COMS30020 - Computer Graphics

Week 6 Briefing

Dr Simon Lock

Use of GitHub

Some peoples' attitude to GitHub is all wrong:
"I'll upload it when I have finished" X

GitHub isn't there as a *submission* platform
It is there to *support ongoing development*
It is where your evolving code should *live*

When you post a request for help on Teams
I'll often take a look at your code on GitHub
I've not been able to help some people fully

When your shower uses github more than you

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...

Immersive Media

VR has been around for many years

It has currently seen a resurgence

Partly due to cheap, high-res, light-weight screens

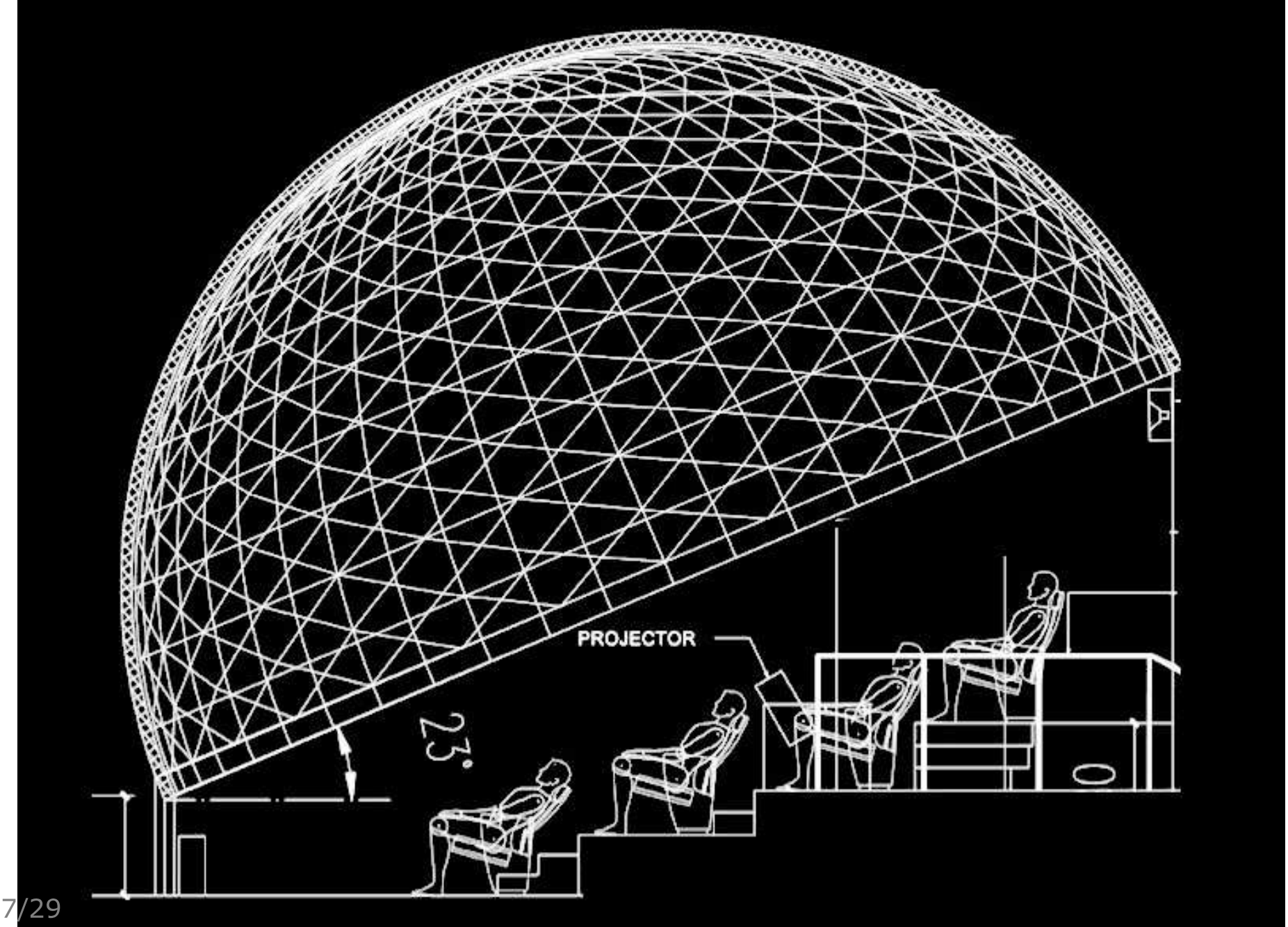
Many brands of VR headset (Oculus, HTC Vive...)

But what if we wanted something more social ?

