COMS30020 - Computer Graphics

Week 6 Briefing

Dr Simon Lock

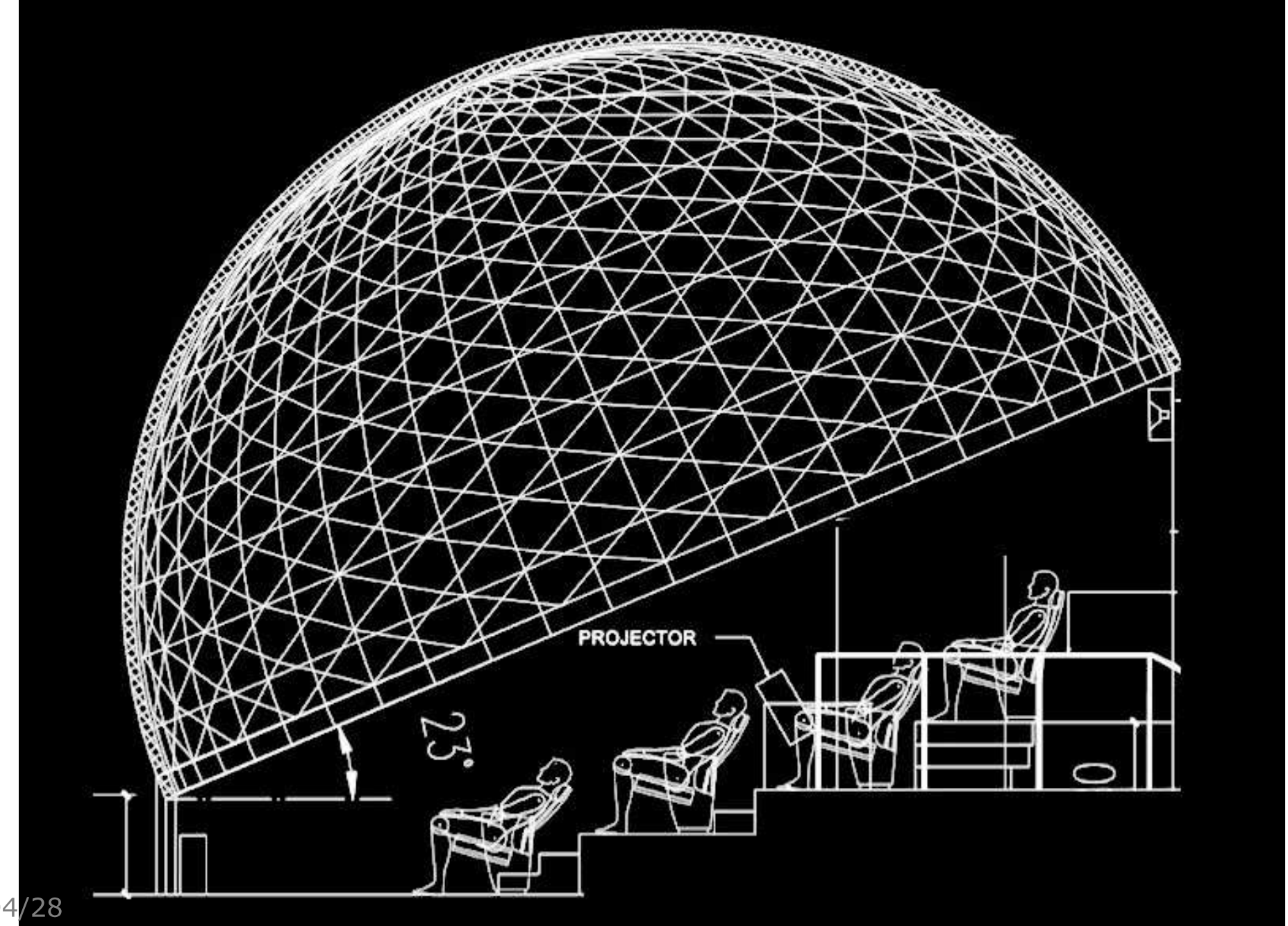
A little digression...

Why learn how to build a rendering engine ?
Surely they exist already ?
Can't we just use one of those ?

