http://upload.wikimedia.org/wikipedia/commons/b/b9/JOGL_Refrection_Demo_Screenshot.png

The OpenGL Pipeline

CS 148, Summer 2012
Introduction to Computer Graphics and Imaging

Justin Solomon

Hi Mom and Dad!

http://upload.wikimedia.org/wikipedia/commons/b/bg/JOGL_Refrection_Demo_Screenshot.png

The OpenGL Pipeline

CS 148, Summer 2012
Introduction to Computer Graphics and Imaging

Justin Solomon

<clarification>

Nonlinearity in Rasterized Depths

$$z_{world} \mapsto z = z_{near} + z_{far} - \frac{z_{near} z_{far}}{z_{world}}$$

$$\frac{dz}{dz_{world}} = \frac{z_{near} z_{far}}{z_{world}^2}$$

$$\Delta z \approx \frac{z_{near} z_{far} \Delta z_{world}}{z_{world}^2}$$

Nonlinearity in Rasterized Depths

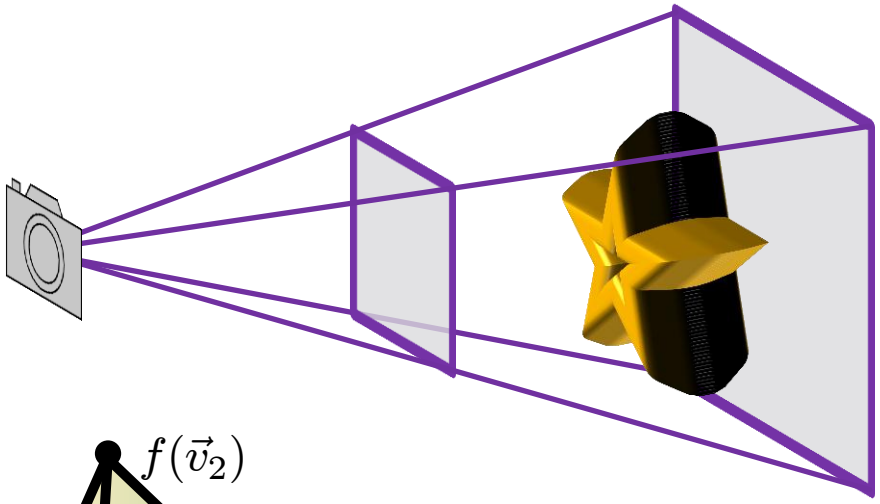
Better resolution when closer to the screen.

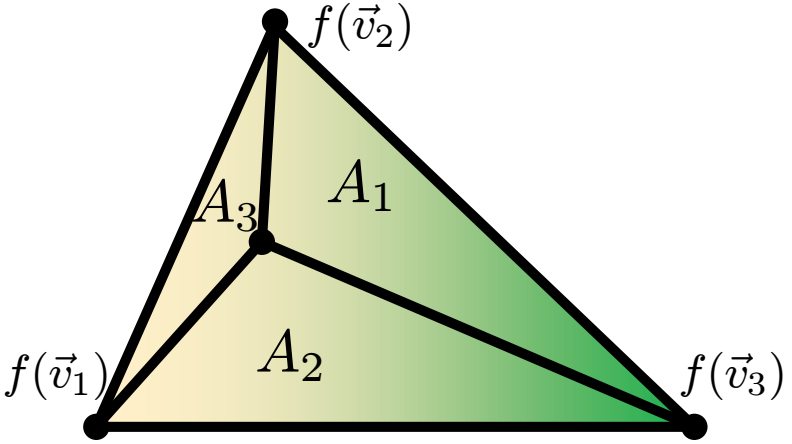
$$\Delta z \approx \frac{z_{near} z_{far} \Delta z_{world}}{z_{world}^2}$$

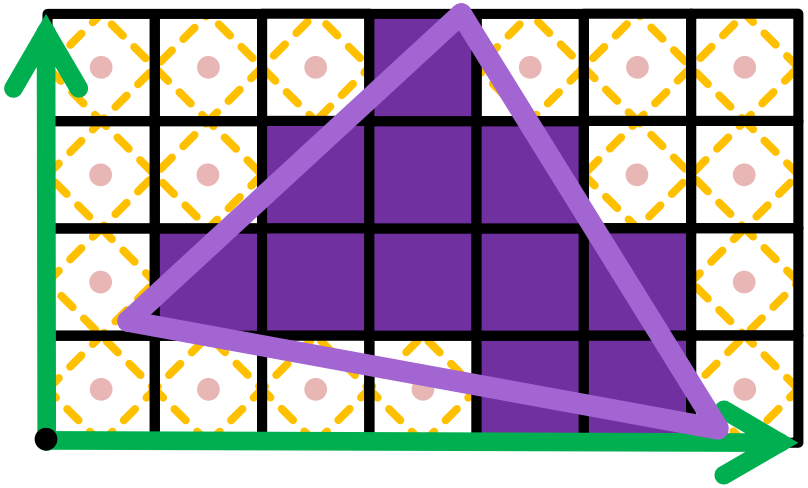
$$\Delta z_{world} \approx \frac{z_{world}^2 \Delta z}{z_{near} z_{far}}$$

</clarification>

What We Have







Theory of rasterization

Advice

Leave implementation of low-level features to the experts.

Why?