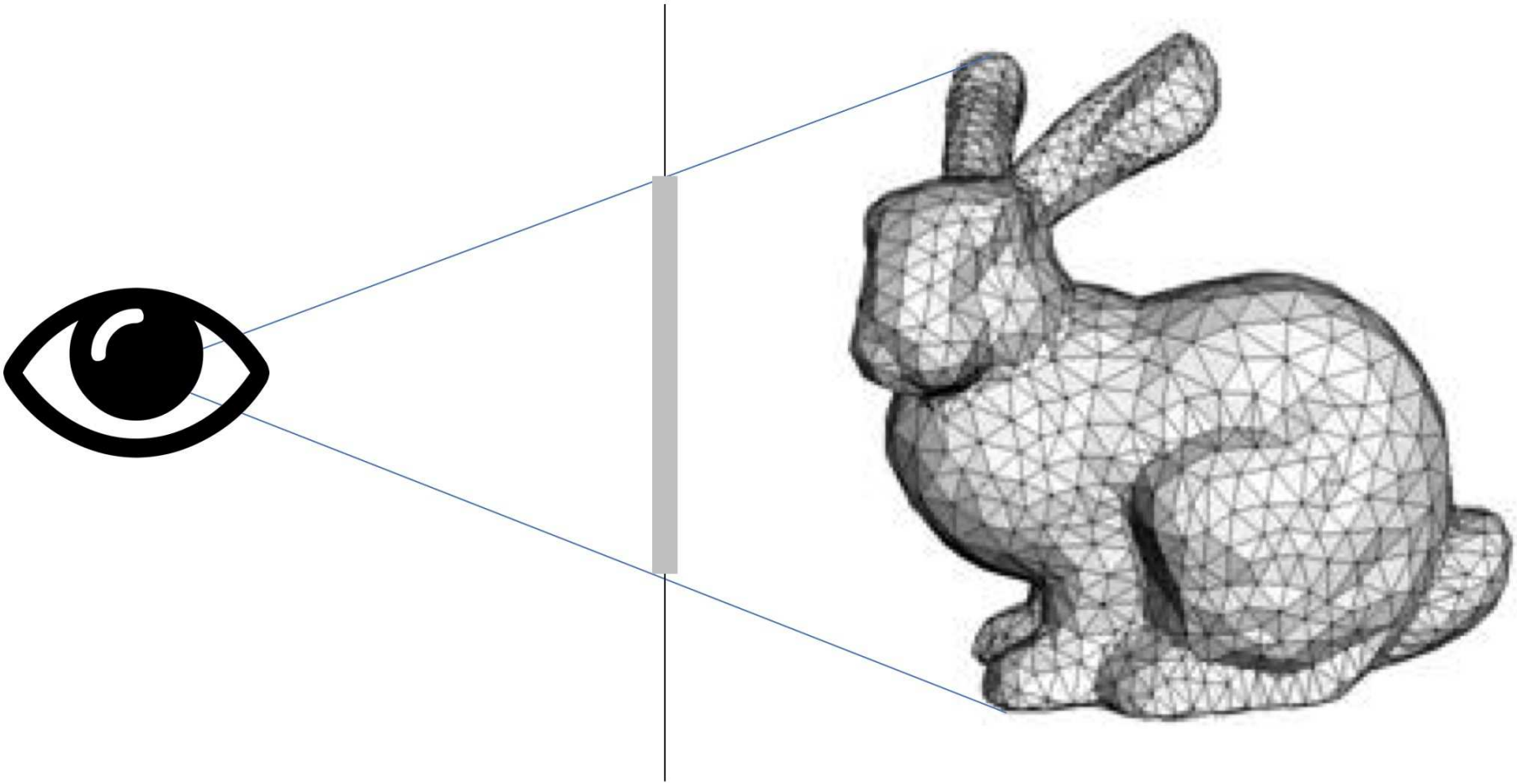
Some kind of "shared" group experience ?

Not isolated in separate headsets

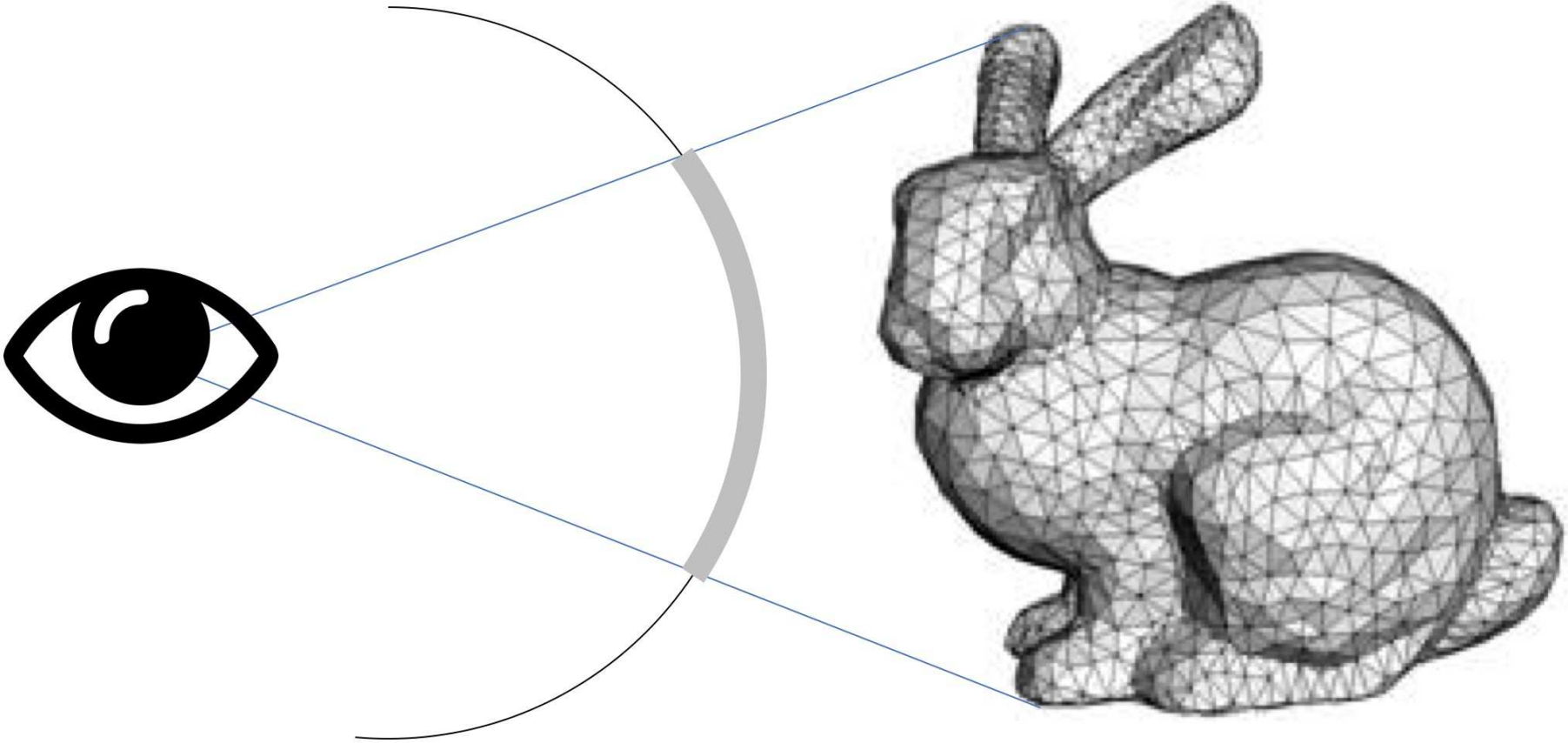
But together in the same room...


