COMP30019 – Graphics and Interaction

Project 1 Feedback

Dr Jarrod Knibbe
Alex Zable

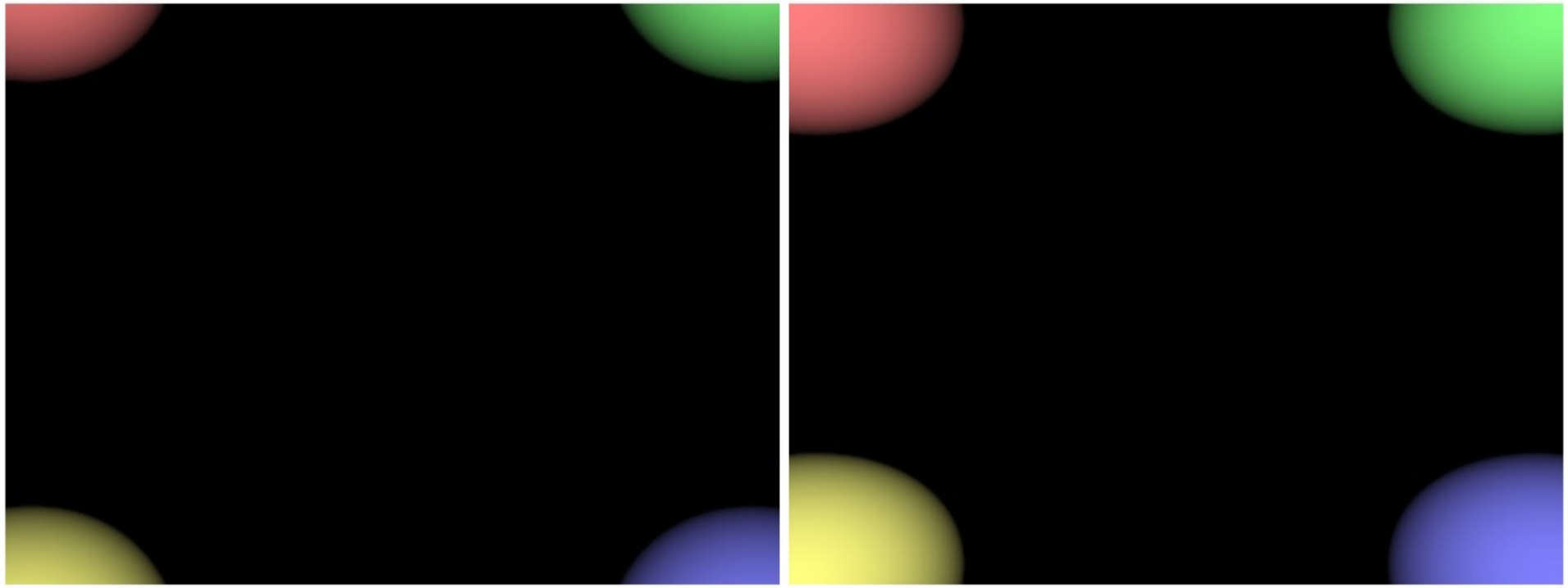
THE UNIVERSITY OF MELBOURNE

Overview

- Overall well done - congrats!
- Approximately 25% H1s (24+/30)
- Hope it was both fun and a great learning experience!
- Be sure to transfer your skills to project 2…
  - Project 1 developed your graphics **toolbox**
  - Vector maths
  - Dot product
  - Cross product
  - Theory of lighting
  - And lots more…

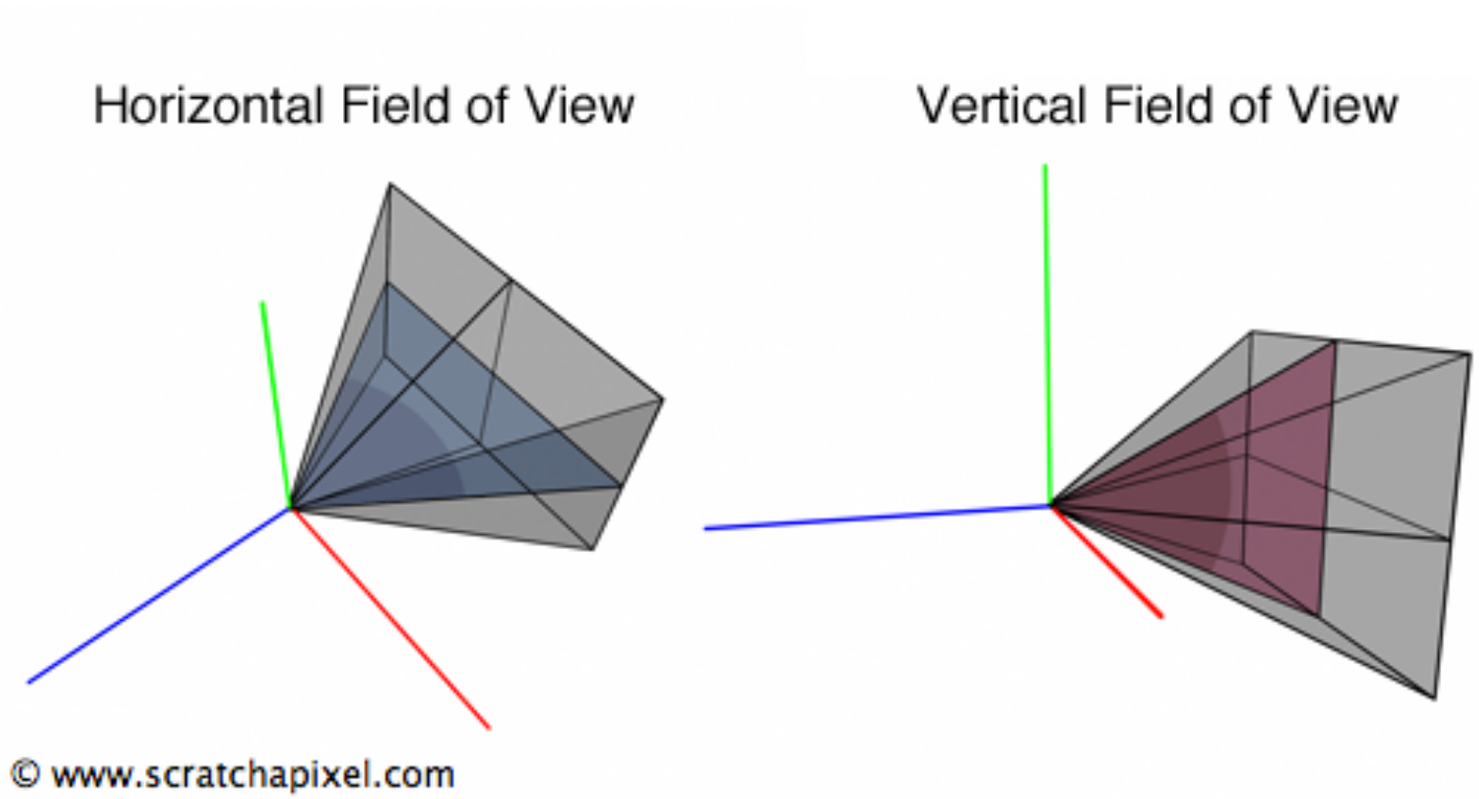
Stage 1

- **Camera, Shape, Colour, Depth**
- Overall strong performance from most
- *Very* common issue: Camera configurations!