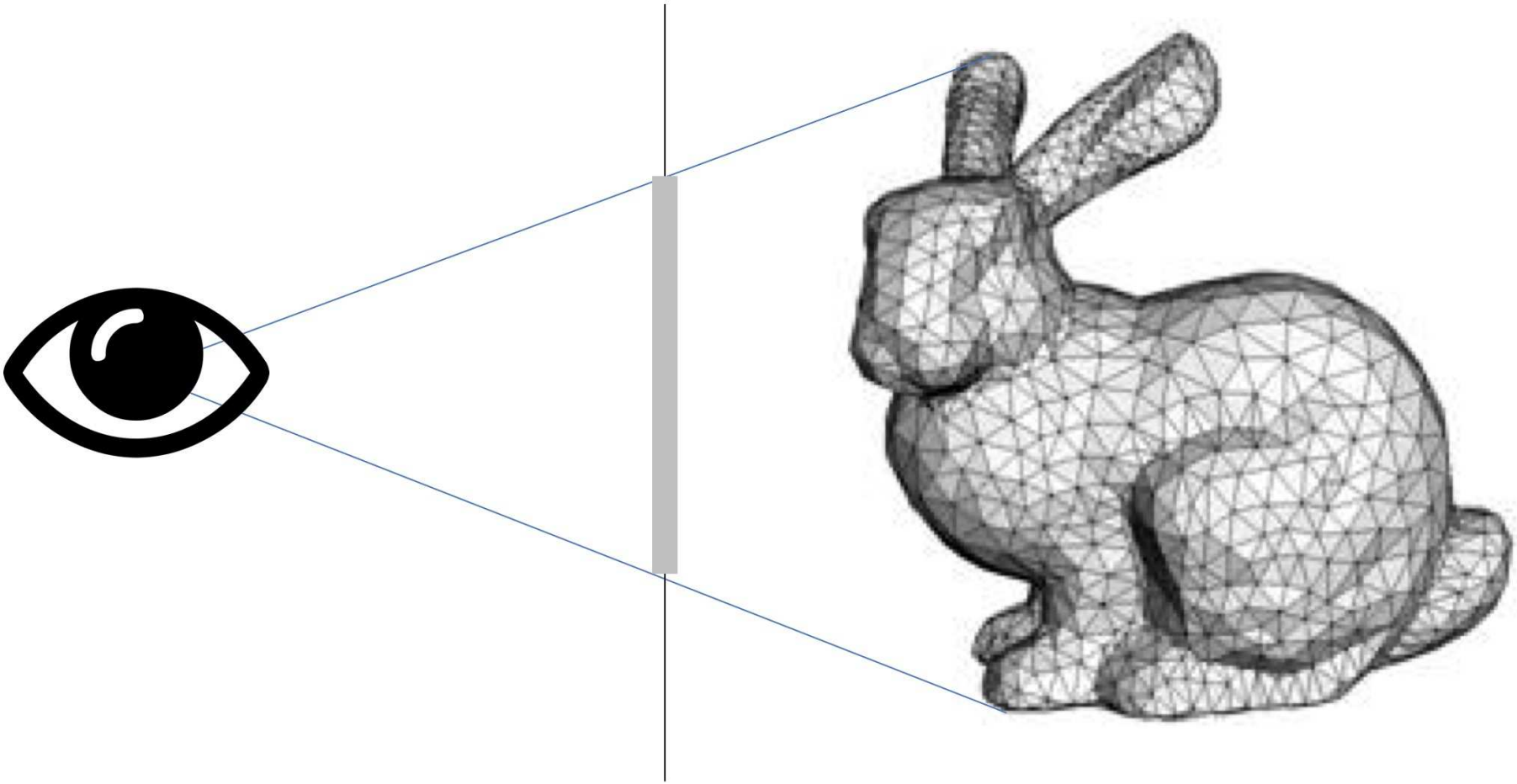
well...

It is useful to understand HOW these engines work
This gives us a deeper fundamental understanding