PROJECTOR
23'

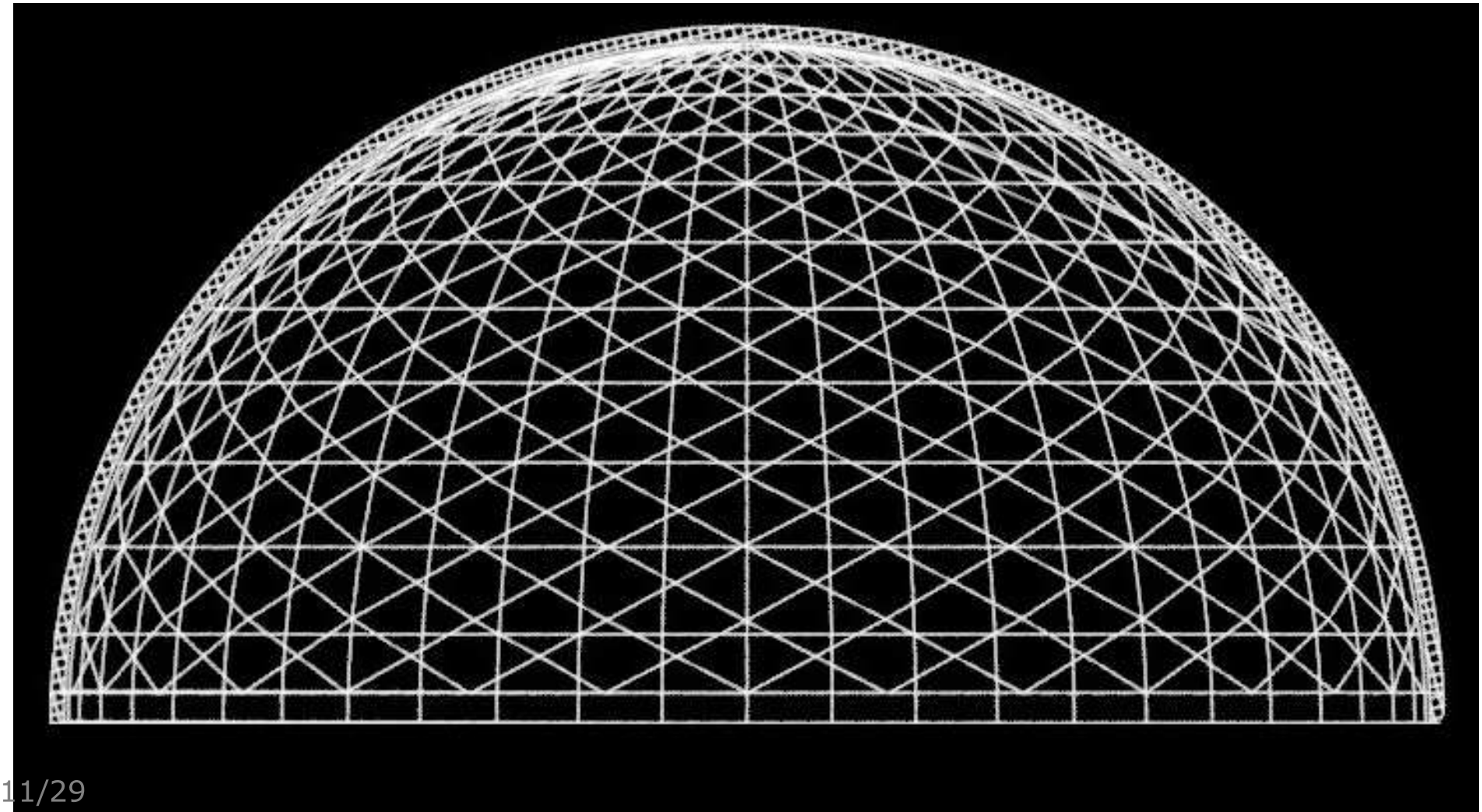
Traditional Renderers



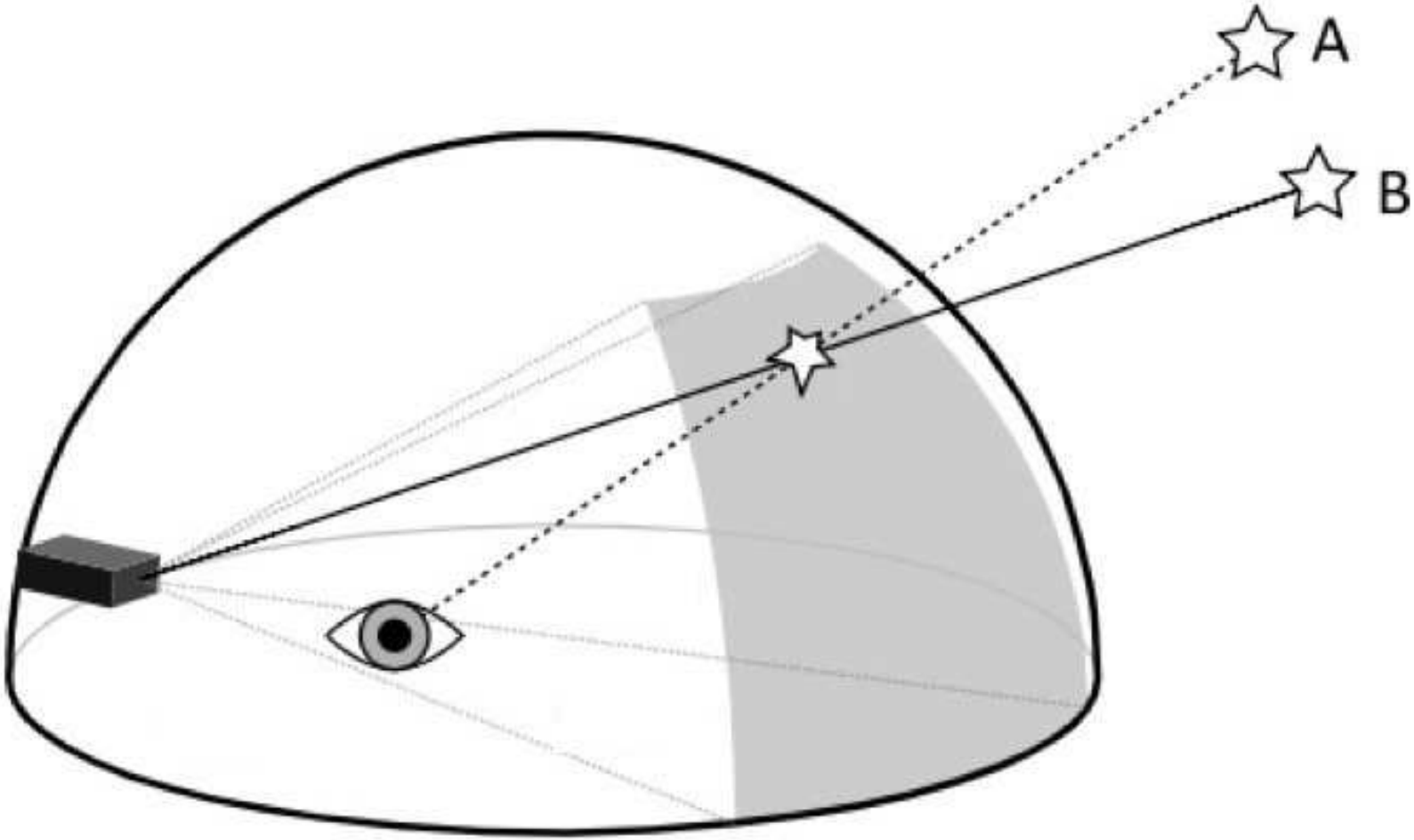
Custom Renderer !