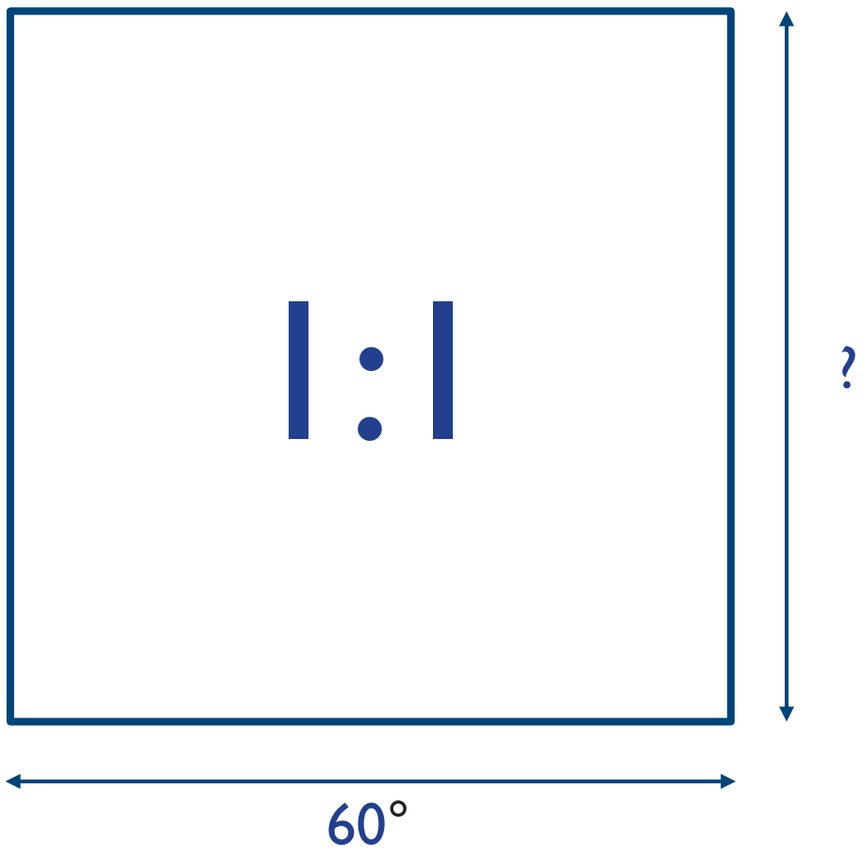
Field of View

- *Horizontal* FOV of 60°
- What does this mean for the *vertical* field of view?



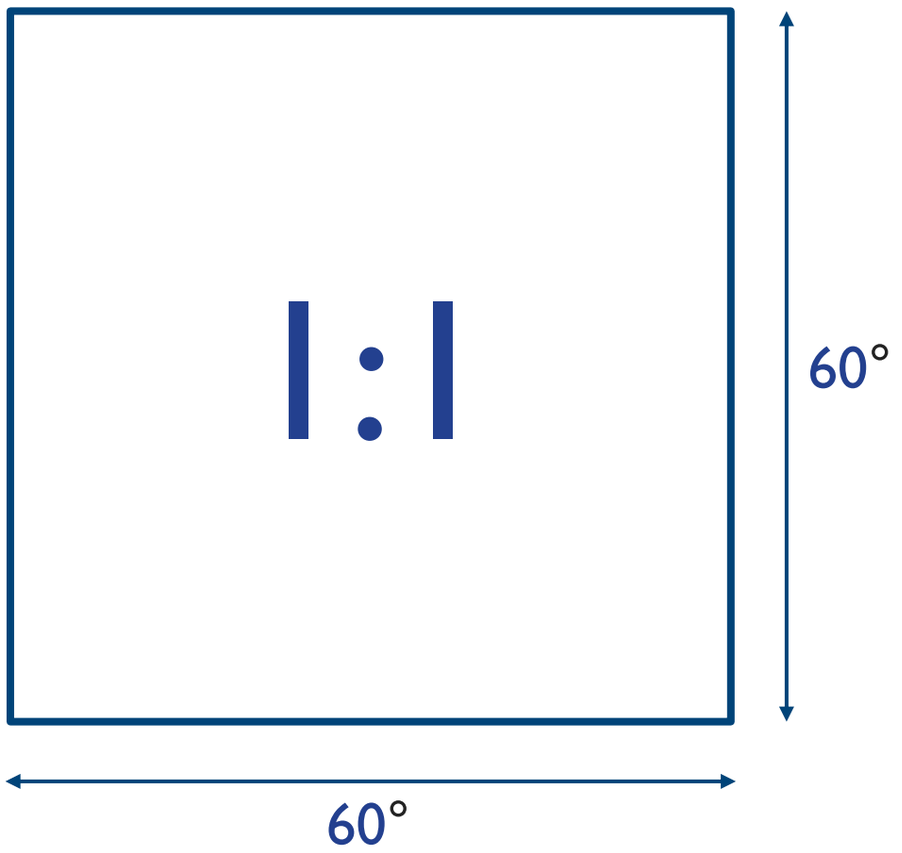
Field of View

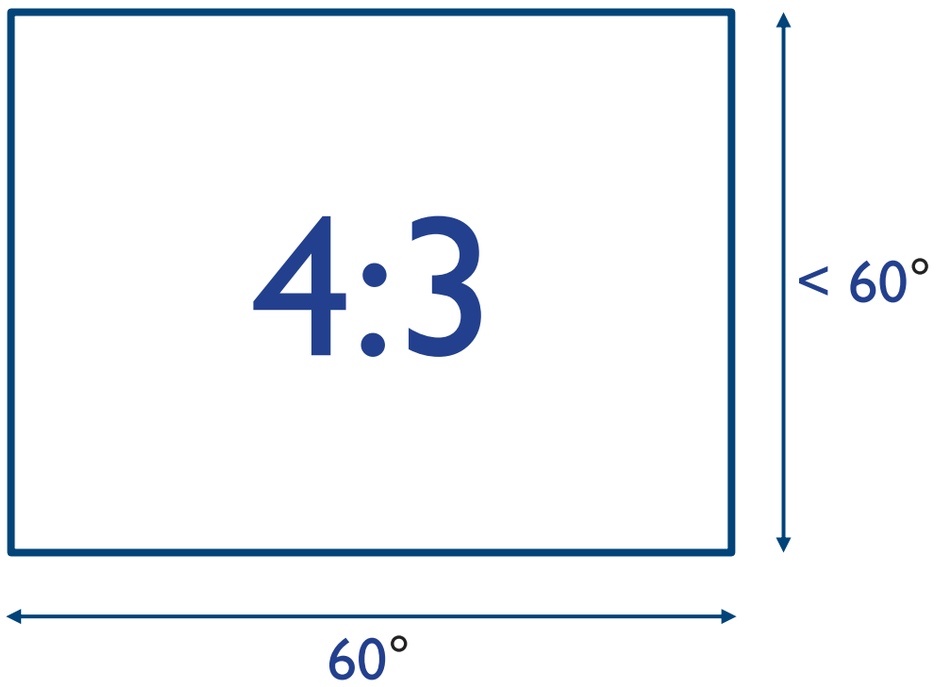
- How does FOV relate to the aspect ratio?



Field of View

- How does FOV relate to the aspect ratio?





Field of View

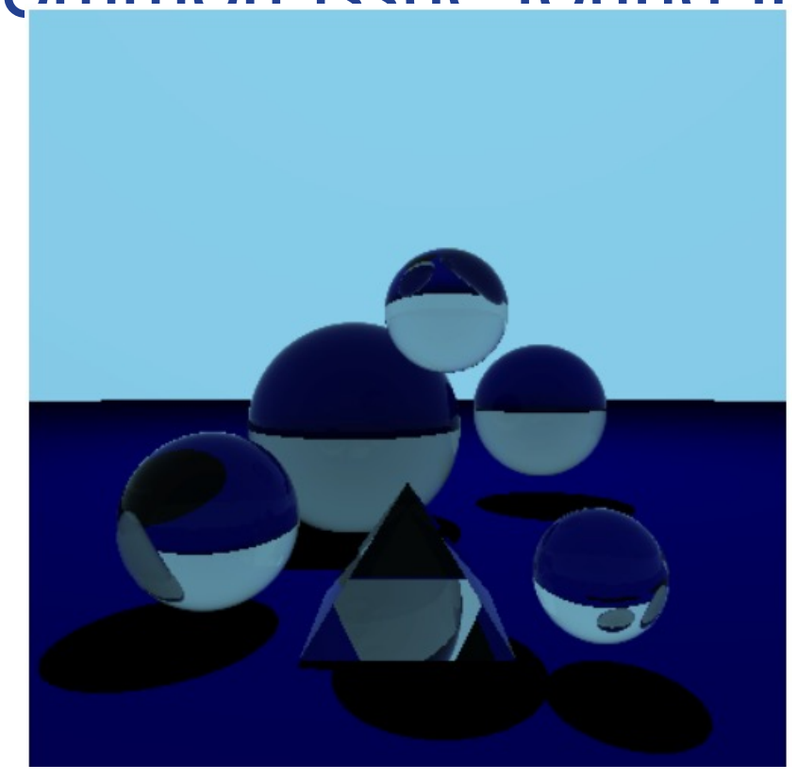
- Aspect ratio and FOV are important concepts! Why?

