OpenGl Course

IT ENGINEERING

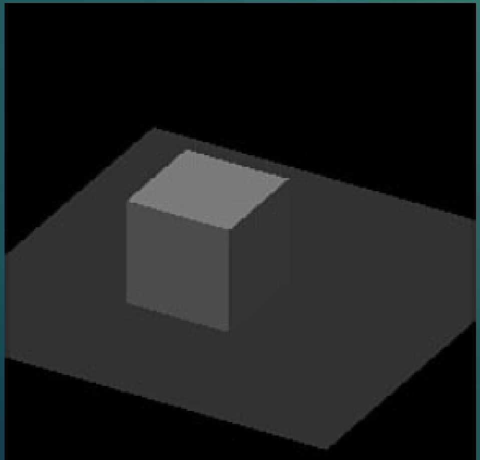
3TH YEAR

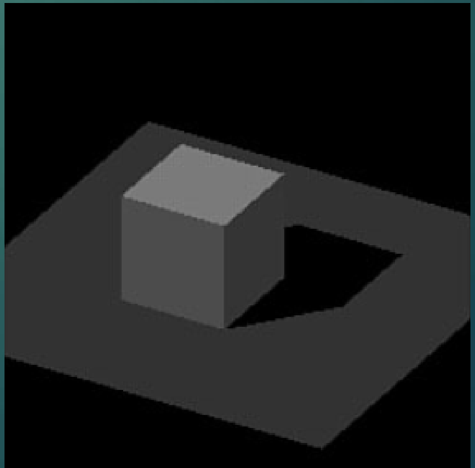
Shadow

Shadow

Why Shadow?

- A chapter on color and lighting naturally calls for a discussion of shadows. Adding shadows to your scenes can greatly improve their realism and visual effectiveness.
- In Figures below, you see two views of a lighted cube. Although both are lit, the one with a shadow is more convincing than the one without the shadow:





What's Shadow In OpenGL

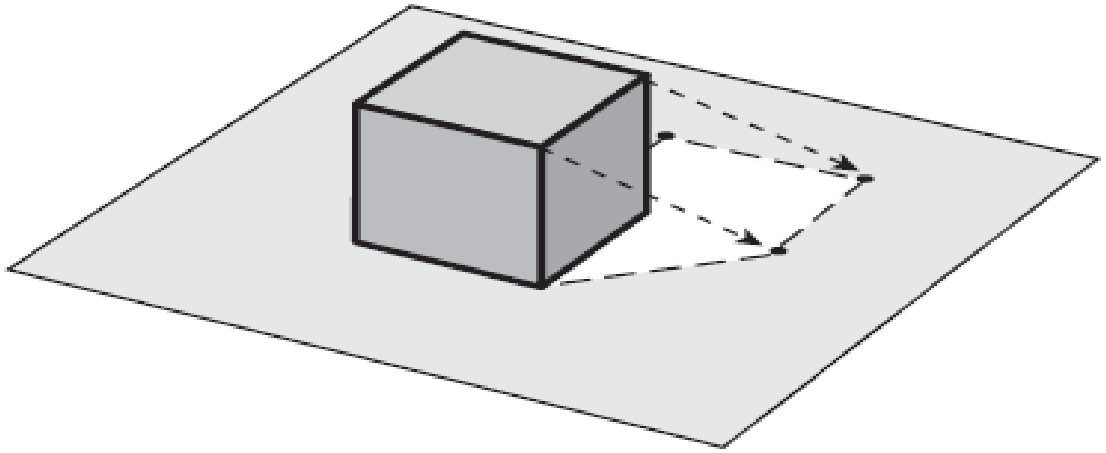
- There are two main types of graphics in general:
  - Raster graphics: which uses the idea of coloring pixels or lighting pixels when the corresponding primitive is obvious. Like OpenGL. This type of graphics never creates shadows for objects by default. But it's faster than the other.
  - Ray tracing graphics: this type of graphics uses the idea of tracing or chasing the rays of light which are reflected from objects to show the colors of shapes and objects which is affected with the colors of objects that the ray reflected from. This type of graphics gives more realistic scenes but in the other hand it's very slow. It contains shadows for objects by default.

What’s Shadow In OpenGL

Conceptually, drawing a shadow is quite simple.

A shadow is produced when an object keeps light from a light source from striking some object or surface behind the object casting the shadow. The area on the shadowed object’s surface, outlined by the object casting the shadow, appears dark. We can produce a shadow programmatically by flattening the original object into the plane of the surface in which the object lies. The object is then drawn in black or some dark color, perhaps with some translucence. There are many methods and algorithms for drawing shadows, some quite complex.