- Abstract away hardware differences
- Rasterization should be *fast*

Introducing…

Industry standard API for computer graphics

Not the Only One

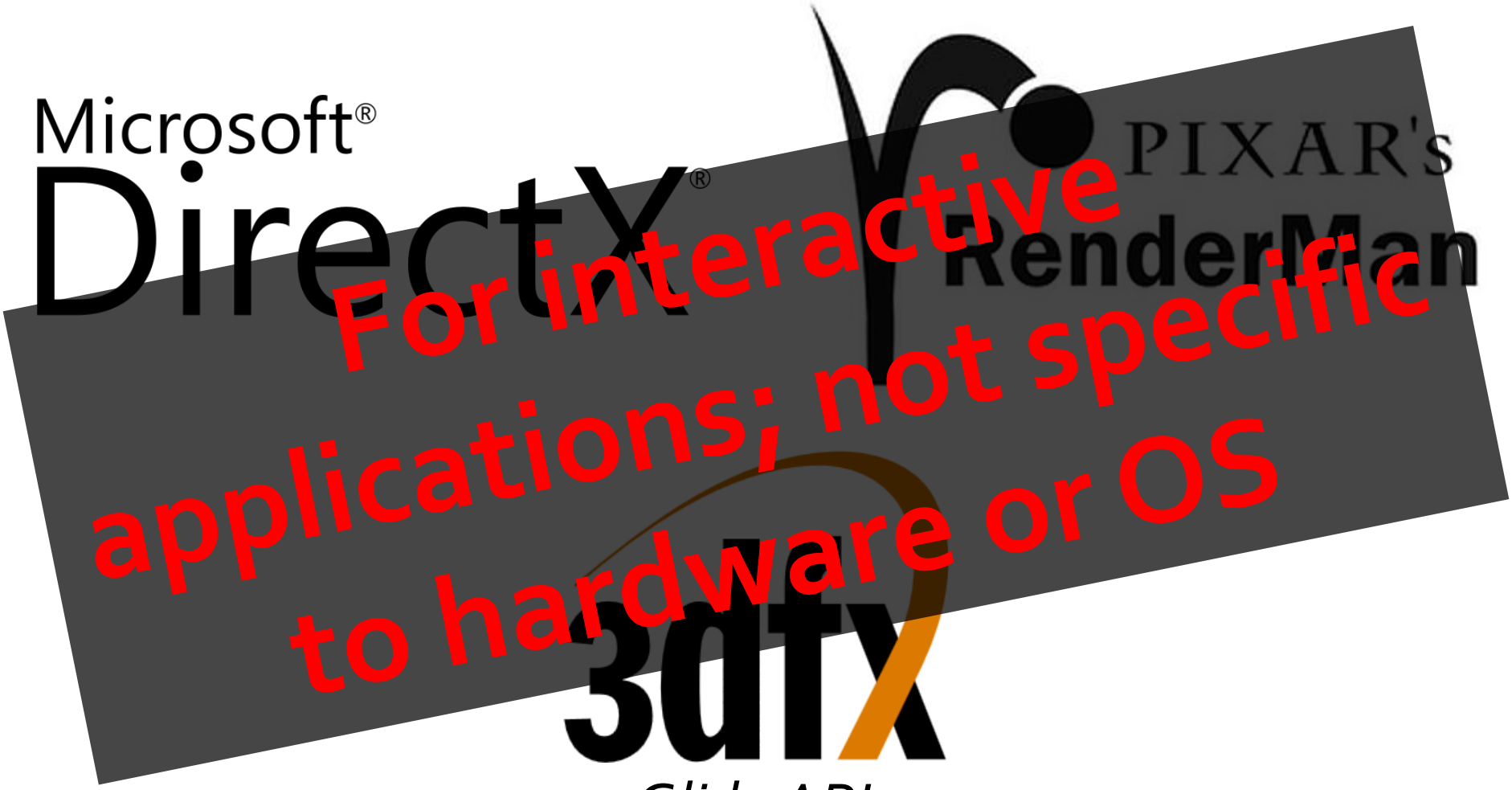
Microsoft® DirectX®

PIXAR's RenderMan

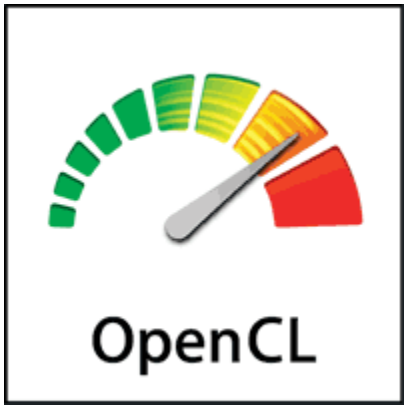
3dfx

Glide API

Not the Only One

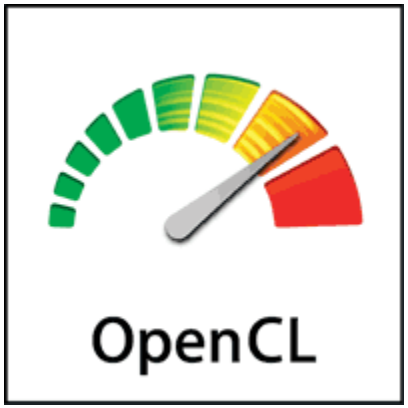


Related APIs in the OpenGL Family

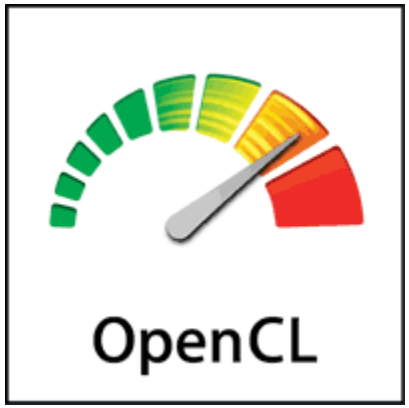


Related APIs in the OpenGL Family

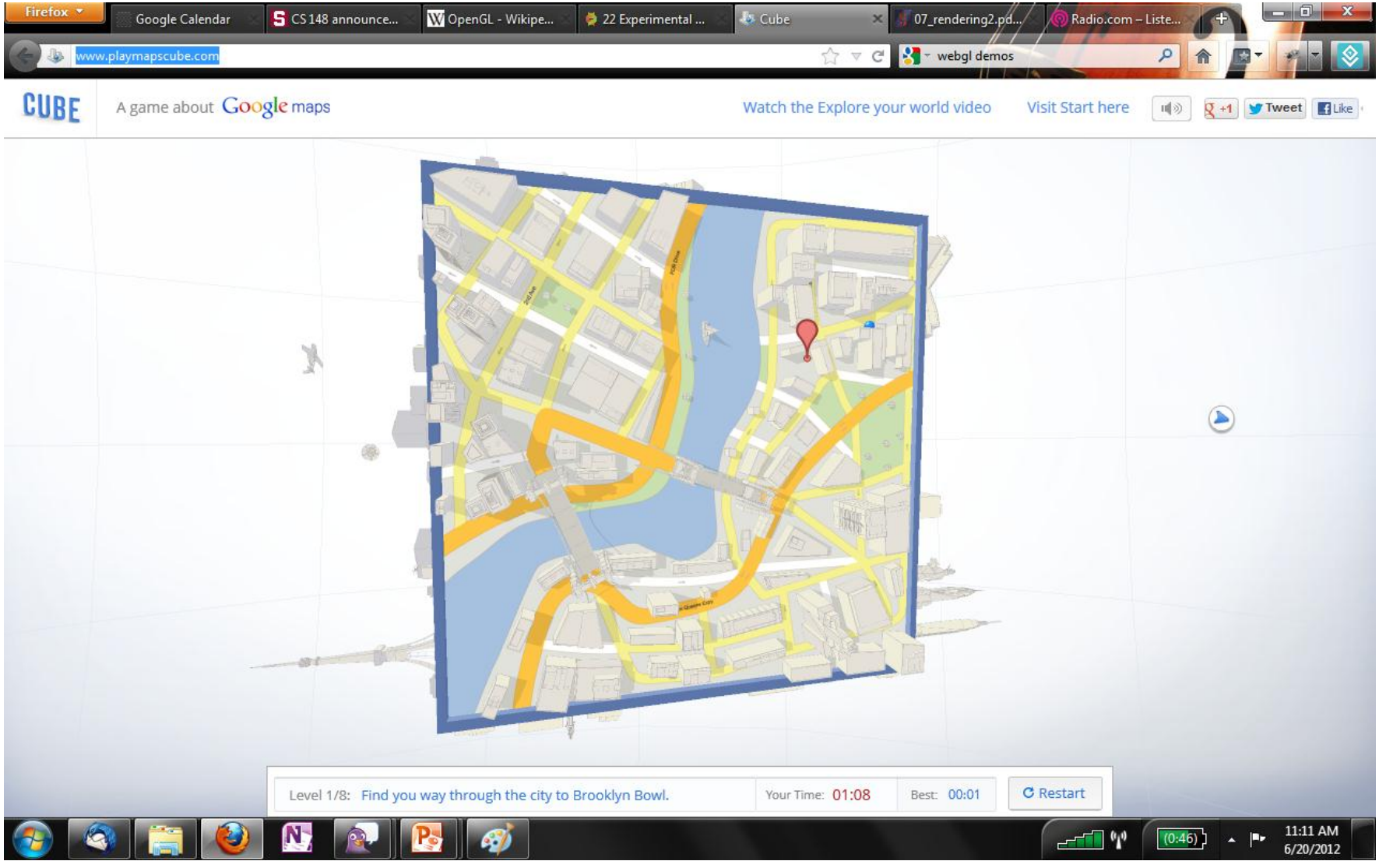
Related APIs in the OpenGL Family



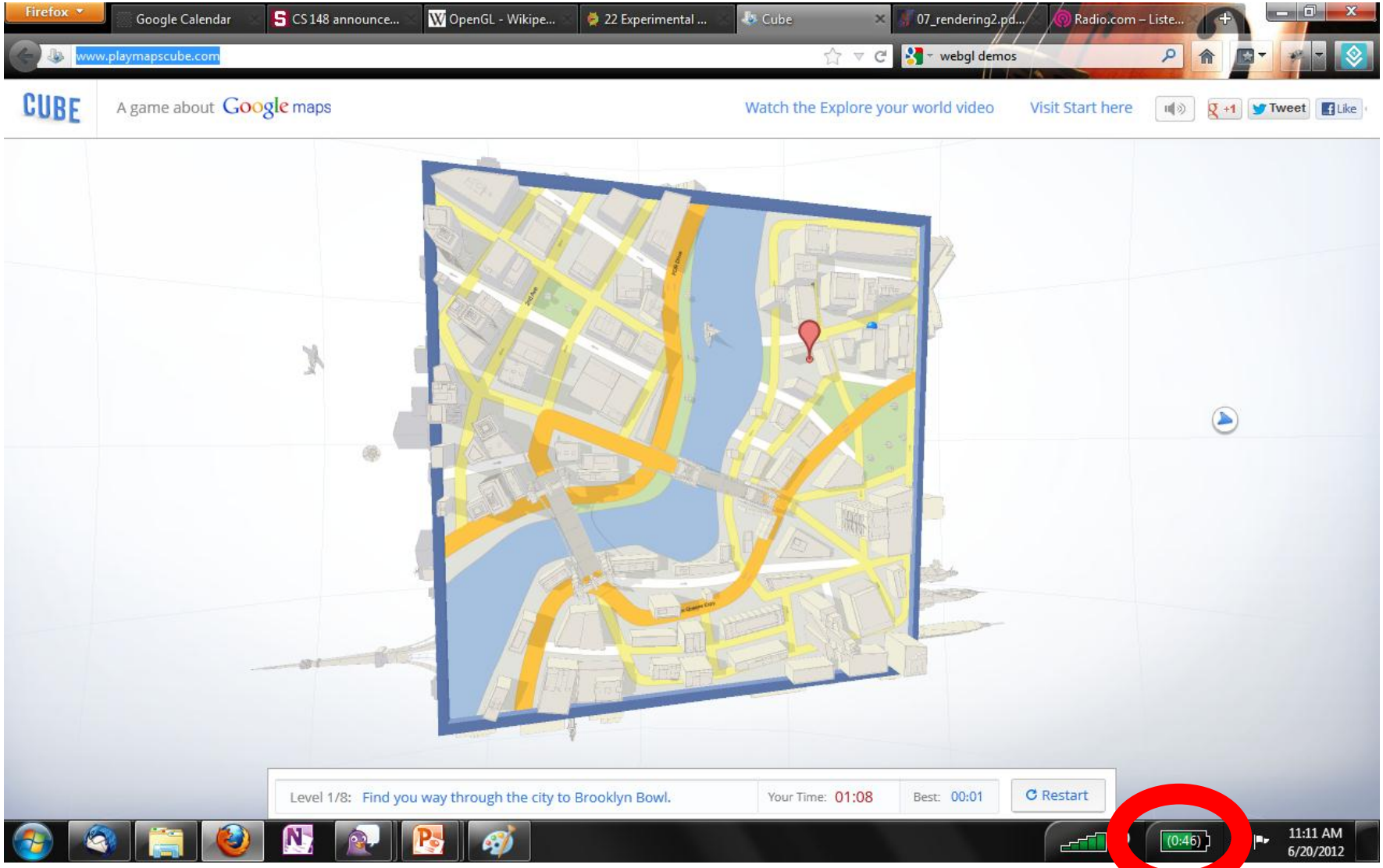
Related APIs in the OpenGL Family



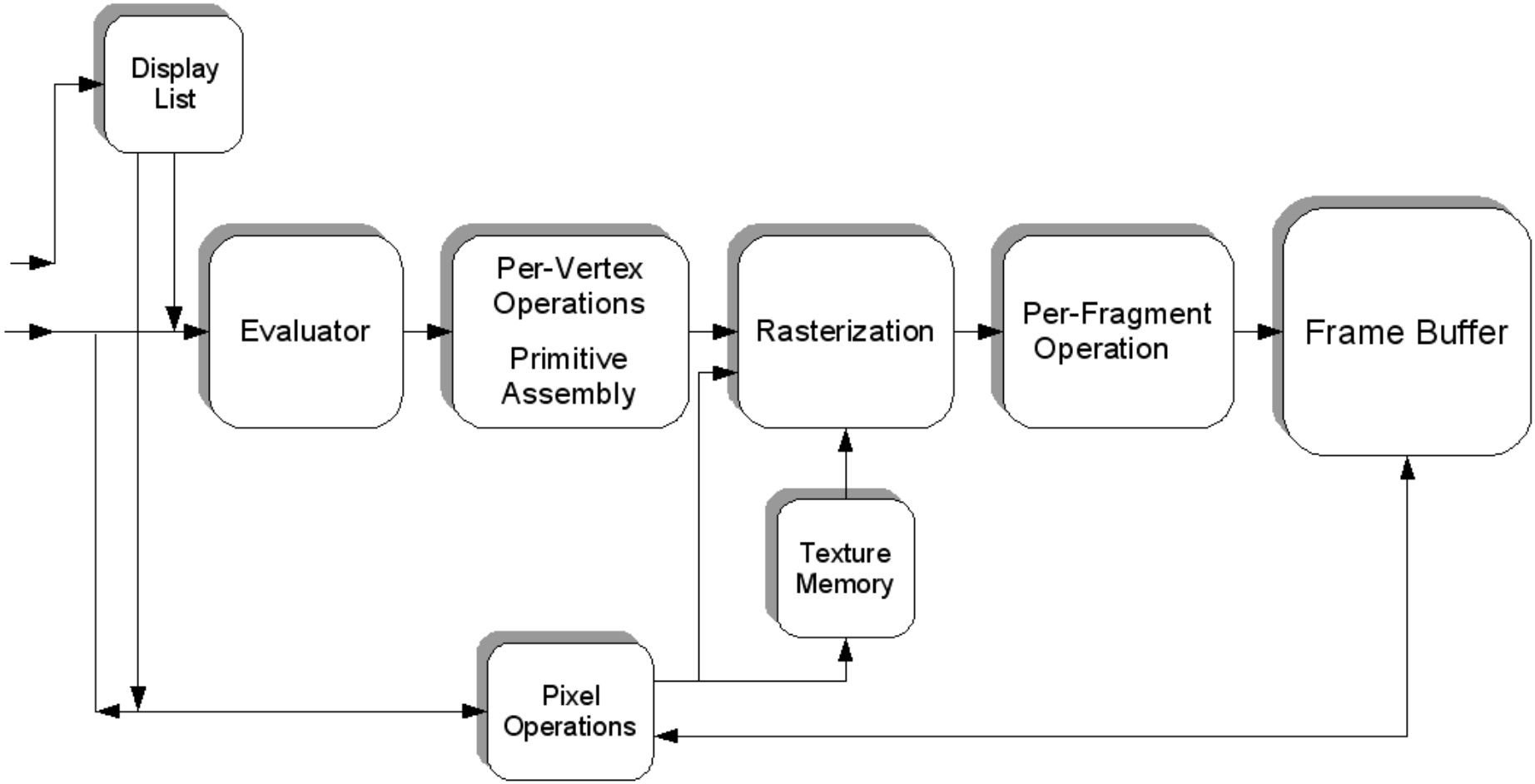
WebGL


Firefox
Google Calendar
CS 148 announce...
OpenGL - Wikipe...
22 Experimental ...
Cube
07_rendering2.pd...
Radio.com – Liste...
www.playmapscube.com
webgl demos
CUBE
A game about Google maps
Watch the Explore your world video
Visit Start here
+1
Tweet
Like
Level 1/8: Find you way through the city to Brooklyn Bowl.
Your Time: 01:08
Best: 00:01
Restart
(0:46)
11:11 AM
6/20/2012

WebGL


Firefox
Google Calendar
CS 148 announce...
OpenGL - Wikipe...
22 Experimental ...
Cube
07_rendering2.pd...
Radio.com – Liste...
www.playmapscube.com
webgl demos
CUBE
A game about Google maps
Watch the Explore your world video
Visit Start here
+1
Tweet
Like
Level 1/8: Find you way through the city to Brooklyn Bowl.
Your Time: 01:08
Best: 00:01
Restart
(0:46)
11:11 AM
6/20/2012

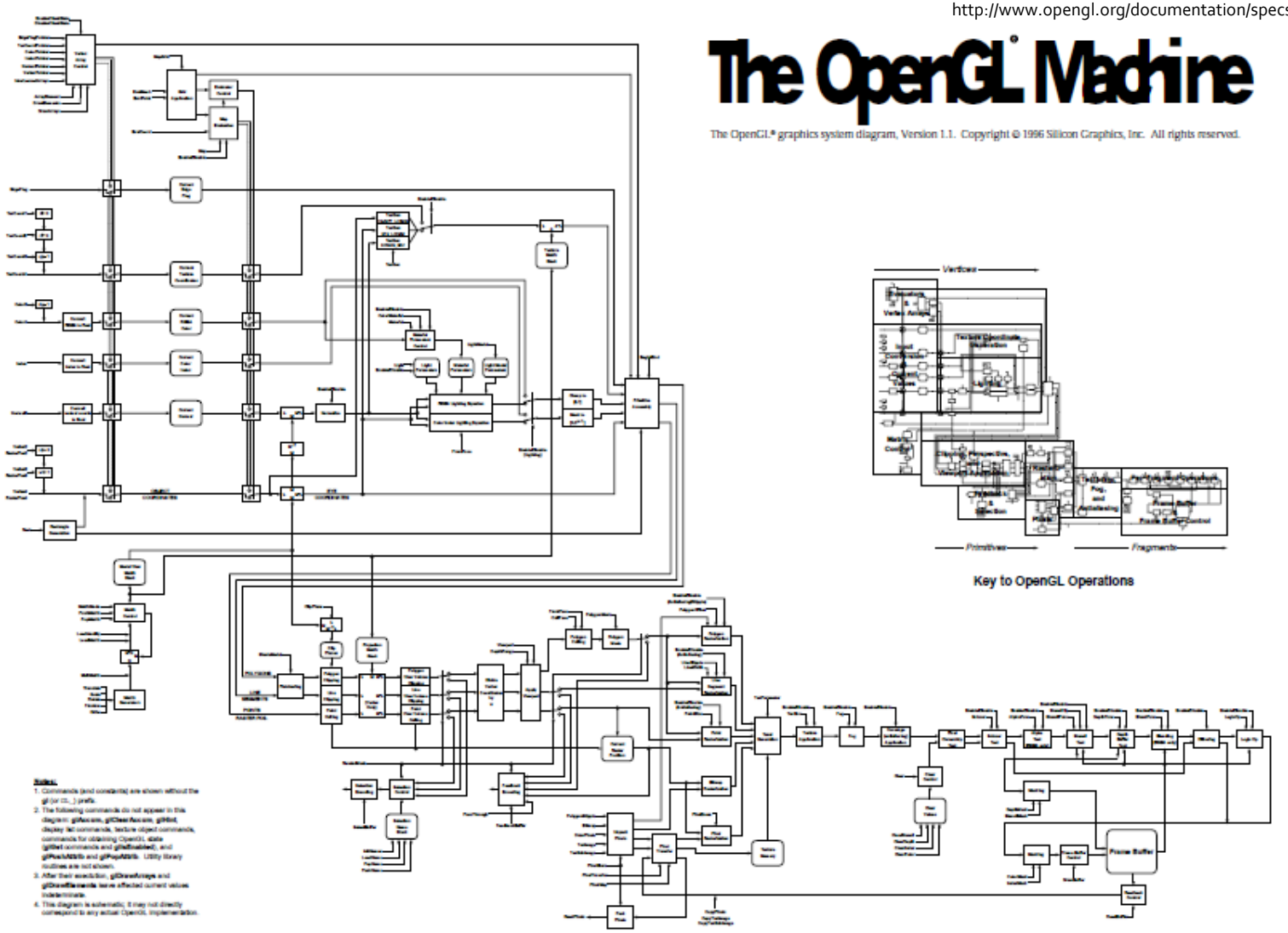
(Simplified) OpenGL Pipeline



http://upload.wikimedia.org/wikipedia/commons/b/bb/Pipeline_OpenGL_%28en%29.png

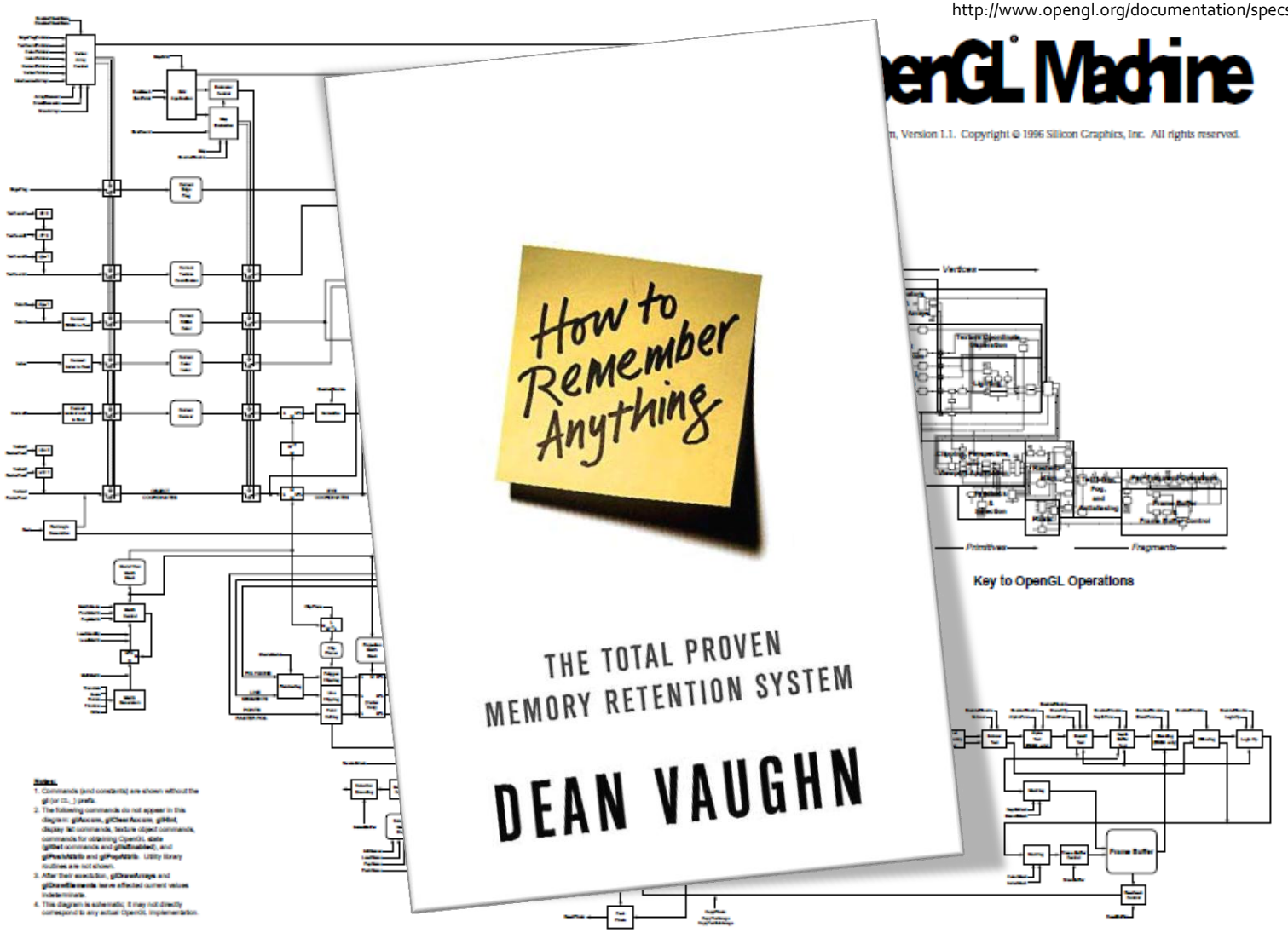
(Unsimplified) OpenGL Pipeline

http://www.opengl.org/documentation/specs/version1.1/state.pdf

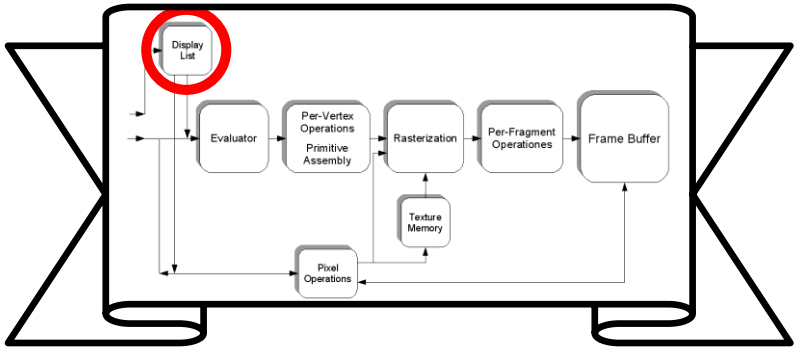


(Unsimplified) OpenGL Pipeline

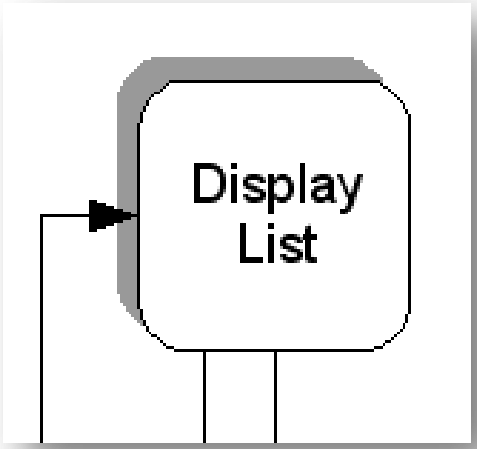
http://www.opengl.org/documentation/specs/version1.1/state.pdf



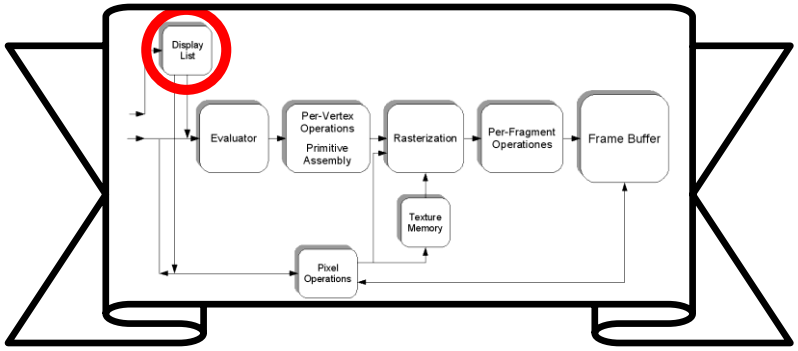
Pieces of the Pipeline



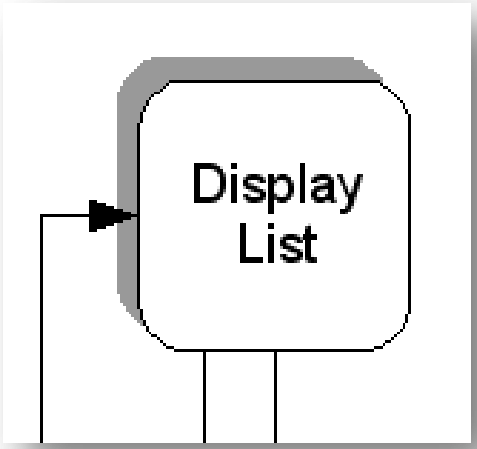
Stores "subroutines"



Pieces of the Pipeline



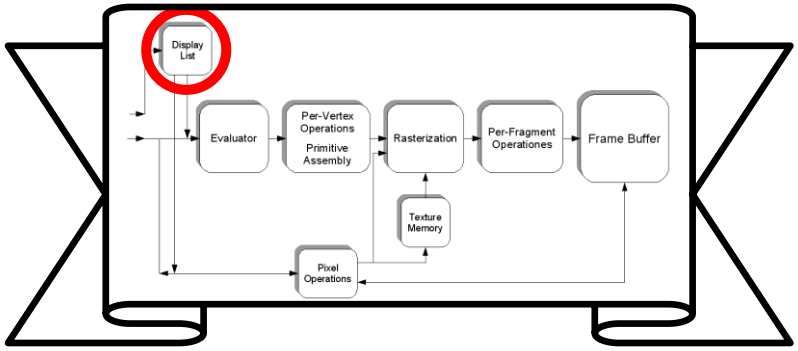
Stores "subroutines"



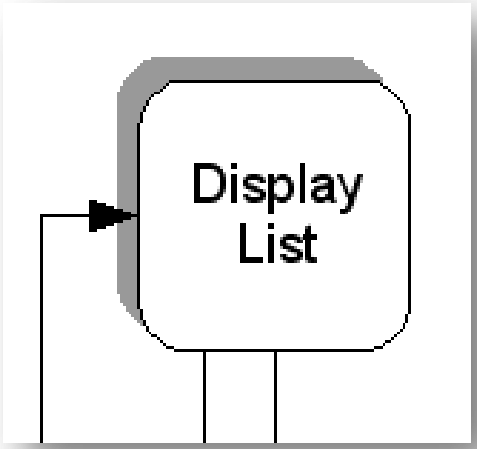
Faster!

- Pre-compiled
- Store on GPU
- Pre-compute transformations

Pieces of the Pipeline



Stores "subroutines"

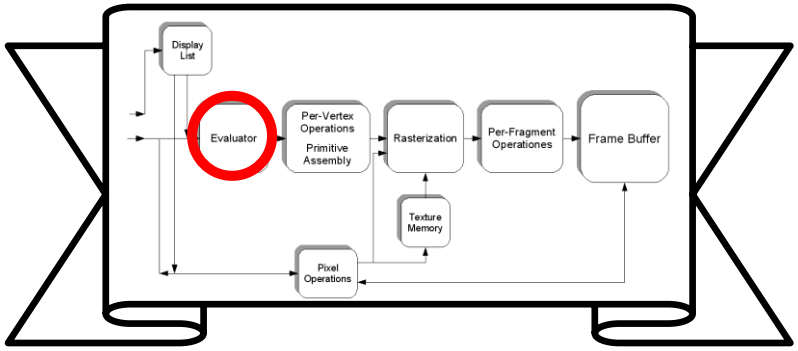


Faster!