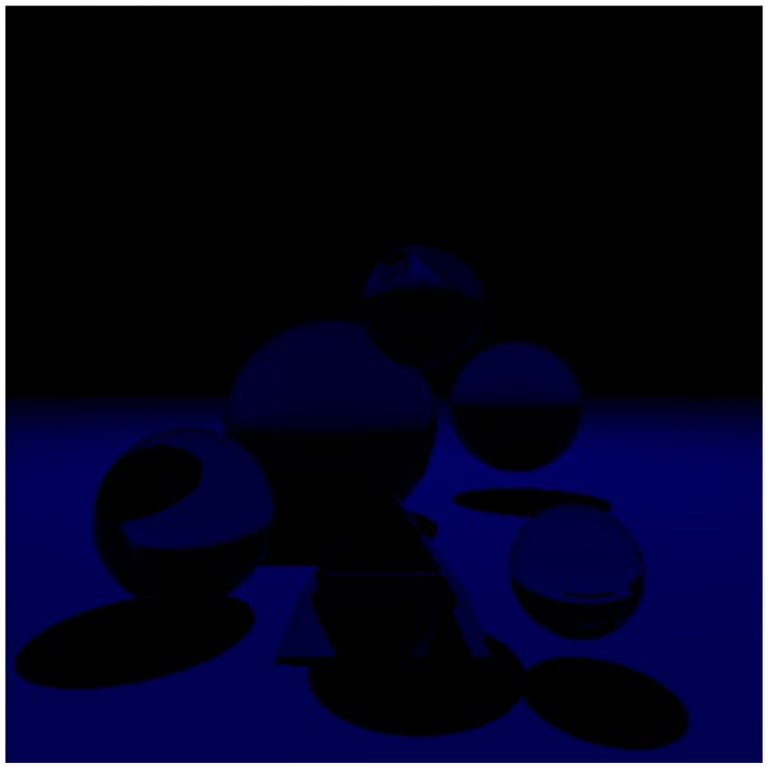
Field of View

- Aspect ratio and FOV are important concepts! Why?
- Ultra (21:9) and super-ultra (32:9!!) wide screens…
- Considerations in game development?

Stage 2

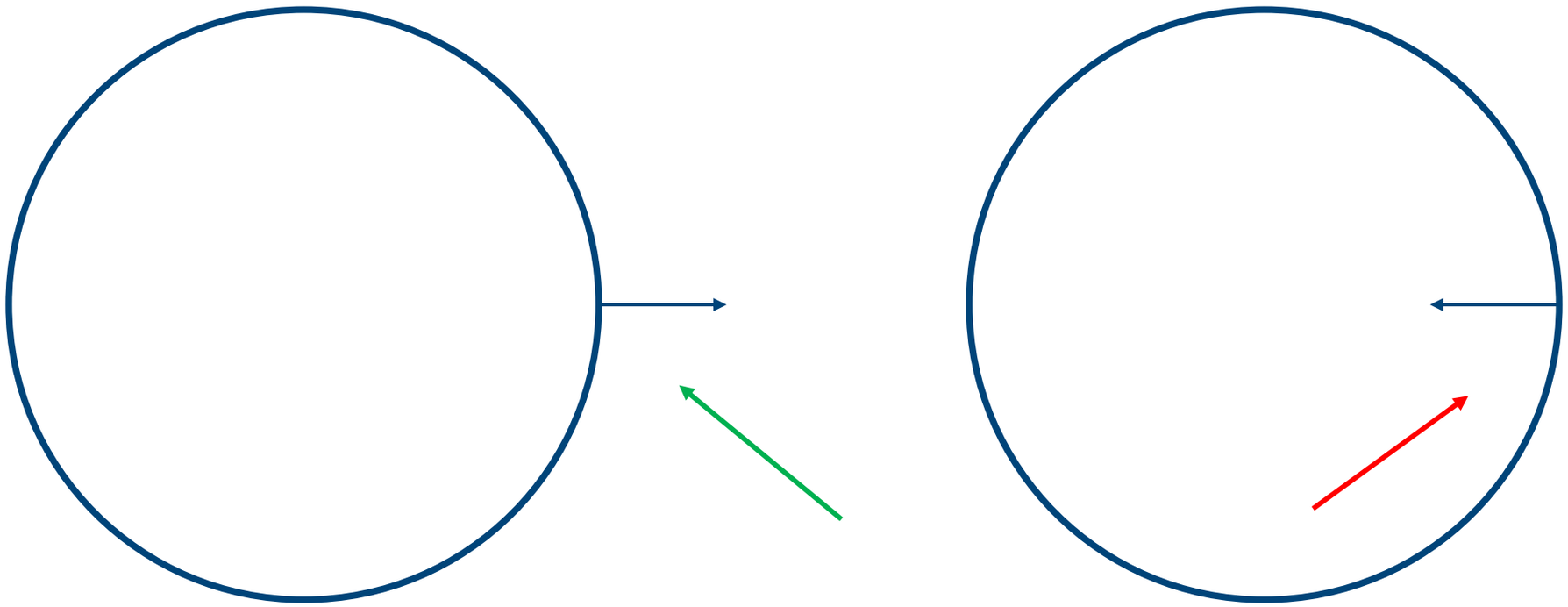
- **Lighting, reflection, refraction (+Fresnel), anti-aliasing**
- Getting harder… but still generally well done!
- Most common issue found in test case 3!