What's Shadow In OpenGL



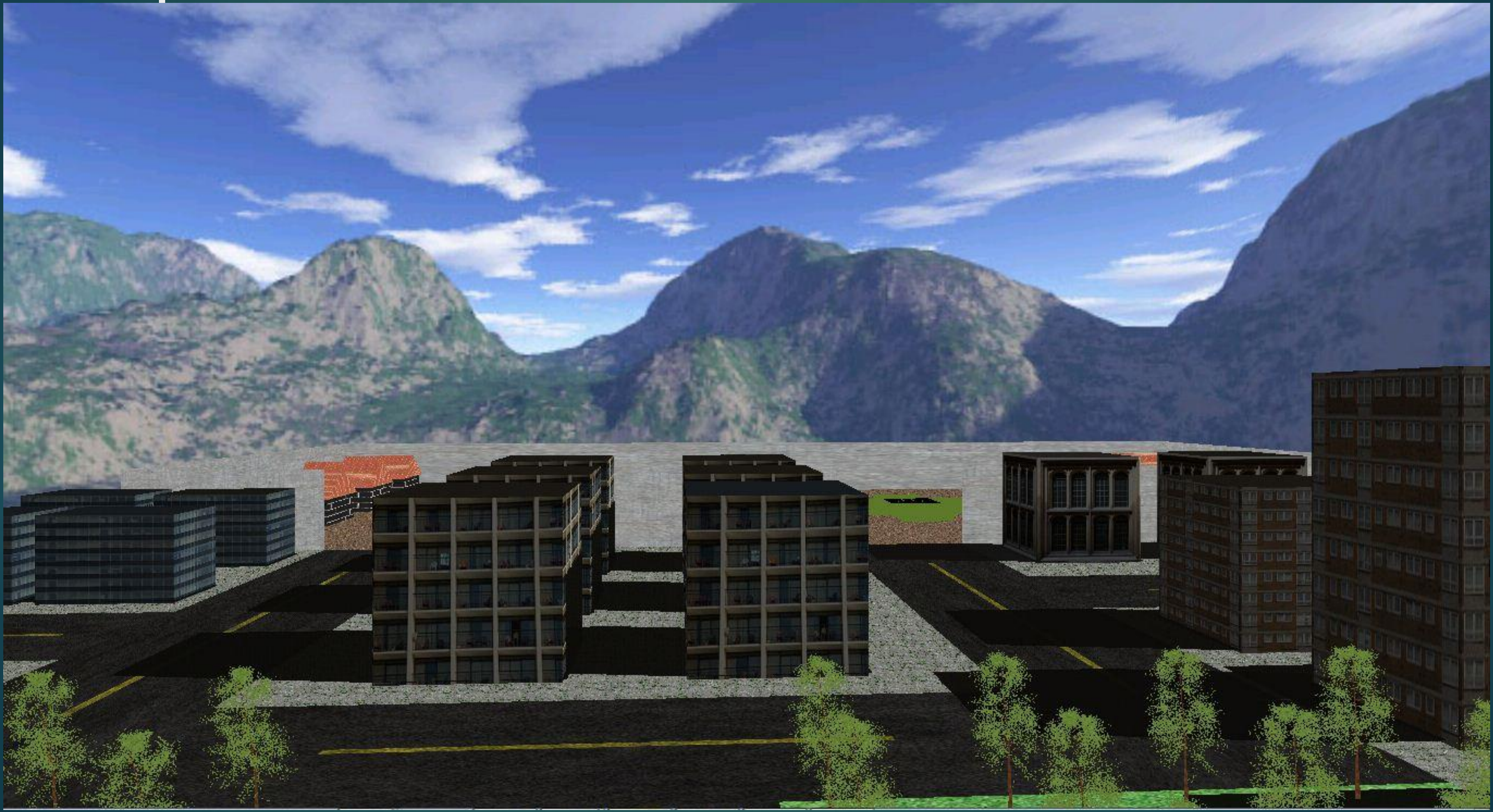
Flattening an object to create a shadow.

- We squish an object against another surface by using some of the advanced matrix manipulations. we boil down this process to make it as simple as possible.

