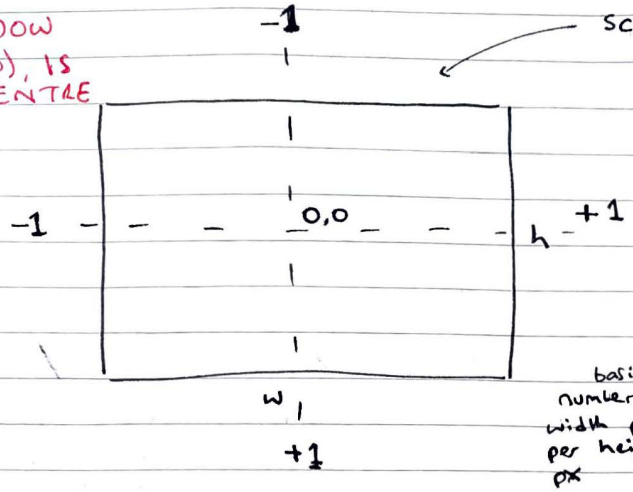
Projection Matrix Transformation 3D → 2D Screen

13/04/19

ENSURE SCREEN WINDOW ORIGIN, (0,0), IS IN SCREEN CENTRE



- First we normalise screen coordinates, as screens come in different sizes

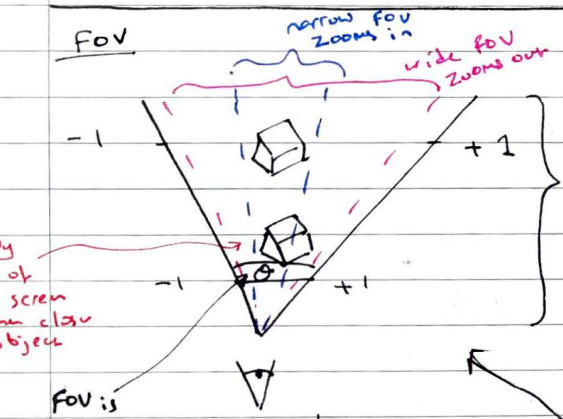
- $a = \frac{h}{w}$ ← screen height / screen width

basically number of width px per height px

- To scale x to convert x value into y value terms y coordinate we multiply by a, aspect ratio.
  - ↳ NOTE: Assumption that (x, y, z) are between 0 and 1 already.

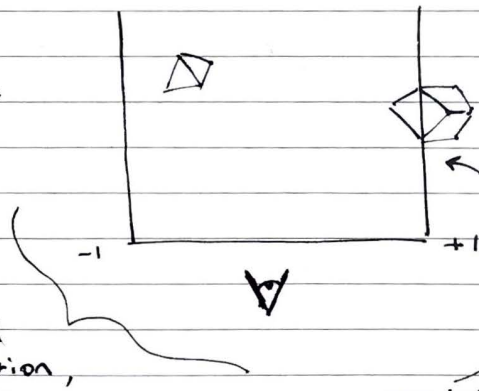
$$[x, y, z] \rightarrow \left[\frac{h}{w}x, y, z\right]$$

FoV



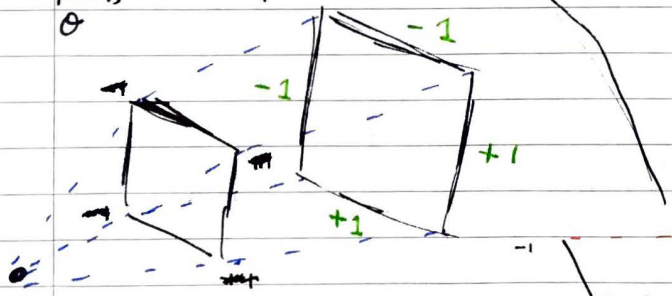
Top down view

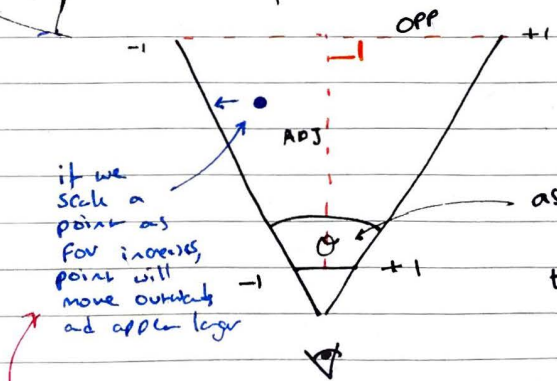
In reality 3D Frustum, perspective projection. (how humans see)



