ECE4893A/CS4803MPG: Multicore and GPU Programming for Video Games

3D to 2D Projection

Prof. Aaron Lanterman
(Based on slides by Prof. Hsien-Hsin Sean Lee)
School of Electrical and Computer Engineering
Georgia Institute of Technology

Georgia Institute of Technology

Specifying the view transformation

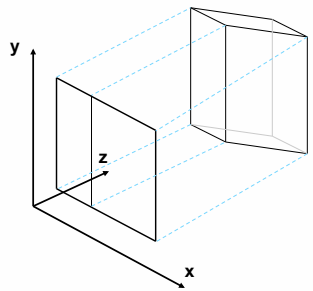
- Most commonly parameterized by:
  - Position of camera
  - Position of point to look at
  - Vector indicating "up" direction of camera
- In Direct3D: `D3DXMatrixLookAtLH`
  - D3D uses a LHS, but also have `D3DXMatrixLookAtRH`
- In XNA: `Matrix.CreateLookAt` (RHS)
- In OpenGL: `gluLookAt` (RHS)
- Can also build a rotation+translation matrix as if the camera was an object in scene, then take the inverse of that matrix

msdn.microsoft.com/en-us/library/bb205342(VS.85).aspx
msdn.microsoft.com/en-us/library/bb205343(VS.85).aspx

Georgia Institute of Technology

Projection from 3D space

- Projection transforms 3D geometry into a form that can be rendered as a 2D image

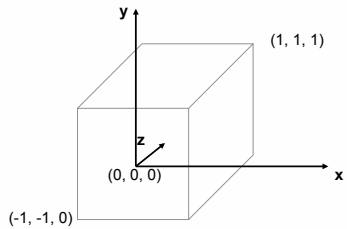


Much discussion adapted from Joe Farrell's article (http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__1/)

Georgia Institute of Technology

Canonical view volume

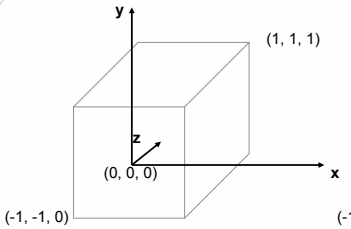
- Projection transforms your geometry into a **canonical view volume** in *normalized device coordinates*
- Only X- and Y-coordinates will be mapped onto the screen
- Z will be almost useless, but used for depth test



Canonical view volume (LHS)
(-1, -1, 0) to (1,1,1) used by Direct3D

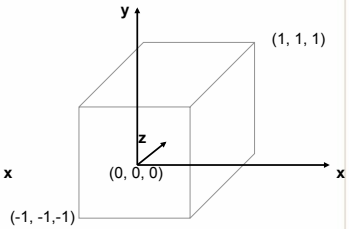
Georgia Institute of Technology

Strange "conventions"



Canonical view volume (LHS)
(-1, -1, 0) to (1,1,1) used by D3D/XNA

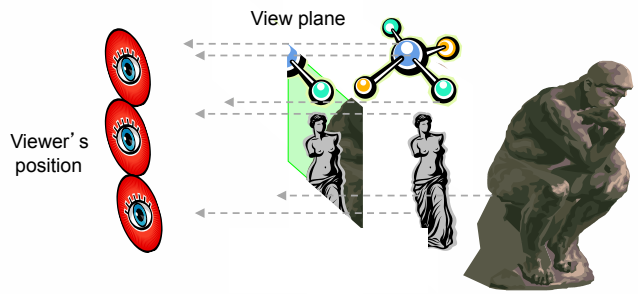
(remember eye-space coordinates in XNA are in a RHS!!!)



Canonical view volume (LHS)
(-1, -1, 1) to (1,1,1) used by OpenGL

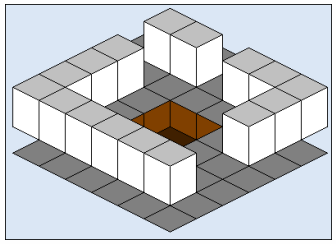
(remember eye-space coordinates in OpenGL are in a RHS!!!)