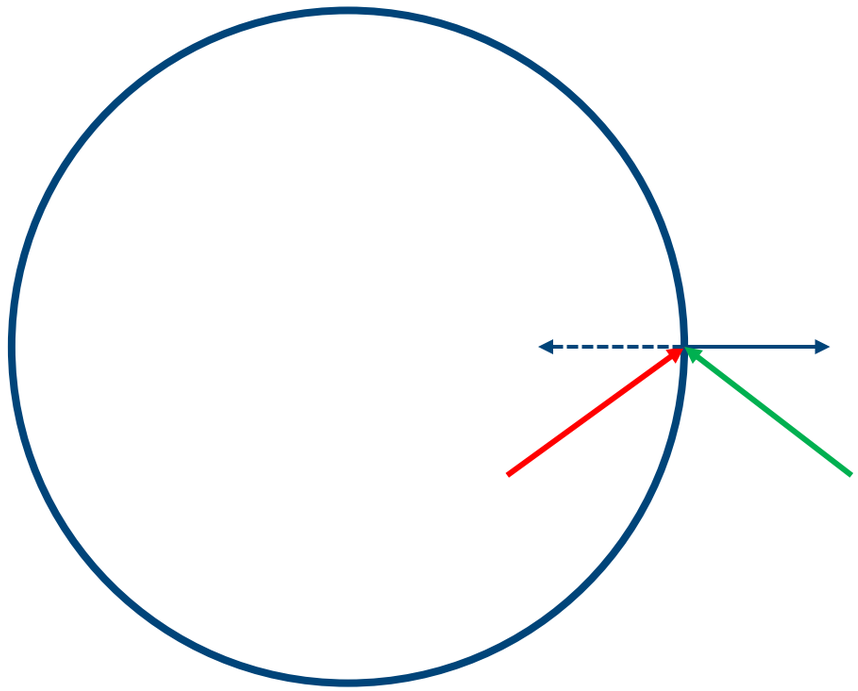
Stage 2

- Sphere intersections and normals



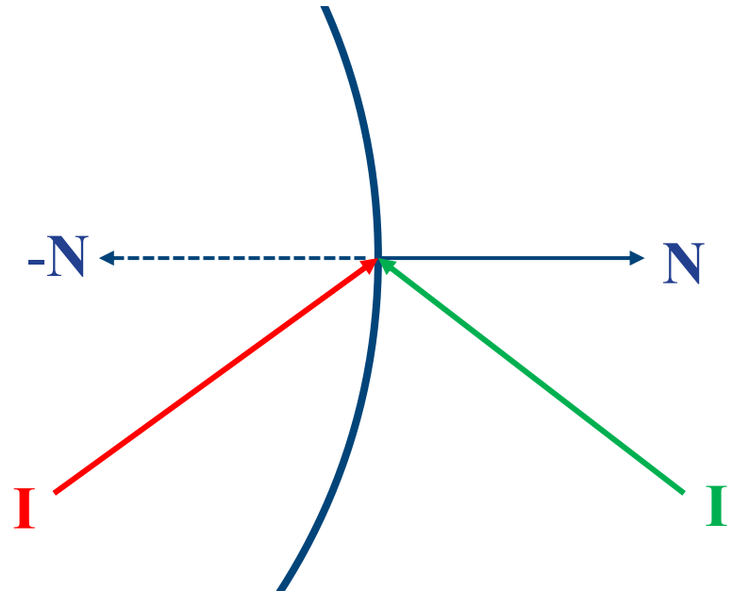
Stage 2

- How can you figure out the normal's direction?
- What maths operation can we use?



Stage 2

- **Dot product** to the rescue!



Dot(**I**, **N**) < 0 … use **N**!
Dot(**I**, **N**) > 0 … use **-N**!

Stage 2

- You probably thought about this with refraction!
- But it *generalises* for all lighting calculations!

- The sphere's `Intersect()` method should handle this transparently, *not* the ray-tracing logic!

Stage 2

- Always think about **edge cases**
- It's super important, not just 'academically'.
- Come up with your own test cases!

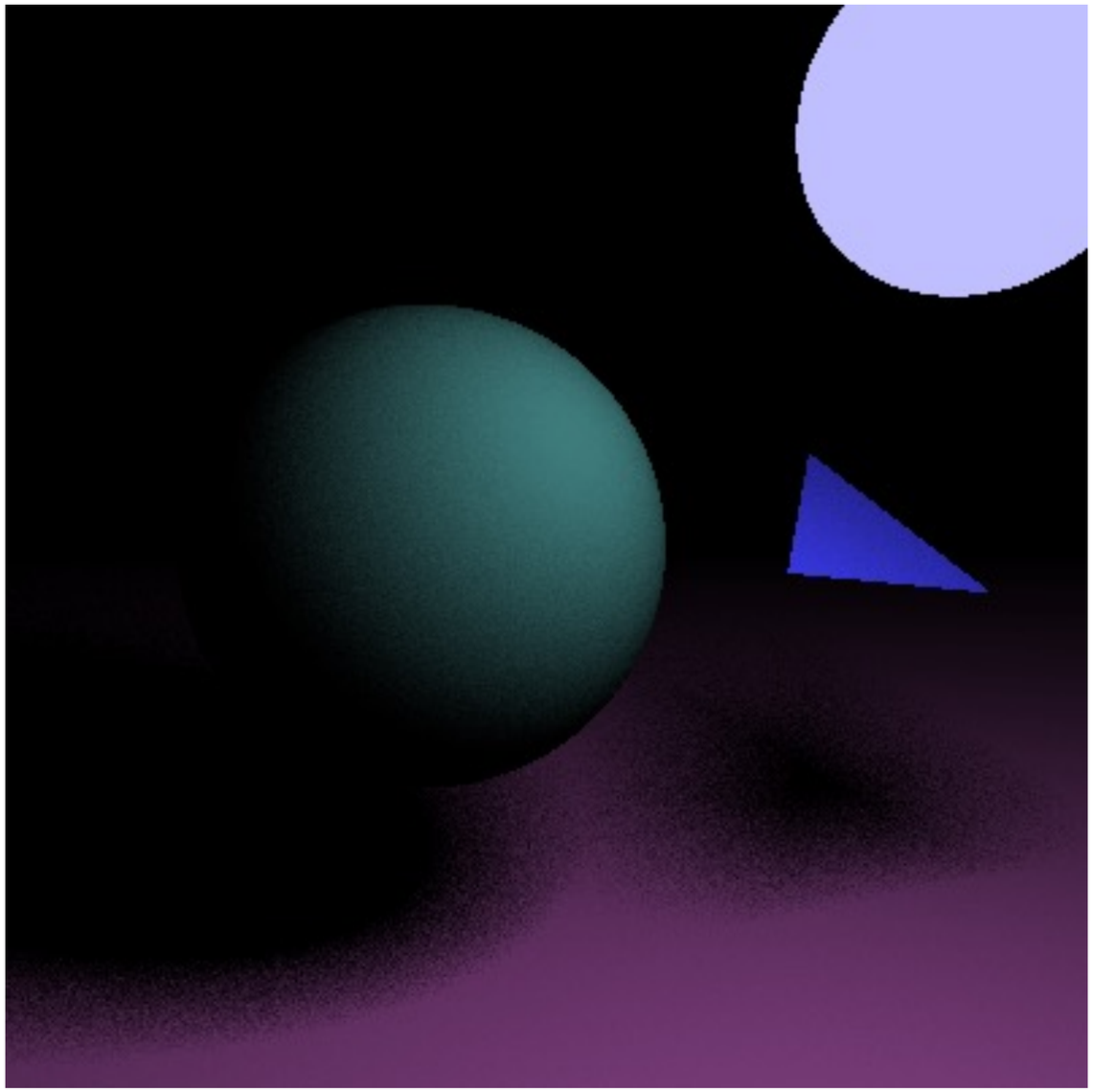
- The refractive pyramid also tripped up many…
- Why should a sphere be the only type of refractive object?