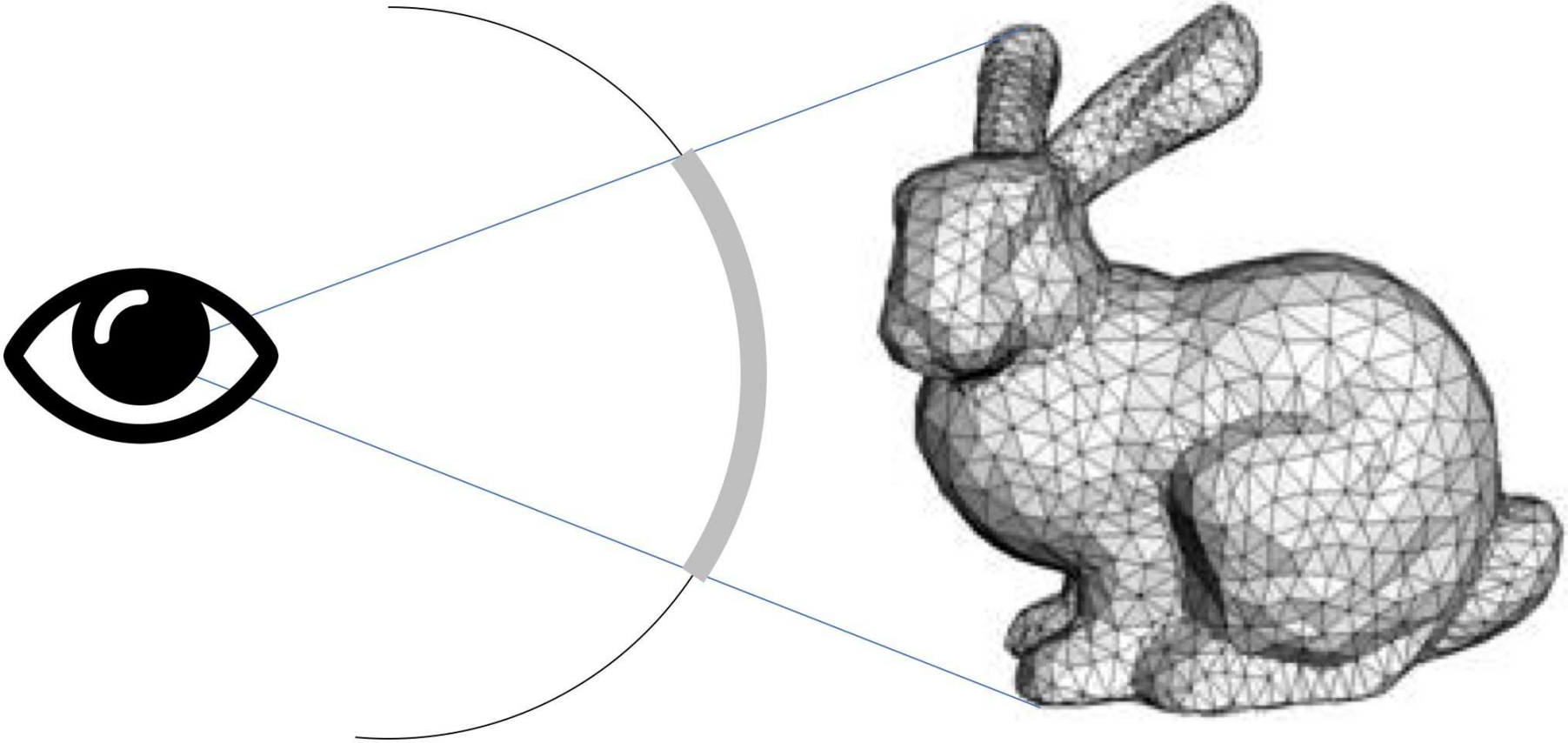
Which is useful if we want to do something different...


PROJECTOR
23'