Orthographic (or parallel) projection



- Project from 3D space to the viewer's 2D space

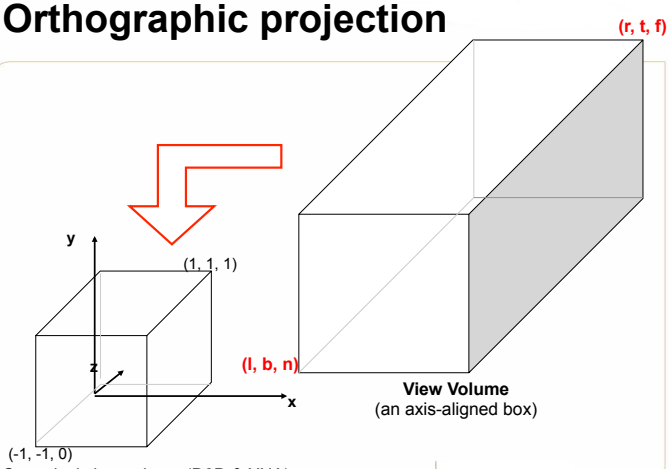
Style of orthographic projection



- Same size in 2D and 3D
- No sense of distance
- Parallel lines remain parallel
- Good for tile-based games where camera is in fixed location (e.g., Mahjong or 3D Tetris)

See http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__2/Deriving-Projection-Matrices.htm

Orthographic projection



Canonical view volume (D3D & XNA)

Orthographic projection math (1)

- Derive x' and y'

$x \in [l, r] \quad x' \in [-1, 1]$

$l \le x \le r$

$0 \le x - l \le r - l$

$0 \le \frac{x-l}{r-l} \le 1$

$0 \le \frac{2(x-l)}{r-l} \le 2$

$-1 \le \frac{2(x-l)}{r-l} - 1 \le 1$

$-1 \le \frac{2x - 2l - r + l}{r-l} \le 1$

$-1 \le \frac{2x}{r-l} - \frac{r+l}{r-l} \le 1$

$\boxed{\therefore \; x' = \frac{2x}{r-l} - \frac{r+l}{r-l}}$ $\boxed{y' = \frac{2y}{t-b} - \frac{t+b}{t-b}}$

See http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__2/Deriving-Projection-Matrices.htm

Orthographic projection math (2)

- Derive z' (slightly different for the range in D3D)

$z \in [n, f] \quad z' \in [0, 1]$

$n \le z \le f$

$0 \le z - n \le f - n$

$0 \le \frac{z-n}{f-n} \le 1$

$0 \le \frac{z}{f-n} - \frac{n}{f-n} \le 1$

$\boxed{\therefore \; z' = \frac{z}{f-n} - \frac{n}{f-n}}$

- OpenGL transform for z looks more like x & y transforms

See http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__2/Deriving-Projection-Matrices.htm

Ortho projection matrix (LHS)

- Put all together

$$[x', y', z', 1] = [x, y, z, 1] \cdot P \text{ where } P = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{1}{f-n} & 0 \\ -\frac{r+l}{r-l} & -\frac{t+b}{t-b} & -\frac{n}{f-n} & 1 \end{bmatrix}$$

See http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__2/Deriving-Projection-Matrices.htm

Ortho proj (LHS) Microsoft style

- Rearranging to look like Microsoft documentation

$$[x', y', z', 1] = [x, y, z, 1] \cdot P \text{ where } P = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{1}{f-n} & 0 \\ \frac{l+r}{l-r} & \frac{t+b}{b-t} & \frac{n}{n-f} & 1 \end{bmatrix}$$

- In Direct3D: D3DXMatrixOrthoOffCenterLH(*o,l,r,b,t,n,f)
  - LHS is default system in Direct3D

http://msdn.microsoft.com/en-us/library/bb205347(VS.85).aspx

Orthographic projection (RHS)

- Math the same, but z clipping plane inputs in most API calls are negated so z input parameters are positive

$$[x',y',z',1] = [x,y,z,1]\cdot P \text{ where } P = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \boxed{\frac{1}{n-f}} & 0 \\ \frac{l+r}{l-r} & \frac{t+b}{b-t} & \frac{n}{n-f} & 1 \end{bmatrix}$$

- In Direct3D: D3DXMatrixOrthoOffCenterRH(*o,l,r,b,t,n,f)
- In XNA: Matrix.CreateOrthographicOffCenter(l,r,b,t,n,f)
- In OpenGL: glOrtho(l,r,b,t,n,f) (matrix is different)
  - OpenGL maps z to [-1,1] & uses column vectors

http://msdn.microsoft.com/en-us/library/bb205348(VS.85).aspx
http://www.cs.utk.edu/~vose/c-stuff/opengl/glOrtho.html

Georgia Institute of Technology 13

Simpler ortho projection (LHS)

- In most orthographic projection setups
  - Z-axis passes through the center of your view volume
  - Field of view (FOV) extends equally far
    - To the *left* as to the *right* **(i.e., r = -l)**
    - To the *top* as to the *below* **(i.e., t=-b)**

$$[x',y',z',1] = [x,y,z,1]\cdot P \text{ where } P = \begin{bmatrix} \frac{2}{w} & 0 & 0 & 0 \\ 0 & \frac{2}{h} & 0 & 0 \\ 0 & 0 & \frac{1}{f-n} & 0 \\ 0 & 0 & \frac{n}{n-f} & 1 \end{bmatrix}$$

