ACTIVITY: MODELLING A CUBE (from CMU lecture 01)

Suppose a cube is:

- centered at origin
- has dimensions 2 × 2 × 2
- edges aligned with x/y/z axis

coordinates:

A :	(1, 1, 1)	D :	(-1, -1, 1)
B :	(-1, 1, 1)	E :	(1, 1, -1)
C :	(+1, -1, 1)	F :	(-1, 1, -1)
		G :	(1, -1, -1)
		H :	(-1, -1, -1)

We now have to find the edges:

AB, CD, EF, GH, AC, BD, EG, FH, AE, CG, BF, DH.

This is called the modelling stage, where we find out what to place where and how.

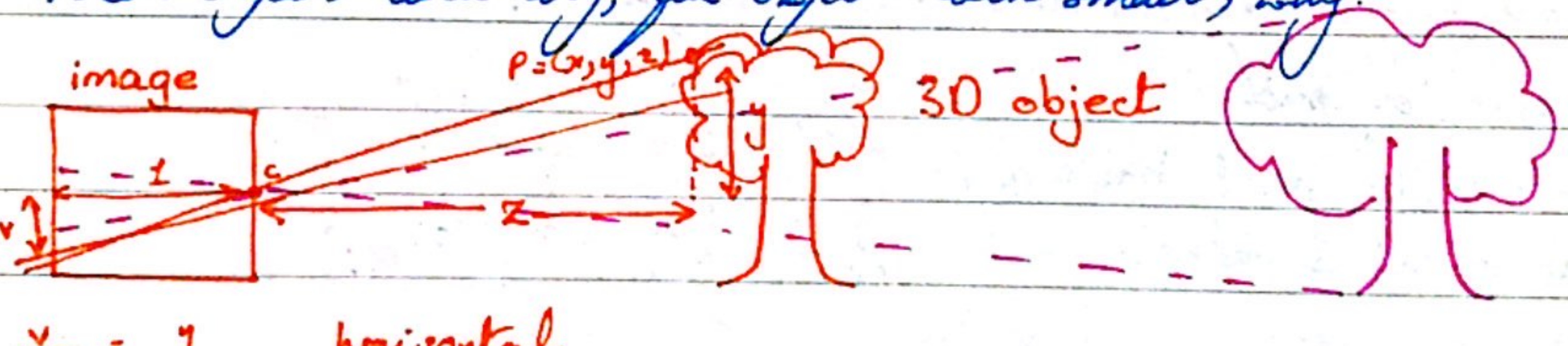
How do we draw this 3D shape as a 2D flat image?

This process is called rendering.

3D coordinates $\xrightarrow[\text{how?}]{\text{map}}$ 2D coordinates

1. Perspective Projection

- Near objects look big, far objects look small, why?



$\frac{v}{1} = \frac{y}{z}$ horizontal coordinate

$v = y/z$ $u = x/z$

This means that to go from 3D to 2D, all you have to do is divide by z.

Let's try it

- Assume camera c is at (2, 3, 5)
- Convert (X, Y, Z) to (u, v)
  1. Subtract c from (X, Y, Z) to get (x, y, z) (3D location of points relative to the camera).
  2. divide (x, y) by z to get (u, v)
- Draw a line between (u1, v1) and (u2, v2)

