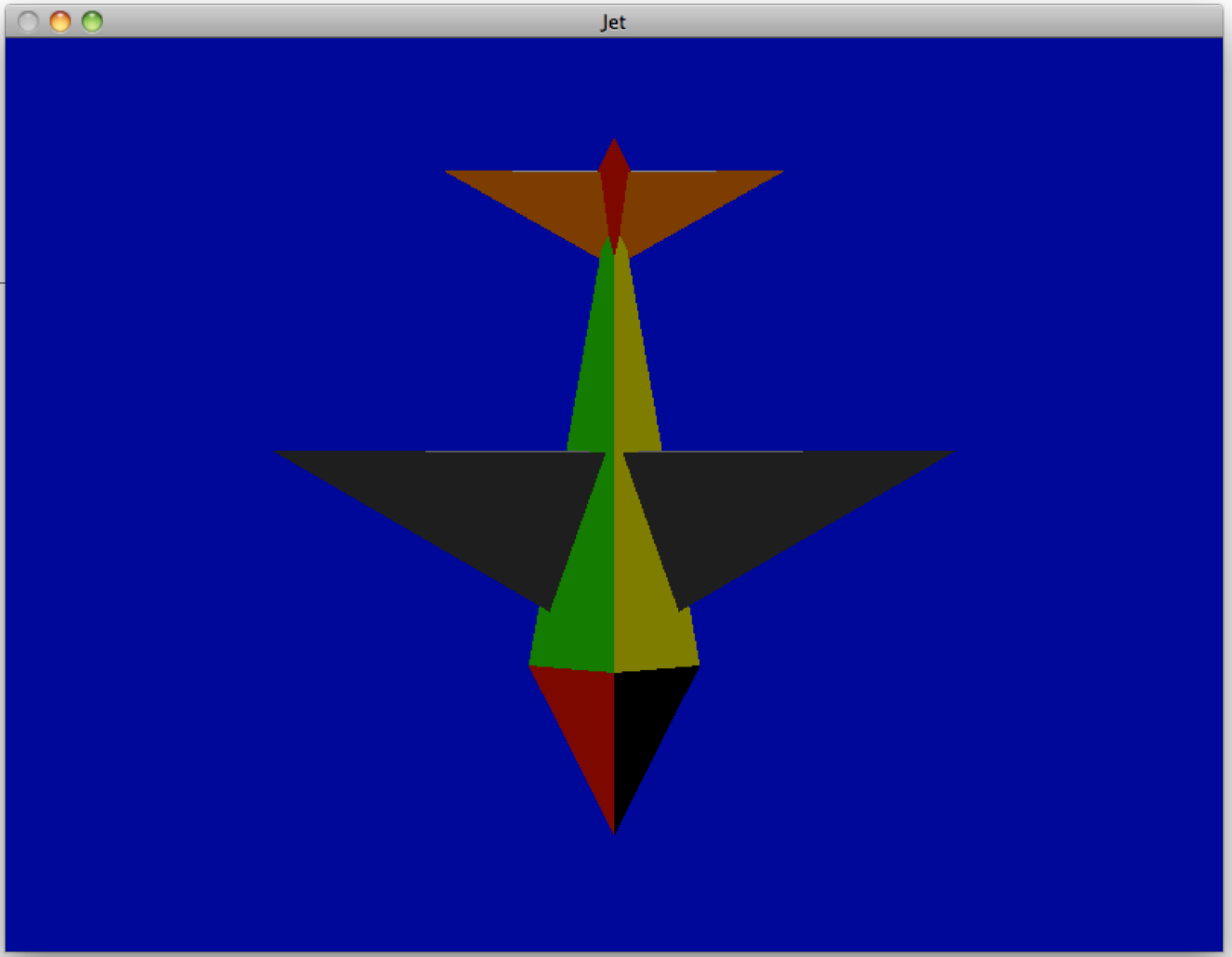
```
float  ambientLightFull[] = { 1.0f, 1.0f, 1.0f, 1.0f };

glEnable(GL_LIGHTING);
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLightFull);

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

Colour Tracking - Default Ambient Light



```
float  ambientLightDefault[] = { 0.2f, 0.2f, 0.2f, 1.0f };

glEnable(GL_LIGHTING);
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLightDefault);

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

Colour Tracking - Half Ambient Light



```
float  ambientLightHalf[] = { 0.5f, 0.5f, 0.5f, 1.0f };

glEnable(GL_LIGHTING);
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLightDefault);

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