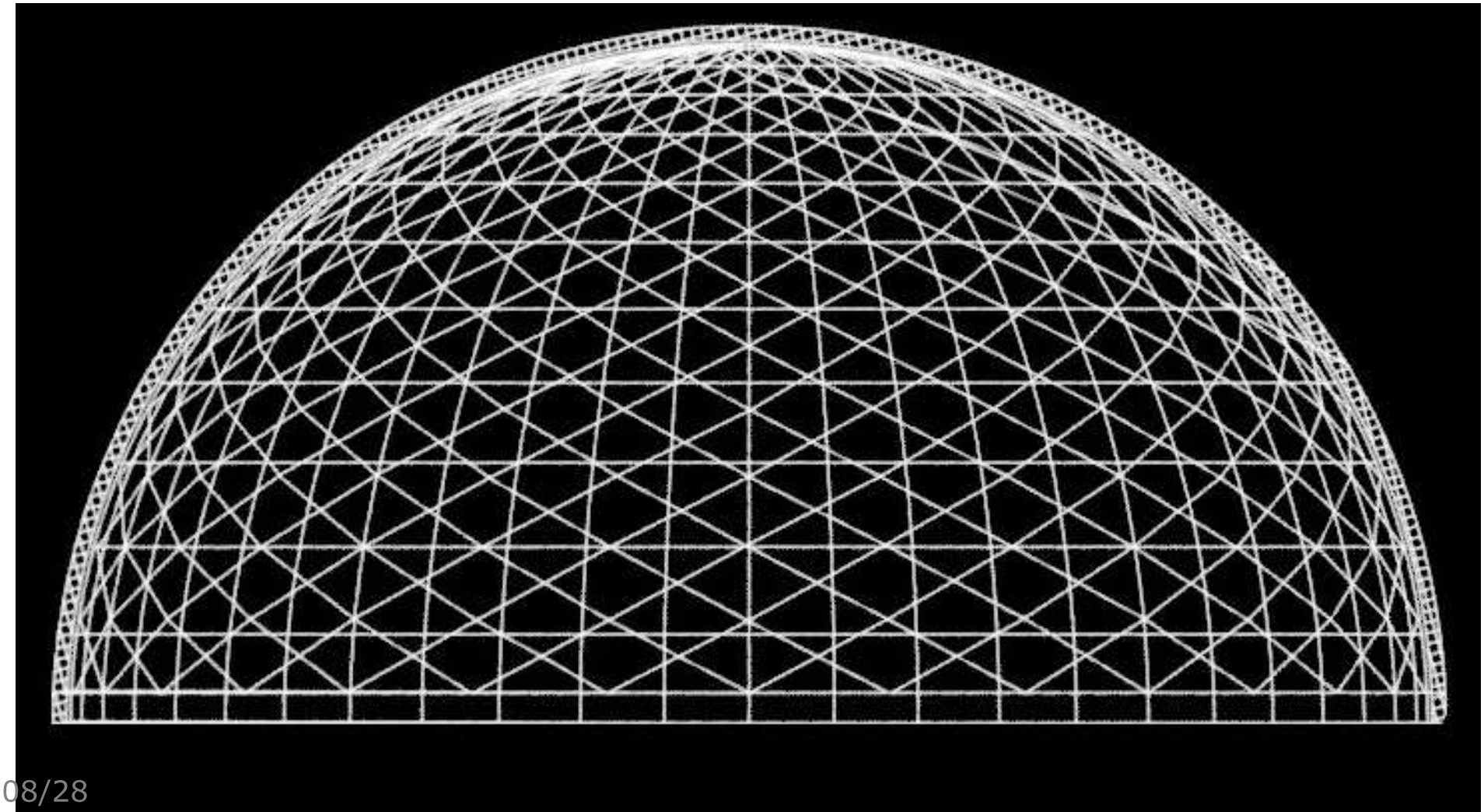
Traditional Renderers



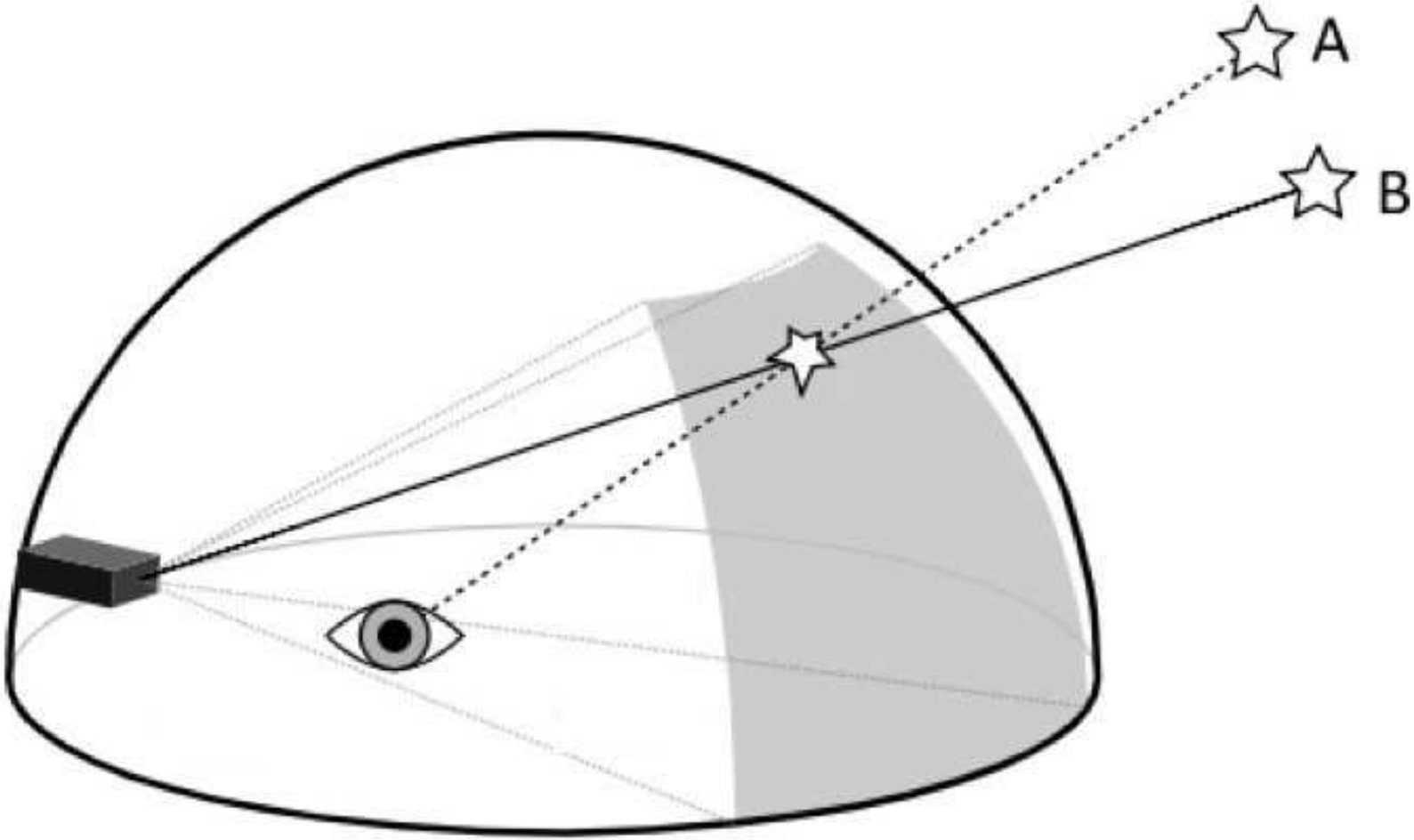
Custom Renderer !