Let's try some of them

A $(1, 1, 1) - (2, 3, 5)$

$= (-1, -2, -4)$

$(-1, -2) \div -4 = (\frac{1}{4}, \frac{1}{2})$

B. $(-1, 1, 1) - (2, 3, 5)$

$(-3, -2, -4)$

$(-3, -2) \div -4 = (\frac{3}{4}, \frac{1}{2})$

C. $(1, -1, 1) - (2, 3, 5)$

$(-1, -4, -4)$

$(-1, -4) \div -4 = (\frac{1}{4}, 1)$

D. $(-1, -1, 1) - (2, 3, 5)$

$(-3, -4, -4)$

$(-3, -4) \div -4 = (+\frac{3}{4}, 1)$

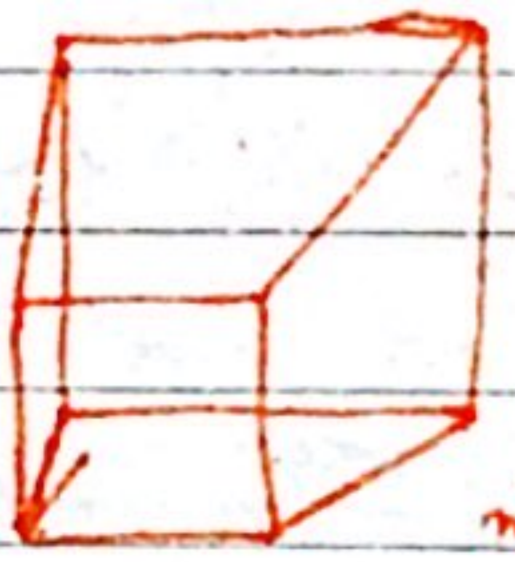
and so on.

E: $(\frac{1}{6}, \frac{1}{3})$

F: $(\frac{1}{2}, \frac{1}{3})$

G: $(\frac{1}{6}, \frac{2}{3})$

H: $(\frac{1}{2}, \frac{2}{3})$

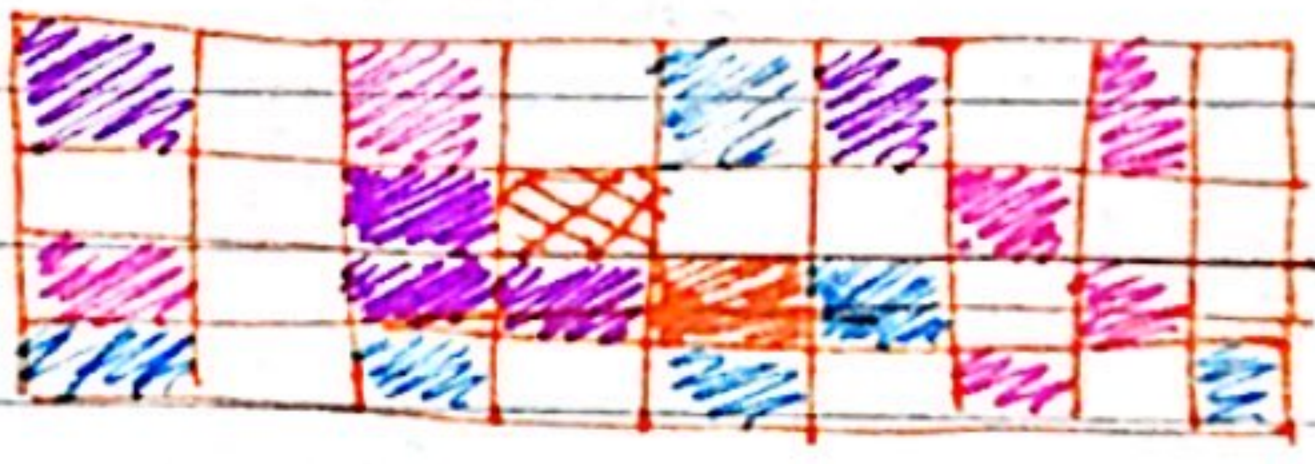


you come up with something like this by implementing the algorithm

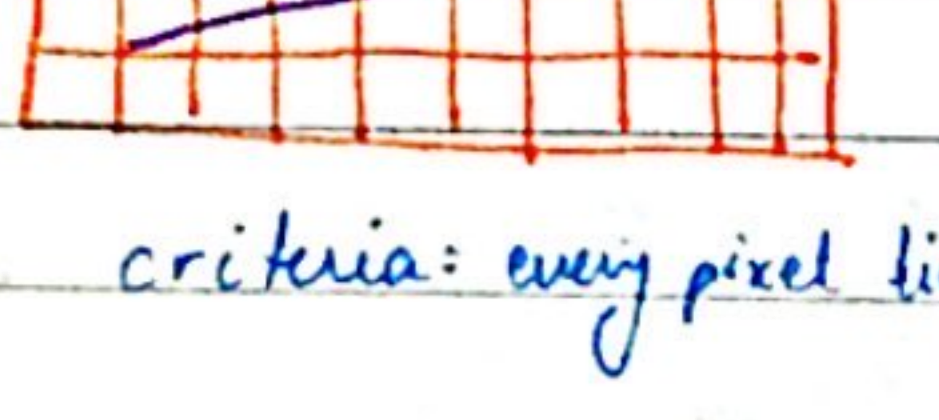
How to draw lines on a computer?