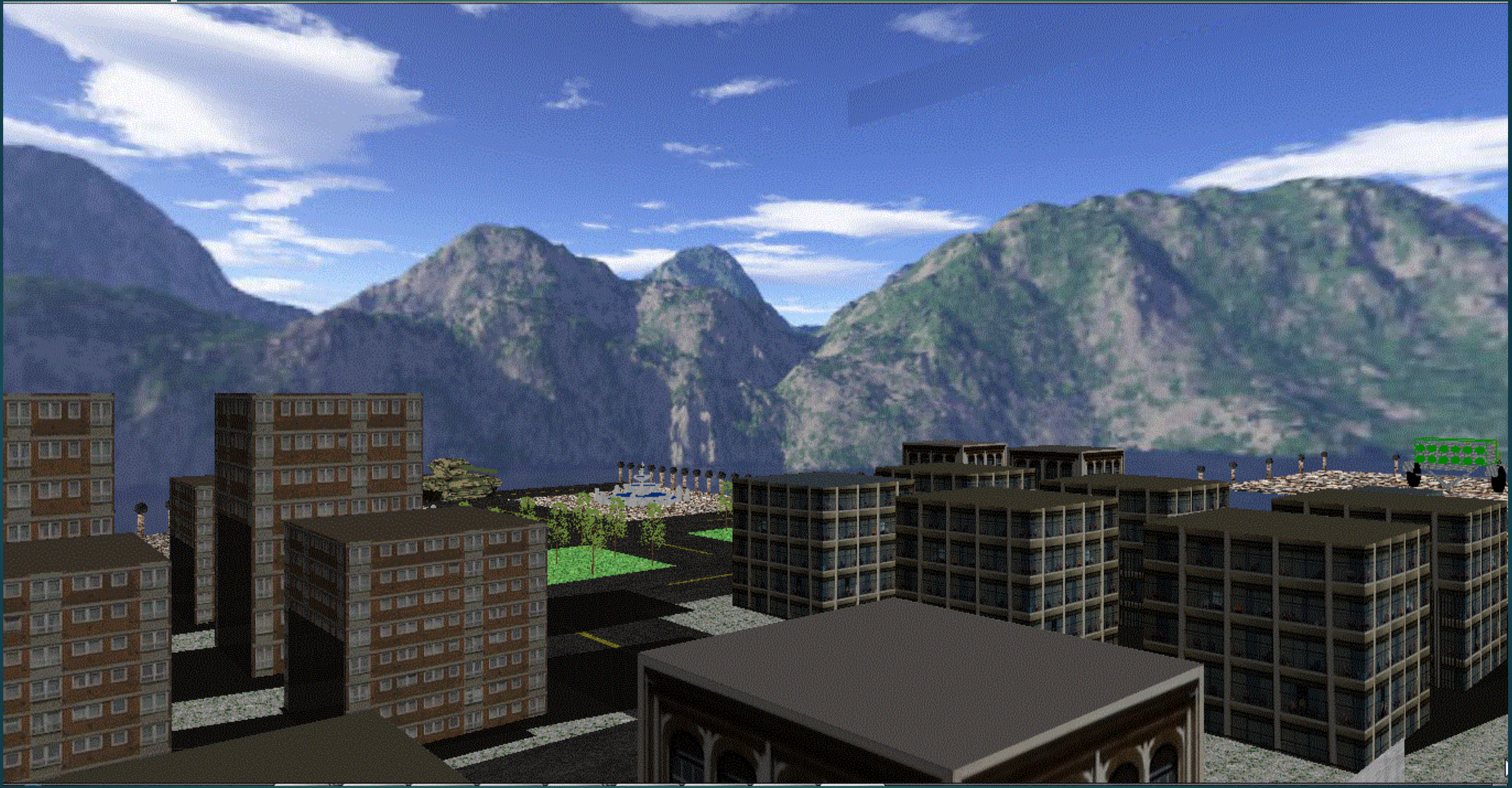
What's Shadow In OpenGL

- We need to flatten the modelview projection matrix so that any and all objects drawn into it are now in this flattened two-dimensional world. so the position of the plane is the first important think to know when we want to draw the shadow.
- The next two considerations are the distance and direction of the light source. The direction of the light source determines the shape of the shadow and influences the size.

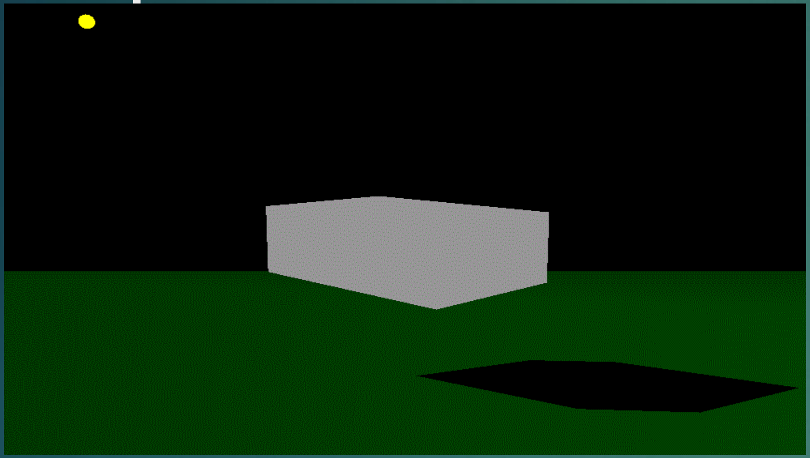
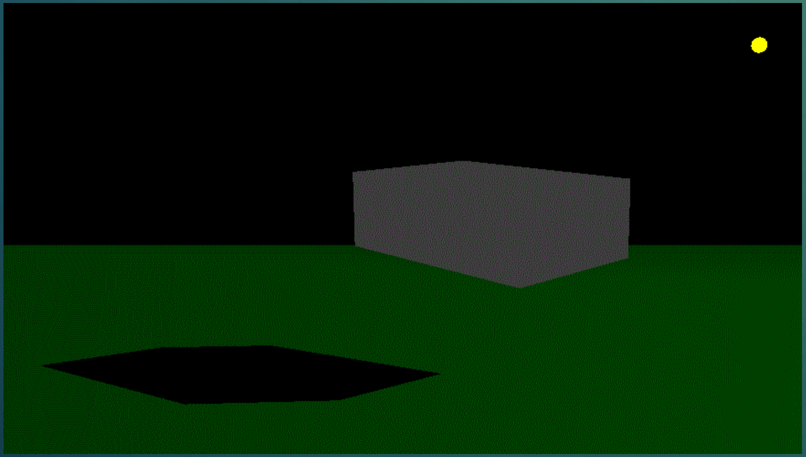
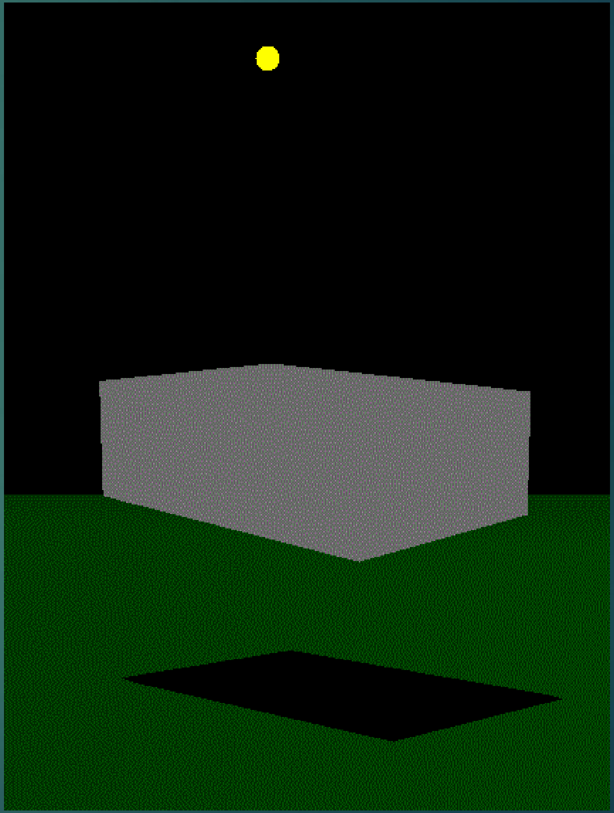
What's Shadow In OpenGL