Parallel projection, not how humans see

normalising view cuts off objects outside of view





SOHCAHTOA

$\tan\theta = \frac{OPP}{ADJ}$ ↑ as this

if we scale a point as FoV increases, point will move outwards and appear larger

as $\frac{\theta}{2}$ increases, OPP increases

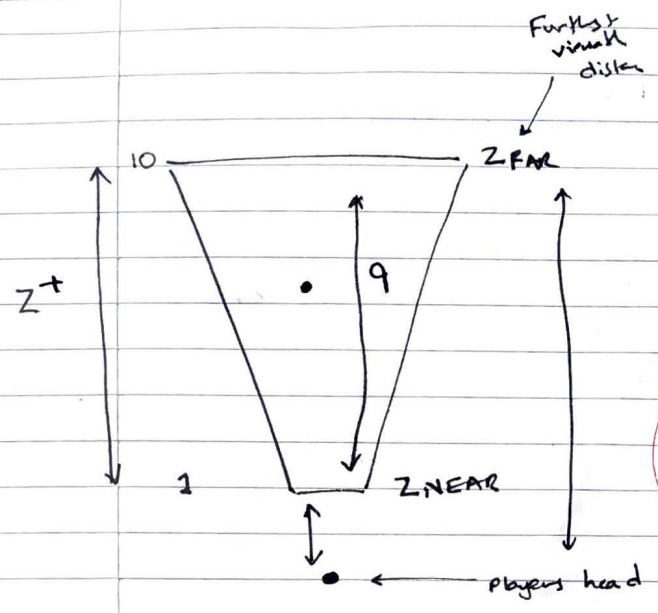
$\tan\left(\frac{\theta}{2}\right)$ when $\tan\theta = \frac{OPP}{ADJ}$

appear smaller ↑

To we want opposite therefore relationship is $\frac{1}{\tan\left(\frac{\theta}{2}\right)}$, happens in both x and y

$$[x, y, z] \rightarrow \left[\frac{h}{w} fx, fy, z\right]$$

$$\frac{1}{\tan\left(\frac{\theta}{2}\right)}$$



- Take z pos and divide by $Z_{FAR} - Z_{NEAR}$ to scale into range normalise system between 0 and 1
- Need to scale back up by multiply by Z_{FAR}

$\frac{9}{9} \times 10$ = However we've decided far plane is 10 ∴ scale back up again.

$$\frac{Z_{FAR}}{Z_{FAR} - Z_{NEAR}} - \frac{Z_{NEAR} Z_{FAR}}{Z_{FAR} - Z_{NEAR}}$$

OFFSET FROM PLAYERS HEAD

e.g.

$Z = 10$

$$\frac{10 \times 10}{9} - \frac{1 \times 10}{9} = \frac{90}{9} = 10$$

$Z = 1$

$$\frac{1 \times 10}{9} - \frac{1 \times 10}{9} = 0$$

$[x, y, z] \rightarrow \left[\frac{h}{w} fx, fy, z\right]$

$$[x, y, z] \rightarrow \left[\frac{h}{w} fx, fy, z \times \frac{Z_{FAR}}{Z_{FAR} - Z_{NEAR}} - \frac{Z_{NEAR} Z_{FAR}}{Z_{FAR} - Z_{NEAR}}\right]$$

We also need to factor in movement in x and y, where further away changes in x are smaller proportionally to the distance z.

$$\therefore \quad [x, y, z] \rightarrow \left[\frac{\frac{h}{w} f x}{z}, \frac{fy}{z}, \frac{z \times z_{FAR}}{z_{FAR} - z_{NEAR}} - \frac{z_{NEAR} z_{FAR}}{z_{FAR} - z_{NEAR}} \right]$$

To summarize, f, handles how objects in distance appear to converge into middle.
$\frac{1}{z}$ factors how x and y in objects will get closer together, i.e. objects get smaller in distance.

IMPLEMENTATION

x, y, z of shape coordinates are between 0 and 1.

h = window height w = window width $a = \frac{h}{w} = \frac{480}{640}$

$z_{FAR} = 1000$ $z_{NEAR} = 0.1$ $\theta = 90^\circ$ (FOV angle)

$$\frac{1}{\tan\left(\underbrace{\frac{90}{2} \times 0.017}_{\text{IN RADIANS}}\right)} = 0.785 \quad \begin{matrix}(\text{FOV OPP Length}) \\ (f)\end{matrix}$$

↳ First we perform the following operations on each component of each vertice:

$$\text{new } x = (x \times a \times f) / z$$
$$\text{new } y = (y \times f) / z$$
$$\text{new } z = \frac{(z \times z_{FAR})}{(z_{FAR} - z_{NEAR})} - \frac{(z_{NEAR} z_{FAR})}{(z_{FAR} - z_{NEAR})}$$

↳ As these new vertice component values are between -1 and 1 we need to scale to ~~between~~ screen coordinates.

↳ First we take each x, y, z component add 1.0 to divide by 2 to get each value between 0 and 1

↳ Next, we multiply x by screen width, y we multiply by screen height and z we leave alone.

↳ We now need to translate objects back away from camera & as origin is currently at (0,0) the same as the camera. ∴ we are correctly inside the cube.

↳ We translate before any conversion from 3D - 2D coordinates, by simply adding to the z component of each vertice of each triangle