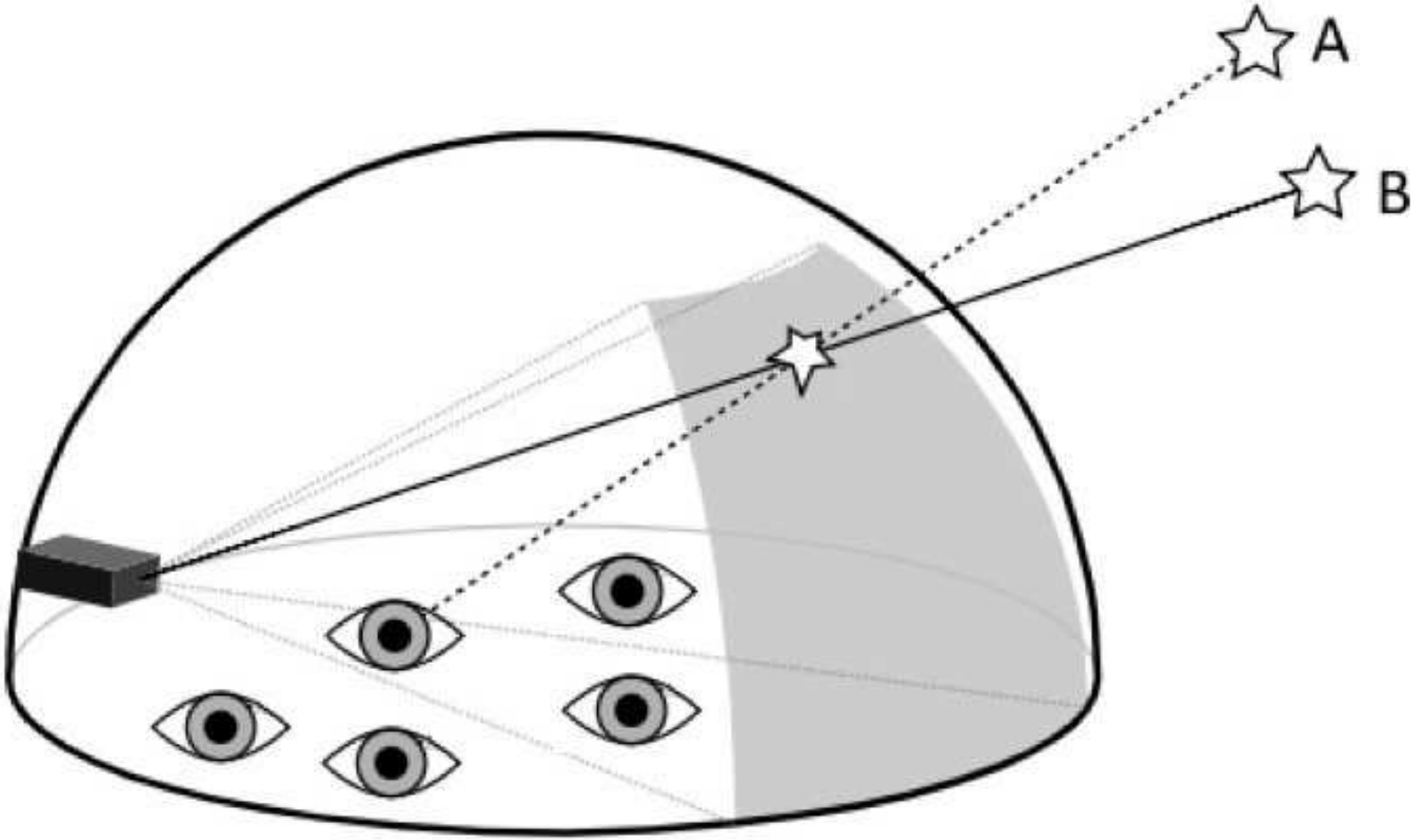
It's a Dome - so curved in 2 dimensions !