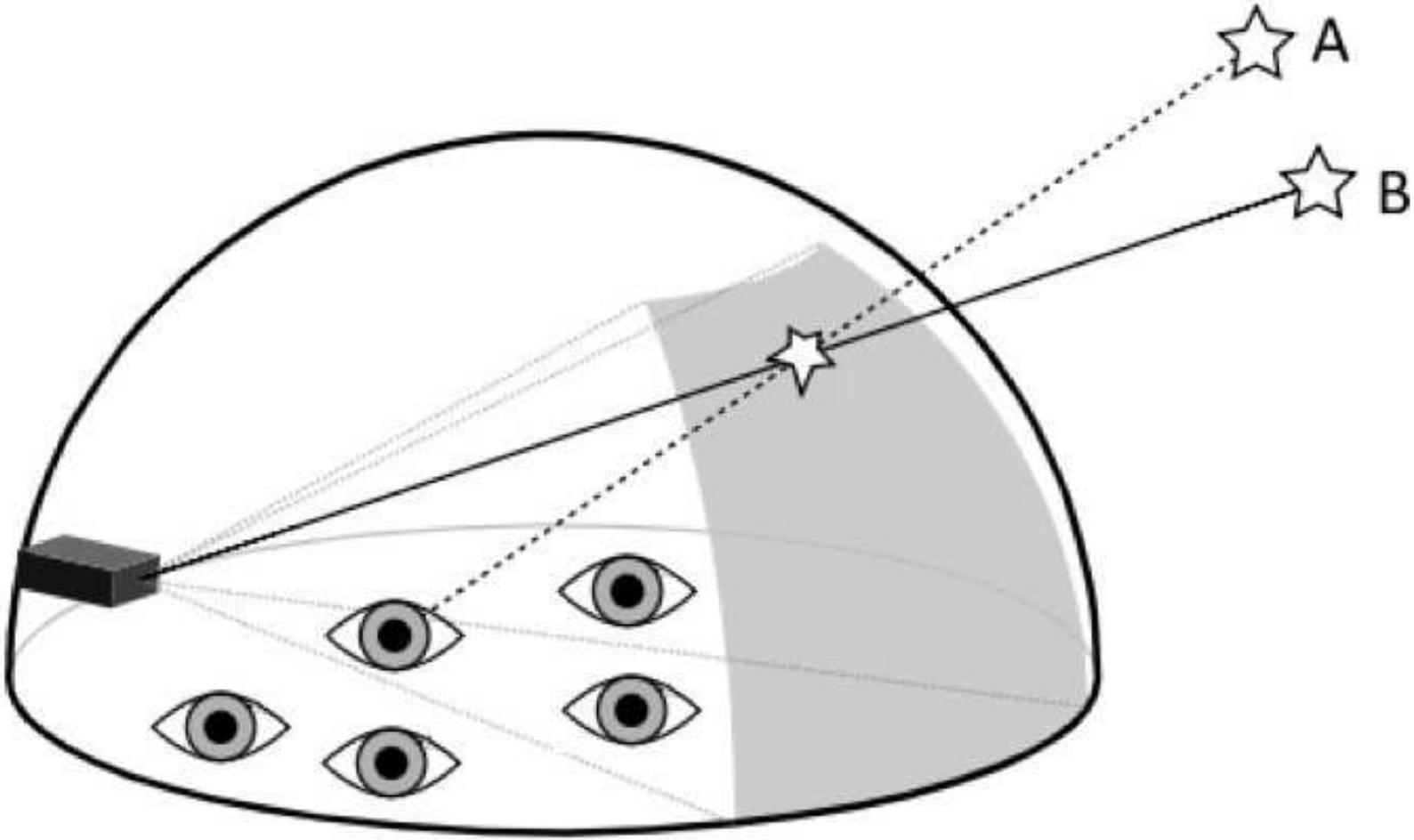
It's a Dome - so curved in 2 dimensions !