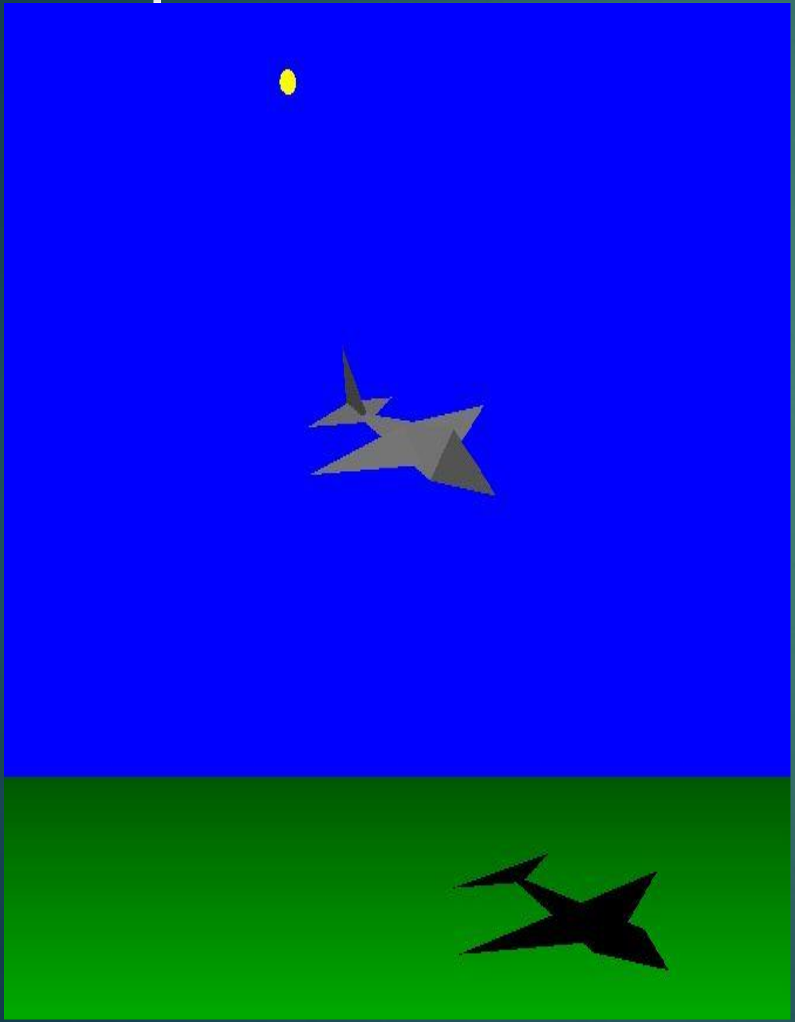
What’s Shadow In OpenGL

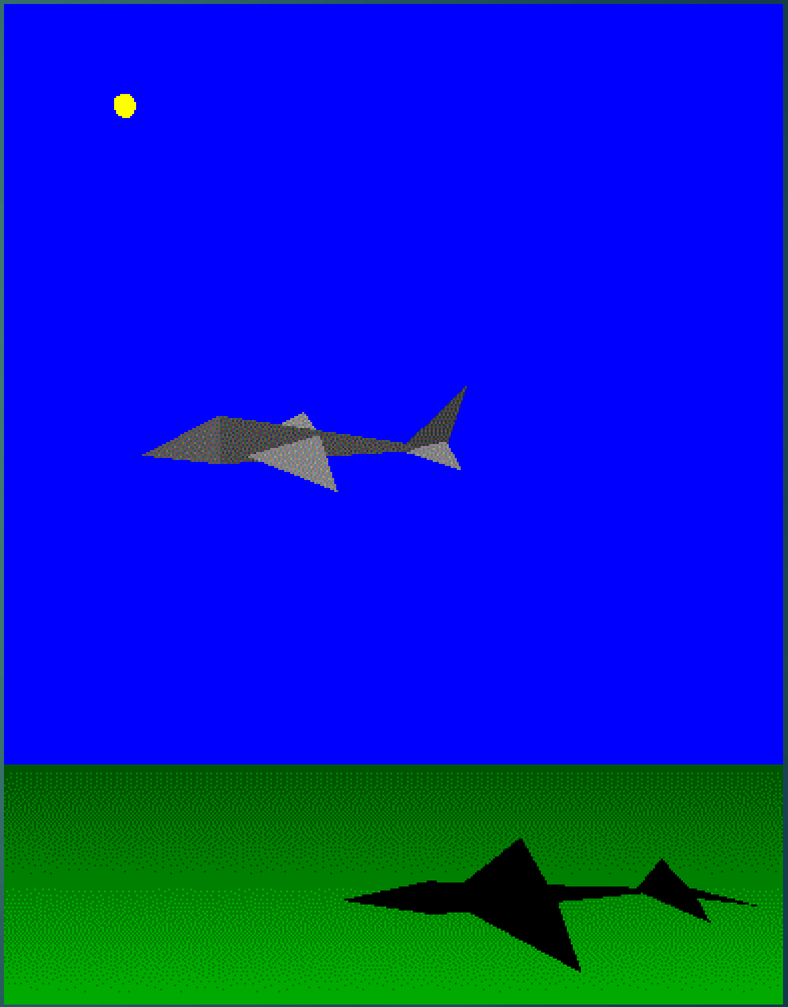


What's Shadow In OpenGL

What's Shadow In OpenGL

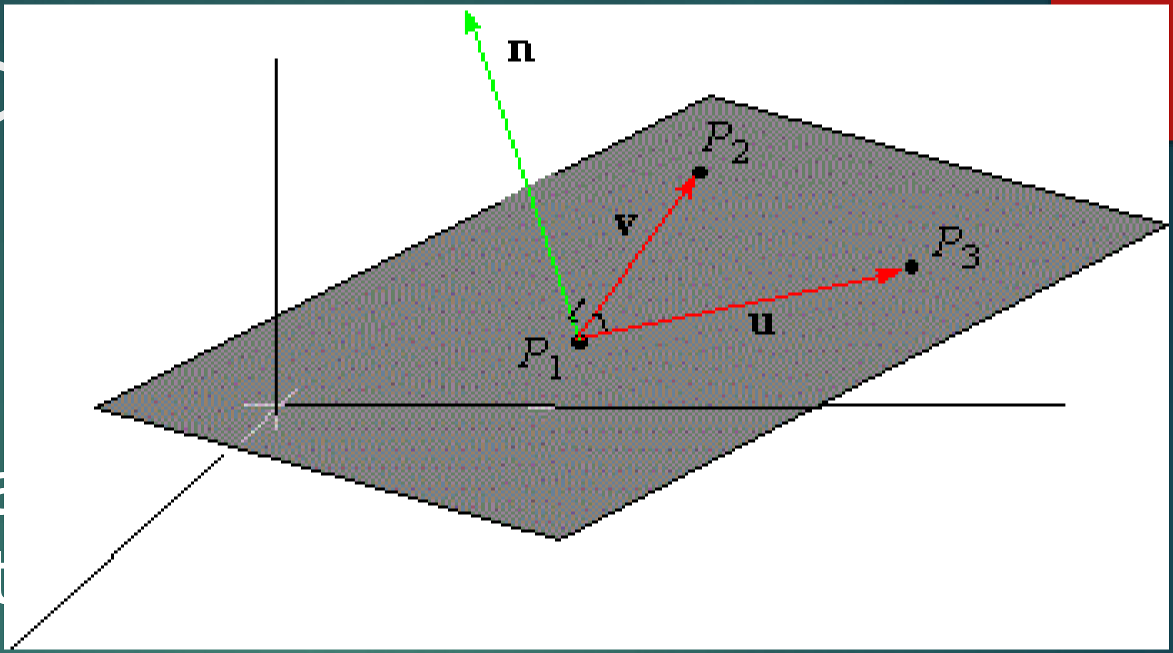




Plane Equation

Plane Equation

Plane Ec



- In order some pla its equat
- A plane in 3 dimensions can be defined by the equation:
- $$Ax + By + Cz + D = 0$$
- Examples:

Math3d Library Functions

Math3d Library Functions

Math3d Library Functions

```
m3dGetPlaneEquation(M3DVector4f vPlaneEquation,
            points[0], points[1], points[2]);
```

- We define points vector in this way:

```
M3DVector3f points[3] = {{x1, y1, z1}, {x2, y2, z2}, {x3,
y3, z3}};
```

Math3d Library Functions

- We calculate the shadow matrix with:

```
m3dMakePlanarShadowMatrix (M3DMatrix44f shadowMat,
                M3DVector4f vPlaneEquation, GLfloat lightPos[4]);
```