Extra Challenge: Eye != Camera



It gets worse !



Key Message

Full domes are all different (no standard spec)

There are times when it's useful to be able to build our own custom rendering engines !

Advanced Warning of Test !

Mid-Term In-Class Test

Test will take place in week *after* reading week
Wise man: "The second mouse gets the cheese"

Test will take place during the normal lab session
Arrive in MVB 2.11 promptly (unless AEAs)
Test will last 50 minutes (unless AEAs)

Only for students taking MAJOR variant of the unit
Students taking MINOR unit should join at 12 noon

Details

Test covers content of workbooks 2 to 5 inclusive
There will be NO programming required (yay !)
Test focuses on theoretical mathematical concepts

You will need a non-programmable calculator
Bring a black or blue pen (NOT red or green)

It will be "closed book" (no access to notes)
Single page containing all questions + answer boxes
Additional blank paper provided for "workings"

Reminder of Workbook Topics

- Position and Colour Interpolation
- Line and Triangle Drawing
- Triangle Rasterising
- 2D Texture Mapping (NOT 3D mapping)
- OBJ Geometry & Material files
- 3D Projection onto Image Plane
- Occlusion Problem
- Camera Position & Orientation
- Orbit & LookAt

Typical Question Scenario

You are given a single triangle
With vertices at specified locations
Camera at a particular point in 3D space

You will be asked to calculate various aspects:

`positions, orientations, colours, texture points`

Focus will be on SINGLE pixel/location/vertex
NOT all pixels on a screen - would take too long !

Most important tip !

Before attempting any question in the test
Draw an *appropriate* diagram of the situation
(Something that helps YOU visualise the problem)

These are typically quite complex 3D situations
Very difficult to visualise whole thing in your head

Simon's top tip: Draw the Situation !

Any Questions About The Test ?

Past Paper Questions ?

This is the first time we are running mid-term test
So there aren't examples of previous past papers

However...

selected questions from past exams are relevant
(halfway through unit, haven't covered all topics yet)

PDF of selected questions is available on GitHub
Provides good indication of what to expect in test

What Next ?

After in-class test, we'll continue with workbooks
Won't get chance for another briefing before then
So it's worth providing a few hints and tips today

Note: Next workbook is for AFTER the in-class test
(i.e. you won't find questions about it in the test !)

Next Workbook

In the next workbook, we will be changing tack
Good timing: in-class test first, then new direction !

Up until now, we have approached rendering by...
Projecting 3D vertices DOWN onto 2D image plane

In future, we will switch things around completely...
Reaching OUT from camera INTO the 3D scene