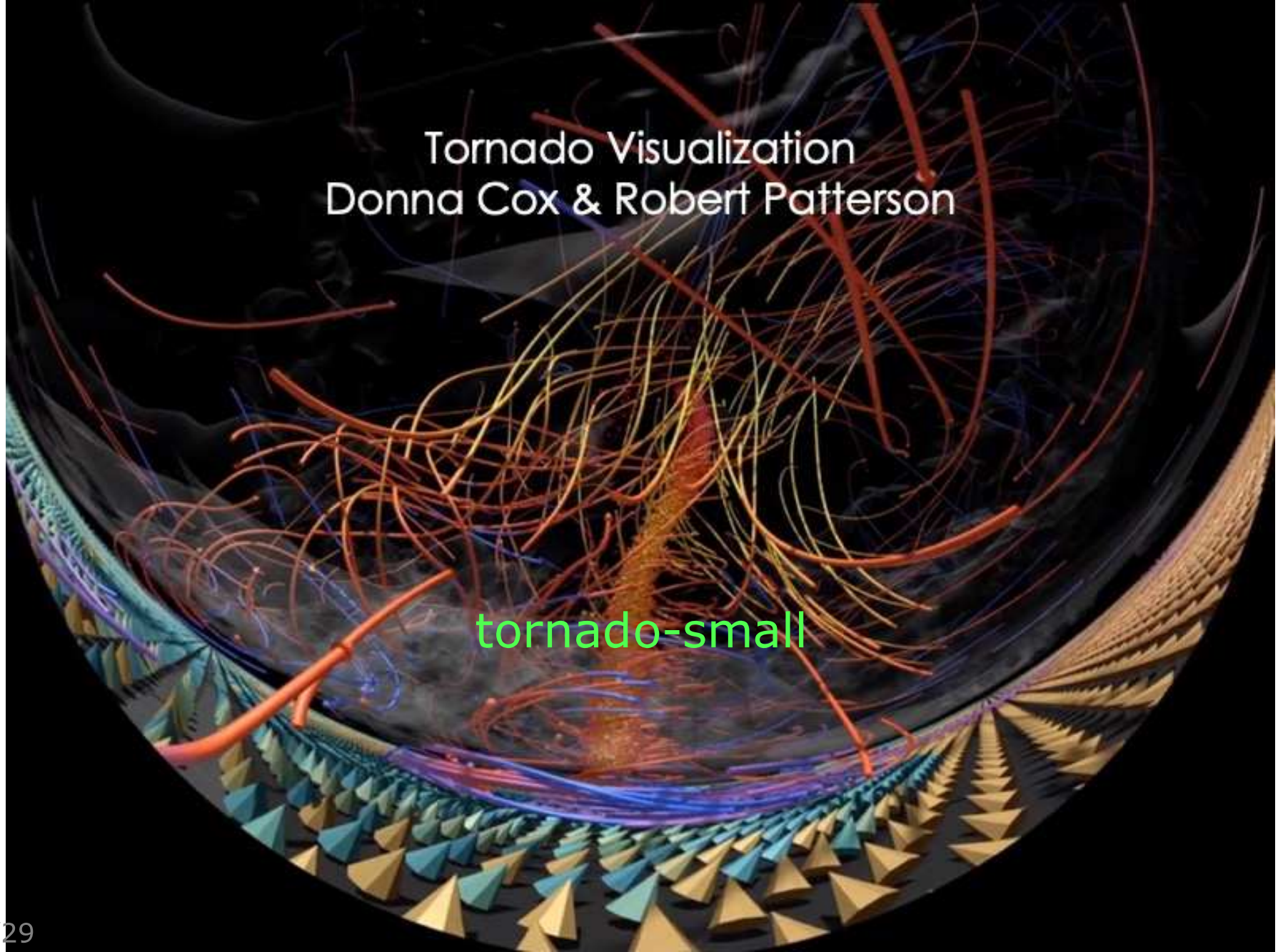


Extra Challenge: Eye != Camera



It gets worse !