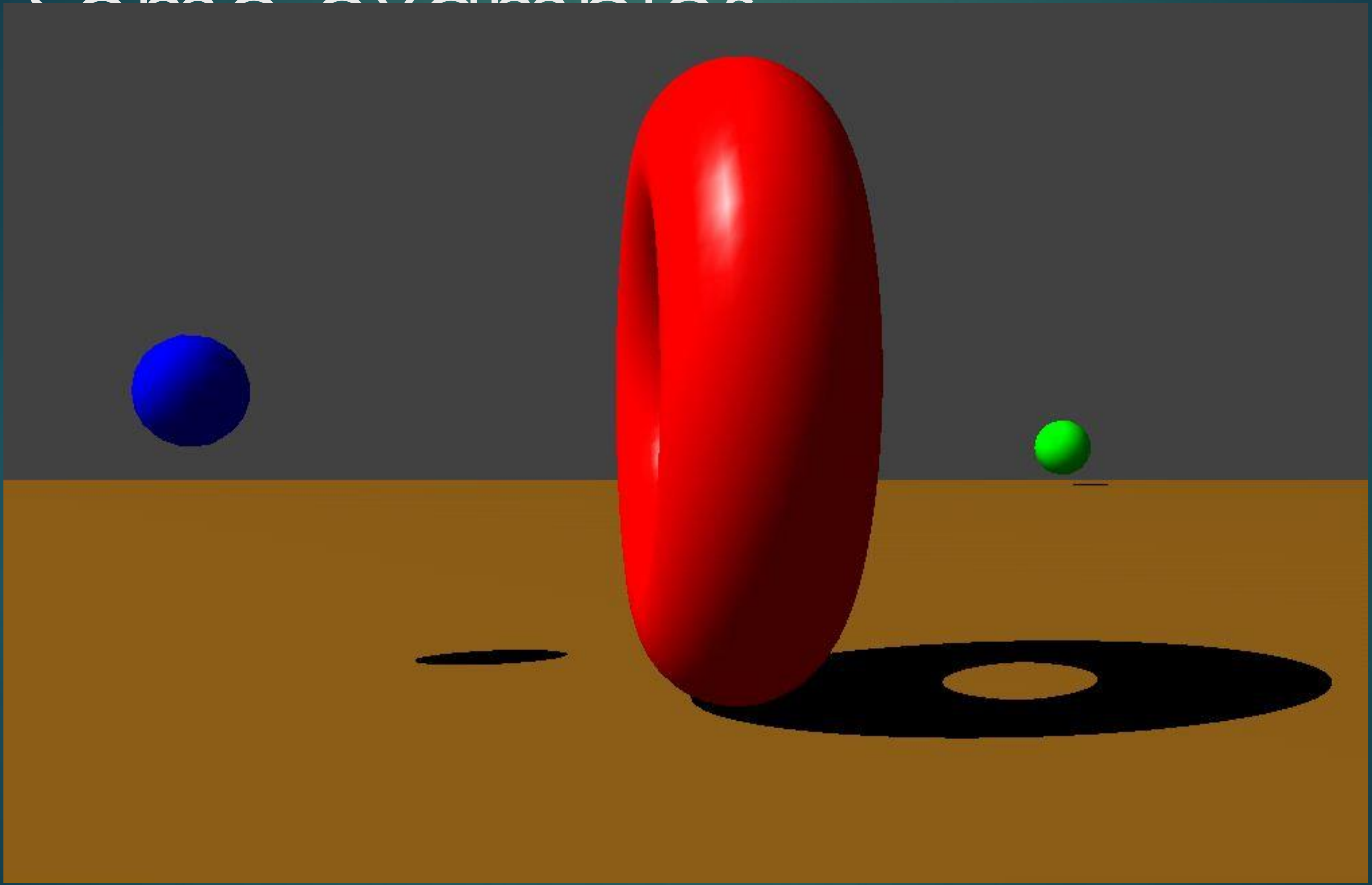
The Shadow In OpenGL

To draw the shadow ,draw the original shape ,after that multiply modelview matrix by shadow matrix.