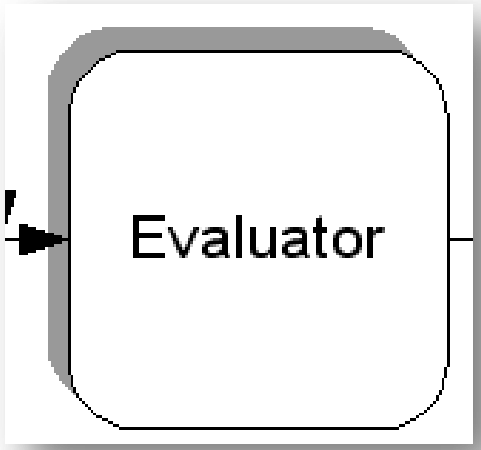
- *Pre-compiled*
- *Store on GPU*
- *Pre-compute transformations*

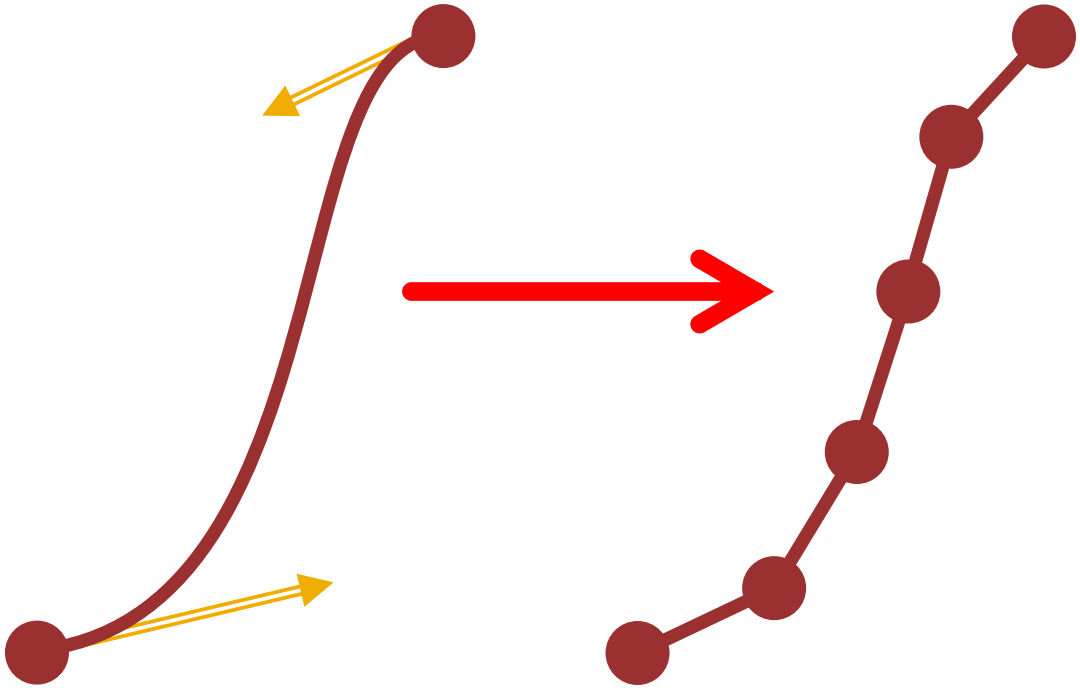
...sometimes

Pieces of the Pipeline

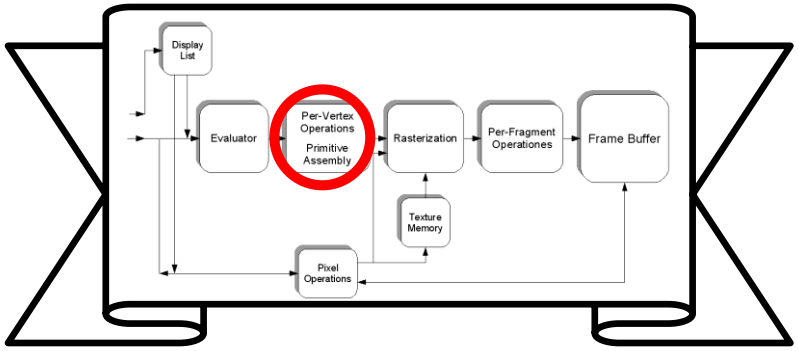


Construct geometric objects

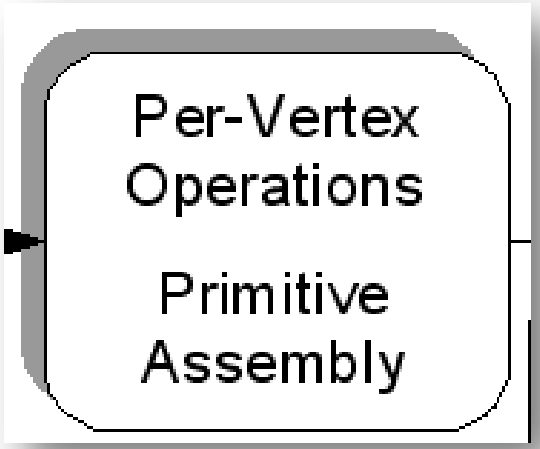




Pieces of the Pipeline

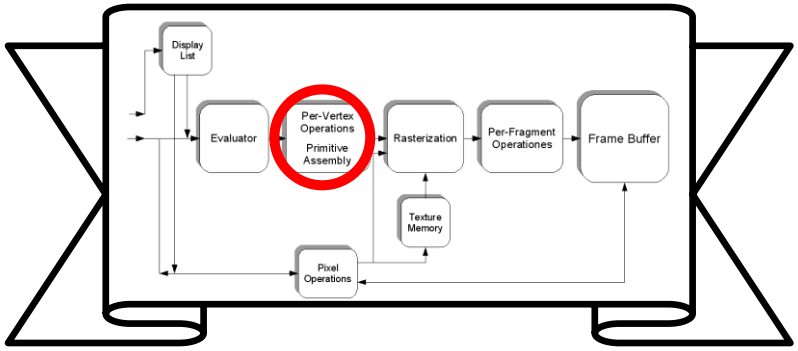


Change meshed geometry

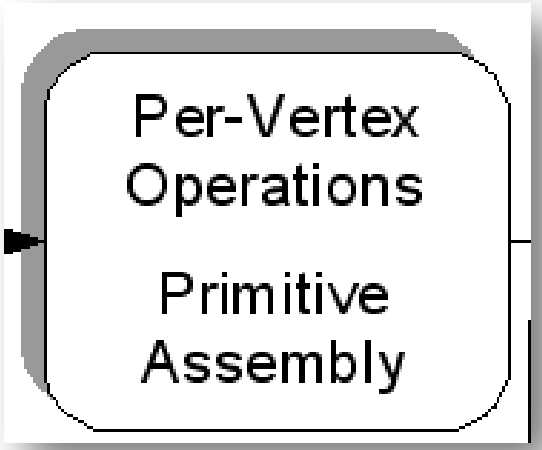


Store primitive shapes

Pieces of the Pipeline



Change meshed geometry



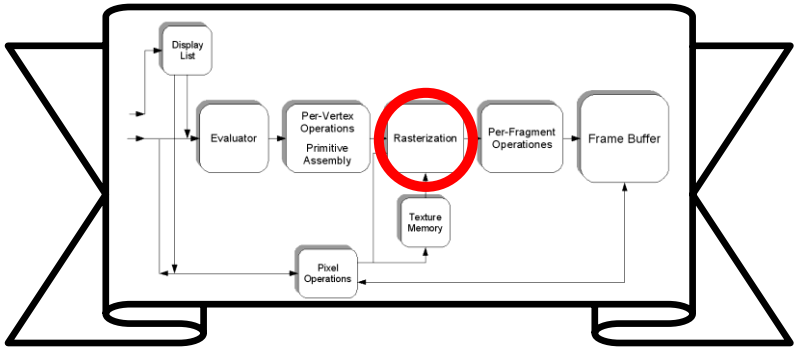
Store primitive shapes

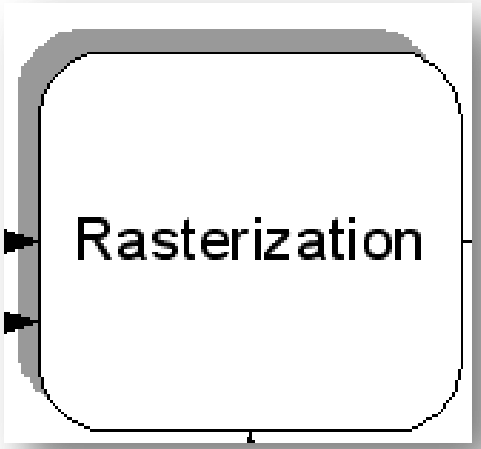
Includes clipping!

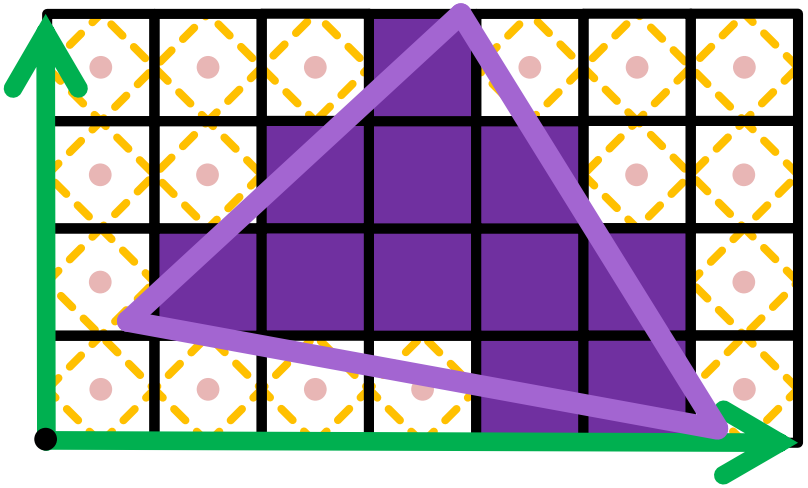
Primitive [**prim**-i-tiv]:

A small piece of geometry that can be rendered; in OpenGL, triangles, quads, lines, points (usually).

Pieces of the Pipeline





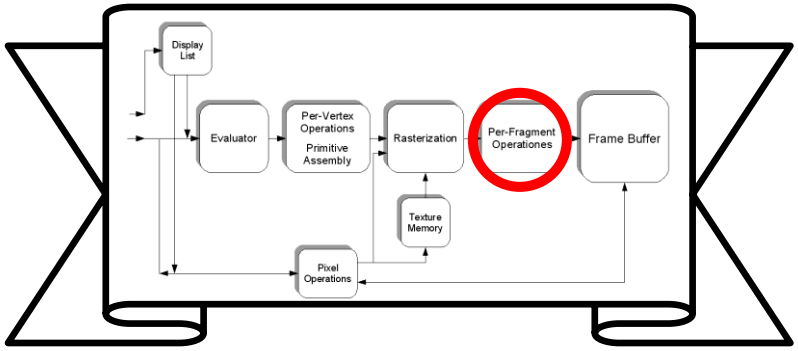


Fragment [**frag**-m*uh*nt]:

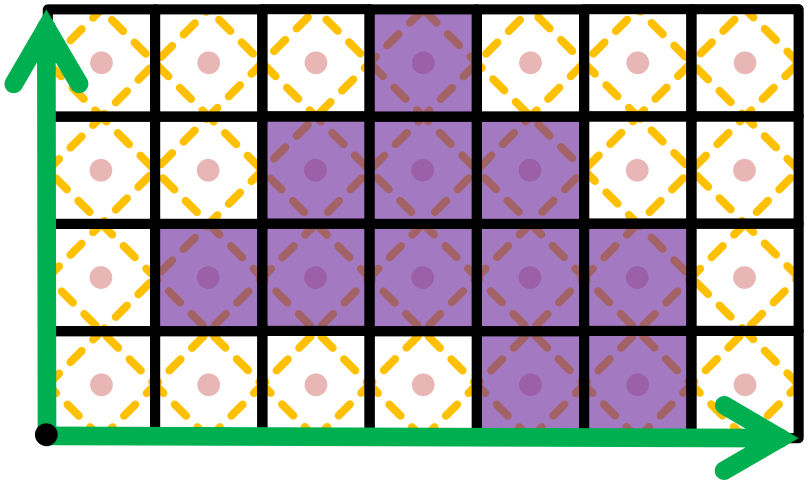
The data necessary to generate a single pixel's worth of a primitive.

http://en.wikipedia.org/wiki/Fragment_%28computer_graphics%29

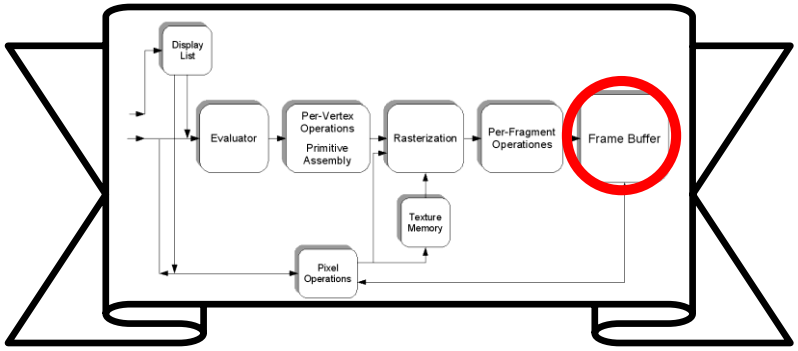
Pieces of the Pipeline



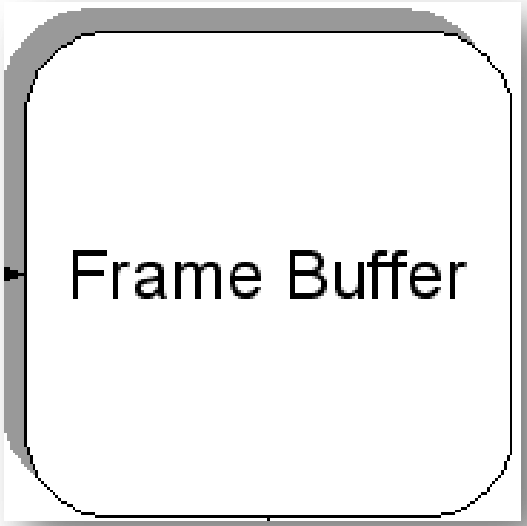
Modify and combine per-pixel information



Pieces of the Pipeline



Prepare image to be displayed



⚠ WARNING
OUT OF DATE.

cf. next lecture!

Programmer's Perspective

`glColor3f(…)`	`glGetFloatv(…)`
`glEnable(…)`	`glIsEnabled(…)`
`glLineStipple(…)`	`glGetLineStipple(…)`

OpenGL is a state machine

What OpenGL Is *Not*

A windowing system

Platform independent!

GLUT

- Simple cross-platform windowing API
- Bindings: C, C++, Fortran, Ada, …
- Features:
  - Multiple windows, menus
  - Keyboard/mouse/other input
  - Assorted callbacks: idle, timers
  - Basic font support
  - `glutSolidTeapot`, `glutSolidSphere`, `glutSolidCube`, …

GL Utility Toolkit

GLUT

- Simple cross-platform windowing API
- Bindings: C, C++, Fortran, Ada, …
- Features:
  - Multiple windows, menus
  - Keyboard/mouse/other input
  - Assorted callbacks: idle, timers
  - Basic font support
  - `glutSolidTeapot`, `glutSolidSphere`, `glutSolidCube`, …

GL Utility Toolkit

GLUT

- Simple cross-platform windowing API
- Bindings: C, C++, Fortran, Ada, …
- Features:
  - Multiple windows, menus
  - Keyboard/mouse/other input
  - Assorted callbacks: idle, timers
  - Basic font support
  - `glutSolidTeapot`, `glutSolidSphere`, `glutSolidCube`, …

Homework 2

freeglut

GL Utility Toolkit

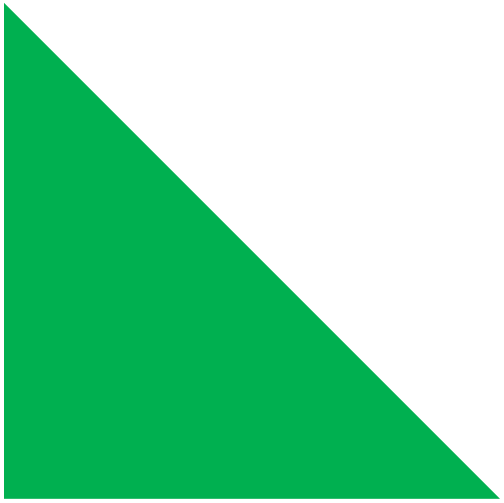
GLU

- High-level graphics commands
- *Not* included in OpenGL ES
- Some interesting features:
  - Mapping between world and screen coordinates
  - Texturing support
  - Tessellation and other geometric utilities
  - OpenGL error code lookup
  - More primitives: spheres, cylinders, disks, …
  - Camera support: `gluLookAt`, `gluOrtho2D`, …

GL Utility *Library*

Adding Geometry to the Scene

```
glBegin(GL_TRIANGLES);
    glVertex3d(0, 0, 0);
    glVertex3d(1, 0, 0);
    glVertex3d(0, 1, 0);
glEnd(/*GL_TRIANGLES*/);
```

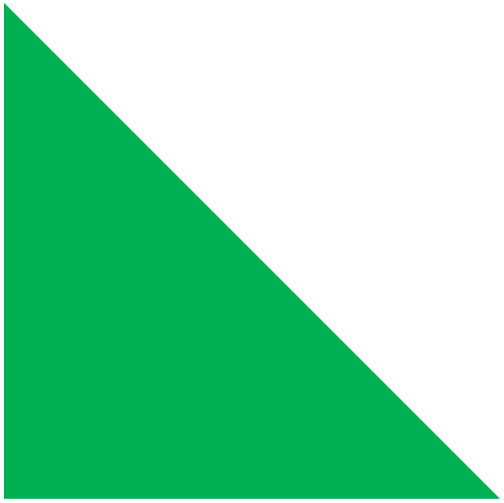


Option 1: Immediate mode

Adding Geometry to the Scene

dimension 3, double type

```
glBegin(GL_TRIANGLES);
    glVertex3d(0, 0, 0);
    glVertex3d(1, 0, 0);
    glVertex3d(0, 1, 0);
glEnd(/*GL_TRIANGLES*/);
```

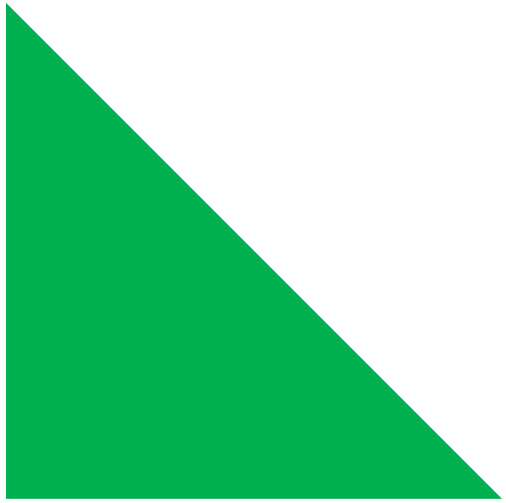


Option 1: Immediate mode

Adding Geometry to the Scene

```
glBegin(GL_TRIANGLES);
    glVertex3d(0, 0, 0);
    glVertex3d(1, 0, 0);
    glVertex3d(0, 1, 0);
glEnd(/*GL_TRIANGLES*/);
```