- In Direct3D: D3DXMatrixOrthoLH(*o,w,h,n,f)

See http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__2/Deriving-Projection-Matrices.htm

Georgia Institute of Technology 14

Simpler ortho projection (RHS)

- Math the same, but z clipping plane inputs in most API calls are negated so z input parameters are positive

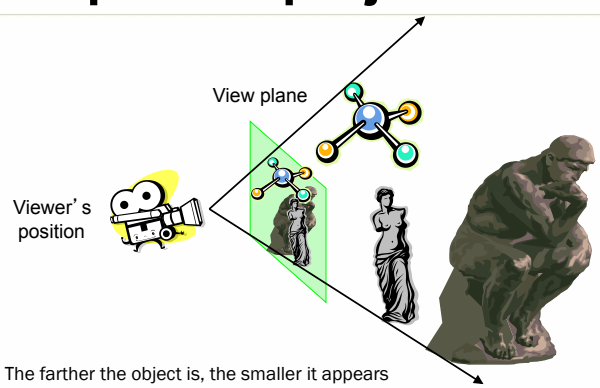
$$[x',y',z',1] = [x,y,z,1]\cdot P \text{ where } P = \begin{bmatrix} \frac{2}{w} & 0 & 0 & 0 \\ 0 & \frac{2}{h} & 0 & 0 \\ 0 & 0 & \boxed{\frac{1}{n-f}} & 0 \\ 0 & 0 & \frac{n}{n-f} & 1 \end{bmatrix}$$

- In Direct3D: D3DXMatrixOrthoRH(*o,w,h,n,f)
- In XNA: Matrix.CreateOrthographic(w,h,n,f)

See http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__2/Deriving-Projection-Matrices.htm

Georgia Institute of Technology 15

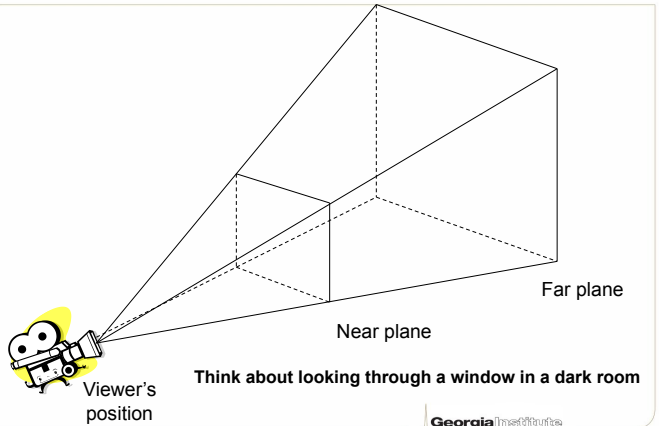
Perspective projection



- The farther the object is, the smaller it appears
- Some photo editing software allows you to perform "Perspective Correction"