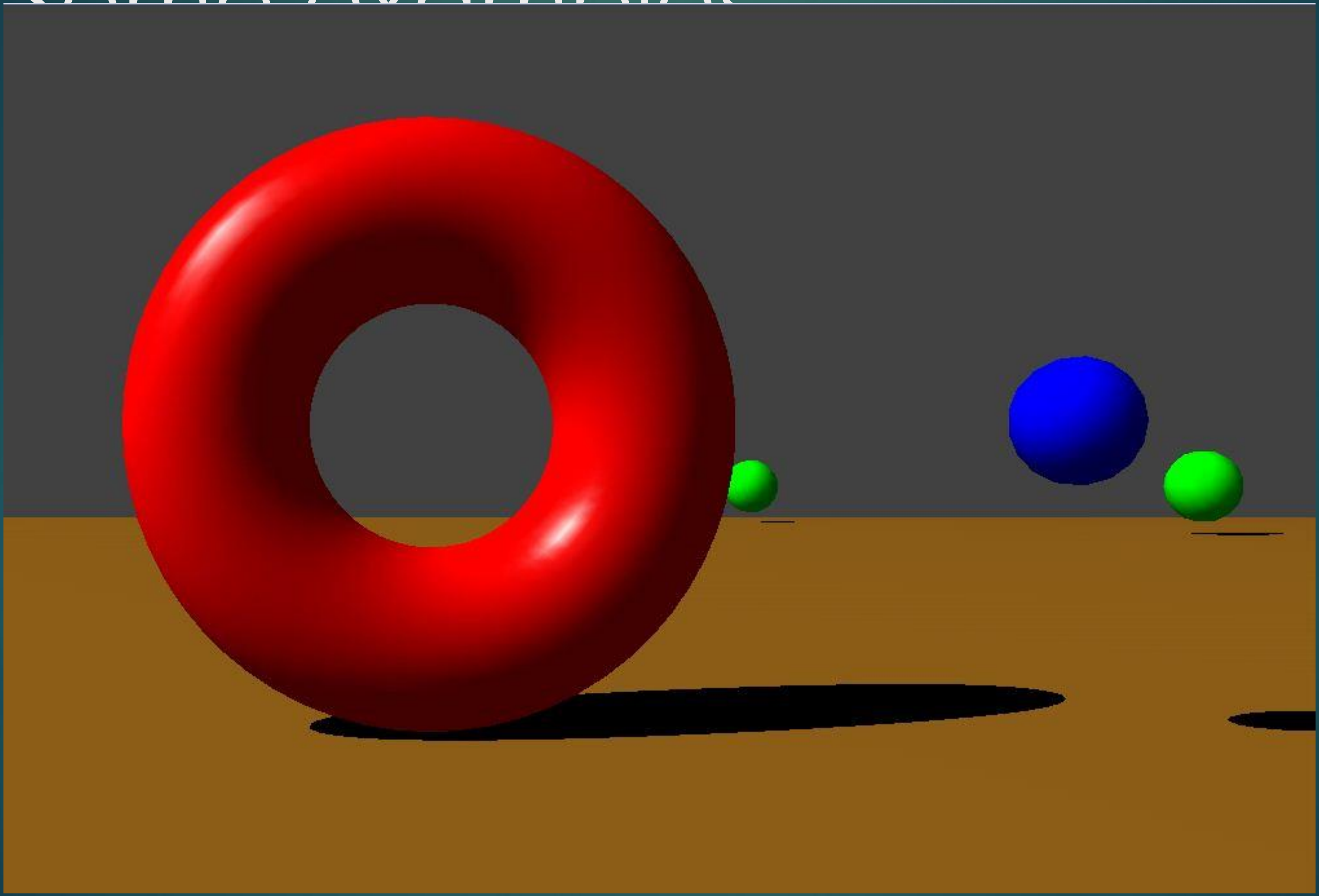
Some examples

Some examples

The Shadow In OpenGL
Some examples



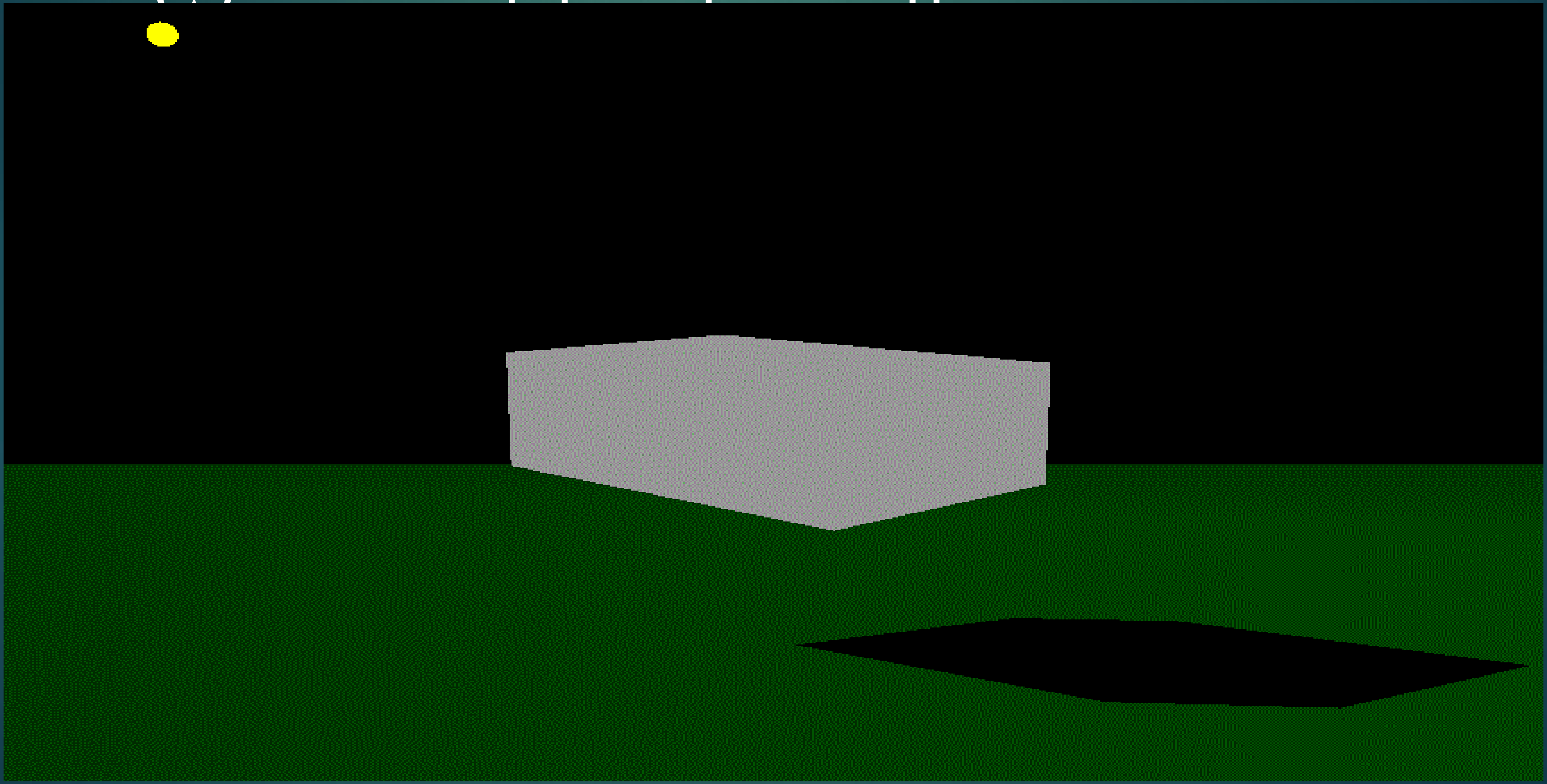
The Shadow In OpenGL
Some examples



Example

Example

Example



Example

- Firstly, we'll declare the following variables before *InitGL* function:
  - The properties of light:

```
GLfloat lightPos[] = { -1.0f, 3.0f, 0.0f, 0.0f };
GLfloat LightAmb[] = {1.0f,1.0f,1.0f,1.0f};
GLfloat LightDiff[]  = {0.6f,0.6f,0.6f,1.0f};
```

  - Transformation matrix to project shadow:

```
M3DMatrix44f shadowMat;
M3DVector4f vPlaneEquation;
```

Example

```
M3DVector3f points[3] = {{ -30.0f, -2.0f, -20.0f }, { -30.0f, -2.0f,
20.0f },
                                    { 40.0f, -2.0f, 20.0f }};
glEnable(GL_DEPTH_TEST); // Hidden surface removal
// Enable lighting
glEnable(GL_LIGHTING);
glLightfv(GL_LIGHT1,GL_POSITION,lightPos);
glLightfv(GL_LIGHT1,GL_AMBIENT,LightAmb);
glLightfv(GL_LIGHT1,GL_DIFFUSE,LightDiff);
glEnable(GL_LIGHT1);
m3dGetPlaneEquation(vPlaneEquation, points[0], points[1],
points[2]);
// Calculate projection matrix to draw shadow on the ground
m3dMakePlanarShadowMatrix(shadowMat, vPlaneEquation,
lightPos);
```