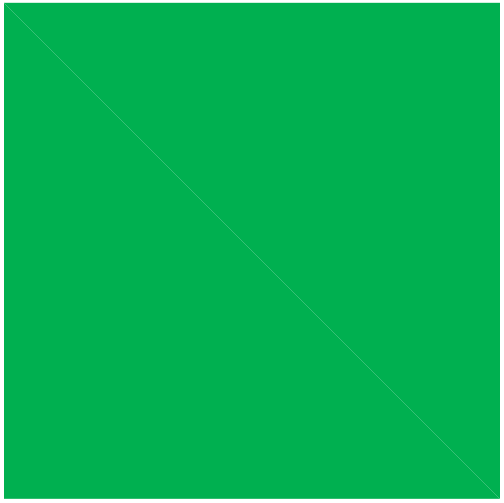
Not needed



Option 1: Immediate mode

Adding Geometry to the Scene

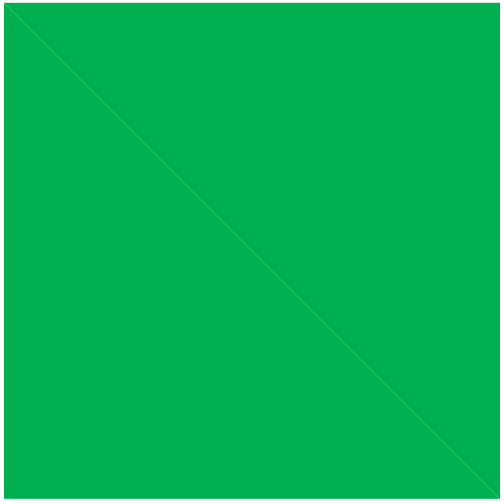
```
glBegin(GL_TRIANGLES);
    glVertex3d(0, 0, 0);
    glVertex3d(1, 0, 0);
    glVertex3d(0, 1, 0);
    glVertex3d(1, 0, 0);
    glVertex3d(1, 1, 0);
    glVertex3d(0, 1, 0);
glEnd(/*GL_TRIANGLES*/);
```



Option 1: Immediate mode

Adding Geometry to the Scene

```
glBegin(GL_TRIANGLE_STRIP);

    glVertex3d(0, 0, 0);
    glVertex3d(1, 0, 0);
    glVertex3d(0, 1, 0);
    glVertex3d(1, 1, 0);

glEnd(/*GL_TRIANGLE_STRIP*/);
```



Option 1: Immediate mode

Adding Geometry to the Scene

```
glBegin(GL_TRIANGLE_STRIP);

    glVertex3d(0, 0, 0);
    glVertex3d(1, 0, 0);
    glVertex3d(0, 1, 0);
    glVertex3d(1, 1, 0);

glEnd(/*GL_TRIANGLE_STRIP*/);
```

Option 1: Immediate mode

Adding Geometry to the Scene

```
GLuint boxList;
boxList = glGenLists(1);
glNewList(boxList, GL_COMPILE);
    // draw box
glEndList(boxList);
…
glCallList(boxList);
```

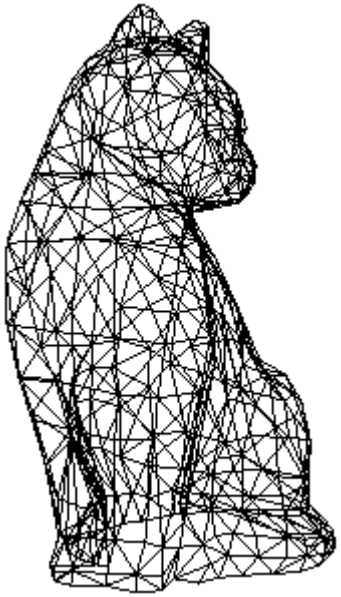
Option 2: Display lists

Adding Geometry to the Scene

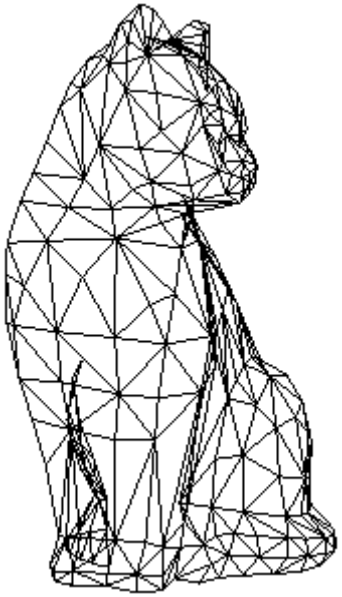
```
GLfloat vertices[] = {...};

glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(3, GL_FLOAT, 0, vertices);
glDrawArrays(GL_TRIANGLES, 0, 36);
glDisableClientState(GL_VERTEX_ARRAY);
```

Option 3: Vertex buffer

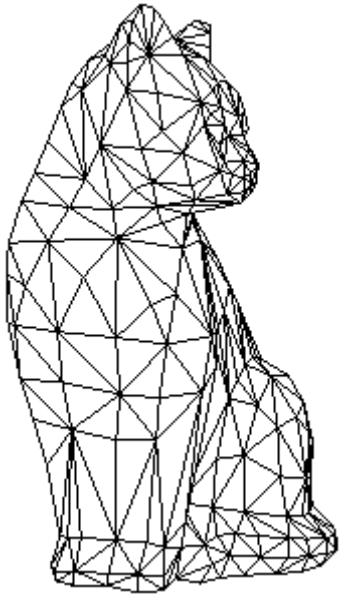
Recall: Backface Culling



None



Backface culling

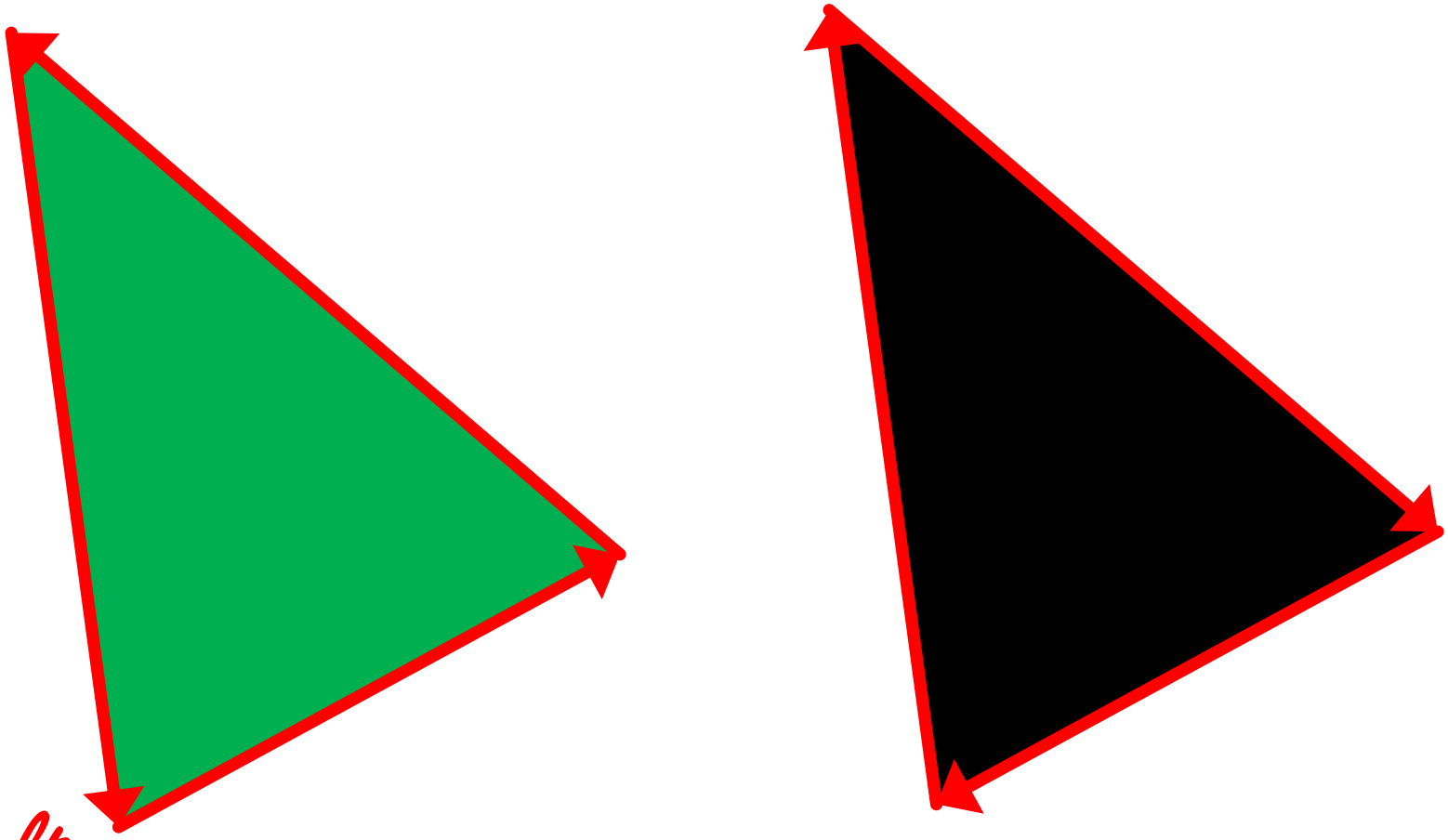


Hidden surface removal

http://medialab.di.unipi.it/web/IUM/Waterloo/node70.html

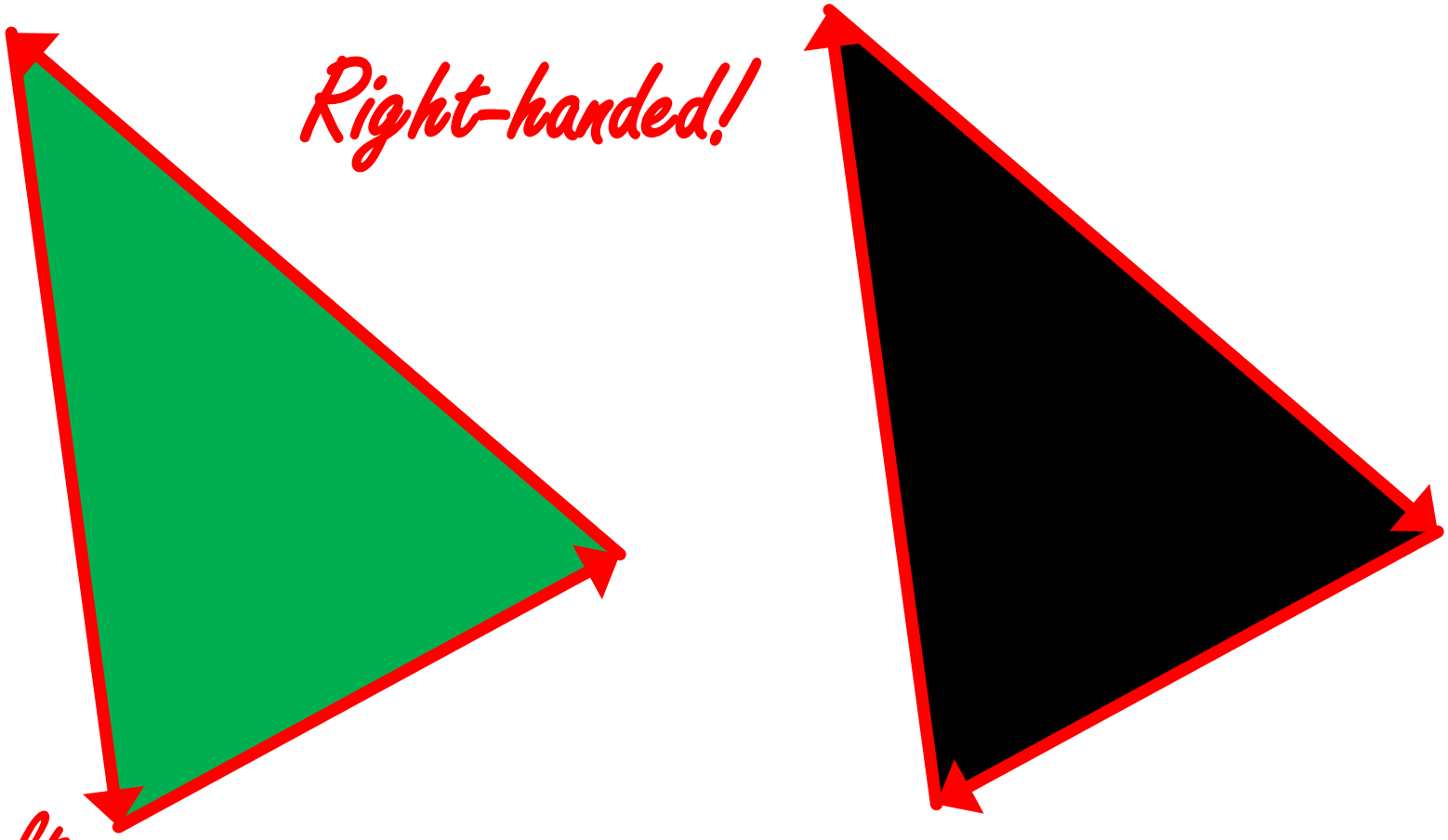
Cull triangles with back-facing normals

Specifying Triangle Orientation



Default: `glFrontFace(GL_CCW)`

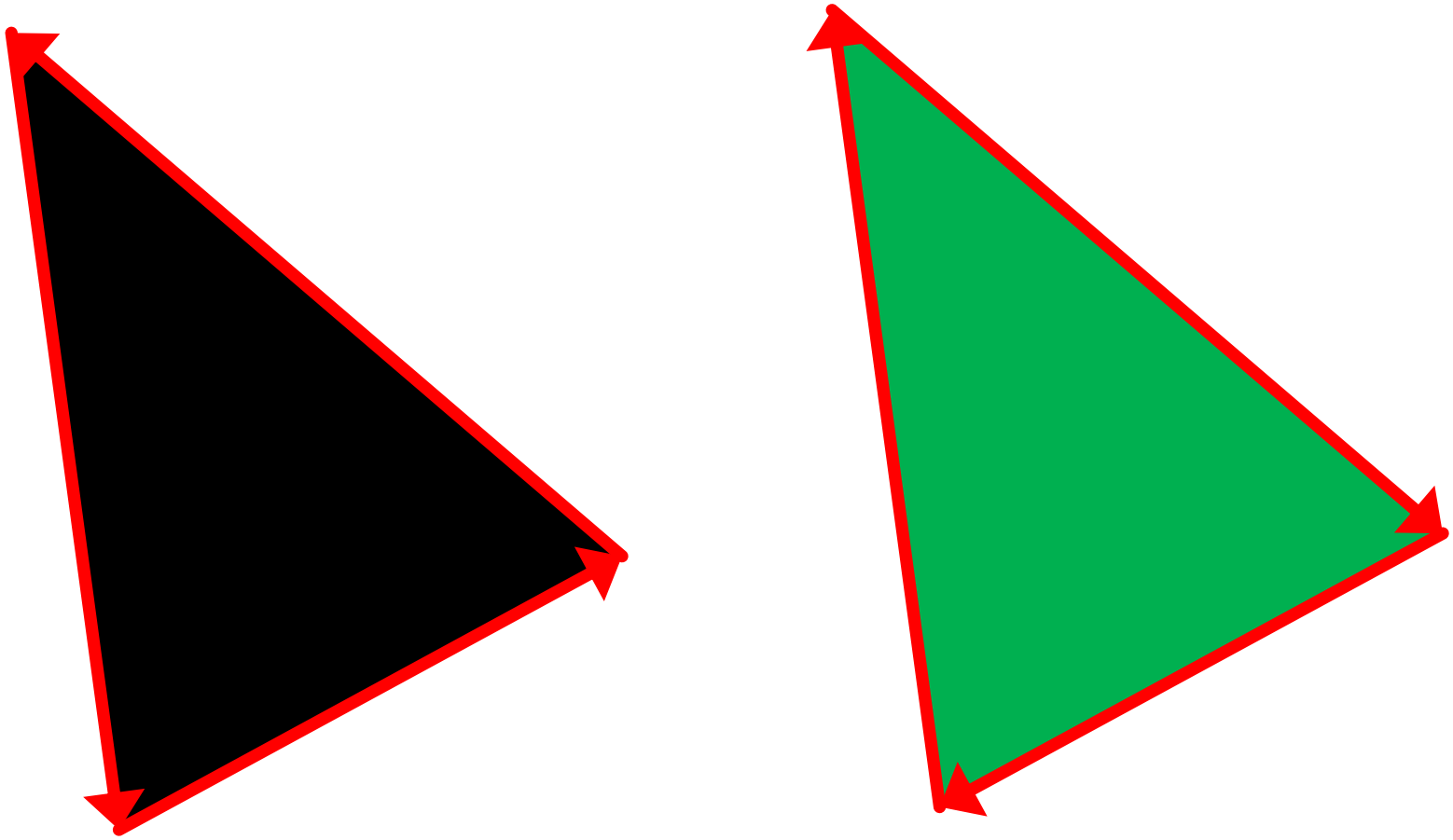
Specifying Triangle Orientation



Default:

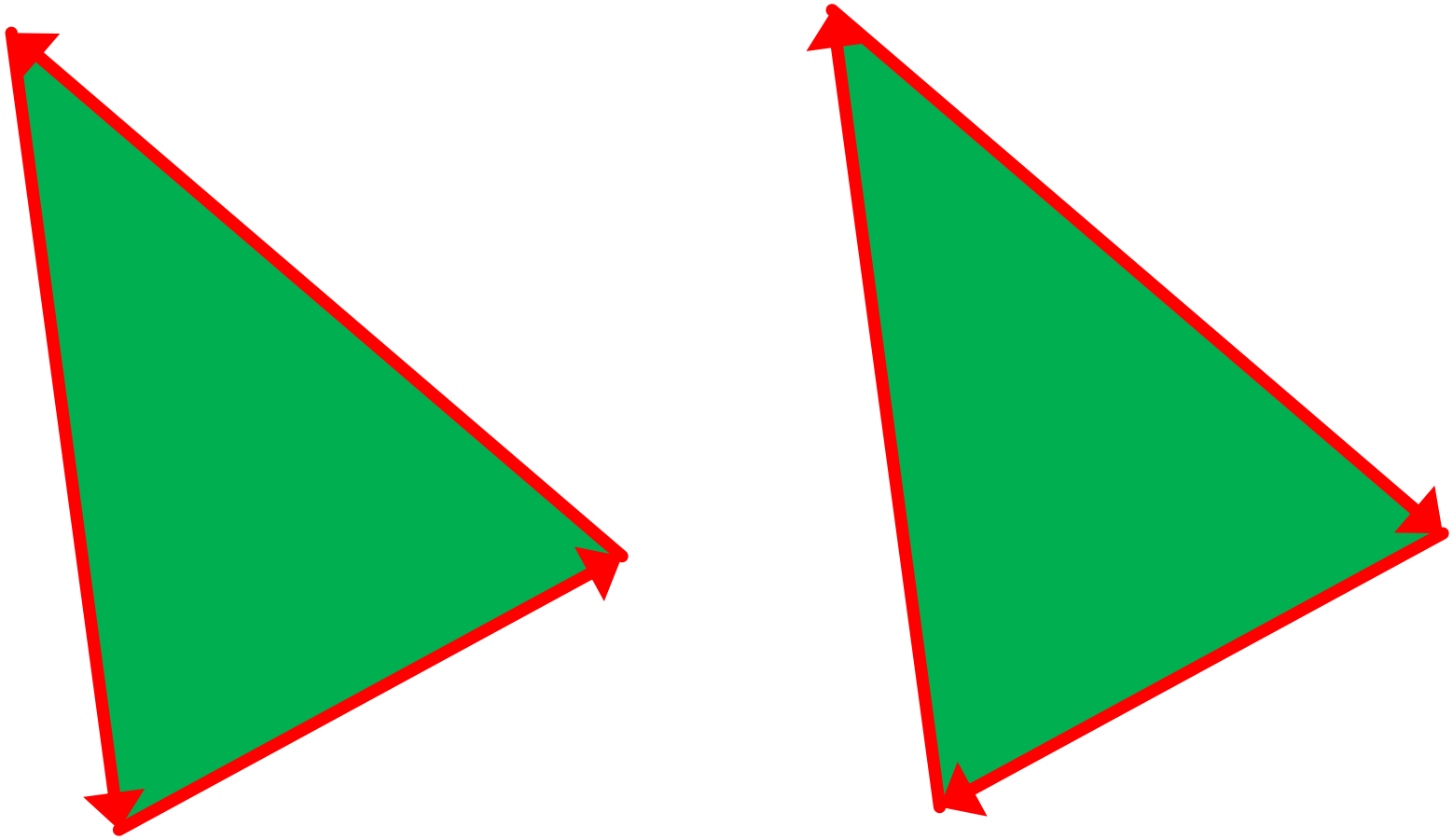
```
glFrontFace(GL_CCW)
```

Specifying Triangle Orientation



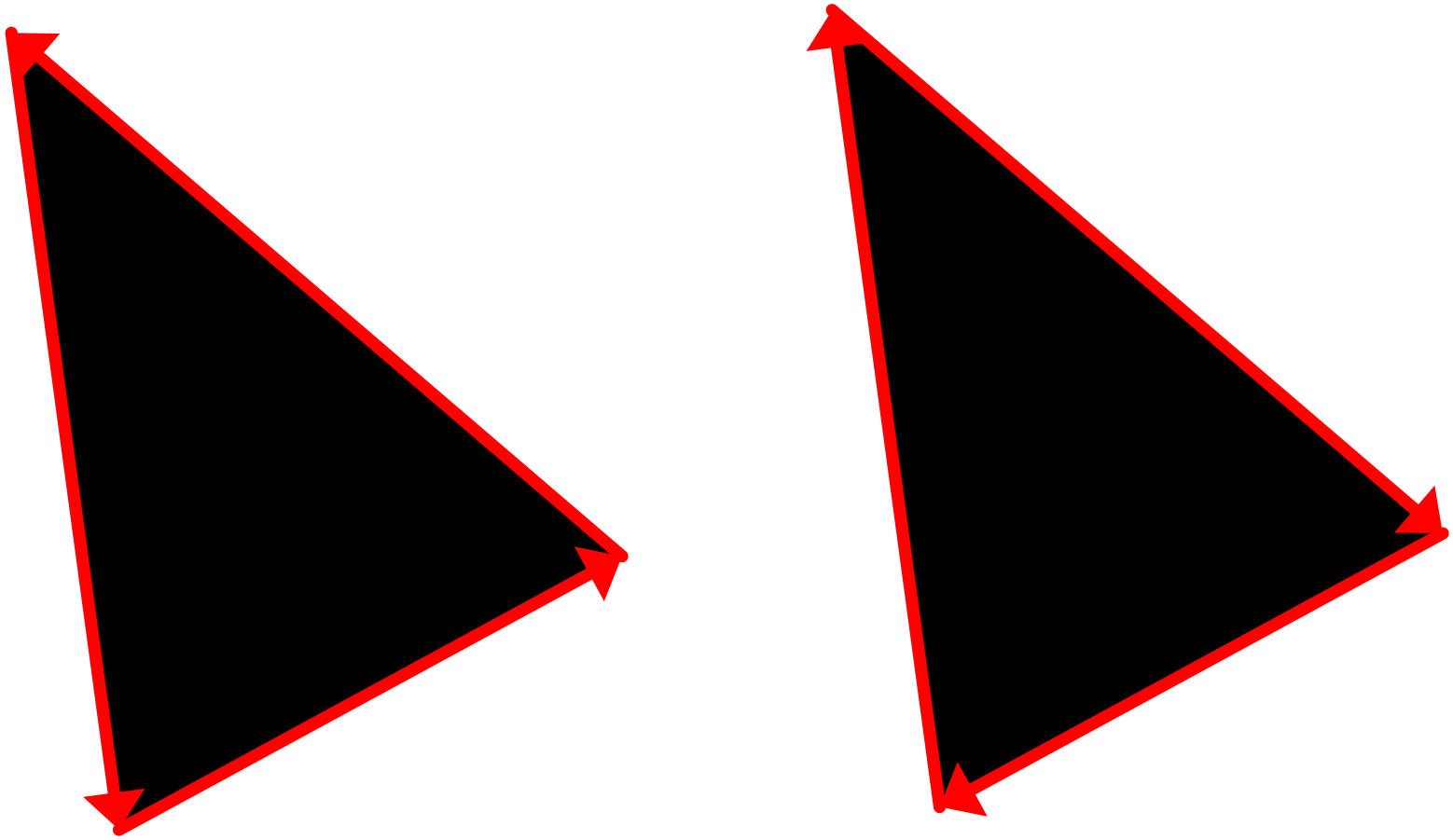
`glFrontFace(GL_CW)`

Specifying Triangle Orientation



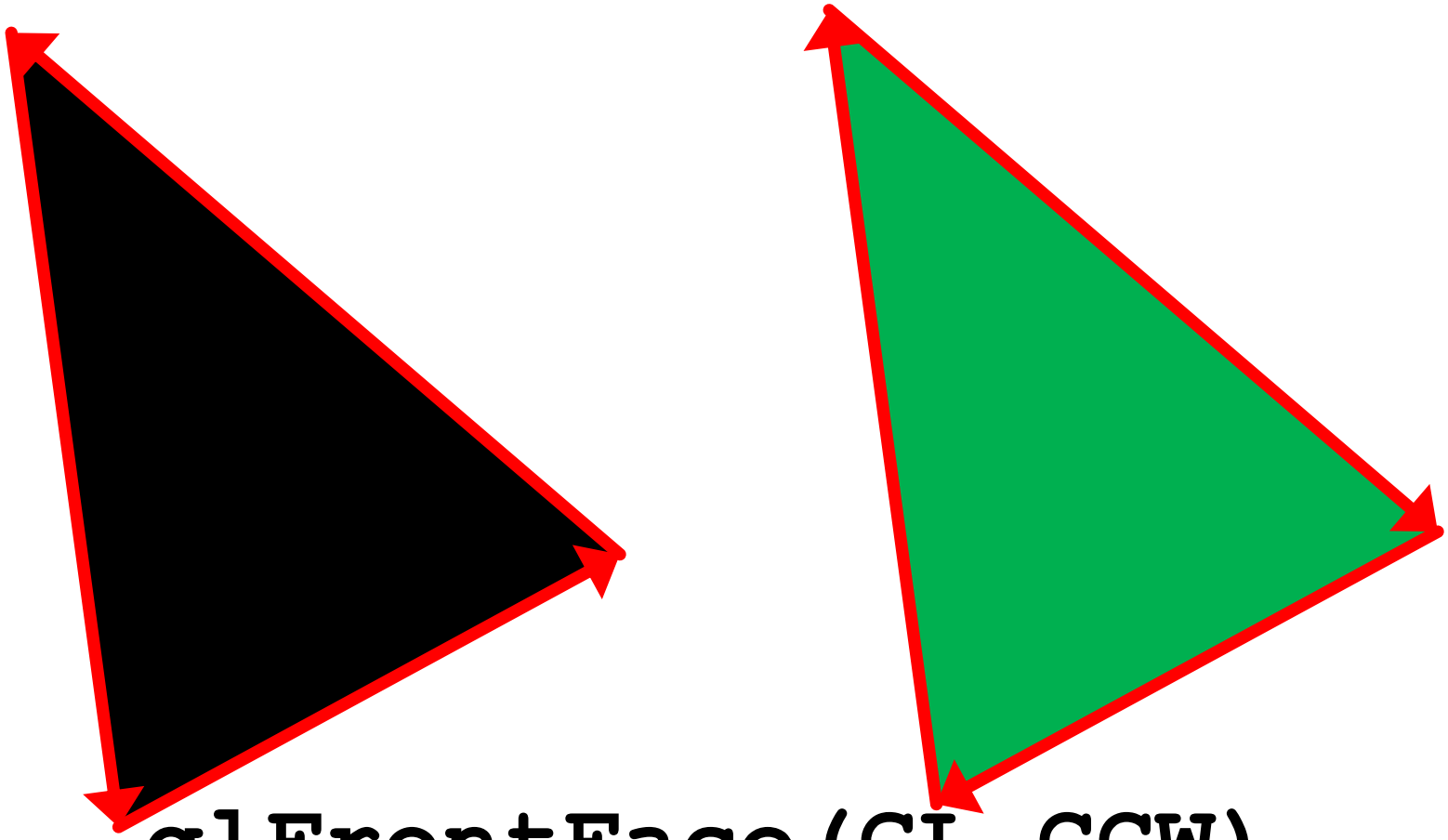
```
glDisable(GL_CULL_FACE)
```

Specifying Triangle Orientation



```
glCullFace(GL_FRONT_AND_BACK)
```

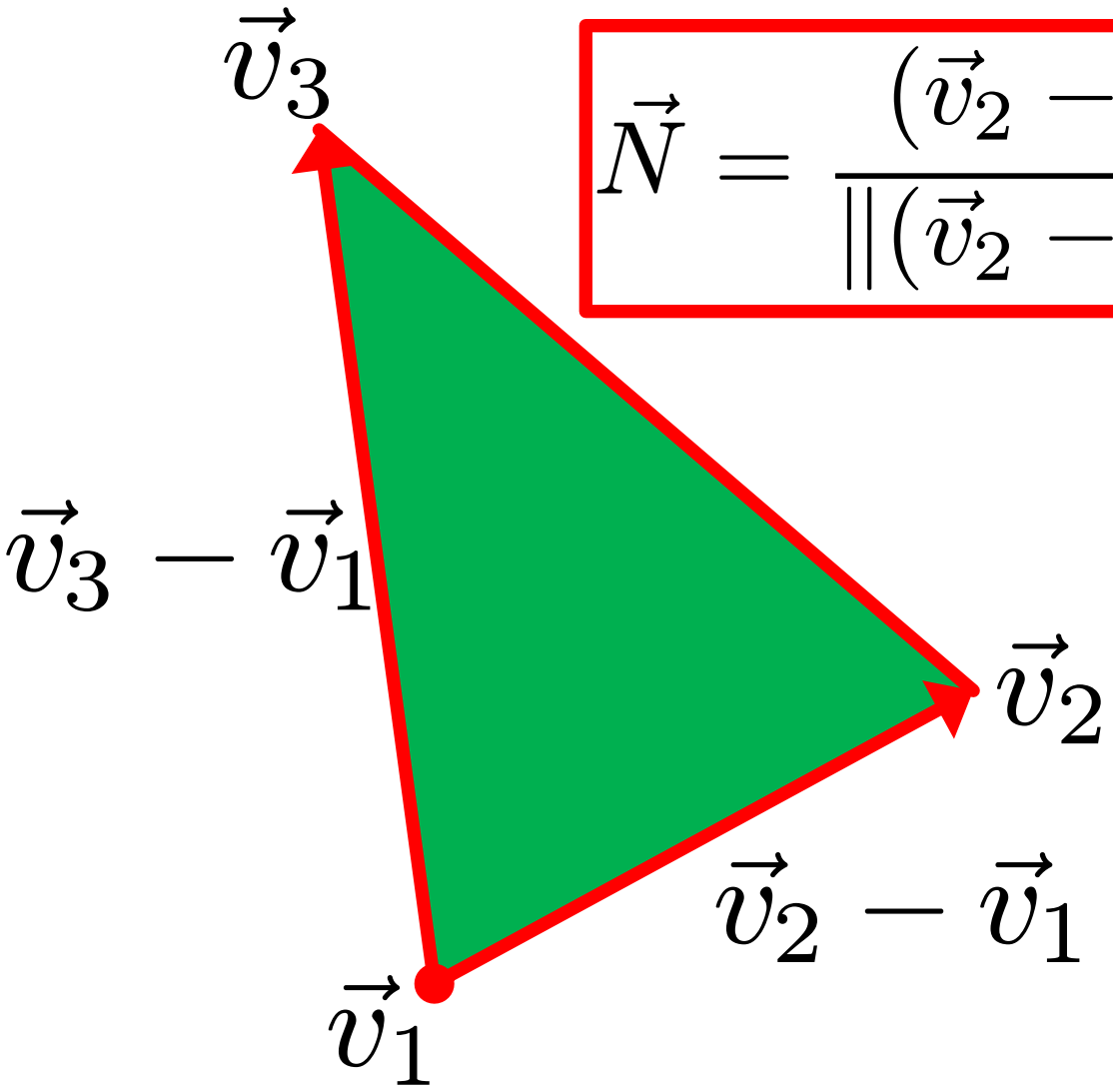
Specifying Triangle Orientation



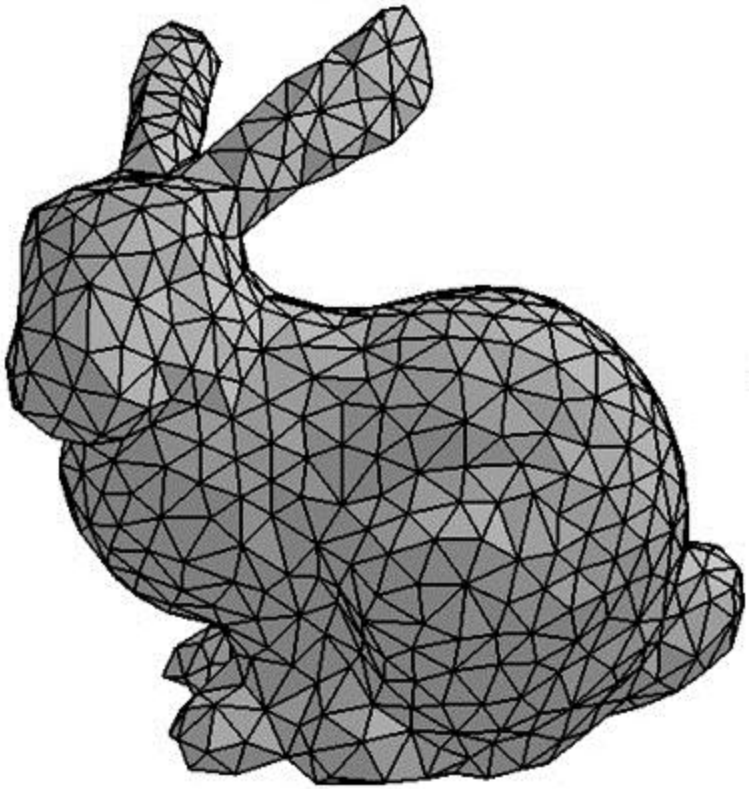
```
glFrontFace(GL_CCW)
glCullFace(GL_FRONT)
```

Computing Face Normals

$$\vec{N} = \frac{(\vec{v}_2 - \vec{v}_1) \times (\vec{v}_3 - \vec{v}_1)}{\|(\vec{v}_2 - \vec{v}_1) \times (\vec{v}_3 - \vec{v}_1)\|}$$



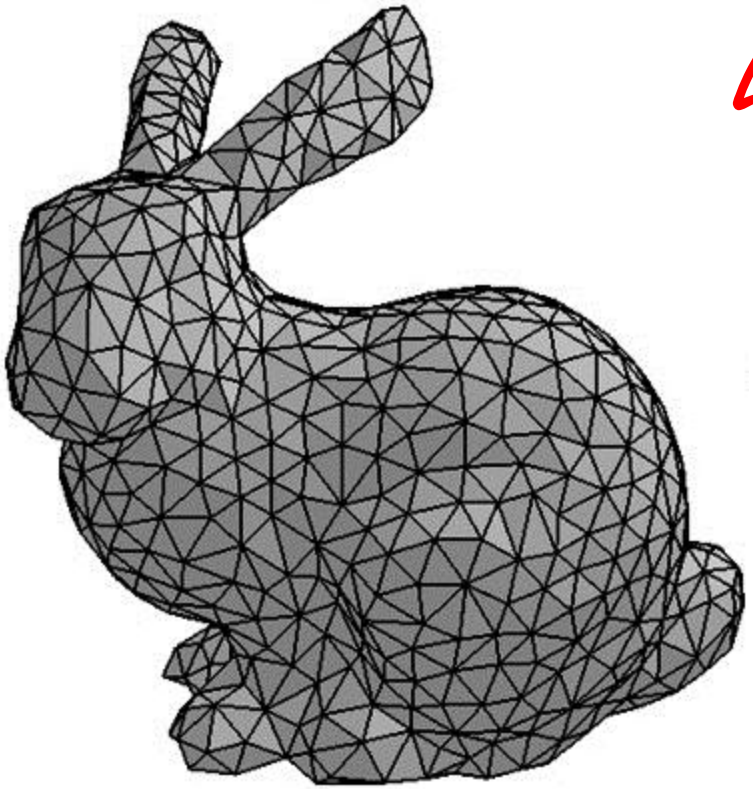
Aside: Storing Meshes



```
xyz/xyz/xyz
xyz/xyz/xyz
xyz/xyz/xyz
...
```

Low level: "Triangle soup"

Aside: Storing Meshes

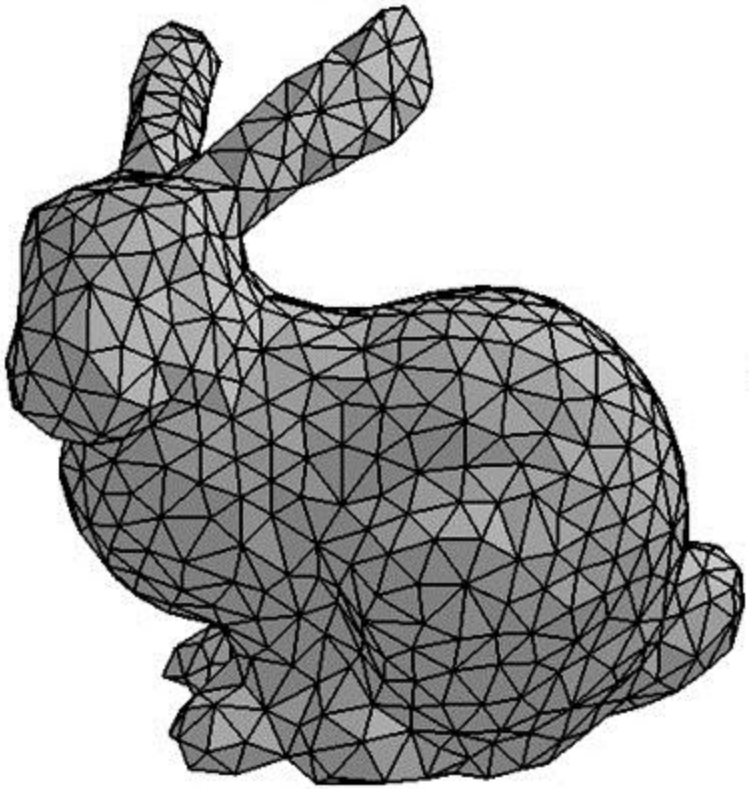


Likely to repeat

```
xyz/xyz/xyz
xyz/xyz/xyz
xyz/xyz/xyz
...
```

Low level: "Triangle soup"