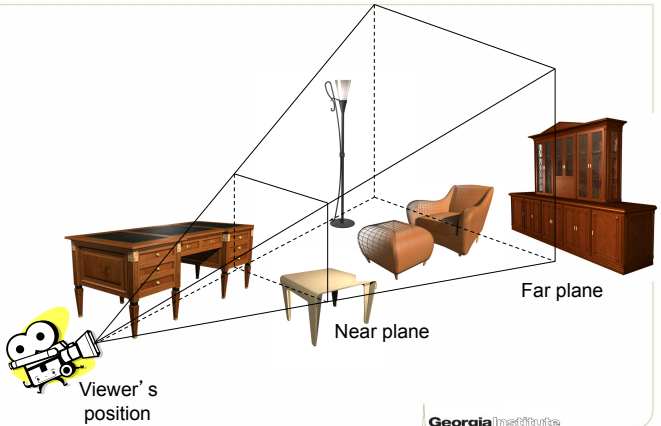
Georgia Institute of Technology 16

Viewing frustum

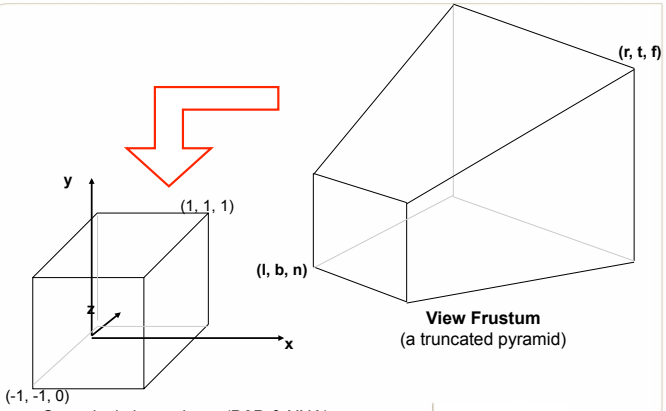


Think about looking through a window in a dark room

Viewing frustum with furniture

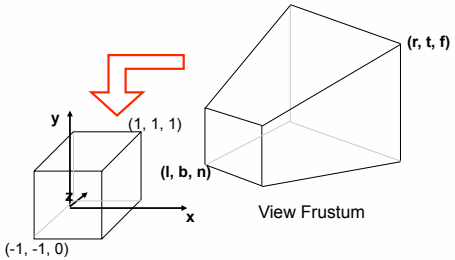


Perspective projection



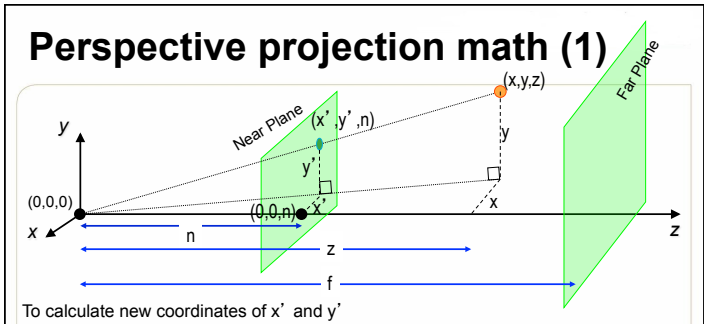
Perspective projection mapping

- Given a point (x,y,z) within the view frustum, project it onto the near plane z=n
  - x ∈ [l, r] and y ∈ [b, t]
- We will map x from [l,r] to [-1,1] and y from [b,t] to [-1,1]



See http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123_3/Deriving-Projection-Matrices.htm

Perspective projection math (1)



To calculate new coordinates of x' and y'

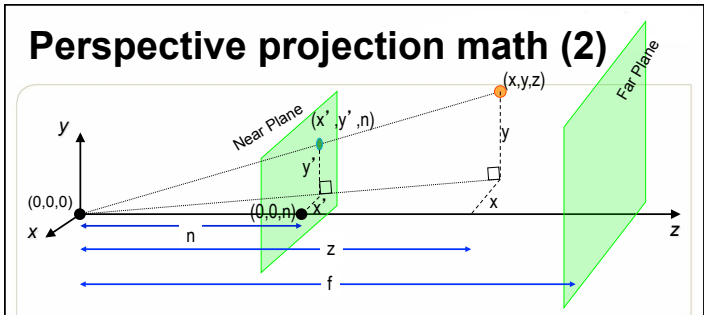
$$\frac{x'}{x} = \frac{n}{z} \Rightarrow x' = \frac{nx}{z}$$

$$\frac{y'}{x'} = \frac{y}{x} \Rightarrow y' = \frac{yx'}{x} = \frac{y}{x} \cdot \frac{nx}{z} = \frac{ny}{z}$$

Next apply our orthographic projection formulas

See http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__3/Deriving-Projection-Matrices.htm

Perspective projection math (2)



$$x' = \frac{2}{r-l} \cdot \frac{nx}{z} - \frac{r+l}{r-l} \qquad y' = \frac{2n}{t-b} \cdot \frac{ny}{z} - \frac{t+b}{t-b}$$

$$x' \cdot z = \frac{2n}{r-l} \cdot x - \frac{r+l}{r-l} \cdot z \qquad y' \cdot z = \frac{2n}{t-b} \cdot y - \frac{t+b}{t-b} \cdot z$$

Now let's tackle the z' component

See http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__3/Deriving-Projection-Matrices.htm

Perspective projection math (3)

$$x' \cdot z = \frac{2n}{r-l} \cdot x - \frac{r+l}{r-l} \cdot z$$

$$y' \cdot z = \frac{2n}{t-b} \cdot y - \frac{t+b}{t-b} \cdot z$$

$$z' \cdot z = p \cdot z + q \quad \text{where } p \text{ and } q \text{ are constants}$$

- We know z (depth) transformation has nothing to do with x and y

See http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__3/Deriving-Projection-Matrices.htm

