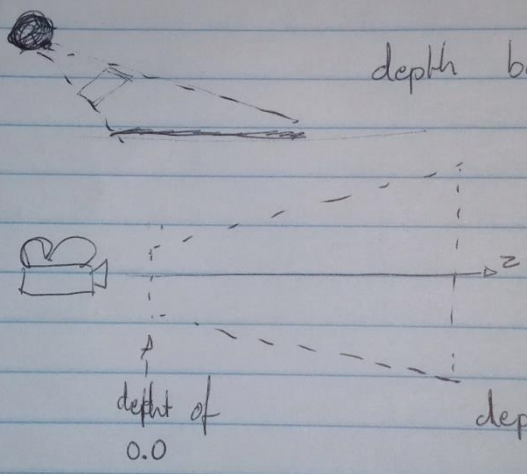
Further down the rabbit hole.

- Render canvas.
  Tina gave piece of advice
  → look at shadows.

- Using code instead.

- Render cameras & coding shaders were my largest gaps in knowledge.

- Unity uses a camera to determine shadows.



Depth is non-linear as closer objects need more detail → ask tim.

It struggles when triangles are ontop of each other.
- ↳ make sure tris not together
- ↳ increase res. of depth buffer

Note Unity uses Vulkan, not open GL

Z-buffer is an image space approach.
→ implemented in screen co-ordination system.

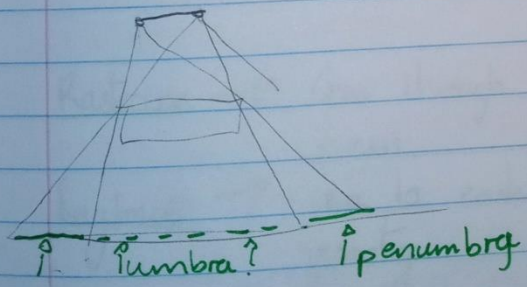
The basic idea is ~~that~~ to find closest z-depth surface.
Cross through every pixel, frame buffer.
↳ stores the intensity value of color (x,y)

depth buffer
↳ values for (x, y)
(0≤depth≤1).

These processes happen for each polygon ~~each of its pixel~~
then for each pixel.
→ is this each pixel of camera / screen. How does this translate for shadow rendering?

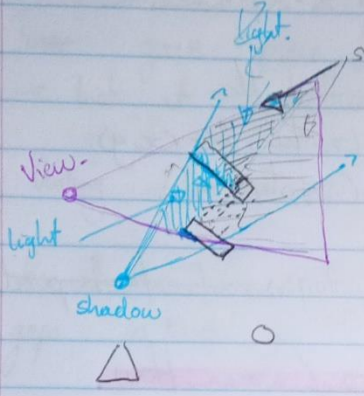
Fun fact: Couldn't do it before (1MB was so big) but now we have Gigs of memory.

Softness in shadows



Depth buffer

Depth buffer + view camera.



- In each pixel we find if it is in shade or not, change color accordingly.

How do view & shadow transfer / talk to each other I am struggling with storage and use of shadow map information.

Using math magic the depth from the source is converted to position in view space.

HSL GLSL → These are what we use in Unity

EDX Course - 2022/03/31
Computer Graphics

Using OpenGl GLSL.
He developed ~~the~~ spherical Harmonic lighting.
This course covers basics of computer graphics.

- We will write systems
- Theory of algorithms.

3D Graphics pipeline.

Modelling → Animation → Rendering
↓ ↓ Rasterize
Transformations HW2: View scene, lighting, Shading
HW3: Ray Tracer
- Realistic images with ray tracing. Raytrace.
~~Raster.~~

Rasterize → Goes through geometry & dermines place on screen.
Raytrace → Goes to each point & finds corrisponding geometry
Opposites of one another.

Game engines ~~makes~~ uses rasterization

Coding will be in C++ Yay!!!

Vectors & Matrices.