Aside: Storing Meshes



Floats

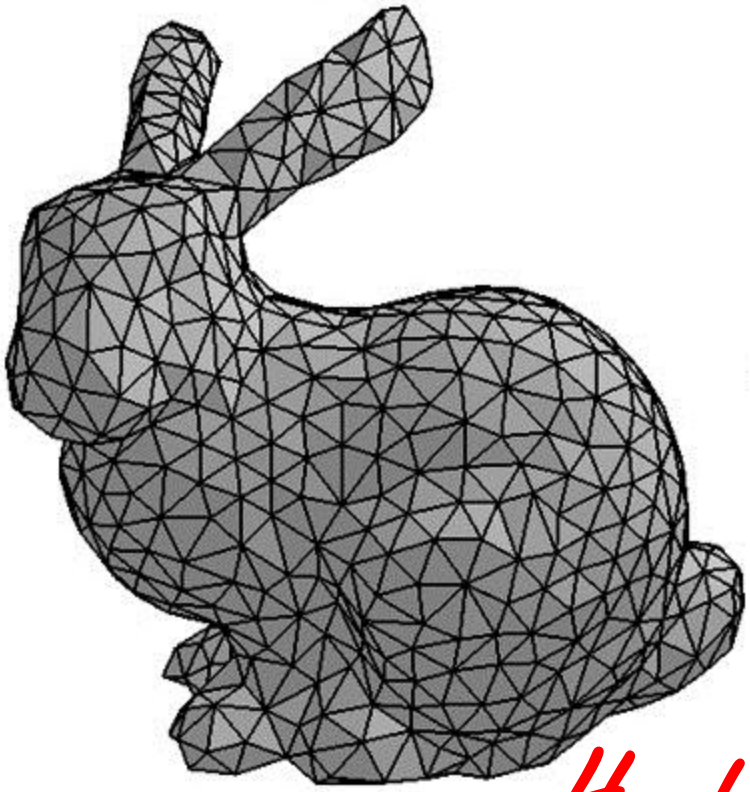
```
xyz
xyz
...
```

Integers

```
v1v2v3
v1v2v3
...
```

Medium level: Triangle/vertex lists

Aside: Storing Meshes



Floats

```
x y z
x y z
...
```

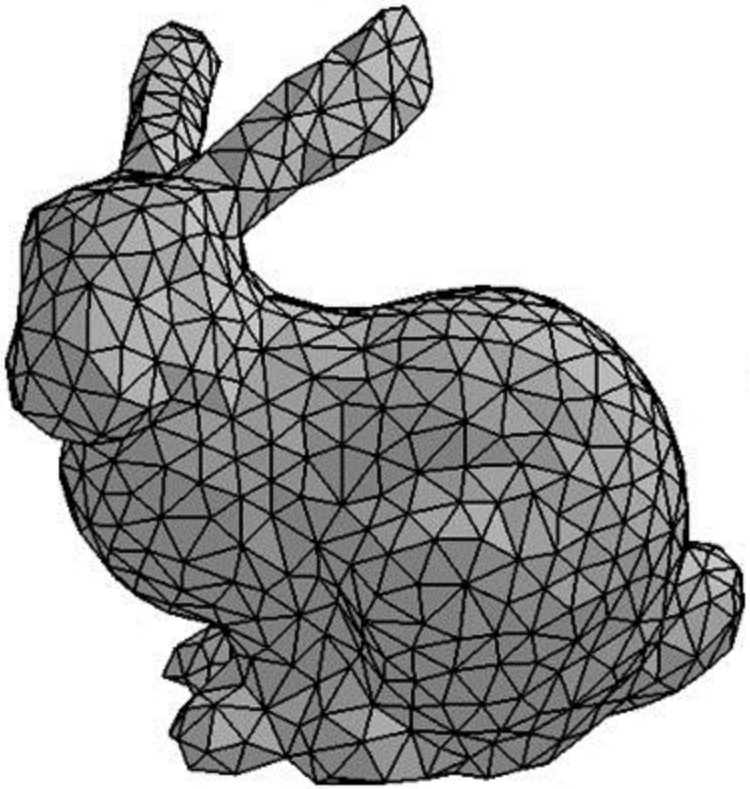
Integers

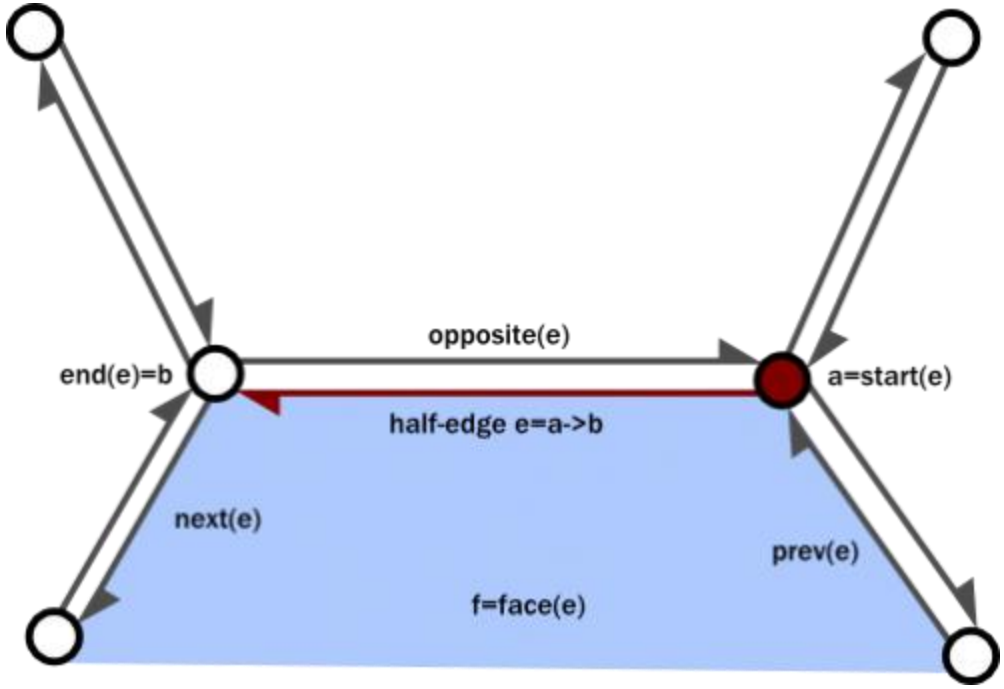
```
v1 v2 v3
v1 v2 v3
...
```

Hard query: "Neighbors of vertex v"

Medium level: Triangle/vertex lists

Aside: Storing Meshes

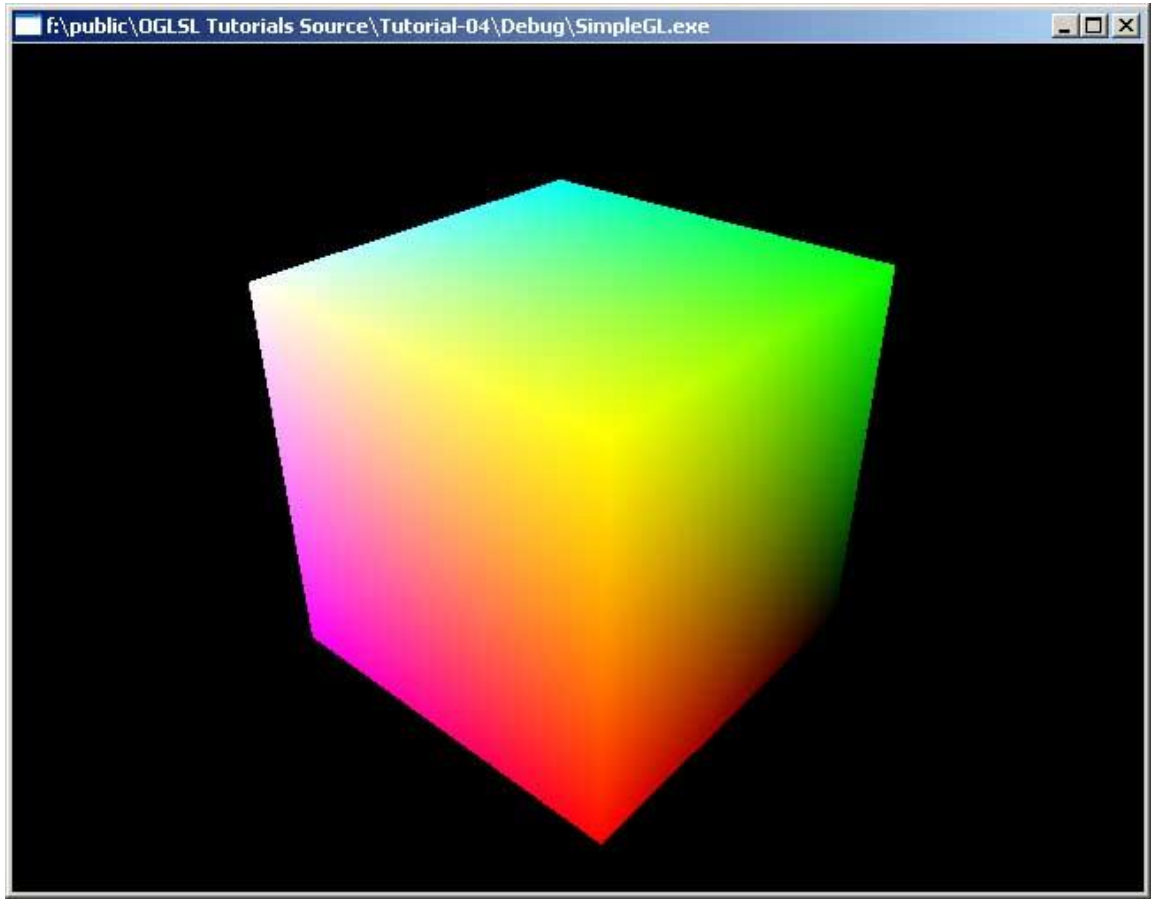




http://fgiesen.wordpress.com/2012/02/21/half-edge-based-mesh-representations-theory/