Perspective projection math (4)

$$z' \cdot z = p \cdot z + q \quad \text{where } p \text{ and } q \text{ are constants}$$

$$\boxed{\begin{aligned} 0 &= p \cdot n + q \\ f &= p \cdot f + q \end{aligned}} \quad \Rightarrow \quad \therefore p = \frac{f}{f-n} \quad \text{and} \quad q = -\frac{fn}{f-n}$$

$$z' \cdot z = \frac{f}{f-n} \cdot z - \frac{fn}{f-n}$$

- We know (boxed equations above)
  - z' = 0 when z=n (near plane)
  - z' = 1 when z=f (far plane)

See http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__3/Deriving-Projection-Matrices.htm

Perspective projection math (5)

$$x' \cdot z = \frac{2n}{r-l} \cdot x - \frac{r+l}{r-l} \cdot z$$

$$y' \cdot z = \frac{2n}{t-b} \cdot y - \frac{t+b}{t-b} \cdot z$$

$$z' \cdot z = \frac{f}{f-n} \cdot z - \frac{fn}{f-n}$$

$$w' \cdot z = z$$

$$[x'z, y'z, z'z, w'z] = [x, y, z, 1] \cdot P \text{ where } P = \begin{bmatrix} \frac{2n}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2n}{t-b} & 0 & 0 \\ -\frac{r+l}{r-l} & -\frac{t+b}{t-b} & \frac{f}{f-n} & 1 \\ 0 & 0 & -\frac{fn}{f-n} & 0 \end{bmatrix}$$

See http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__3/Deriving-Projection-Matrices.htm

Simpler perspective projection

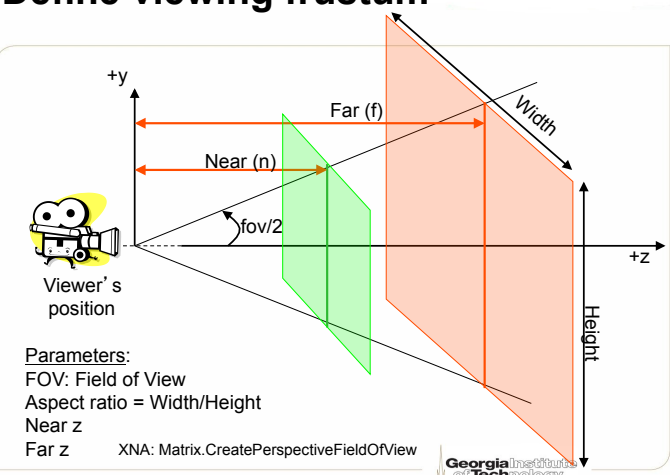
- Similar to orthographic projection, if ***l=-r and t=-b***, we can simplify to

$$[x'z, y'z, z'z, w'z] = [x, y, z, 1] \cdot P \text{ where } P = \begin{bmatrix} \frac{2n}{w} & 0 & 0 & 0 \\ 0 & \frac{2n}{h} & 0 & 0 \\ 0 & 0 & \frac{f}{f-n} & 1 \\ 0 & 0 & -\frac{fn}{f-n} & 0 \end{bmatrix}$$

- In any case, we will have to divide by z to obtain **[x',y', z', w']**
  - Implemented by dividing by the fourth (w'z) coordinate

See http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__3/Deriving-Projection-Matrices.htm

Define viewing frustum



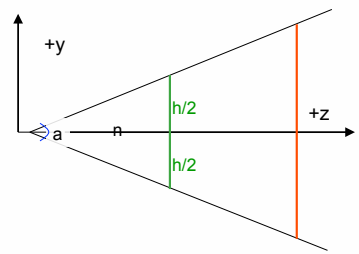
Parameters:
FOV: Field of View
Aspect ratio = Width/Height
Near z
Far z

XNA: Matrix.CreatePerspectiveFieldOfView

See http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__3/Deriving-Projection-Matrices.htm

Reparameterized matrix

$$P = \begin{bmatrix} \frac{2n}{w} & 0 & 0 & 0 \\ 0 & \frac{2n}{h} & 0 & 0 \\ 0 & 0 & \frac{f}{f-n} & 1 \\ 0 & 0 & -\frac{fn}{f-n} & 0 \end{bmatrix}$$



$$\cot\left(\frac{a}{2}\right) = \frac{2n}{h}$$

$$r = \frac{w}{h}$$

$$\frac{2n}{w} = \frac{2n}{rh} = \frac{2n}{r \cdot \frac{2n}{\cot(\frac{a}{2})}} = \frac{1}{r} \cdot \cot\left(\frac{a}{2}\right)$$

Need to replace w and h with FOV and aspect ratio

See http://www.codeguru.com/cpp/misc/misc/math/article.php/c10123__3/Deriving-Projection-Matrices.htm