Stage 3

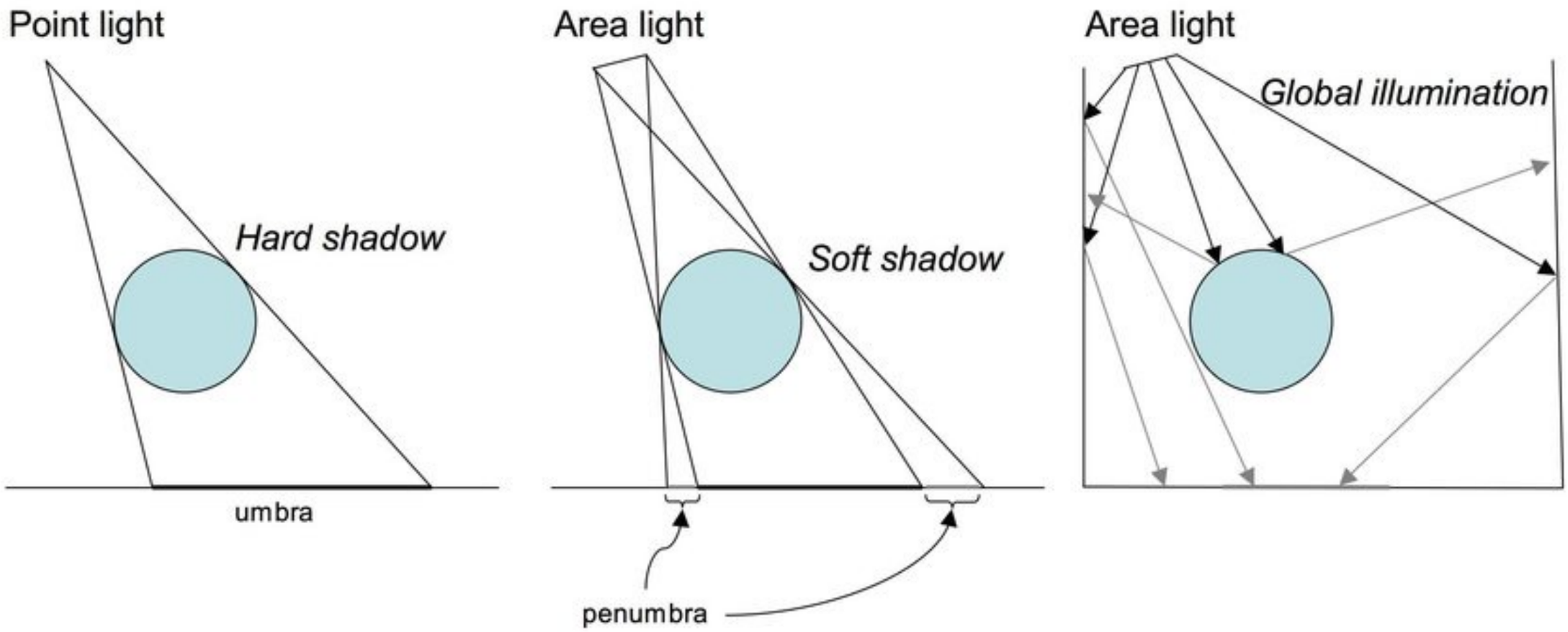
- Naturally a lot more challenging…
- Full marks was very rare!
- These were **research** questions, not just ‘coding’.
- Not all questions had a perfect or ‘correct’ answer.

Stage 3A – Emissive Materials

- Emissive materials
- Soft shadows – why?