High level: Halfedge

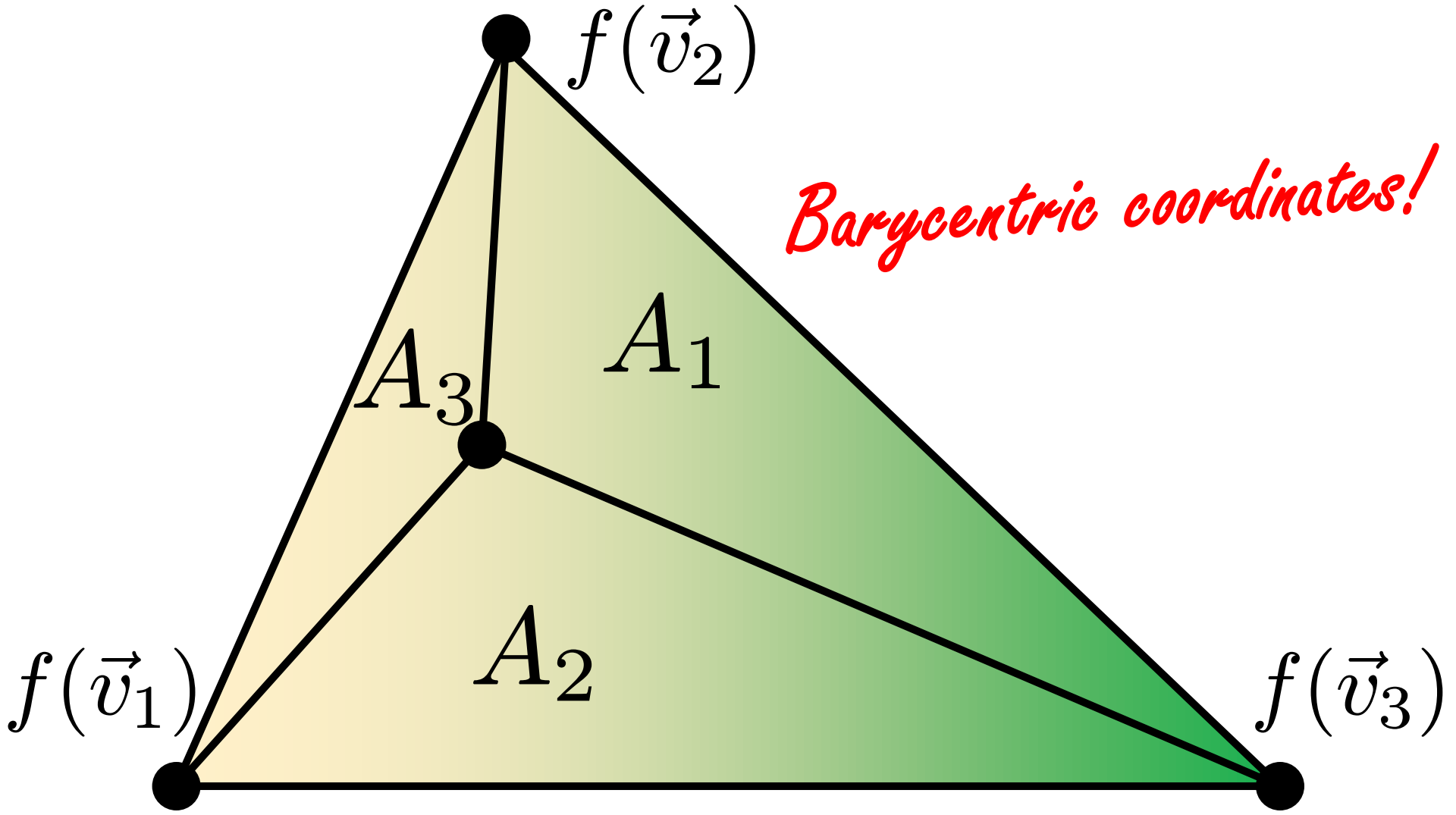
Shading Options



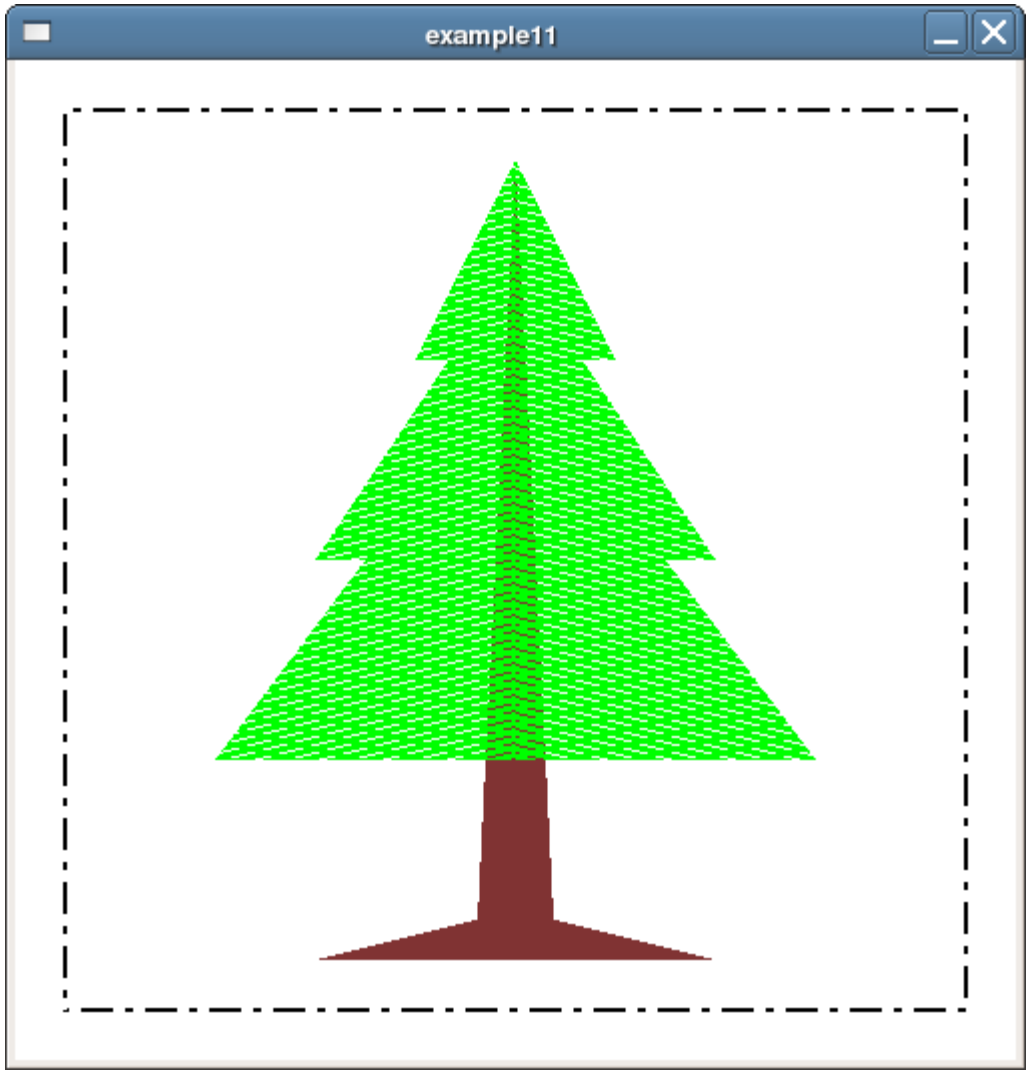
`glColor3f`

http://www.clockworkcoders.com/oglsl/tut4.jpg

Per-Vertex Shading



Shading Options

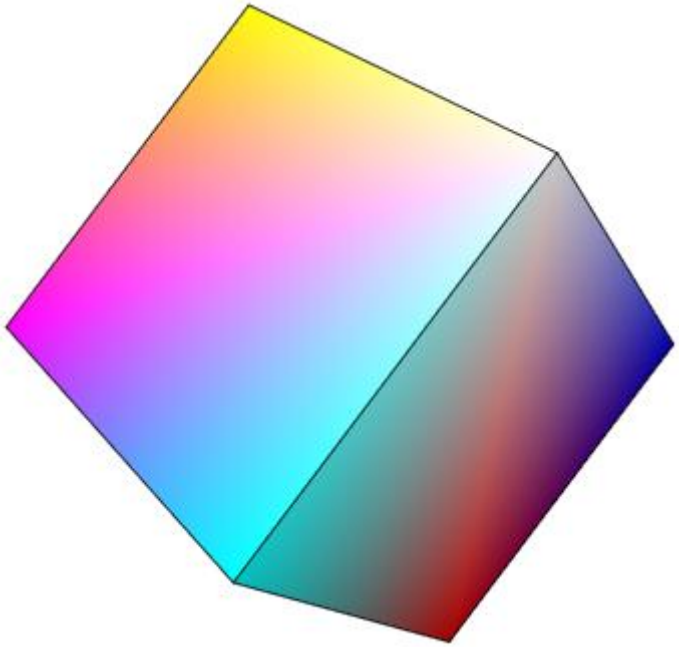


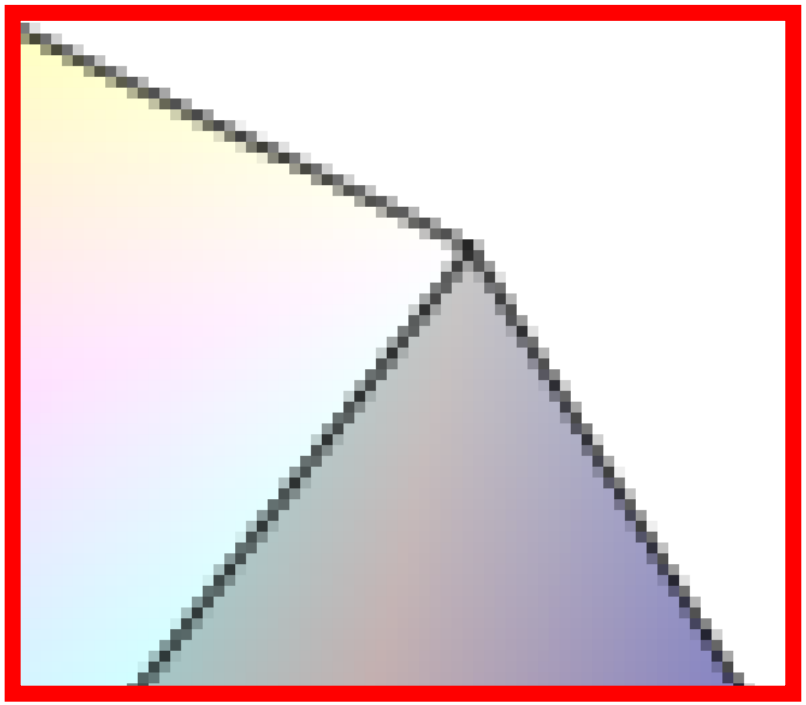
`glLineStipple`

`glPolygonStipple`

http://www.linuxcenter.ru/lib/images/opengl/example11.png

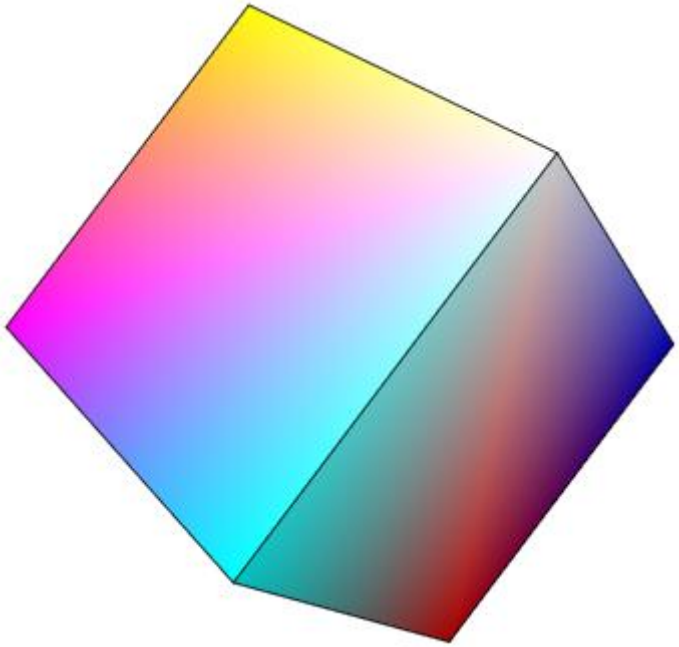
Shading Options

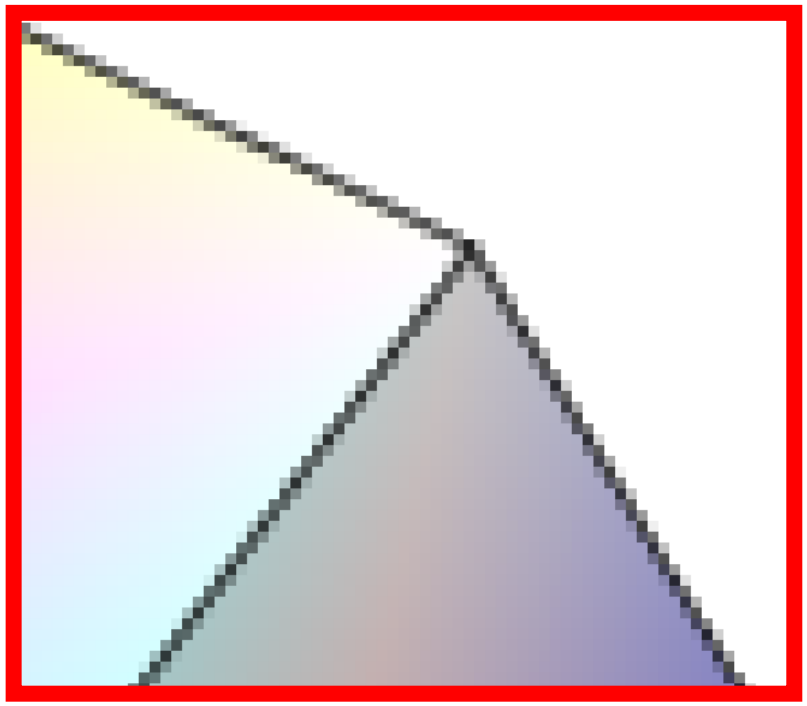




```
glHint(GL_LINE_SMOOTH_HINT, GL_NICEST)
```

Shading Options

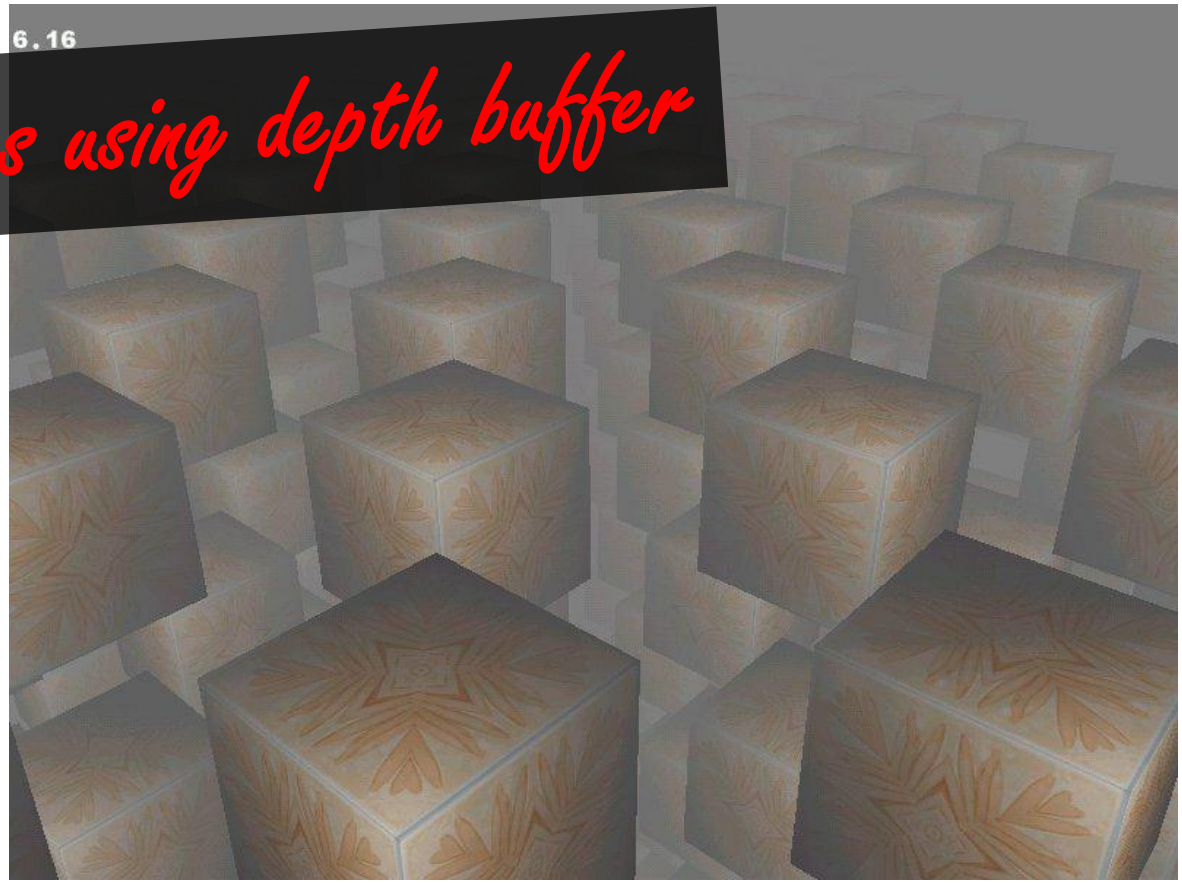




```
glHint(GL_LINE_SMOOTH_HINT, GL_NICEST)
```

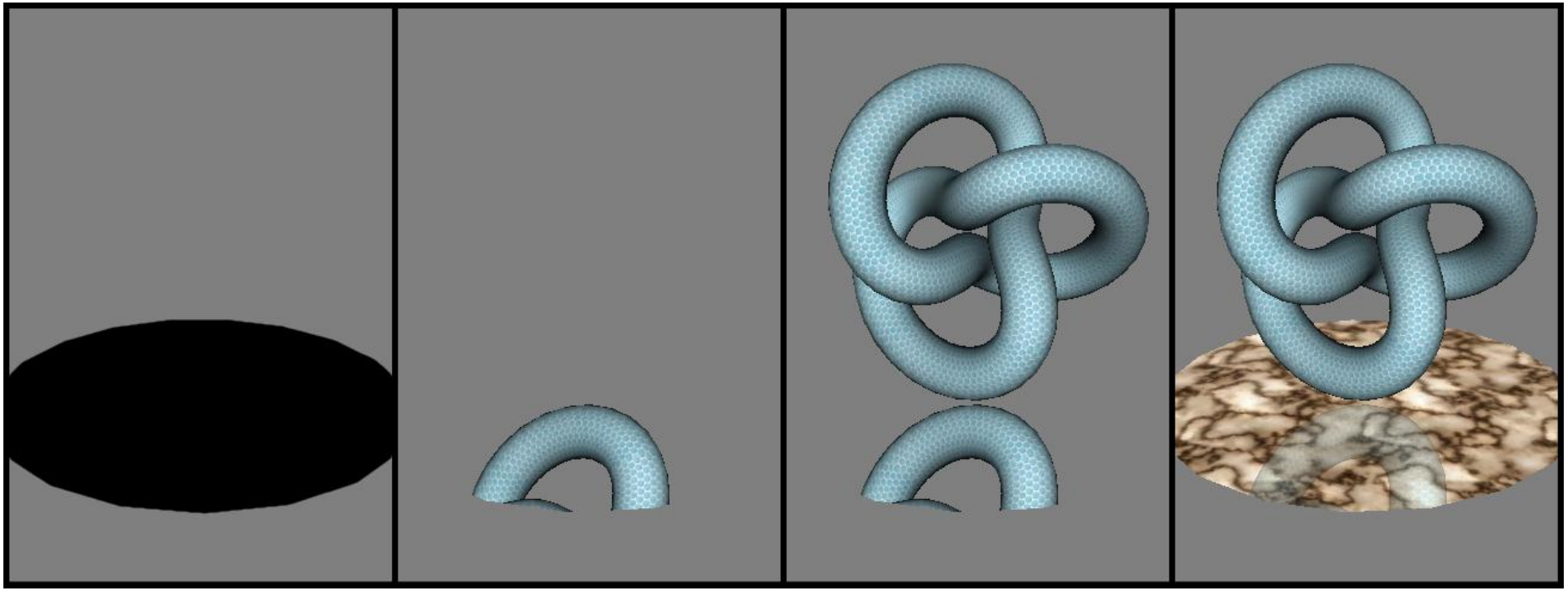
Shading Options

```
glFogi(GL_EXP,GL_FOG_START)
```



http://zanir.wz.cz/programming/opengl/img/004_Diplaylist_fog.jpg

Shading Options



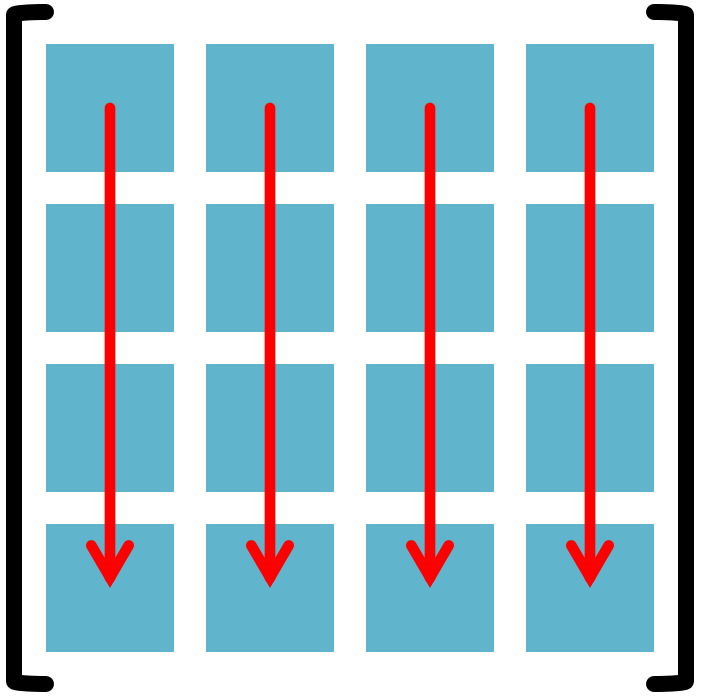
http://ofps.oreilly.com/titles/9780596804824/chblending.html

Stencil buffer

Explicit Transformations

```
void glLoadMatrixd(const GLdouble *m)
```

- Loads a 4x4 projective transformation matrix
- Stored in *column-major* format



Composing Transformations

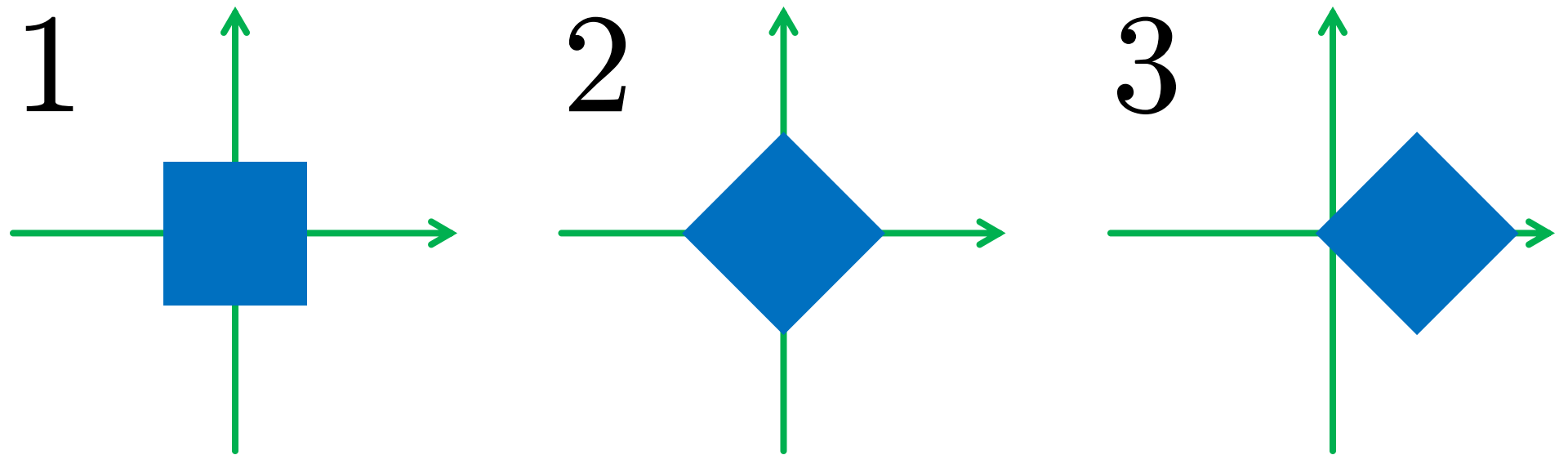
```
glLoadMatrixf(A);
glMultMatrixf(B);
glMultMatrixf(C);
```

$$\vec{v}_{new} = A \cdot B \cdot C \cdot \vec{v}$$

Composing Transformations

```
glLoadMatrixf(A); // translate right
glMultMatrixf(B); // rotate 45°
```

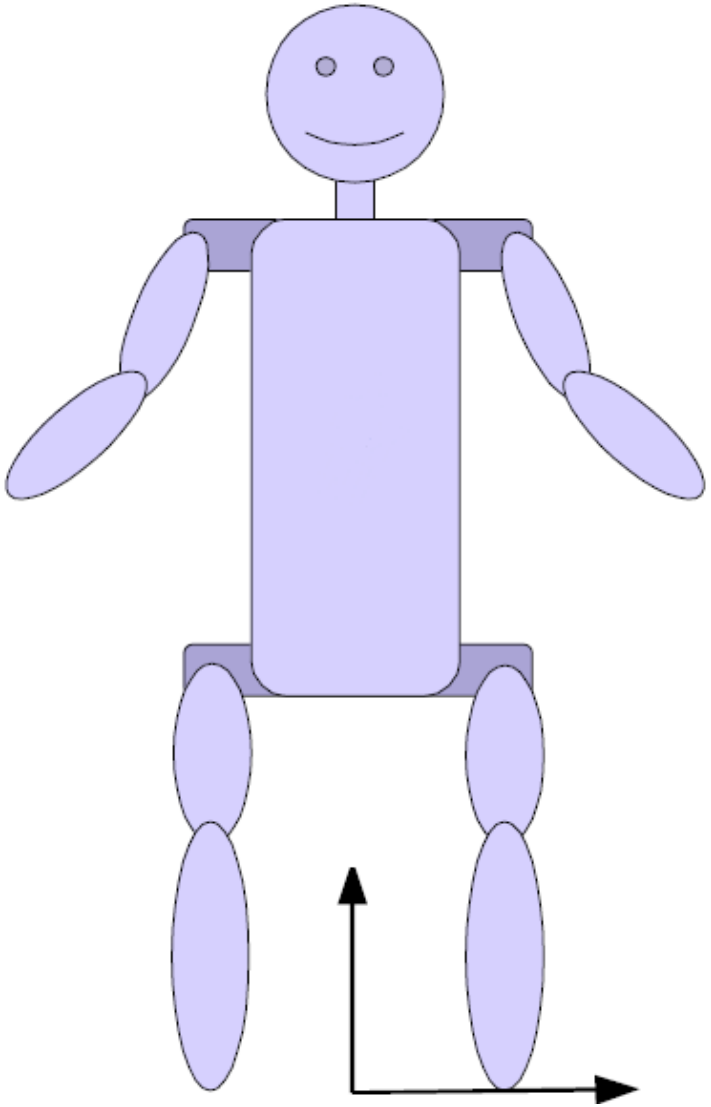
$$\vec{v}_{new} = A \cdot B \cdot \vec{v}$$



Convenience Functions

- `glTranslatef(tx, ty, tz)`
- `glRotatef(degrees, x, y, z)`
- `glScalef(sx, sy, sz)`