Example

```
gluLookAt(0,0,10,0,0,0,0,1,0);
glBegin(GL_QUADS); //ground
	glColor3f(0,0.2,0);
	glVertex3f(40.0f, -2.0f, -40.0f);
	glVertex3f(-40.0f, -2.0f, -40.0f);
	glColor3f(0,1,0);
	glVertex3f(-40.0f, -2.0f, 40.0f);
	glVertex3f(40.0f, -2.0f, 40.0f);
glEnd();
glPushMatrix(); //light
glTranslatef(lightPos[0],lightPos[1], lightPos[2]);
glColor3ub(255,255,0);
auxSolidSphere(0.1f);
glPopMatrix();
```

Example

- Inside *DrawGLScene* (cont'd):

```
//the original shape
glPushMatrix();
glEnable(GL_LIGHTING);
glRotatef(10, 1.0f, 0.0f, 0.0f);
glRotatef(45, 0.0f, 1.0f, 0.0f);
glColor3f(1,0,0);
Draw_Cubic(2,1,3);
glPopMatrix();
```

Example

```
glDisable(GL_DEPTH_TEST);
glDisable(GL_LIGHTING);
glPushMatrix();  //the shadow
glMultMatrixf((GLfloat *)shadowMat);
glRotatef(10, 1.0f, 0.0f, 0.0f);
glRotatef(45, 0.0f, 1.0f, 0.0f);
glColor3f(0,0,0);
Draw_Cubic(2,1,3);
glPopMatrix();
glEnable(GL_DEPTH_TEST);
```

Example

Depth testing is some means to determine

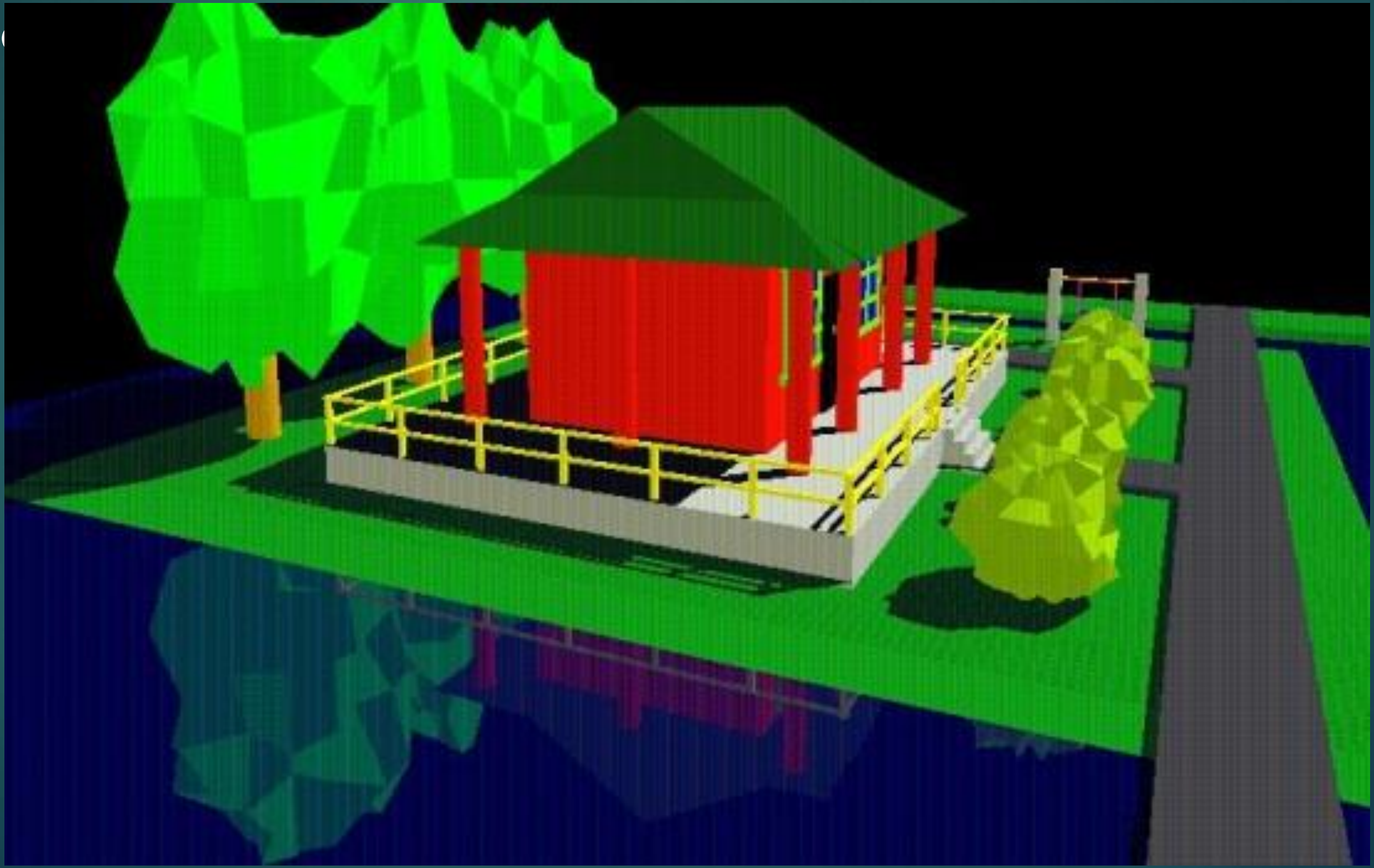
what is drawn in front of what. However, If two objects occupy the same location, usually the last one drawn is shown when depth testing is disabled.

Homework

Homework ☺

Homework ☺

- Add shadow to the house which you



The End