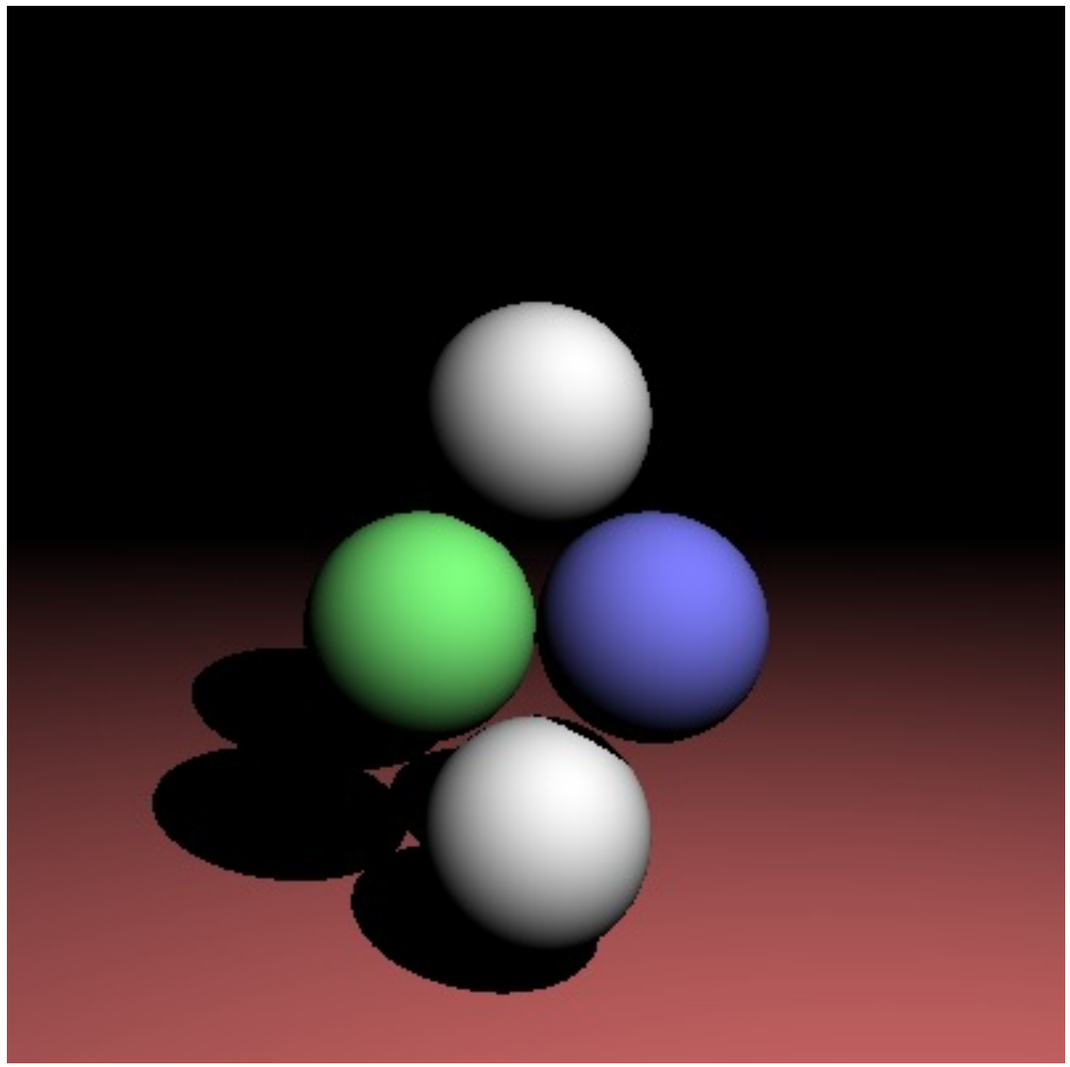
Stage 3A - Emissive Materials

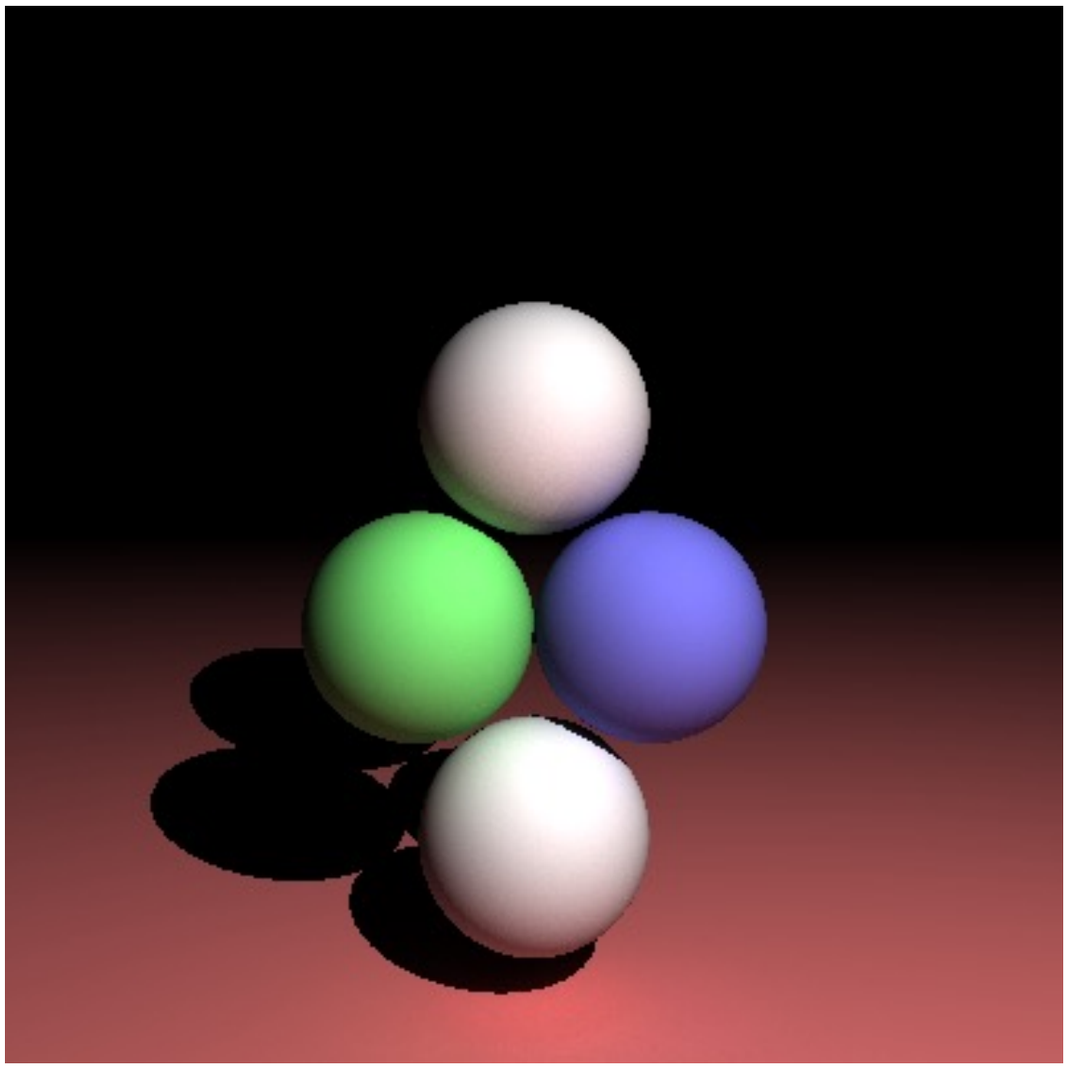


https://www.researchgate.net/figure/Different-approaches-to-shadow-creation-Hard-shadows-are-generated-by-approximating_fig11_265514880

Stage 3B – Ambient Lighting

- Ambient lighting





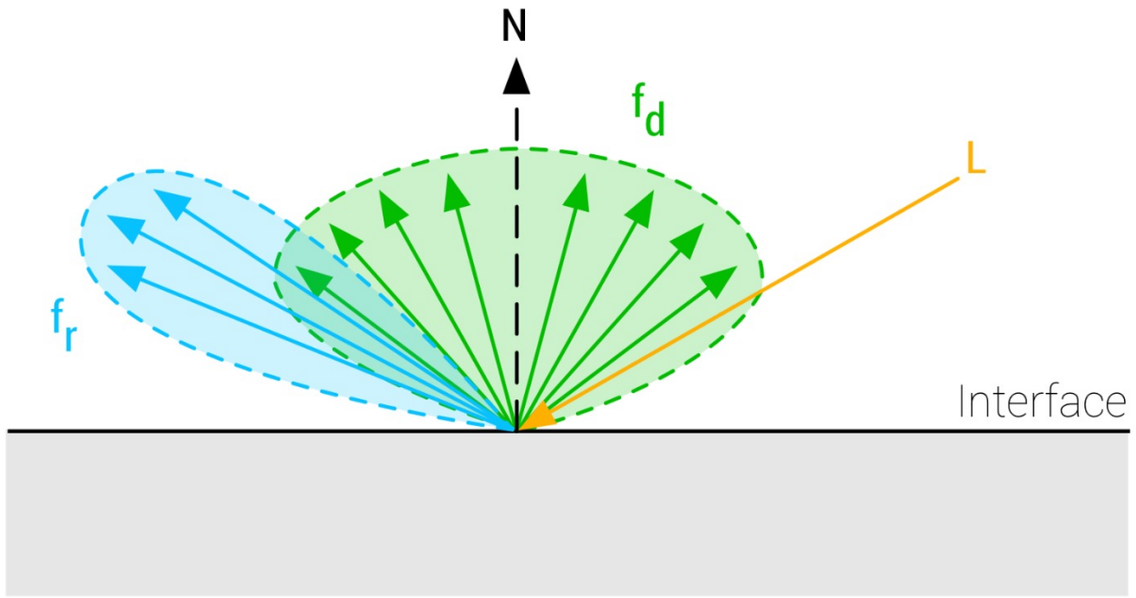
Stage 3B – Ambient Lighting

- Infeasible to simulate perfectly. Why?

Stage 3B – Ambient Lighting

- Infeasible to simulate perfectly. Why?

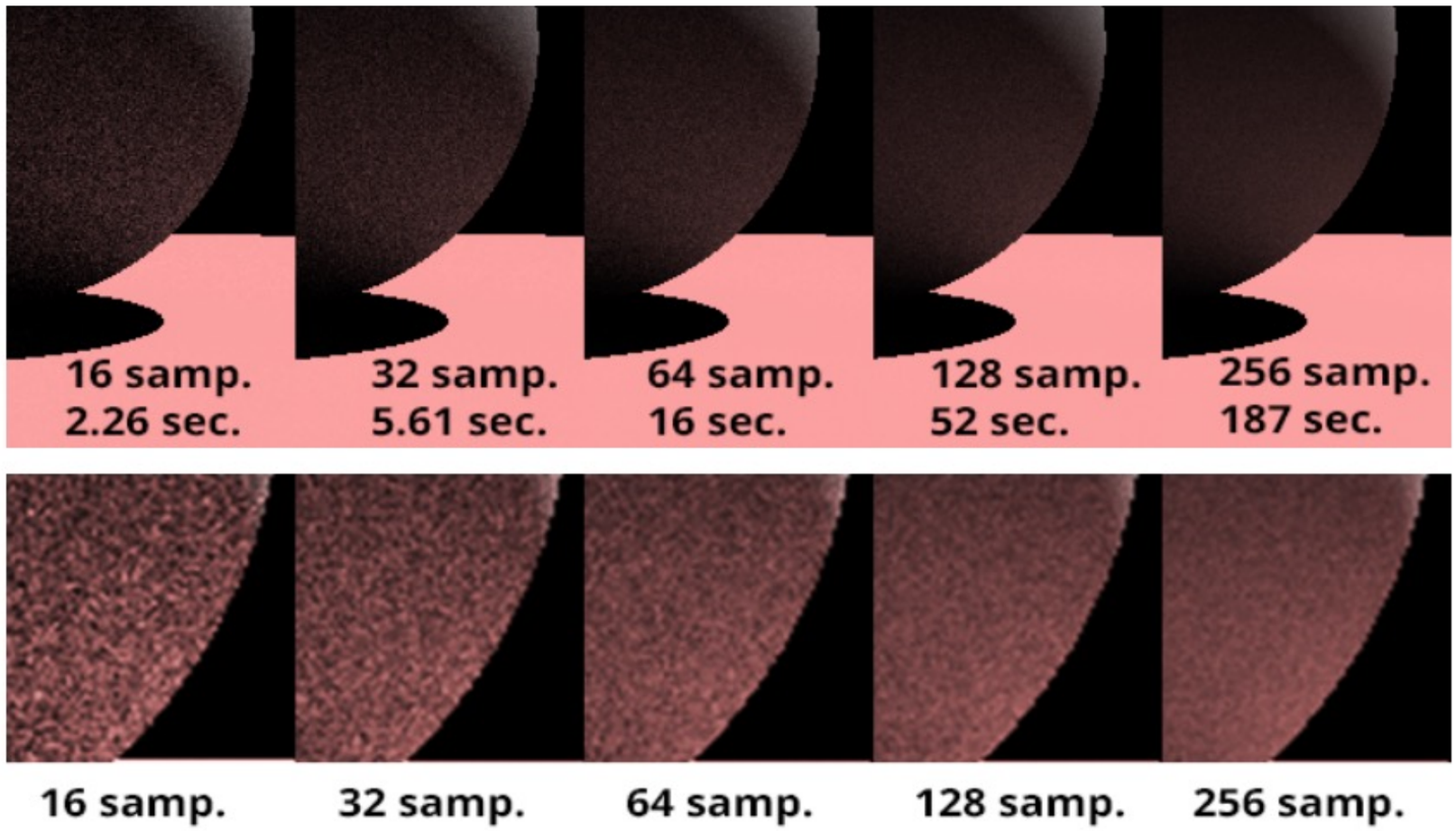
- Need to use ‘Monte-Carlo’ techniques – ‘path tracing’
- Random simulation of rays in the scene
- ‘BRDF’ functions can model reflection for different surfaces