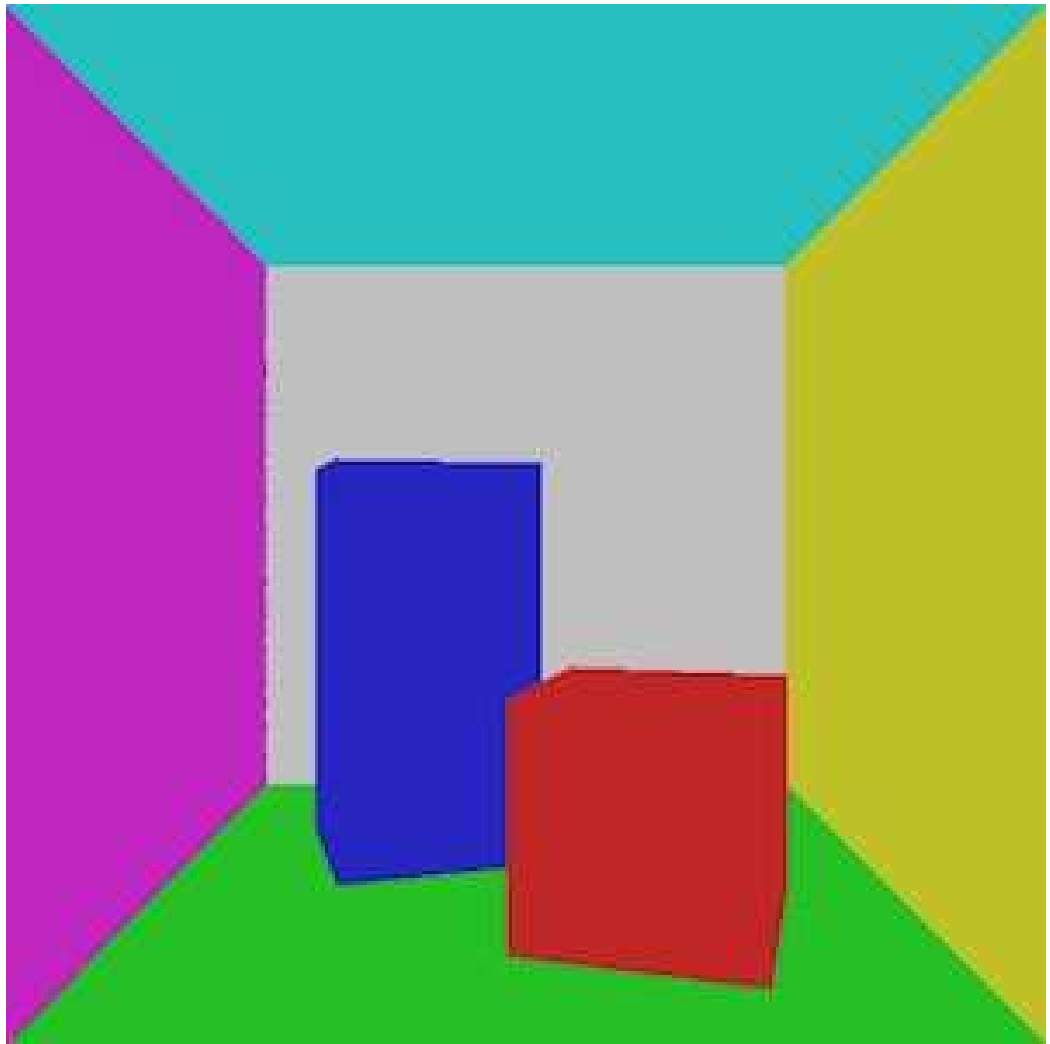



Tornado Visualization
Donna Cox & Robert Patterson
tornado-small

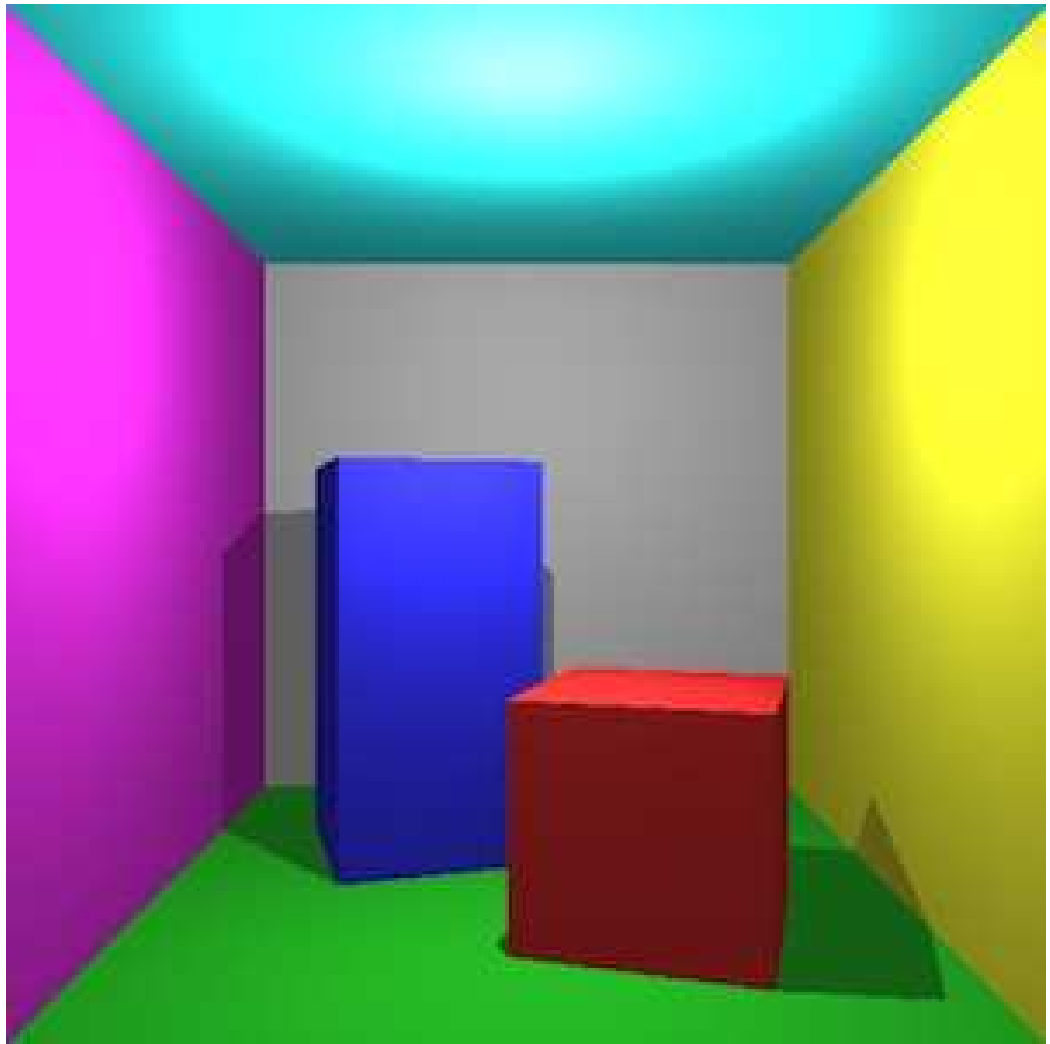
There are times when it's useful to be able to build our own custom rendering engines !

Lets move on

Where we are now: How could we improve it ?



How about this ?



What do we need to do this ?

We'll need to know location of light(s) in 3D space
So that we can calculate:

- Reachability/Visibility of surfaces from lights
- Distance of surfaces from lights (dissipation)
- Angle that light falls onto surfaces (obliqueness)

Calculating all of these requires 3D spatial data
The problem is we lost a lot of this data...
When we "flatten" the scene onto the image plane

In order to solve this problem

We need a whole new approach to rendering

We need to switch things around completely:

Reaching OUT from the camera & image plane...

INTO the 3D scene

(which is where the data is !)

RayTracer

Don't worry...

We won't just throw away your rasterising code !

We are STILL going to need it

It's just that we'll build a Ray Tracer AS WELL

The two approaches are in fact complementary

Let's take a look at the next workbook:

https://github.com/COMS30020/CG2023

Don't Panic !

This may seem like a whole new departure
But don't panic - it's not that different

But we'll still be using a lot of our existing code
There's a fair amount of common calculation

It may seem like there is a lot of material
But there is only 19 minutes of video
(and some of that is derivation of formulae)

Remember:

This workbook is intended for *Week 7*

We introduce it now since there's no briefing next week
Use reading week to get up-to-date with workbooks

We WON'T be releasing another workbook next week !