$a \cdot b = ?$

$\|\vec{a}\| \|\vec{b}\| \cos\phi$

$\phi = \cos^{-1}\left(\frac{a \cdot b}{\|a\| \|b\|}\right)$

$a \cdot b = \begin{pmatrix} x_a \\ x_b \end{pmatrix} \cdot \begin{pmatrix} x_b \\ y_b \end{pmatrix} = ?$

$= x_a x_b (\vec{x} \cdot \vec{x}) + (\vec{x} \cdot \vec{y})^{0}$ 6

$y_a y_b (\vec{y} \cdot \vec{y})$

$\therefore = x_a x_b + y_a y_b.$

$\|b \to a\| = ?$ $\quad \|b \to_a\| = \|b\| \cos\phi = \frac{a \cdot b}{\|a\|}$ ... ①

$b \to a = ?$ $\quad = \frac{a \cdot b}{\|a\|^2} a$ ... ② (~~last x~~ ② = ① × $\frac{a}{\|a\|}$

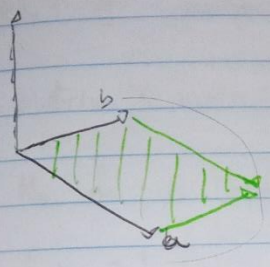
~~Question~~ L 2 P1

mag of $i = 1$.

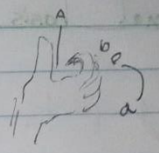
mag of $k = 1$.

Cross product

$a \times b = -b \times a$ = the area of the parallelogram made by a & b.



it has a dir. $a \to b$ (right hand)



$a \times a = 0$

$a \times (b+c) = a \times b + a \times c$

$a \times (kb) = k(a \times b)$.

Derive ~~cross~~ prod.

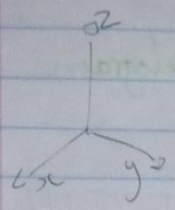
$$(X_a \bar{X} + Y_a \bar{Y} + Z_a \bar{Z}) \times (X_b \bar{X} + Y_b \bar{Y} + Z_b \bar{Z}).$$

$$= X_a Y_b \vec{Z} + Y_a X_b \vec{Z} = \cancel{0} X_a$$

$$a \times b = \begin{vmatrix} \bar{X} & \bar{Y} & \bar{Z} \\ X_a & Y_a & Z_a \\ X_b & Y_b & Z_b \end{vmatrix} = \begin{bmatrix} y_a z_b - y_b z_a \\ z_a x_b - x_a z_b \\ x_a y_b - y_a x_b \end{bmatrix} \begin{matrix} \vec{X} \\ \vec{Y} \\ \vec{Z} \end{matrix}$$

$$a \times b = A^* b = \begin{bmatrix} 0 & -z_a & y_a \\ z_a & 0 & -x_a \\ -y_a & x_a & 0 \end{bmatrix} \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix}$$

L2P2



Orthonormal basis

- Representing points, positions, locations

- There are many co-ord systems eg local, world.

Co-ordinate frame

↳ Any set of 3 vectors so that

$\|u\| = \|v\| = \|w\| = 1$

$u \cdot v = v \cdot w = w \cdot u = 0$

$w = u \times v$

$p = \ldots$

Matrices

- transform, translation, rotation
Area of numbers m-rows, n-columns.

Matrix × matrix. multiplication.

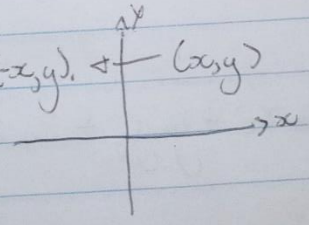
No. of n in first = No. m in second. → Rule to mult.

$$\begin{matrix} & 1 & 2 & 3 & 4 \\ 1 & & & & \\ 2 & & & & \\ 3 & & & & \end{matrix}\begin{bmatrix} 1 & 3 \\ 5 & 2 \\ 0 & 4 \end{bmatrix} * \begin{bmatrix} 3 & 6 & 9 & 4 \\ 2 & 7 & 8 & 3 \end{bmatrix} = \begin{matrix} (1,1) & (1,2) & (1,3) & (1,4) \\ (2,1) & (2,2) & (2,3) & (2,4) \\ (3,1) & (3,2) & (3,3) & (3,4) \end{matrix}\begin{bmatrix} 9 & 27 & 33 & 13 \\ 19 & 24 & 61 & 26 \\ 8 & 28 & 32 & 12 \end{bmatrix}$$

Not commutative.