Source: https://google.github.io/filament/Filament.html

Stage 3B (and 3A)

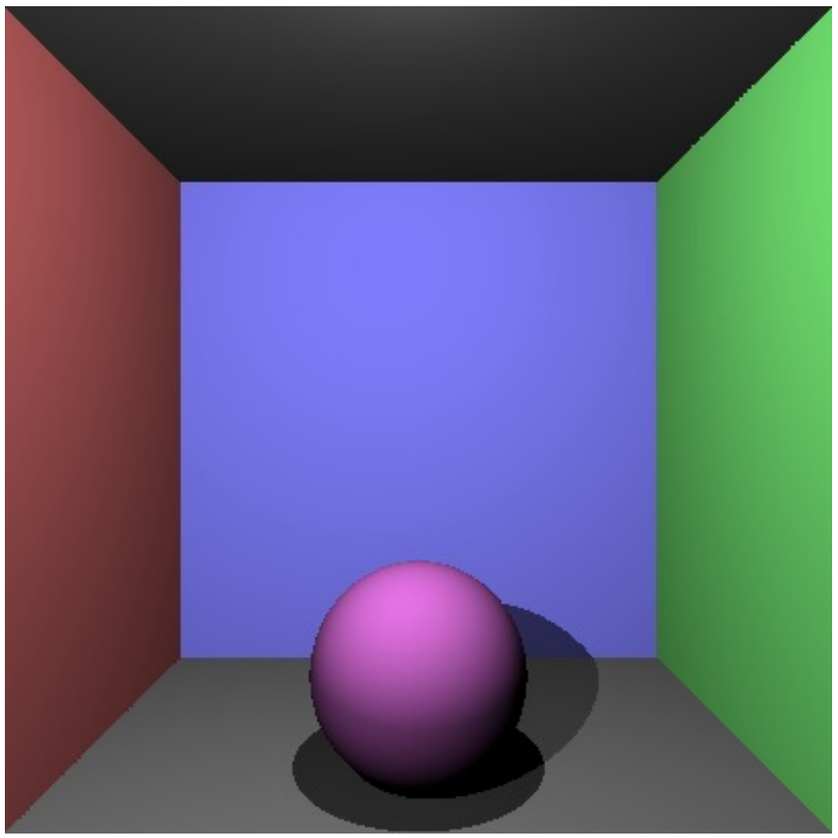
- **Noise** is also a key consideration!
- Trade-off between computation time and noise…

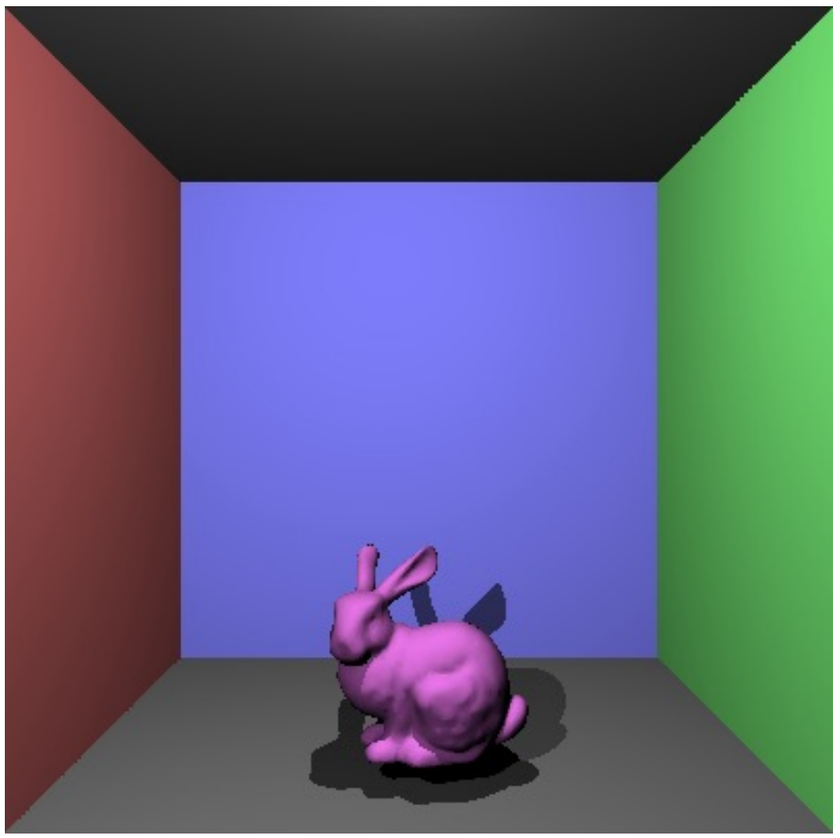


Credit: scratchapixel.com

Stage 3C – OBJ Models

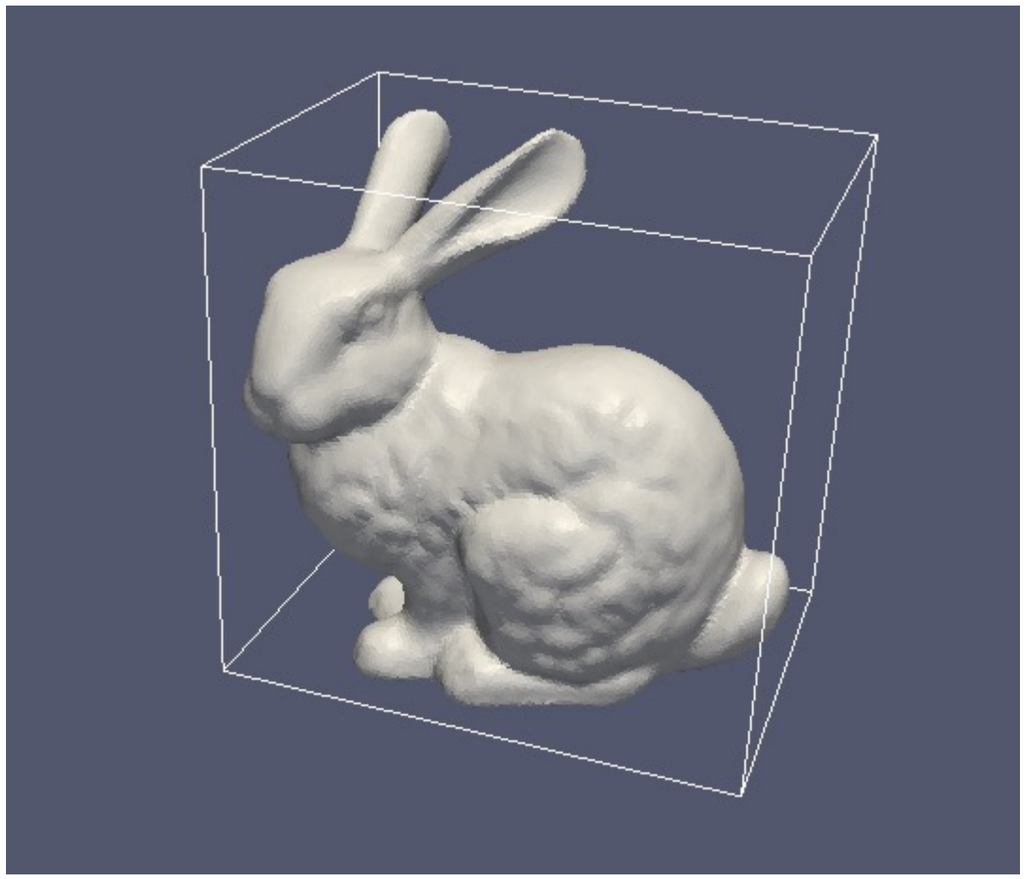
- More ‘straightforward’ than A/B, but lots of details!
- Vertices, normals, triangles, coordinate systems, etc!
- Also: Need to consider correctness **and** *efficiency*!





