Computer Graphics

• (Realistic) pictorial synthesis of real and/or imaginary objects from their computer-based models (datasets)
• It typically includes modeling, rendering (graphics pipeline), and human-computer interaction
• So, we are focusing on computer graphics hardware, software, and mathematical foundations
• Computer Graphics is computation
  – A new method of visual computing
• Why is Computer Graphics useful and important?
• Course challenges: more mathematics oriented, programming requirements, application-driven, inter-disciplinary in nature, etc.
Computer Graphics Systems

Graphical Models → Rendering

Rendering Parameters → Rendering

Rendering → Output Device
Output Devices

• **Vector Devices**
  – Lasers (for example)

• **Raster Devices**
  – CRT, LCD, bitmaps, etc.
  – Most output devices are 2D
  – Can you name any 3D output devices?
Graphical Models

- 2D and 3D objects
  - Triangles, quadrilaterals, polygons
  - Spheres, cones, boxes

- Surface characteristics
  - Color, reaction to light
  - Texture, material properties

- Composite objects
  - Other objects and their relationships to each other

- Lighting, fog, etc.

- Much, much more...
Rendering

• Conversion of 3D model to 2D image
  – Determine where the surfaces “project” to
  – Determine what every screen pixel might see
  – Determine the color of each surface
Rendering Parameters

- **Camera parameters**
  - Location
  - Orientation
  - Focal length
2D Graphics vs. 3D Graphics

• 2D
  – X, Y - 2 dimensions only
  – We won’t spend time on 2D graphics in this course

• 3D
  – X, Y, and Z
  – Space

• Rendering is typically the conversion of 3D to 2D
3D Coordinate Systems

Right-Hand Coordinate System

OpenGL uses this!
Left-Hand Coordinate System

Direct3D uses this!
How to Model/Render This?
Render/Display a Box in OpenGL

- We render the 6 faces as **polygons**
  - Polygons are specified as a list of vertices
  - Vertices are specified in counter-clockwise order looking at the surface of the face!
Visualizing in 3D

Counter-clockwise
OpenGL Conventions

• C library
  – All function names start with gl

• OpenGL is a retained mode graphics system
  – It has a state
  – glBegin(GL_POLYGON) puts us into a polygon rendering state
OpenGL Polygon Rendering

```c
GLdouble size = 1.0;

glBegin(GL_POLYGON);  // front face
    glVertex3d(0.0, 0.0, size);
    glVertex3d(size, 0.0, size);
    glVertex3d(size, size, size);
    glVertex3d(0.0, size, size);
glEnd();
```
OpenGL Types

• Basic numeric types
  – GLdouble = double
  – GLfloat = float
  – GLint = int
  – GLshort = short

• Mostly, you’ll use GLdouble and GLfloat
Defined \texttt{glVertex3fv}

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Function</th>
<th>#Parms</th>
<th>Type</th>
<th>Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{gl}</td>
<td>Vertex</td>
<td>1</td>
<td>\texttt{f (float)}</td>
<td>\texttt{v (vector)}</td>
</tr>
<tr>
<td>\texttt{glu}</td>
<td>Begin</td>
<td>2</td>
<td>\texttt{d (double)}</td>
<td></td>
</tr>
<tr>
<td>\texttt{wgl}</td>
<td>End</td>
<td>3</td>
<td>\texttt{i (integer)}</td>
<td></td>
</tr>
<tr>
<td>\texttt{agl}</td>
<td>Lighting</td>
<td>4</td>
<td>\texttt{b (byte)}</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>\texttt{s (short)}</td>
<td></td>
</tr>
</tbody>
</table>
Function Suffixes

• Many functions have alternatives
  – Alternatives are specified by the suffix
  – glVertex2d
    • 2 double parameters
    • void glVertex2d(GLdouble x, GLdouble y);
  – glVertex3f
    • 3 float parameters
    • void glVertex3f(GLfloat x, GLfloat y, GLfloat z);
  – glVertex3fv
    • void glVertex3fv(const GLfloat *v);
All of Them...

• glVertex2d, glVertex2f, glVertex2i, glVertex2s,
  glVertex3d, glVertex3f, glVertex3i, glVertex3s,
  glVertex4d, glVertex4f, glVertex4i, glVertex4s,
  glVertex2dv, glVertex2fv, glVertex2iv, glVertex2sv,
  glVertex3dv, glVertex3fv, glVertex3iv, glVertex3sv,
  glVertex4dv, glVertex4fv, glVertex4iv, glVertex4sv
Vector Parameters

GLdouble a[ ] = {0, 0, 1};
GLdouble b[ ] = {1, 0, 1};
GLdouble c[ ] = {1, 1, 1};
GLdouble d[ ] = {0, 1, 1};

glBegin(GL_POLYGON); // front face
  glVertex3dv(a);
  glVertex3dv(b);
  glVertex3dv(c);
  glVertex3dv(d);
glEnd();
Specify a Color (No Lighting)

- `glColor3f(red, green, blue);`
- **Most of the same suffixes apply...**

```c
GLdouble size = 1.0;

glColor3d(1.0, 0.0, 0.0); // red

glBegin(GL_POLYGON); // front face
    glVertex3d(0.0, 0.0, size);
    glVertex3d(size, 0.0, size);
    glVertex3d(size, size, size);
    glVertex3d(size, 0.0, size);
    glVertex3d(size, size, size);
    glVertex3d(size, 0.0, size);
    glVertex3d(size, size, size);
    glVertex3d(size, 0.0, size);
    glVertex3d(size, size, size);
    glVertex3d(size, 0.0, size);
    glVertex3d(size, size, size);
    glVertex3d(size, 0.0, size);
    glVertex3d(size, size, size);
    glVertex3d(size, 0.0, size);
    glEnd();
```

Colors range from 0 to 1
How to Model/Render This?
Top View

Front View

Side View

X Z

Y

a b
c d
e f
gh
i j
e f
m n
f g
n o
3.00
1.00 1.00 1.00
YY
X Z
Z
X
3.00
1.00
1.00
1.00
1.00
Labels

Top View

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>c</td>
</tr>
</tbody>
</table>

h
g
e
f

Bottom labels

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>l</td>
<td>k</td>
</tr>
</tbody>
</table>

p
o
m
n

Front View

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>j</td>
</tr>
</tbody>
</table>

m
n

Side View

<table>
<thead>
<tr>
<th>f</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>o</td>
</tr>
</tbody>
</table>
The Basic Idea

• **Describe an object using surfaces**
• **Surfaces are polygons**
  – Triangles, quadrilaterals, whatever
  – Important thing is that they are flat
  – They must also be convex
• **Provide points in counter-clockwise order**
  – From the visible side