Matrix mult. is key for transformation.

e.g Reflection about Y-axis (2-D).



$$= \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -x \\ y \end{bmatrix}$$

Trans

Transpose

Transpose of Matrix

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

$(AB)^T = B^T A^T$ (have to flip order) *

The identity matrix

$$I_{3\times 3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$AA^{-1} = A^{-1}A = I$

$(AB)^{-1} = B^{-1}A^{-1}$ → (change order).

- **Dot product.**

$$(x_a, y_a, z_a) \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = (x_a x_b + y_a y_b + z_a z_b).$$

$a \cdot b = a^T b.$

- **Cross product:**

$$a \times b = A^* b = \begin{bmatrix} 0 & -z_a & y_a \\ z_a & 0 & -x_a \\ -y_a & x_a & 0 \end{bmatrix} \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix}$$

- Wasted time getting VSCode C++ compatible with mingw64. VS 13 already C++ compatible and got HW0 to work.

HW0 ✓ COMPLETE

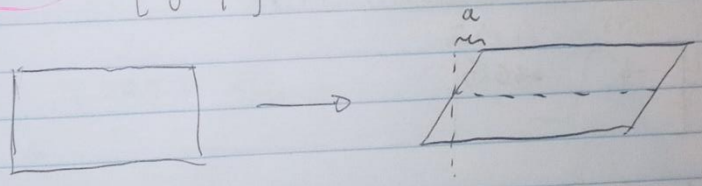
~~L3V~~ L3V1: Transforms 1: Basic 2D Transforms

We have many different ~~transf~~ co-ord systems
- World, object, screen space.

- To relate them, we transform between

Scale, rotation, translation, shear.

- Shear = $\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}$



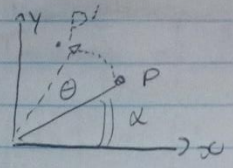
$y' = y$ (y doesn't change)

$x' = x + ay$

- Rotations $R(x+y) = R(x) + R(y)$ 2D.

In 2D they are commutative, in 3D they are not.

Rotations.



$P = r\cos\alpha, r\sin\alpha$

$P' = r\cos(\alpha+\theta), r\sin(\alpha+\theta)$

polar co-ords

$P'_x = r\cos\alpha\cos\theta - r\sin\alpha\sin\theta$

$P'_x = x\cos\theta - y\sin\theta$

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

$\vec{i} = \begin{pmatrix} 4.0 \\ 3.0 \end{pmatrix}$

Q: $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} \cos(183) & -\sin(183) \\ \sin(183) & \cos(183). \end{bmatrix} \begin{pmatrix} 4 \\ 3 \end{pmatrix}$.

$\vec{j} = \begin{bmatrix} -4. & -3.05 \end{bmatrix}$ (1,1) (1,2)

A: $\vec{j} = \begin{pmatrix} -3.83751 \\ -3.205 \end{pmatrix}$

polar co-ords

Composing Transformations L3V2

e.g. scale it, move it, rotate it.

NB: Not commutative.

$x_3 = Rx_2 \qquad x_2 = Sx_1$

$x_3 = R(Sx_1) = (RS)x_1$

$x_3 \neq SRx_1$ → Cannot change order

Convert Composite matrix

$M = M_1 M_2 M_3$ (M_3 is the first transformation).

$M^{-1} = M_3^{-1} M_2^{-1} M_1^{-1}$

$M^{-1}M = I.$

Qu $\frac{(a+n)}{2} \times n = \frac{1+180}{2} \times 180$

$16290 \div 360$

$=$

3D Rotations L3V3

$$R_p = \begin{bmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \\ x_w & y_w & z_w \end{bmatrix} \begin{pmatrix} x_p \\ y_p \\ y z_p \end{pmatrix}$$

$$= \begin{pmatrix} u \cdot p \\ v \cdot p \\ w \cdot p \end{pmatrix}$$

- Rows of 3 unit vectors
- rotation matrix of 3 orthonormal.