Testing your code

You can't FULLY test your code after each task
You will have to complete a few tasks first:

- Calculate Closest Intersection
- Validate the Intersection
- Ray Trace the Scene

Before you can extensively test your code

This is because individual test cases might work
But you won't know if it works in all situations
Until you render the whole scene

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 an additional strategy...
One solution is "selective" rendering/printing:

SingleTriangleRenderer

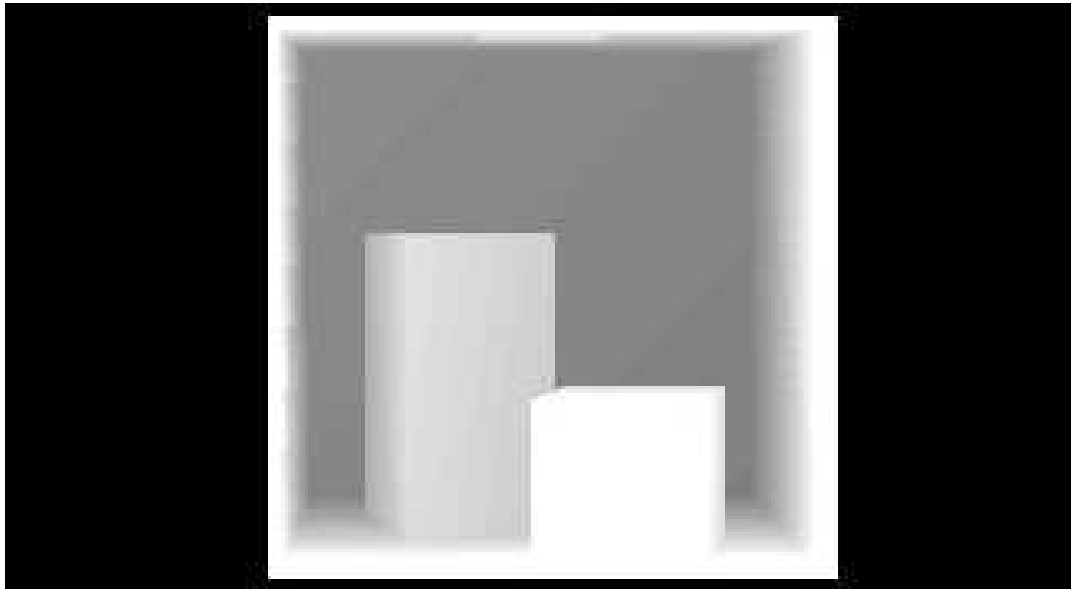
"Visual" Debugging

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

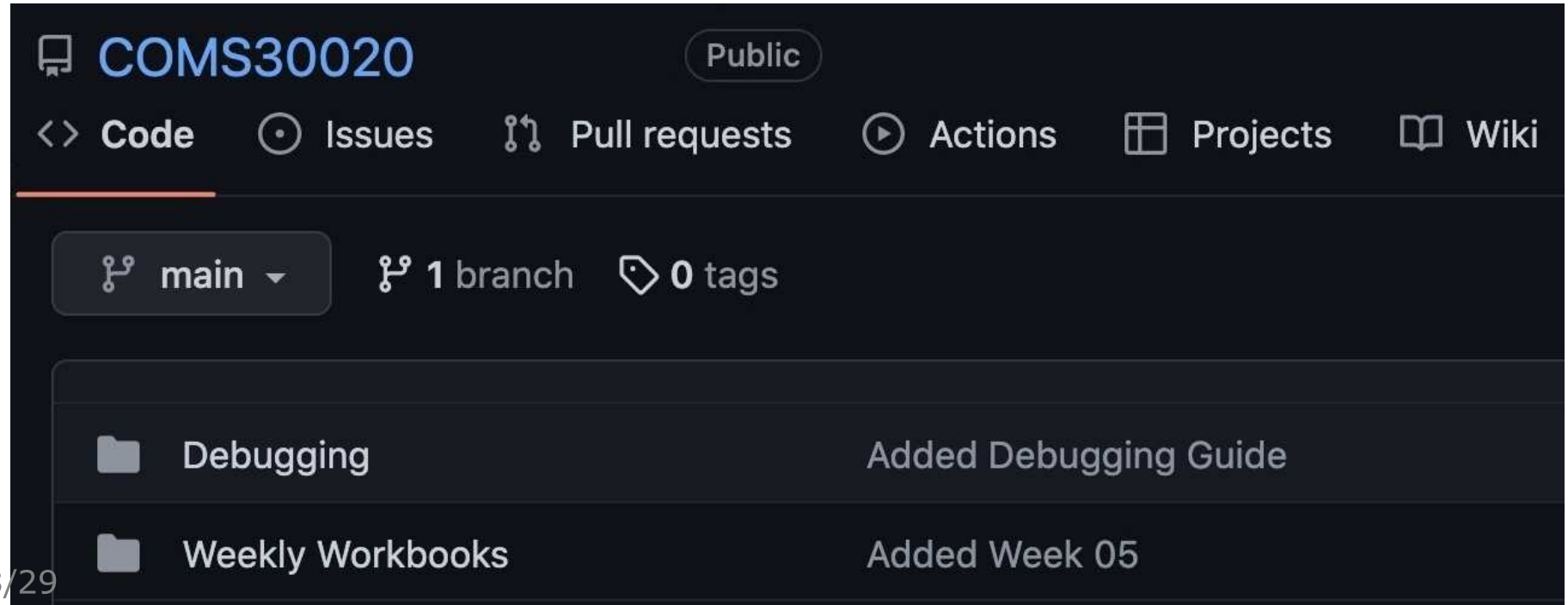


Follow me for more tips !

We've put together a document of debugging tips

README

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



Questions ?