RayTracer

Calculating Intersections

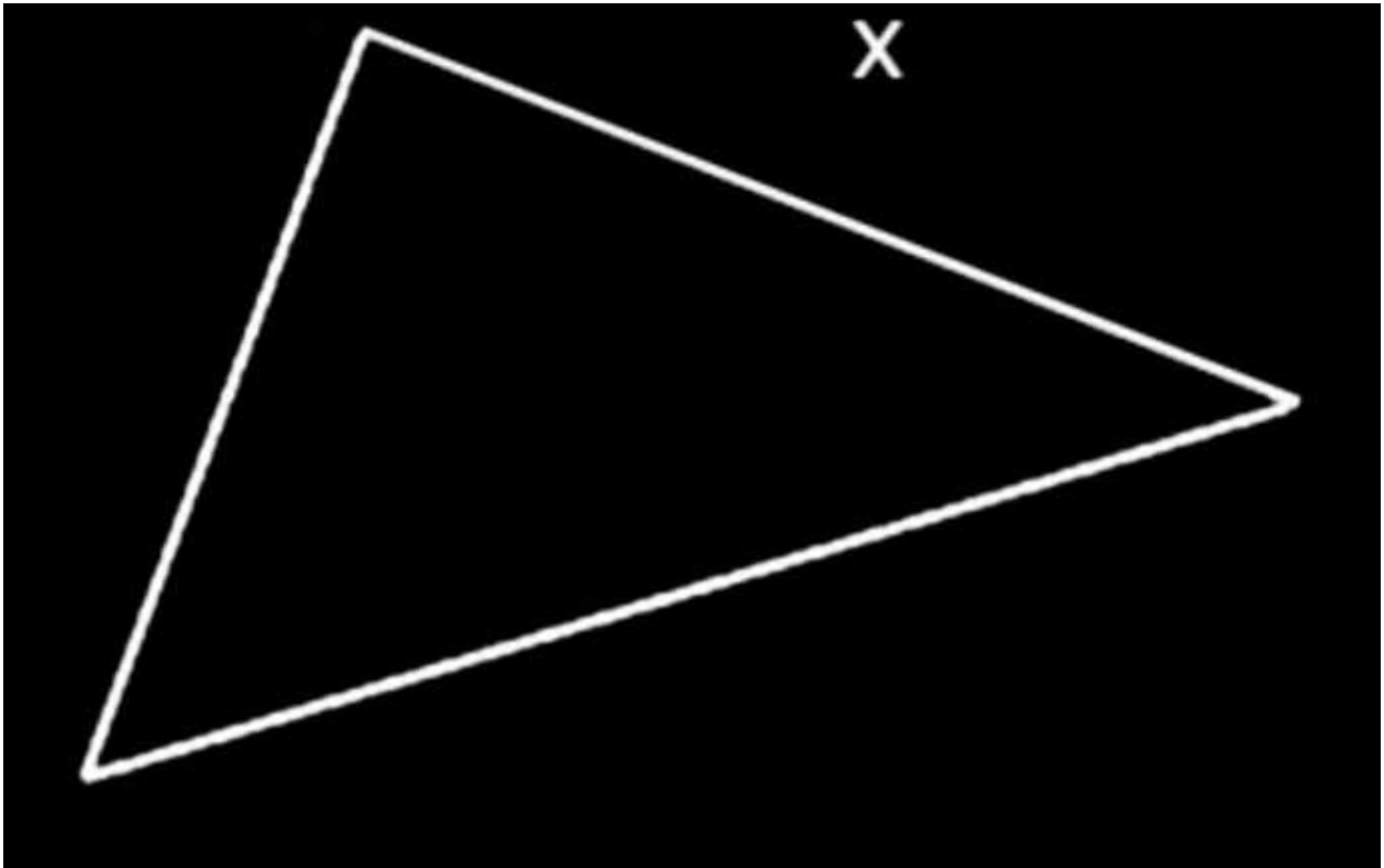
Rays represented as a start point + direction vector

Each triangle as a bounded area on a 3D plane

We can apply standard mathematical techniques...

In order to determine position of any intersection

Check that intersection is within bounds !