Final matrix of perspective proj (LHS)

$$[x'\cdot z, y'\cdot z, z'\cdot z, w'\cdot z] = [x, y, z, 1]\cdot P \quad \text{where } P = \begin{bmatrix} \frac{1}{r}\cdot\cot(\frac{a}{2}) & 0 & 0 & 0 \\ 0 & \cot(\frac{a}{2}) & 0 & 0 \\ 0 & 0 & \frac{f}{f-n} & 1 \\ 0 & 0 & -\frac{fn}{f-n} & 0 \end{bmatrix}$$

a : Field of View (FOV) r : aspect ratio $= \frac{\text{width}}{\text{height}}$ n : near plan f : far plane

- In Direct3D: D3DXMatrixPerspectiveFovLH(*o,a,r,n,f)
  - LHS is default system in Direct3D

http://msdn.microsoft.com/en-us/library/bb205350(VS.85).aspx

Georgia Institute of Technology 29

Final matrix of perspective proj (RHS)

$$[x'\cdot z, y'\cdot z, z'\cdot z, w'\cdot z] = [x, y, z, 1]\cdot P \quad \text{where } P = \begin{bmatrix} \frac{1}{r}\cdot\cot(\frac{a}{2}) & 0 & 0 & 0 \\ 0 & \cot(\frac{a}{2}) & 0 & 0 \\ 0 & 0 & \boxed{\frac{f}{n-f}} & \boxed{-1} \\ 0 & 0 & \frac{fn}{n-f} & 0 \end{bmatrix}$$

a : Field of View (FOV) r : aspect ratio $= \frac{\text{width}}{\text{height}}$ n : near plan f : far plane

- In Direct3D: D3DXMatrixPerspectiveFovRH(*o,a,r,n,f)
- In XNA: Matrix.CreatePerspectiveFieldOfView(a,r,n,f)
- In OpenGL: gluPerspective(a,r,n,f)

http://msdn.microsoft.com/en-us/library/bb205351(VS.85).aspx

Georgia Institute of Technology 30

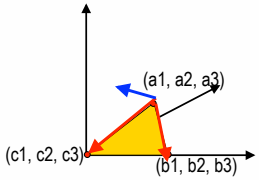
Viewport transformation

- The actual 2D projection to the viewer
- Copy to your back buffer (frame buffer)
- Can be programmed, scaled, …

Georgia Institute of Technology 31

Backface culling

- Determine "facing direction"
- Triangle order matters
- How to compute a normal vector for 2 given vectors?
  - Using Cross product of 2 given vectors



2 Vectors

$$\overrightarrow{V1} = (b1-a1)\mathbf{i} + (b2-a2)\mathbf{j} + (b3-a3)\mathbf{k}$$
$$\overrightarrow{V2} = (c1-a1)\mathbf{i} + (c2-a2)\mathbf{j} + (c3-a3)\mathbf{k}$$

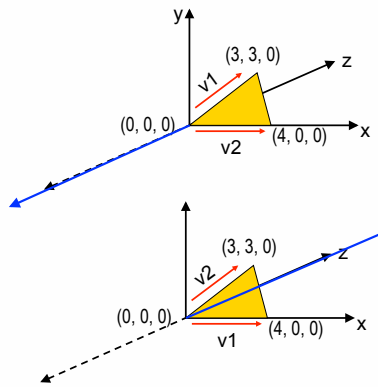
Cross product

$$\overrightarrow{V1} = x_1\mathbf{i} + x_2\mathbf{j} + x_3\mathbf{k}$$
$$\overrightarrow{V2} = y_1\mathbf{i} + y_2\mathbf{j} + y_3\mathbf{k}$$
$$\overrightarrow{V1}\times\overrightarrow{V2} = (x_2y_3 - x_3y_2)\mathbf{i} + (x_3y_1 - x_1y_3)\mathbf{j} + (x_1y_2 - x_2y_1)\mathbf{k}$$

Georgia Institute of Technology 32

Compute the surface normal for a triangle

- Clockwise normals, LHS



$\vec{v1} = 3\mathbf{i} + 3\mathbf{j} + 0\mathbf{k}$

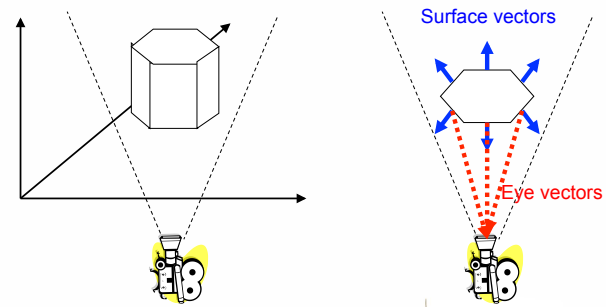
$\vec{v2} = 4\mathbf{i} + 0\mathbf{j} + 0\mathbf{k}$

