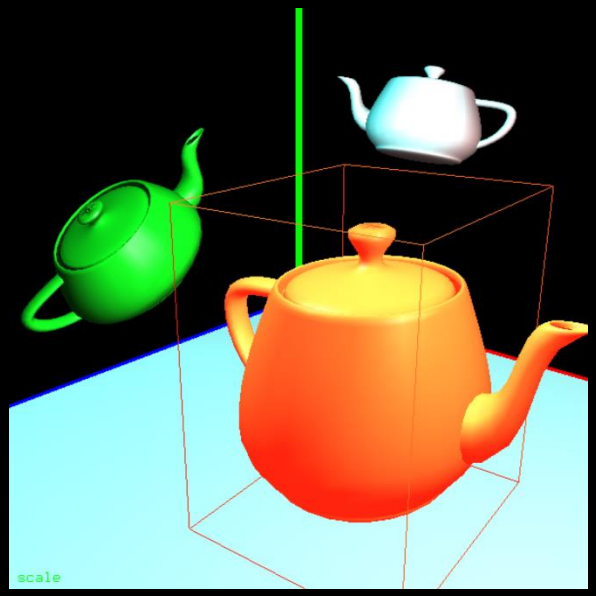
MATRIX STACKS

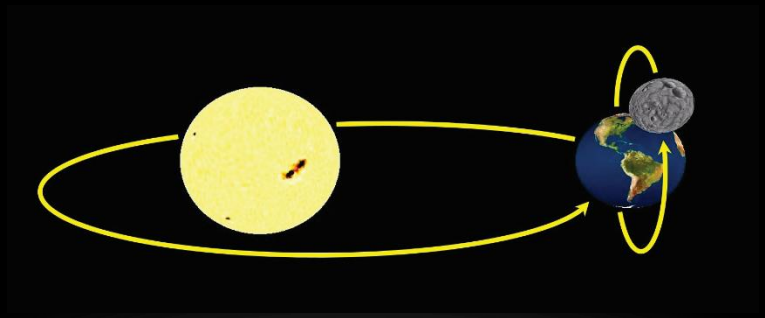


OUTLINE

- Hierarchical Models
- Z-Fighting
- Primitives Besides Triangles
- Back Face Culling

HIERARCHICAL MODELS

- Building scenes where objects move in relation to each other rather than independently
- Building complex objects from simpler shapes



MATRIX STACKS

- Stack of transformation matrices
- OpenGL built in matrix stack deprecated (of course)
- But graphicslib3D library contains MatrixStack with methods:
  - pushMatrix() – make a copy of top matrix and push on stack
  - popMatrix() – return and remove top matrix on stack
  - multMatrix(m) – multiply top matrix on stack by m
  - peek() – return top matrix, but don't remove
  - rotate(d, x, y, z) – apply to top matrix on stack
  - translate(x, y, z) – apply to top matrix on stack
  - scale(x, y, z) – apply to top matrix on stack
- Instead of wrangling multiple model and view matrices in code, can manage them all on a single stack

USING A STACK

- Let's say we want to view our scene from a particular position and watch the objects move
- Strategy:
  - Set our camera location
    - Push a matrix (identity) onto the stack and multiply it by the view matrix
  - Create the first object and its model matrix
    - Do a push (duplicates the previous matrix) and multiply it by the model matrix
  - Create the next object and model matrix
    - Do a push and multiply it by the model matrix
  - You probably see the pattern here…

USING A STACK

- Now, when we draw each object:
  - Call glDrawArrays to draw the object
  - Pop the appropriate matrices off the stack
  - Call glDrawArrays to draw the next object
  - Pop the appropriate matrices off the stack
  - …
  - Another pattern…

JAVA JOGL CODE – IN DISPLAY() (1/4)

```
// push view matrix onto the stack
mvStack.pushMatrix();
mvStack.translate(-cameraX, -cameraY, -cameraZ);
double amt = (double)(System.currentTimeMillis())/1000.0;

// Assuming we have the location of proj_loc already
gl.glUniformMatrix4fv(proj_loc, 1, false,
                      pMat.getFloatValues(), 0);
```

JAVA JOGL CODE – IN DISPLAY() (2/4)

```
// ----------------------  pyramid == sun
mvStack.pushMatrix();
mvStack.translate(pyrLocX, pyrLocY, pyrLocZ);
mvStack.pushMatrix();
mvStack.rotate((System.currentTimeMillis())/10.0,1.0,0.0,0.0);
gl.glUniformMatrix4fv(mv_loc, 1, false,
mvStack.peek().getFloatValues(), 0);
gl.glBindBuffer(GL_ARRAY_BUFFER, vbo[1]);
gl.glVertexAttribPointer(0, 3, GL_FLOAT, false, 0, 0);
gl.glEnableVertexAttribArray(0);
gl.glEnable(GL_DEPTH_TEST);
gl.glDrawArrays(GL_TRIANGLES, 0, 18);
mvStack.popMatrix();
// Note: we only pop the rotation matrix, not translation
```