Rotations not commutative (e.g. x axis then y axis).

* [Axis-angle $R(a, \theta) = I_{3\times3}\cos\theta + aa^T(1-\cos\theta) + A^* \sin\theta$]

Derivation of Axis angle = hard.
Rodrigues rotation formula.

* Not going to spend time on this - we use Quarternion in Unity.

Qu Using the Rodrigues Rotation form, what effect does negating the axis of rotation have?

L4 V1 Trs 2 : Homogeneous Co-ords.

Can start HW1 after this.

Translation. (moving an object.)

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ ? & & \\ ? & & \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x+s \\ y \\ z \end{bmatrix}$$

To solve we add a 4th homogeneous co-ord.

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & s \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w=1 \end{bmatrix} = \begin{bmatrix} x+s \\ y \\ z \\ 1 \end{bmatrix}$$

Homogeneous point doesn't matter ~~this~~ till end. divide by w to get it.

$W = 0$ → This is vector

When we find point to display → we de homogenise.
The advantage is division is only done once.

General translation matrix

$$T = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} I_3 & T \\ 0 & 1 \end{bmatrix}$$

$P' = TP = P + T$ (Point + translation)

Translation & Rot. not commutative.

Note: Nice Course but not getting me lose to shoulder though.

$$P' = (TR)P = MP = RP + T$$

$$\begin{pmatrix} I_{3\times3} & T \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R_{3\times3} & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_{3\times3} & T_{3\times1} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} P_{3\times1} \\ 1 \end{pmatrix} = \begin{pmatrix} R_{3\times3}P + T \\ 1 \end{pmatrix}$$

The conclusion of the lecture is that if ~~rotate before translate~~ rotate before translate, it will affect the translation.

Qui). R^2 $i = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

~~RP + T.~~ $\vec{j} = R(178)T(2,9,0)\vec{i}$

$\vec{k} = T(2,9)R(178)$.

A). $\vec{j} = RTP$

$\simeq$ $RP + RT$ → so we rotate point & translation

~~$RP = \begin{bmatrix} \cos(178) & -\sin(178) \\ \sin(178) & \cos(178) \end{bmatrix}$~~

$$RP = \begin{bmatrix} \cos(178) & -\sin(178) \\ \sin(178) & \cos(178) \end{bmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

$$RT = \begin{bmatrix} \cos(178) & -\sin(178) \\ \sin(178) & \cos(178) \end{bmatrix}\begin{pmatrix} 2 \\ 9 \end{pmatrix}$$

$$= \begin{bmatrix} -2.31 \\ -8.92 \end{bmatrix}$$

$RP + RT = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} -2.31 \\ -8.91 \end{pmatrix}$

$\vec{j} = (-2.31, -8.91)$.

$\vec{k} = (TR)R$

$= R(TP)$.

$\therefore \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} a \\ a \end{pmatrix}$

$TP = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$

Ask for tims help

$R(TP) =$

↳ what we did wrong!!!
Appy transforms from right to left.

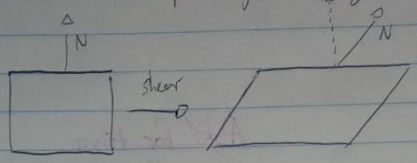
So $\vec{j} = P \xrightarrow{\text{Translate}} P' \xrightarrow{\text{Rotate}} P''$

$\vec{k} = P \xrightarrow{\text{Rotate}} P' \xrightarrow{\text{Translate}} P''$

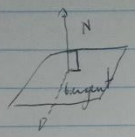
Decision: Going to take a break from maths & code in shaders for a while.