$\vec{v1} = 4\mathbf{i} + 0\mathbf{j} + 0\mathbf{k}$

$\vec{v2} = 3\mathbf{i} + 3\mathbf{j} + 0\mathbf{k}$

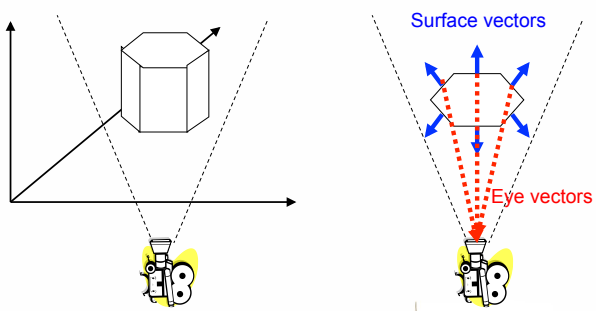
Backface culling method (1)

- Check if the normal is facing the camera
- How to determine that?
  - Use Dot Product

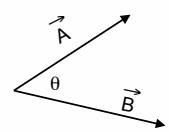


Backface culling method (2)

- Check if the normal is facing the camera
- How to determine that?
  - Use Dot Product

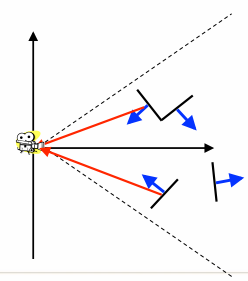


Dot product method (1)

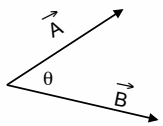


$$\vec{A} \bullet \vec{B} = |A||B|\cos\theta$$

$$\vec{A} \bullet \vec{B} > 0 \;\Rightarrow\; -\frac{\pi}{2} < \theta < \frac{\pi}{2}$$

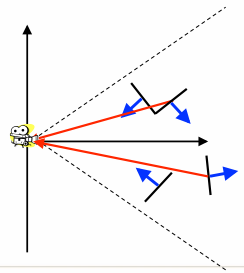


Dot product method (2)

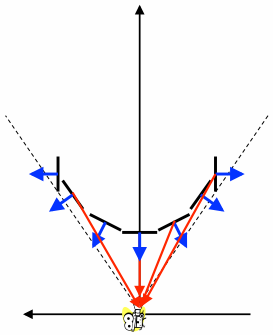


$$\vec{A} \bullet \vec{B} = |A||B|\cos\theta$$

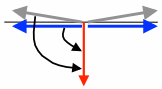
$$\vec{A} \bullet \vec{B} > 0 \;\Rightarrow\; -\frac{\pi}{2} < \theta < \frac{\pi}{2}$$



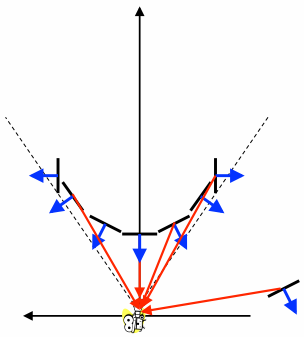
Dot product method (3)



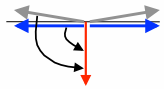
$$\vec{A} \bullet \vec{B} > 0 \;\Rightarrow\; -\frac{\pi}{2} < \theta < \frac{\pi}{2}$$



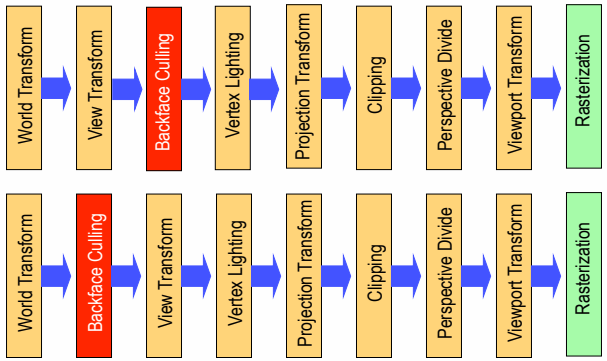
Caution!



$$\vec{A} \bullet \vec{B} > 0 \;\Rightarrow\; -\frac{\pi}{2} < \theta < \frac{\pi}{2}$$

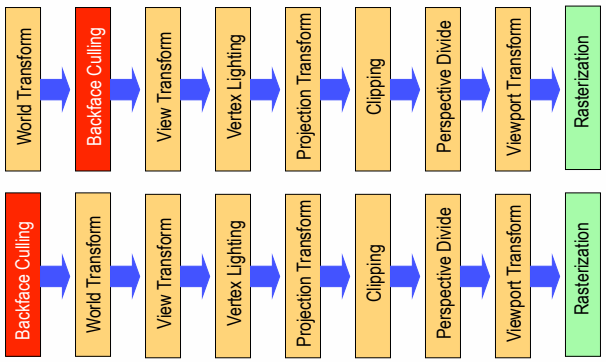


When to perform backface culling?