JAVA JOGL CODE – IN DISPLAY() (3/4)

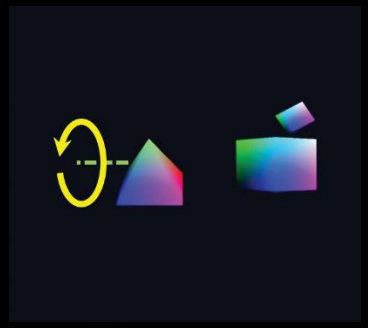
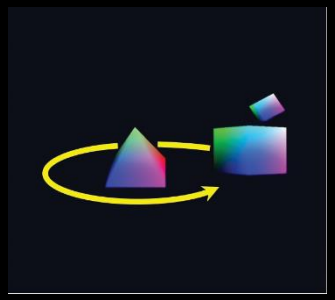
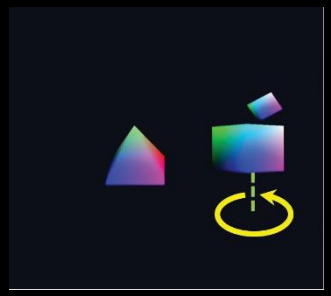
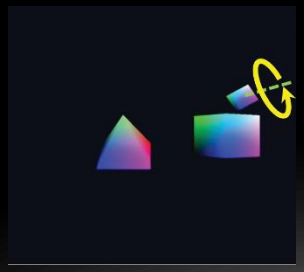
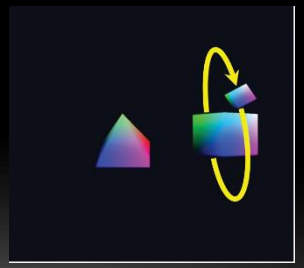
```
//-----------------------  cube == planet
mvStack.pushMatrix();
mvStack.translate(Math.sin(amt)*4.0f, 0.0f,
Math.cos(amt)*4.0f);
mvStack.pushMatrix();
mvStack.rotate((System.currentTimeMillis())/10.0,0.0,1.0,0.0);
gl.glUniformMatrix4fv(mv_loc, 1, false,
mvStack.peek().getFloatValues(), 0);
gl.glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
gl.glVertexAttribPointer(0, 3, GL_FLOAT, false, 0, 0);
gl.glEnableVertexAttribArray(0);
gl.glDrawArrays(GL_TRIANGLES, 0, 36);
mvStack.popMatrix();
// Again, only pop the rotation matrix
```

JAVA JOGL CODE – IN DISPLAY() (4/4)

```
//-----------------------  smaller cube == moon
mvStack.pushMatrix();
mvStack.translate(0.0f, Math.sin(amt)*2.0f, Math.cos(amt)*2.0f);
mvStack.rotate((System.currentTimeMillis())/10.0,0.0,0.0,1.0);
mvStack.scale(0.25, 0.25, 0.25);
gl.glUniformMatrix4fv(mv_loc, 1, false,
mvStack.peek().getFloatValues(), 0);
gl.glBindBuffer(GL_ARRAY_BUFFER, vbo[0]);
gl.glVertexAttribPointer(0, 3, GL_FLOAT, false, 0, 0);
gl.glEnableVertexAttribArray(0);
gl.glDrawArrays(GL_TRIANGLES, 0, 36);
// Here's where we clean up the stack by popping all
mvStack.popMatrix();  mvStack.popMatrix();  mvStack.popMatrix();
mvStack.popMatrix();
```

USING A STACK

- Since the stack code is all in display(), it gets called each time the frame changes
  - So these matrices are different for each new change

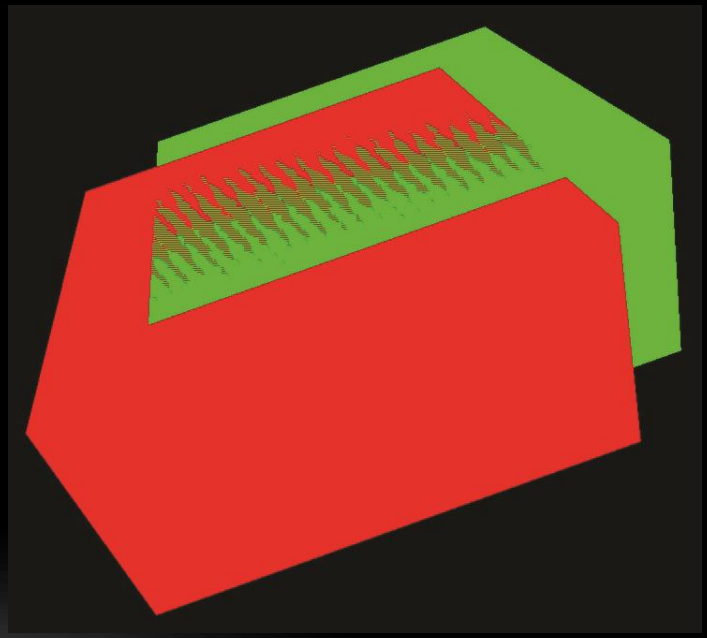






SO.. WHY AREN'T WE DOING THIS WITH SPHERES?