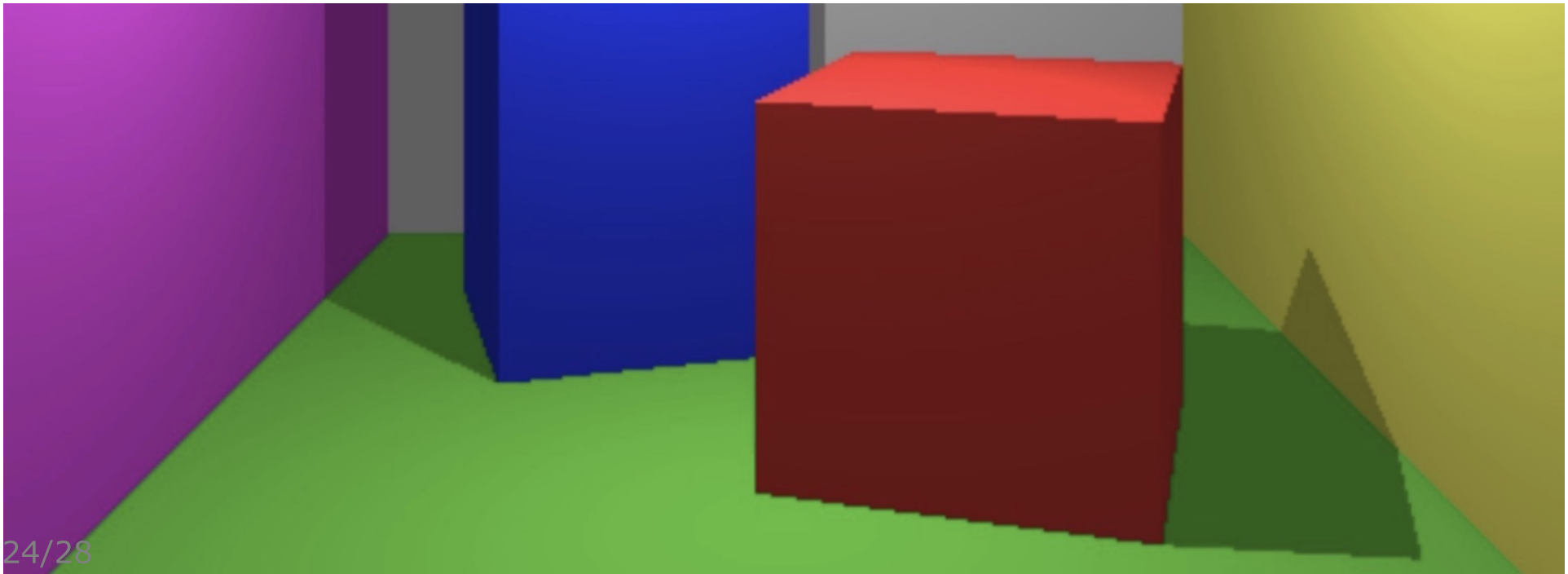


Additional Benefits of Ray Tracing

Ray Tracing permits various lighting effects to be used

Form example, creating Shadows is conceptually easy...

"Can a particular point on a surface *SEE* the light ?"



Full details are provided in the next workbook

Remember: this is for *after* the in-class test

Debugging Tips

As you are probably starting to realise...
Debugging 3D rendering can be particularly tricky !
Problem is the large amount of data washing around
You can't just print everything out !

IDE debuggers can be very helpful
But sometimes you need additional strategies...
A useful approach is "selective" rendering/printing:

SingleTriangleRenderer

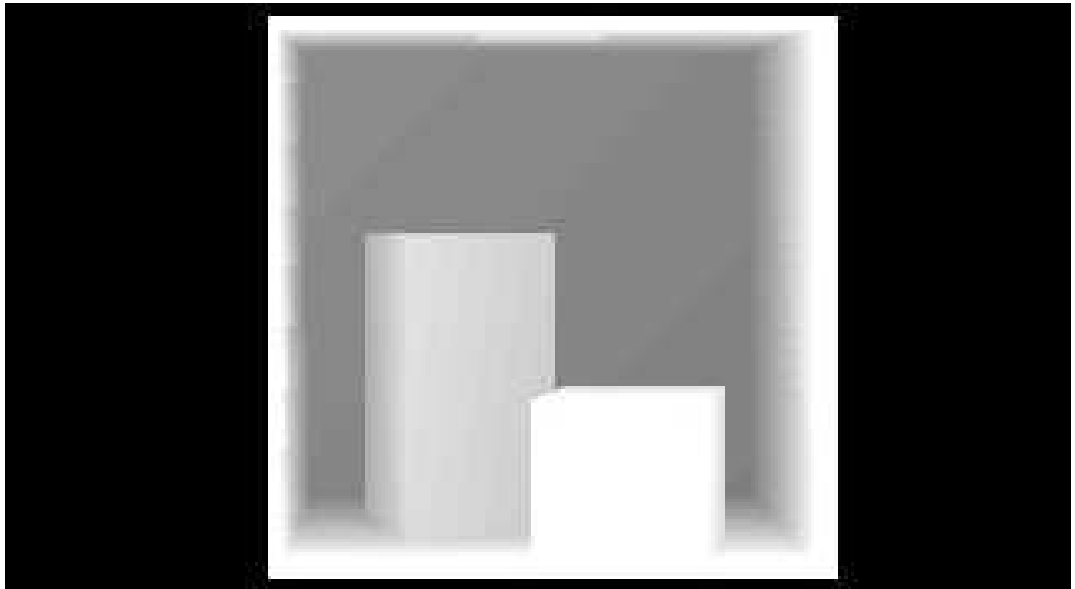
"Visual" Debugging

Rather than drawing pixels using their "true" colour

We can colour them according to a numerical value

For example, distance of points from the camera

Easier to see patterns than with numerical data



Follow me for more tips !

We've put together a document of debugging tips

README

Have put it in the "Debugging" folder on GitHub !