University of British Columbia

Geometric Transformations

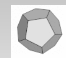
and quick OpenGL intro

Project 0

University of British Columbia

Get used to OpenGL:

- compile and run template
- change draw routine to dodecahedron
- add color change on mouse click



OpenGL State

University of British Columbia

State machine:

- set state variables, including color

Program structure:

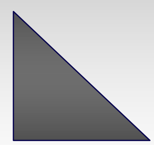
- graphics initialization
- draw routine
  - *specify geometric type*
  - *provide vertices*

Rendering Triangles

University of British Columbia

Example:

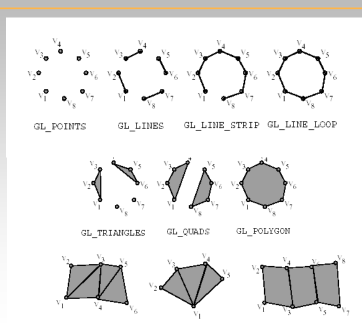
```
glBegin( GL_TRIANGLES );
    glColor3f( 1.0, 0.0, 0.0 );
    glVertex3f( 0.0, 1.0, 0.0 );
    glColor3f( 0.0, 0.0, 1.0 );
    glVertex3f( 0.0, 0.0, 0.0 );
    glVertex3f(1.0, 0.0, 0.0 );
glEnd();
```



Every vertex gets the color, etc. that corresponds to the last specified value.

Points, lines, polygons

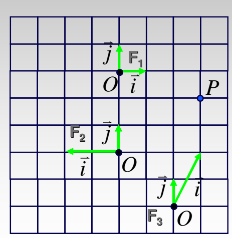
University of British Columbia



Math Review

University of British Columbia

Working with Frames



$$P = O + x\vec{i} + y\vec{j}$$

F_1 P(3,-1)

F_2 P(-1.5,2)

F_3 P(1,2) y=4? no! y=2

j has horiz and vert components

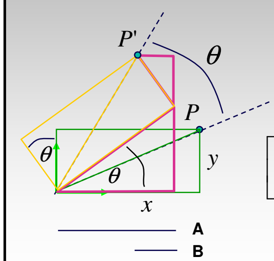
Transformations

University of British Columbia

Rotation

$Rotate(z,\theta)$

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$P_x' = A - B = x\cos\theta - y\sin\theta$

2D Transformations

University of British Columbia

Let $P = \begin{bmatrix} x \\ y \end{bmatrix}$ and $P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$ ← column vectors

Translation:

$T(d_x, d_y) = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$ $\quad P + T(\ldots) = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} x + d_x \\ y + d_y \end{bmatrix} = P'$ (vector addition)

Scaling:

$S(s_x, s_y) = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$ $\quad S(\ldots) \cdot P = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \end{bmatrix}$ (matrix multiplication)

2D transformations

University of British Columbia

Shears:

$SH_x(a) = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}$ $\quad P' = SH_x(a) \cdot P = \ldots = \begin{bmatrix} x + ay \\ y \end{bmatrix}$

$SH_y(b) = \begin{bmatrix} 1 & 0 \\ b & 1 \end{bmatrix}$

Challenge

University of British Columbia

Matrix multiplication

- for everything except translation
- how to do everything with multiplication?
  - *then just do composition, no special cases*

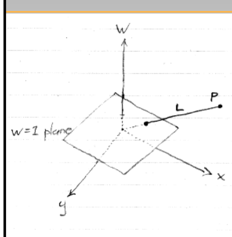
Homogeneous coordinates to the rescue

- 2D cartesian (x,y) --> 3D homogeneous (x,y,w)

homogeneous		cartesian
(x, y, w)	$\xrightarrow{/w}$	$(\frac{x}{w}, \frac{y}{w})$

Homogeneous coordinates

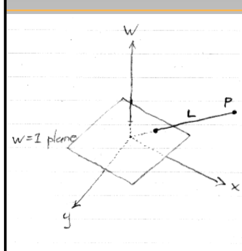
University of British Columbia



- point in 2D cartesian + weight w = point P in 3D homog. coords
- multiples of (x,y,w)
  - represent same point in 2D cartesian
  - a line L in 3D homog
- homogenize a point in 3D:
  - divide by w to get (x/w, y/w, 1)
  - projects point onto w=1 plane

Homogeneous coordinates

University of British Columbia

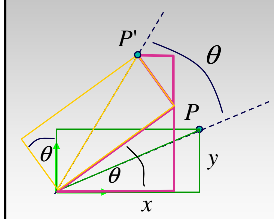


- w=0 denotes point at infinity
  - think of as direction
  - cannot be homogenized
  - lies on x-y plane
- (0,0,0) is not allowed

Transformations

University of British Columbia

Rotation



$Rotate(z,\theta)$

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & & \\ \sin\theta & \cos\theta & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

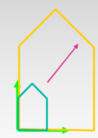
glRotatef(angle,x,y,z);
glRotated(angle,x,y,z);

Transformations

University of British Columbia

Scaling

scale(a,b,c)



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & & & \\ & b & & \\ & & c & \\ & & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

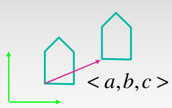
glScalef(a,b,c);
glScaled(a,b,c);

Transformations

University of British Columbia

Translation

translate(a,b,c)



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & & & a \\ & 1 & & b \\ & & 1 & c \\ & & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

glTranslatef(a,b,c);
glTranslated(a,b,c);