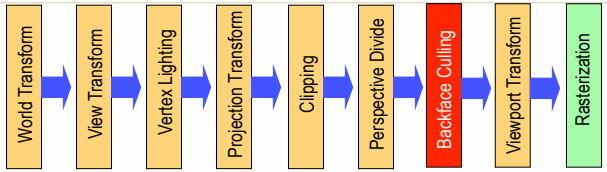
Make sure camera is in correct coordinates!

How about before you even start?

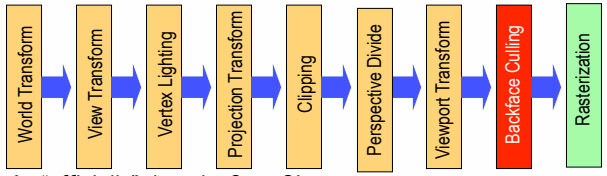


Transform camera vectors into object spaces

Or how about at the very end?

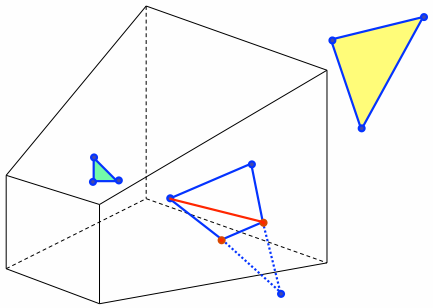


- Now you can just check "winding order" in 2D



- As "officially" done by OpenGL

3D clipping



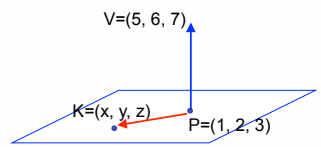
- Test 6 planes if a triangle is inside, outside, or partially inside the view frustum
- Clipping creates new triangles (triangulation)
  - Interpolate new vertices info

Appendix

Clipping against a plane

- Test each vertex of a triangle
  - Outside
  - Inside
  - Partially inside
- Incurred computation overhead
- Save unnecessary computation (and bandwidth) later
- Need to know how to determine a plane
- Need to know how to determine a vertex is inside or outside a plane

Specifying a plane

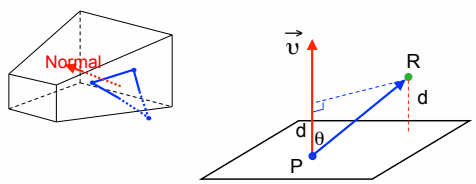


Plane equation

5*(x-1)+6*(y-2)+7*(z-3)=0

- You need two things to specify a plane
  - A point on the plane *(p0, p1, p2)*
  - A vector (normal) perpendicular to the plane *(a, b, c)*
  - Plane → *a*(x – p0) + b*(y – p1) + c*(z - p2) = 0*

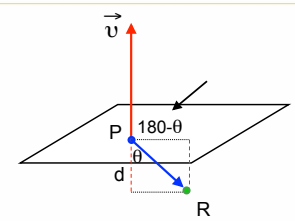
Distance calculation from a plane (1)



$$d = |R - P| \cdot \cos\theta = |R - P| \cdot \frac{\vec{v} \bullet (R - P)}{|\vec{v}| \cdot |R - P|} = \frac{\vec{v} \bullet (R - P)}{|\vec{v}|}$$

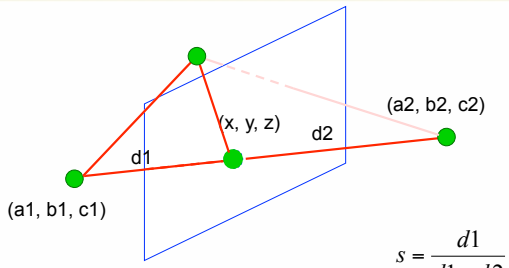
- Given a point R, calculate the distance
  - Distance > 0 inside the plane
  - Distance = 0 on the plane
  - Distance < 0 outside the plane

Distance calculation from a plane (2)



$$d = |R - P| \cdot \cos(180 - \theta)$$

Triangulation using interpolation



$$s = \frac{d1}{d1 + d2}$$

$$x = a1 + s \cdot (a2 - a1)$$

$$y = b1 + s \cdot (b2 - b1)$$

$$z = c1 + s \cdot (c2 - c1)$$