Transforming Normals

This is used for light intensity



But should be dots



$t \to Mt$ — M is surface transform

$n \to Qn$ — $Q = ?$

$n^T t = 0$ — $nt = 0$ (90° to one another)

$(Qn)^T = (n^T Q^T)(Mt) = 0$

$(n^T Q^T Mt) = 0 \Rightarrow Q^T M = I$

$Q = (M^{-1})^T$ — $^{-T}$ applies to upper 3×3 only

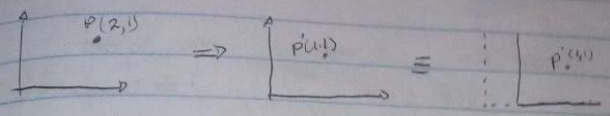
Only effects shear & scaling, not translation

$$* \left[Q = (M^{-1})^T \right] *$$

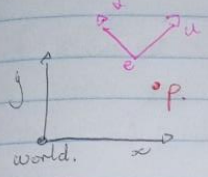
Qu: 3D rotation transform, Q? $M = R(\vec{a}, \theta)$ what is normal

$= R(\vec{a}, \theta)$ $M = Q$ Does rotation affect normals?

L4 V3. Coord frames



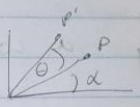
Co-ord ⇒ location }



$e = (-av, -bv)$

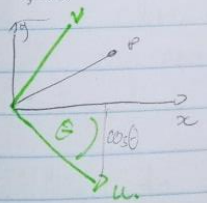
world = $(+cx, +dy)$.

Remember rotation:



$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

But we can rotate coord system.



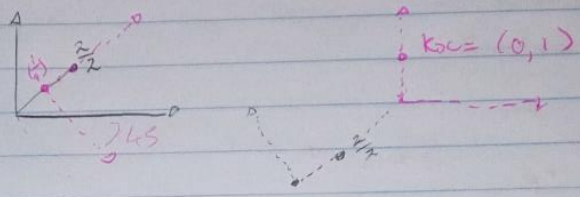
$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{matrix} u \\ v \end{matrix}$$

Axis angle Remember

$$R(a,\theta) = I_{3\times3}\cos\theta + aa^T(1-\cos\theta) + A^*\sin\theta$$

$$R(a,\theta) = \cos\theta \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + (1-\cos\theta) \begin{bmatrix} x^2 & xy & xz \\ xy & y^2 & yz \\ xz & yz & z^2 \end{bmatrix} + \sin\theta \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}$$

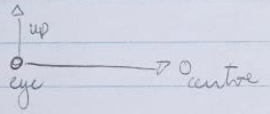
Qu: $i = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$ What are the coords in new coord system with origin $j = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ & axis angle 45°



$\sqrt{1^2 - 1^2} = 1.41$

L4 V4 Constructing a coord frame.

eye x, y, z centre x, y, z up x, y z



1- Create coord frame for camera
2- Define rotation matrix
3- Apply translation for camera

Associate w with a $\vec{b}$, $\vec{b}$.

$w = \frac{a}{\|a\|}$

$u = \frac{b \times w}{\|b \times w\|}$

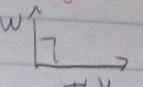
$v = w \times u$

a is eye

b up vector

What is Orthonormal again?

Orthogonal ≡ $v \cdot w = 0$



Unit $\|u\| = 1$ $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ or $\begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$

- all vectors are unit vectors.
- all vectors are orthogonal to one another
- linearly independant

What is linear independance?

if no vectors can be expressed by one another, they are linearly ind.

$\vec{V_1}, \vec{V_2}, \vec{V_2} \ldots V_n$ in vector space V

for linear independence in the foll. eqn. all scalars must be zero.

$$c\vec{V_1} + c_2\vec{V_2} + \ldots c_n\vec{V_n} = \vec{0}$$

If non-zero scalars exist the it is not linearly independant

Why do we want linear indep.

for a basis like x & y to span all of 2D space they must be linearly indep.

if $e_1 e_2 e_3$ are parallel / all in same plane then can't span all of 3D space.

Back to lecture.

Rotation matrix.

$$R_{uvw} = \begin{bmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \\ x_w & y_w & z_w \end{bmatrix}$$

$u = x_u X + y_u Y + z_u Z$

- Cannot apply translation after rotation.
- Translation first.

$$P' = (RT)P = RP + RT$$

$$M = \begin{bmatrix} R_{3\times3} & R_{3\times3}T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}$$

$$\text{gluLookAt} = \begin{bmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$-u \cdot e$
$-v \cdot e$
$-w \cdot e$.