Hierarchical Modeling

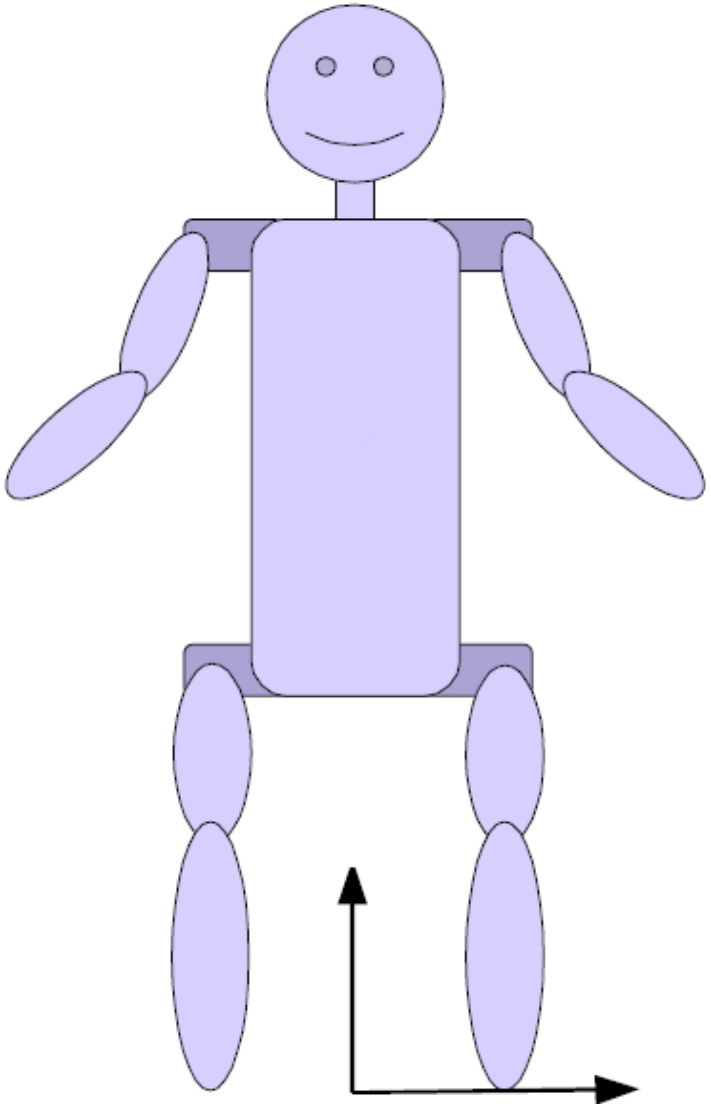


```
Body
  Torso
    Head
    Shoulder
      LeftArm
        UpperArm
        LowerArm
        Hand
      RightArm
        UpperArm
        LowerArm
        Hand
    Hips
      LeftLeg
        UpperLeg
        LowerLeg
        Foot
      RightLeg
        UpperLeg
        LowerLeg
        Foot
```

Work in local coordinate system

http://graphics.stanford.edu/courses/cs148-10-summer/docs/07_rendering2.pdf

Hierarchical Modeling

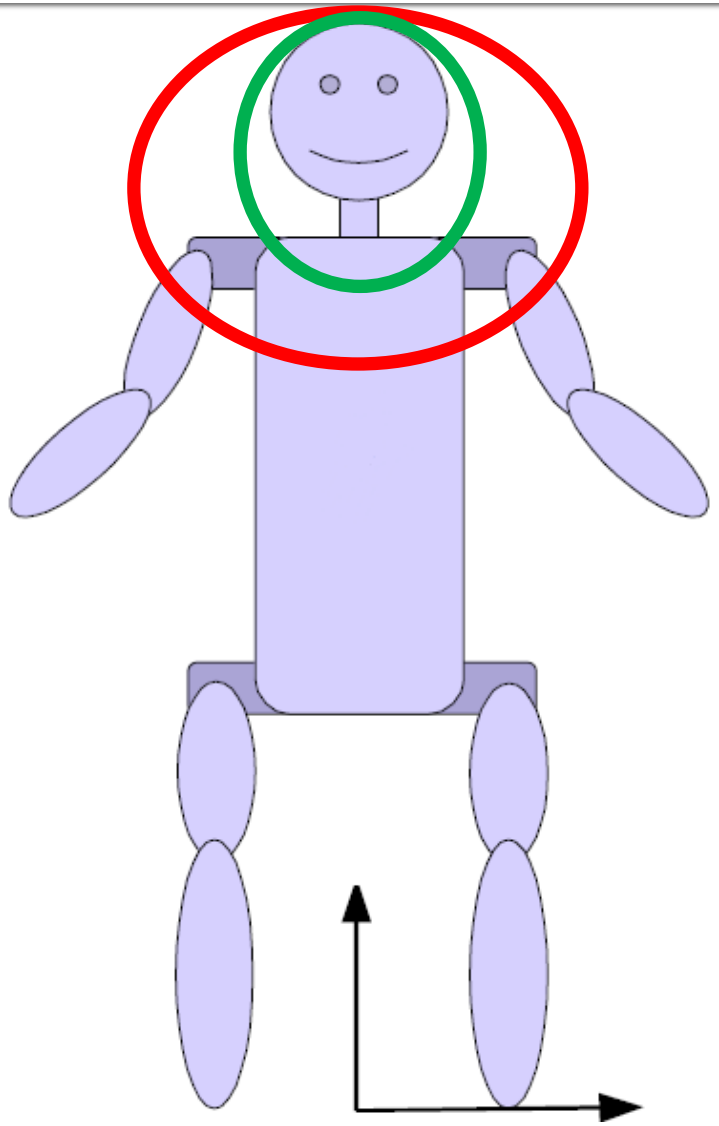


```
Body
  Torso
    Head
    Shoulder
      LeftArm
        UpperArm
        LowerArm
        Hand
      RightArm
        UpperArm
        LowerArm
        Hand
    Hips
      LeftLeg
        UpperLeg
        LowerLeg
        Foot
      RightLeg
        UpperLeg
        LowerLeg
        Foot
```

Repeated elements!

http://graphics.stanford.edu/courses/cs148-10-summer/docs/07_rendering2.pdf

Hierarchical Modeling



```
glTranslatef(0,1.5,0);
drawTorso();
glPushMatrix();
 glTranslatef(0,5,0);
 drawShoulder();
 glPushMatrix();
  glRotatef(neck_y,0,1,0);
  glRotatef(neck_x,1,0,0);
  drawHead();
 glPopMatrix();
 glPushMatrix();
  glTranslatef(1.5,0,0);
   glRotatef(l_shoulder_x);
   drawUpperArm(); ...
```

http://graphics.stanford.edu/courses/cs148-10-summer/docs/07_rendering2.pdf

OpenGL Transformation Model

$$T = M_{projection} \cdot M_{camera} \cdot M_{model}$$

Camera coordinates to normalized device coordinates

Local coordinates to camera coordinates

OpenGL Transformation Model

$$T = M_{projection} \cdot M_{camera} \cdot M_{model}$$

Camera coordinates to normalized device coordinates

GL_PROJECTION

Local coordinates to camera coordinates

GL_MODELVIEW

```
glMatrixMode(GL_PROJECTION);
glMatrixMode(GL_MODELVIEW);
```

Applied to all geometry

OpenGL Transformation Model

$$T = M_{projection} \cdot M_{camera} \cdot M_{model}$$

Camera coordinates to normalized device coordinates

Local coordinates to camera coordinates

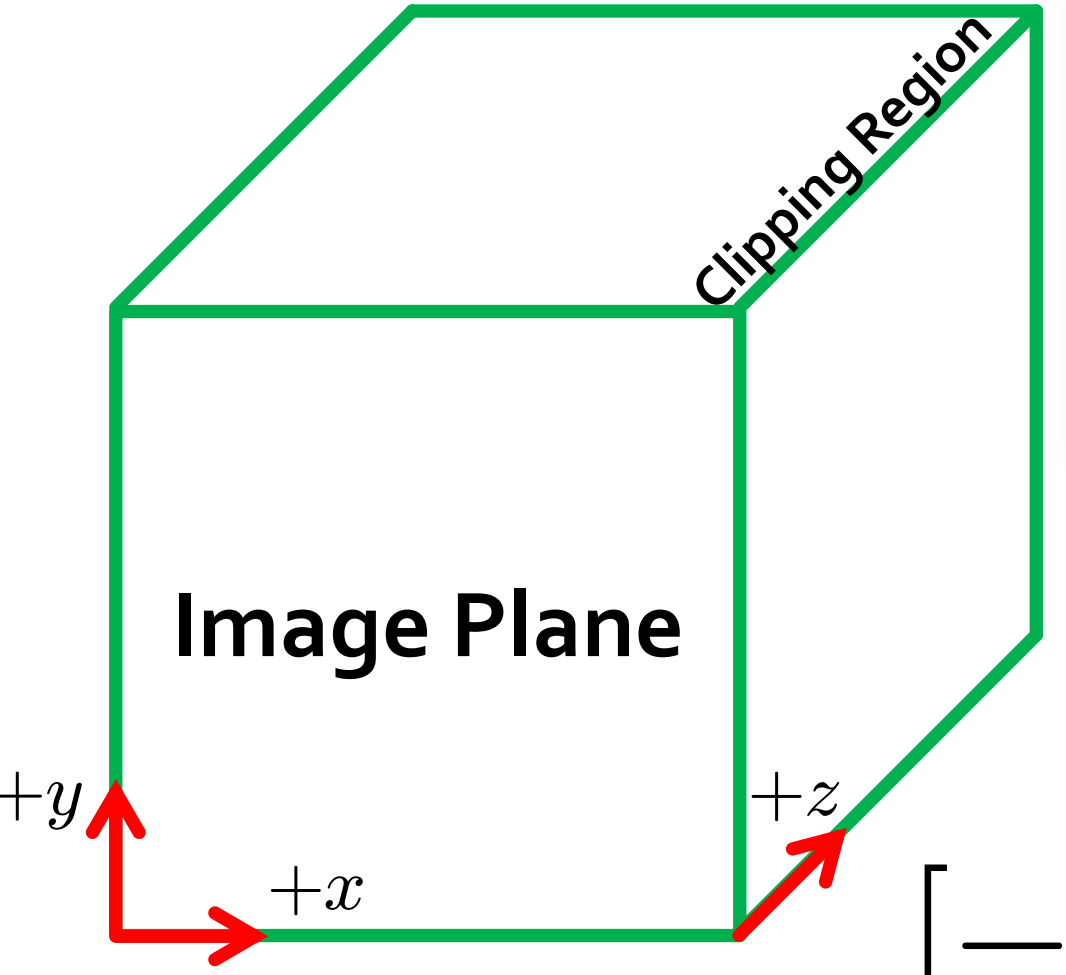
GL_PROJECTION **GL_MODELVIEW**

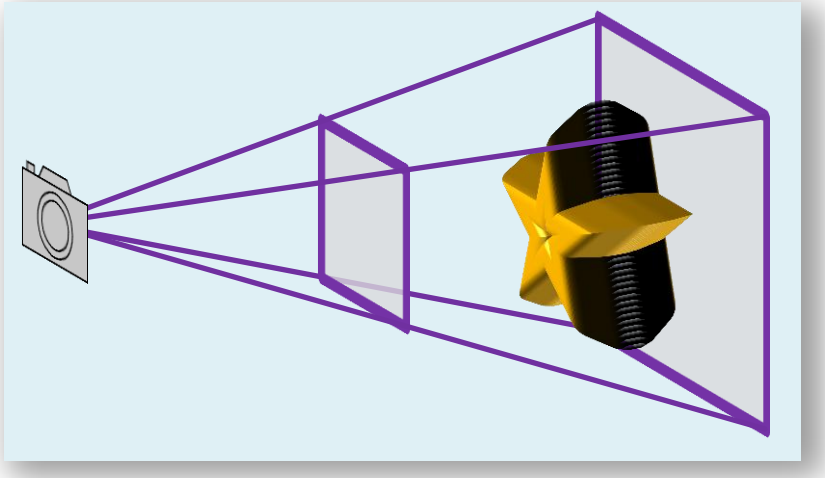
Can do whatever you want with these.

```
glMatrixMode(GL_PROJECTION);
glMatrixMode(GL_MODELVIEW);
```

Applied to all geometry

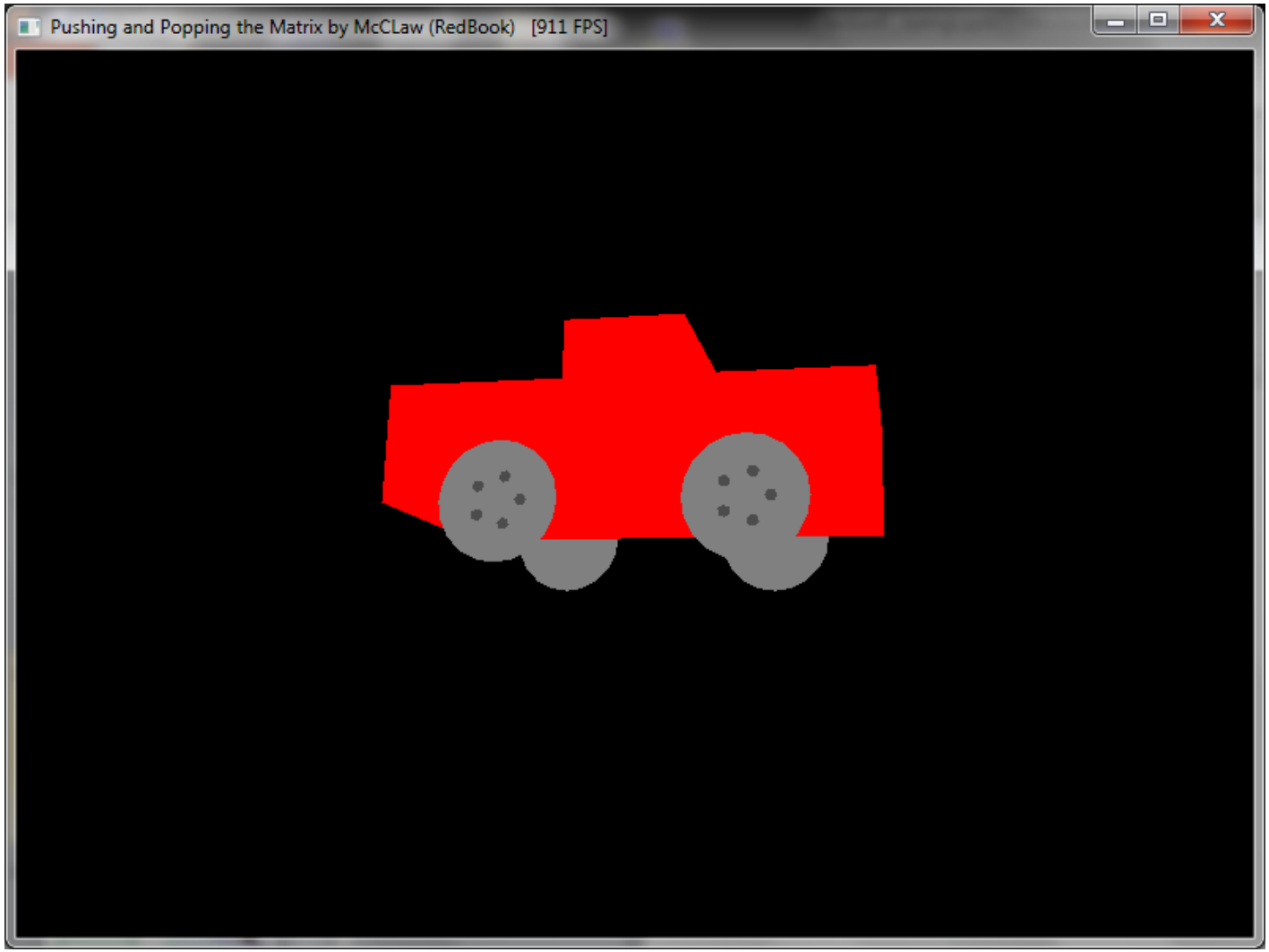
Normalized Device Coordinates





$$[-1, 1]^3$$

Transformation Example

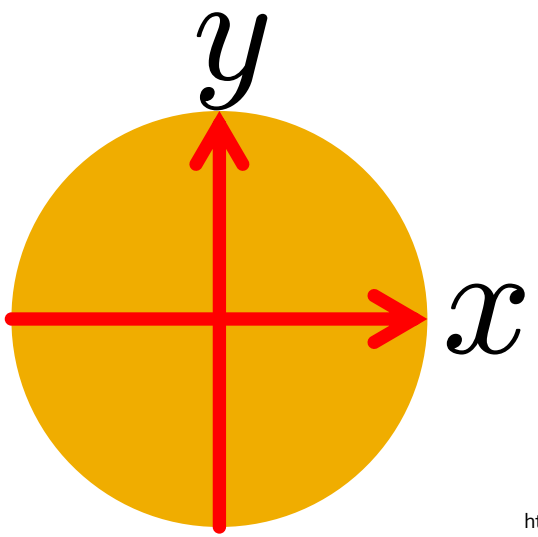


http://www.sulaco.co.za/pushing_and_popping_the_matrix_stack_tutorial.htm

Transformation Example

```
procedure DrawTriangleFan(CirclePoints: GLint; scale : GLfloat );
var i : GLint;
    Angle : GLfloat;
begin
  glBegin(GL_TRIANGLE_FAN);
    for i := 0 to CirclePoints - 1 do
    begin
      Angle := 2*PI*i/CirclePoints;
      glVertex2f(cos(angle)*scale, sin(angle)*scale); end;
  glEnd();
end;
```

Delphi!



http://www.sulaco.co.za/pushing_and_popping_the_matrix_stack_tutorial.htm

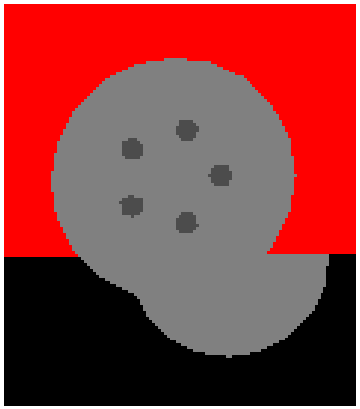
Transformation Example

```
procedure DrawWheel();
begin
  DrawTriangleFan(20,0.5);
end;

procedure DrawBolt();
begin
  DrawTriangleFan(6,0.05);
end;
```

http://www.sulaco.co.za/pushing_and_popping_the_matrix_stack_tutorial.htm

Transformation Example

```
procedure DrawWheelAndBolts(RightSide : boolean);
var i : GLint;
begin
  glColor3f(0.5,0.5,0.5);
  DrawWheel();
  for i := 0 to 4 do
  begin
    glPushMatrix();
      glRotatef(72.0*i,0.0,0.0,1.0);
      if RightSide then
        glTranslatef(0.2,0.0,0.01)
      else
        glTranslatef(0.2,0.0,-0.01);
      glColor3f(0.3,0.3,0.3);
      DrawBolt();
    glPopMatrix();
  end;
end;
```



http://www.sulaco.co.za/pushing_and_popping_the_matrix_stack_tutorial.htm

Transformation Example

```
procedure DrawBodyAndWheelAndBolts();
begin
    glColor3f(1.0,0.0,0.0);
    DrawCarBody();
    glPushMatrix();
       glTranslatef(-1.0,-0.5,1.01);   DrawWheelAndBolts(True);
    glPopMatrix();
    glPushMatrix();
       glTranslatef(-1.0,-0.5,-1.01); DrawWheelAndBolts(False);
    glPopMatrix();
    glPushMatrix();
       glTranslatef(1.0,-0.5,-1.01);   DrawWheelAndBolts(False);
    glPopMatrix();
    glPushMatrix();
       glTranslatef(1.0,-0.5,1.01);    DrawWheelAndBolts(True);
    glPopMatrix();
end;
```

http://www.sulaco.co.za/pushing_and_popping_the_matrix_stack_tutorial.htm

http://upload.wikimedia.org/wikipedia/commons/b/b9/JOGL_Refrection_Demo_Screenshot.png

The OpenGL Pipeline

CS 148, Summer 2012
Introduction to Computer Graphics and Imaging

Justin Solomon