- Well… we haven't looked at how do create sphere objects yet
  - First, sphere vertices are more elegantly defined by a mathematical function than by listing vertices
  - Second, we need to worry about curved surfaces
    - Otherwise our sphere will look faceted, not smooth
  - We will get to all of that soon

Z-FIGHTING

- When the depth (z axis) of two objects is similar, the display can get confused
- Often seen in terrain and shadows
- Caused by floating point rounding error and limited precision in the of z-buffer values

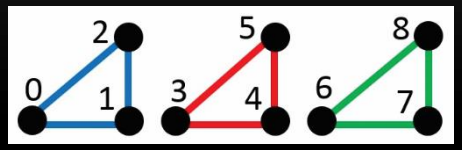


Z-FIGHTING

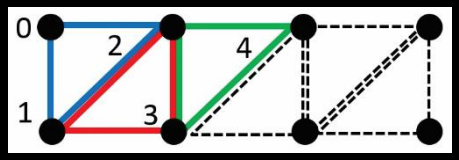
- Some solutions:
  - Move objects apart slightly
  - Don’t make the near clipping plane too close to the camera

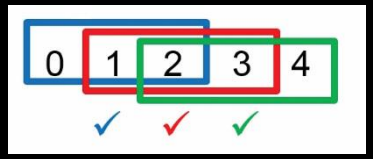
OPENGL PRIMITIVES AND THEIR INTERPRETATIONS

- GL_TRIANGLES

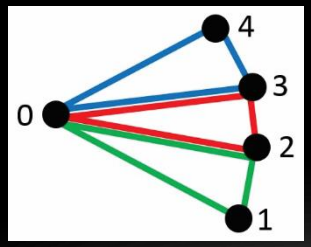


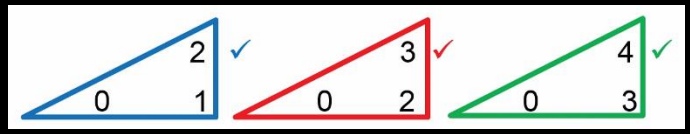
- GL_TRIANGLE_STRIP





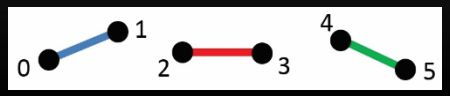
- GL_TRIANGLE_FAN

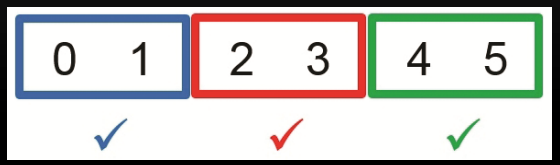




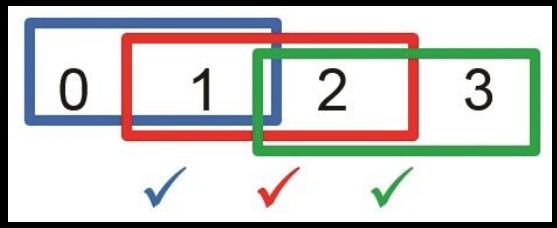
OPENGL PRIMITIVES AND THEIR INTERPRETATIONS

- GL_LINES



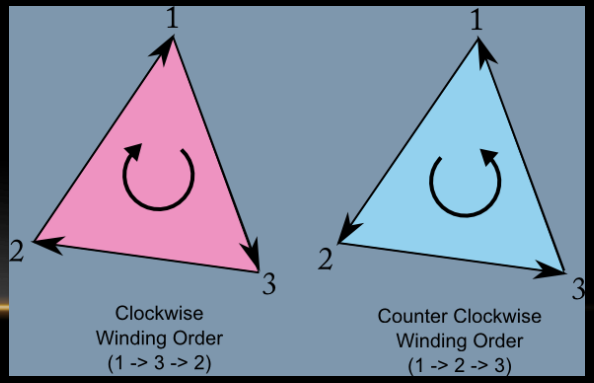


- GL_LINE_STRIP



BACK FACE CULLING

- Affects rendering efficiency – in a good way
- Every primitive we draw has a front face and a back face
  - When we construct a cube out of triangles, only the front face of each triangle shows
    - The cube is a closed object
  - But OpenGL is still interpolating colors, etc. for the back face
- We can enable back face culling by:
  - `glEnable(GL_CULLFACE);`
- We may also need to tell OpenGL how to tell what is a back face:
  - `gl_FrontFace(GL_CCW);`
- Default is CCW – counter clockwise – that is, if the vertices progress in a counter clockwise order
  - Called the winding order



BACK FACE CULLING

- If using back face culling, want to make your objects so that triangles defining the exterior are the same winding order
  - Usually counter-clockwise
- May use culling in interesting ways
  - With transparent objects, may want to see inside, not outside
- Probably don’t want to use back face culling in objects that are not closed…

SUMMARY

- Hierarchical Models
- Z-Fighting
- Primitives Besides Triangles
- Back Face Culling