The gluLookAt matrix can be thought of as translating the world so that the camera is at origin. Then rotate new origin depends we direction camera will be looking at. $-\vec{z}$ in openGL
This would be camera space.

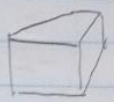
LSV1: Viewing : ORthographic projection

6d 3hours 2.

All of our calculations & workings are in 3D but view in 2D.
So we must project the 3D world to a 2D space.

ortho

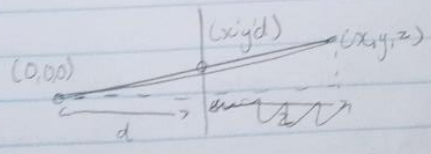
persp

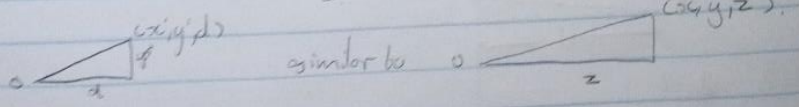


To do this we translate & origin then scale

$$glortho = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

LSV2 : Perspective projection.





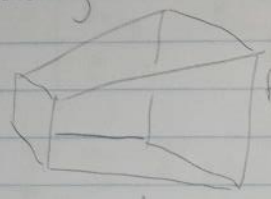
$$\therefore \frac{d}{x'} = \frac{z}{x} \Rightarrow x' = \frac{dx}{z}$$ NB.

$$y' = \frac{d * y}{z}$$

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{d} & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ w=1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ -\frac{z}{d} \end{pmatrix} \quad \text{dehomogenize} = \begin{pmatrix} -x \frac{z}{d} \frac{d}{z} \\ -y \frac{d}{z} \\ -d \\ 1 \end{pmatrix}$$

Derivation of glu Perspective.

Viewing frustum.

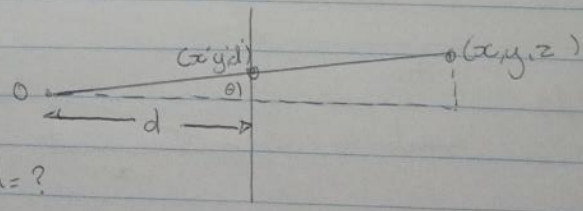


$$\text{Aspect} = \frac{\text{width}}{\text{height}}$$

glu Perspective (fovy, aspect, zNear > 0, zFar > 0).

↳ fovy, aspect fovy in x & y

↳ zNear, zFar control viewing frustum



θ = ? d = ?

~~fovy~~ / 2 = θ

$$\theta = \frac{fovy}{2} \qquad d = \cot(\theta).$$

In Matrices

- Simplest form.

$$P = \begin{bmatrix} \frac{1}{\text{aspect}} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad \text{aspect}$$

perspective.

Multiply by d.

$$Pd = \begin{bmatrix} \frac{d}{\text{aspect}} & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & A & B \\ 0 & 0 & -1 & 0 \end{bmatrix} \rightarrow \begin{matrix} z \\ w \end{matrix}$$

→ A & B selected to map n & f to −1, +1.

$$\begin{pmatrix} A & B \\ -1 & 0 \end{pmatrix} \begin{pmatrix} z \\ 1 \end{pmatrix} = ? = \begin{pmatrix} Az + B \\ -z \end{pmatrix} = -A - \frac{B}{z}$$

(1 ↔ w) ∴ still homogeneous.

$z = -n \rightarrow -1 \quad \therefore -A + \frac{B}{n} = -1$

$z = -f \rightarrow +1 \quad \therefore -A + \frac{B}{n \text{ or } F} = +1.$

$$\therefore A = \frac{f+n}{f-n}$$

$$B = \frac{-2fn}{f-n}$$

* Note need to research this more

Death check

Mapping of z is non-linear.

→ handles large range of depth.
→ Depth res. not uniform.

Do not set near clipping to 0.

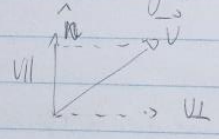
HW1. = This is my hand in.

Step 1 - fix VS, Took 2 hrs boot, no sha.
had to download & install VS 2022.
VS 2016 kept making file paths that don't exist.