↳ how are images represented on a computer? we "rasterise"

- colours on a grid.
- each box, i.e. pixel, holds a numerical value.

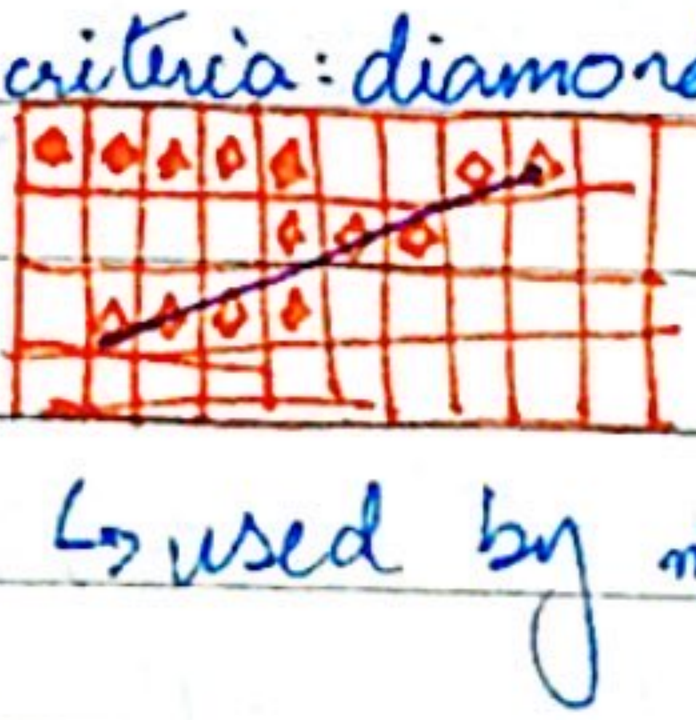


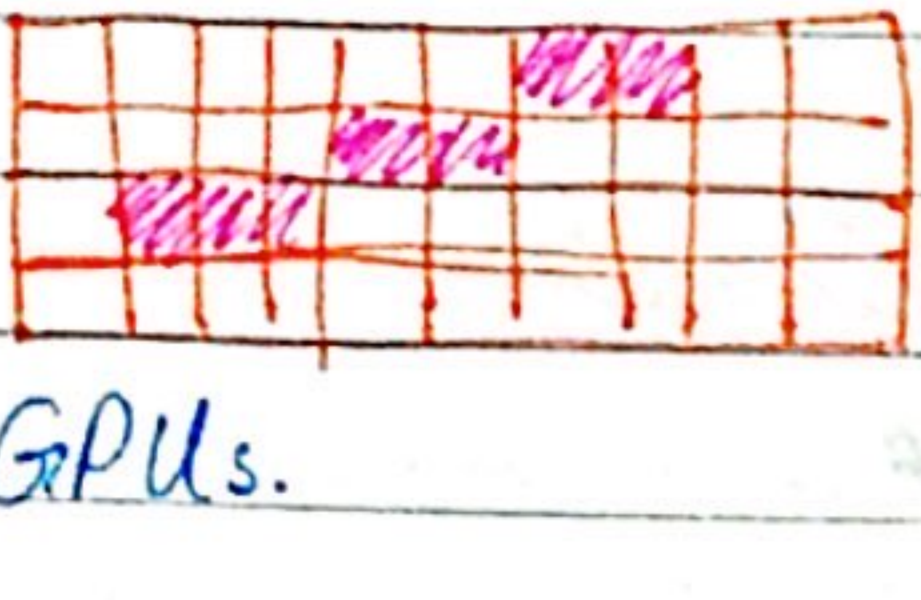
RASTERIZATION



criteria: every pixel line touches:

criteria: diamond rule.





↳ used by modern GPUs.

Incremental Line Rasterization Algorithm

```
v = v1
for (u = u1, u < u2, u ++){
    v + = s
    draw (u, $\underbrace{\text{round}(v)}_{\text{integer closest to current v value}}$)
}
```