Lets rotate ↻

Rotation matrix = $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix}$

Need to use rodrigues rotation.



$\vec{v}' = \vec{v}_{\parallel} + \cos(\theta)\,\vec{v}_{\perp} + \sin(\theta)(\hat{n} \times \vec{v}_{\perp})$

$= \vec{v}_{\parallel} + \cos(\theta)(\vec{v} - \vec{v}_{\parallel}) + \sin(\theta)(\hat{n} \times \vec{v})$

$\hat{n} \times \vec{v} = \hat{n} \times (\vec{v}_{\perp} + \vec{v}_{\parallel})$
$= \hat{n} \times \vec{v}_{\perp} + \hat{n} \times \vec{v}_{\parallel}$ → 0

$= (1 - \cos(\theta))\,\vec{v}_{\parallel} + \cos(\theta)\,\vec{v} + \sin(\theta)(\hat{n} \times \vec{v})$

9-3

6.

$\vec{V_u} = a\hat{n}$ $\quad a = (\vec{V} \cdot \hat{n})$

$\therefore \vec{V_u} = (\vec{V} \cdot \hat{n})\hat{n}$

$\vec{V}' = (1 - \cos(\theta))(\vec{V} \cdot \hat{n})\hat{n} + \cos(\theta)\vec{V} + \sin\theta(\hat{n} \times \vec{V})$

$$R(a,\theta) = \cos\theta \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + (1-\cos\theta)\begin{bmatrix} x^2 & xy & xz \\ xy & y^2 & yz \\ xz & yz & z^2 \end{bmatrix} + \sin\theta \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$$

- Row & column Vectors handled by GLSL.

$(x, y, z)\begin{pmatrix} x \\ y \\ z \end{pmatrix}$

$\begin{pmatrix} x \\ y \\ z \end{pmatrix}(x\ y\ z)$.

$3\times1 * 3\times3$

$x^2 \quad xy \quad xz$

$xy \quad y^2 \quad yz$

There is no support to do a $3\times1 * 3\times3$ so custom matrix.

NOW there is an error mat3[0] = anything

NO documentation on how to set matrix.

$\underline{a}$ is given as eye - center.
$\underline{b}$ = up.

$$w = \frac{a}{\|a\|} \checkmark$$

$$u = \frac{b \times w}{\|b \times w\|}$$

$$v = w \times u$$

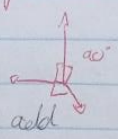
$$R_{uvw} = \begin{pmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \\ x_w & y_w & z_w \end{pmatrix}$$

Most of this is struggling with custom library lol. Like vec3 doesn't have a size().

Loop through a divide by mag = W.

Note: I discovered that the program uses a custom GLM library, which has made coding difficult as it is quite different from C++

U. Find cross, loop & divide.
XD. → there is a normalize function in GLM.
Time - wasted
Many cross product to give orthonormal basis



How do I add a row to a matrix!?

argenthian

Code does not work as expected.
Going to make fuctions to output matrix & vectors.

; 0 0 0 0
; 0 Create 4 func → display vec3
; 0 vec4
; 0 mat3
; 0 mat4.

a) eye = 007.
up = 010.

⇒ w = 001
u = 100
v = 010.

$\frac{0.0.7.}{7^2}$

= (0, 0, 1.43.).
= 0, 0, 01

0 1 0 x 0 0 7.

$c_x = a_y b_z - a_z b_y = 1 \times 7 - 0 \times 0$
$= 7$

$c_y = a_z b_x - a_x b_z = 0 \times 0 - 0 \times 07$
$= 0$

$c_z = a_x b_y - a_y b_x = 0 - 0$
$= 0$

2 0 0 →
1 0 0 →

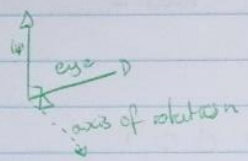
No : Left / Right

Just going to use unchanged rodigues formula don't see why we need to muddy things up with matrices.

Look Up

Quick 3D.T'y.

axis of rotation = up x eye.



Something Funky but need to do reports