Stage 3C – OBJ Models

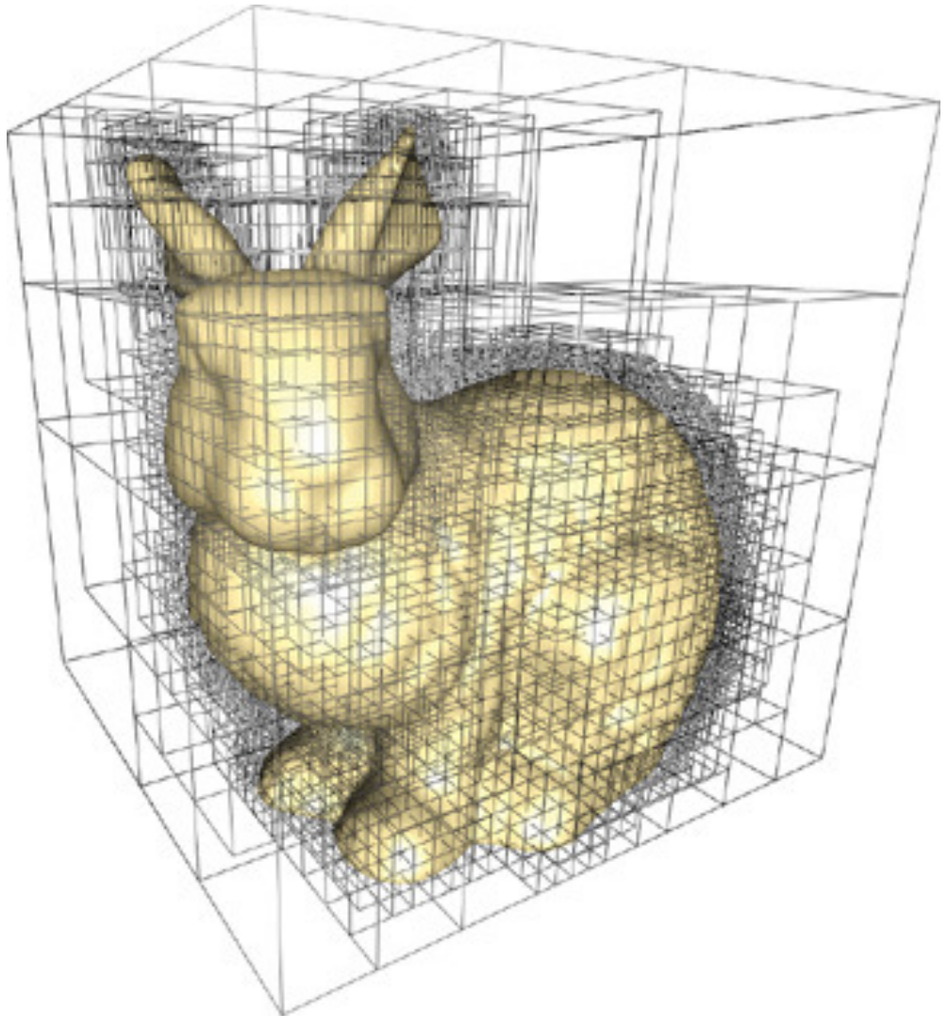
- **Efficiency:** 'Spatial partitioning' data structures
- Bounding volumes



https://jamesgregson.blogspot.com/2011/03/latex-test.html

Stage 3C – OBJ Models

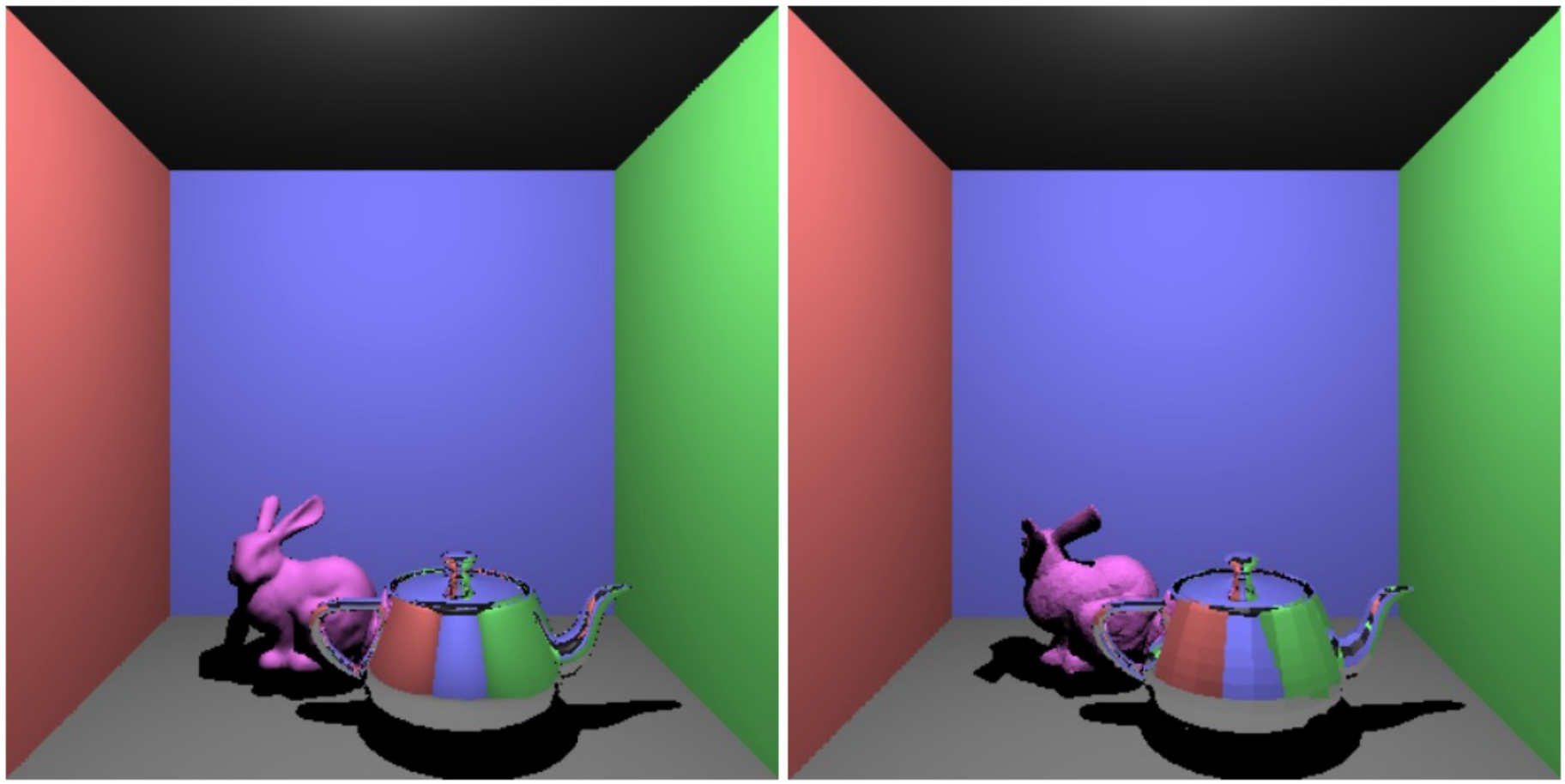
- Even better: Hierachical Space Partitioning (e.g., Octrees)



https://developer.nvidia.com/gpugems/gpugems2/part-v-image-oriented-computing/chapter-37-octree-textures-gpu

Stage 3C – OBJ Models

- Other considerations
  - Coordinate system conversion (OBJ files are right-handed!)
  - Interpolating vertex normals (barycentric coordinates)

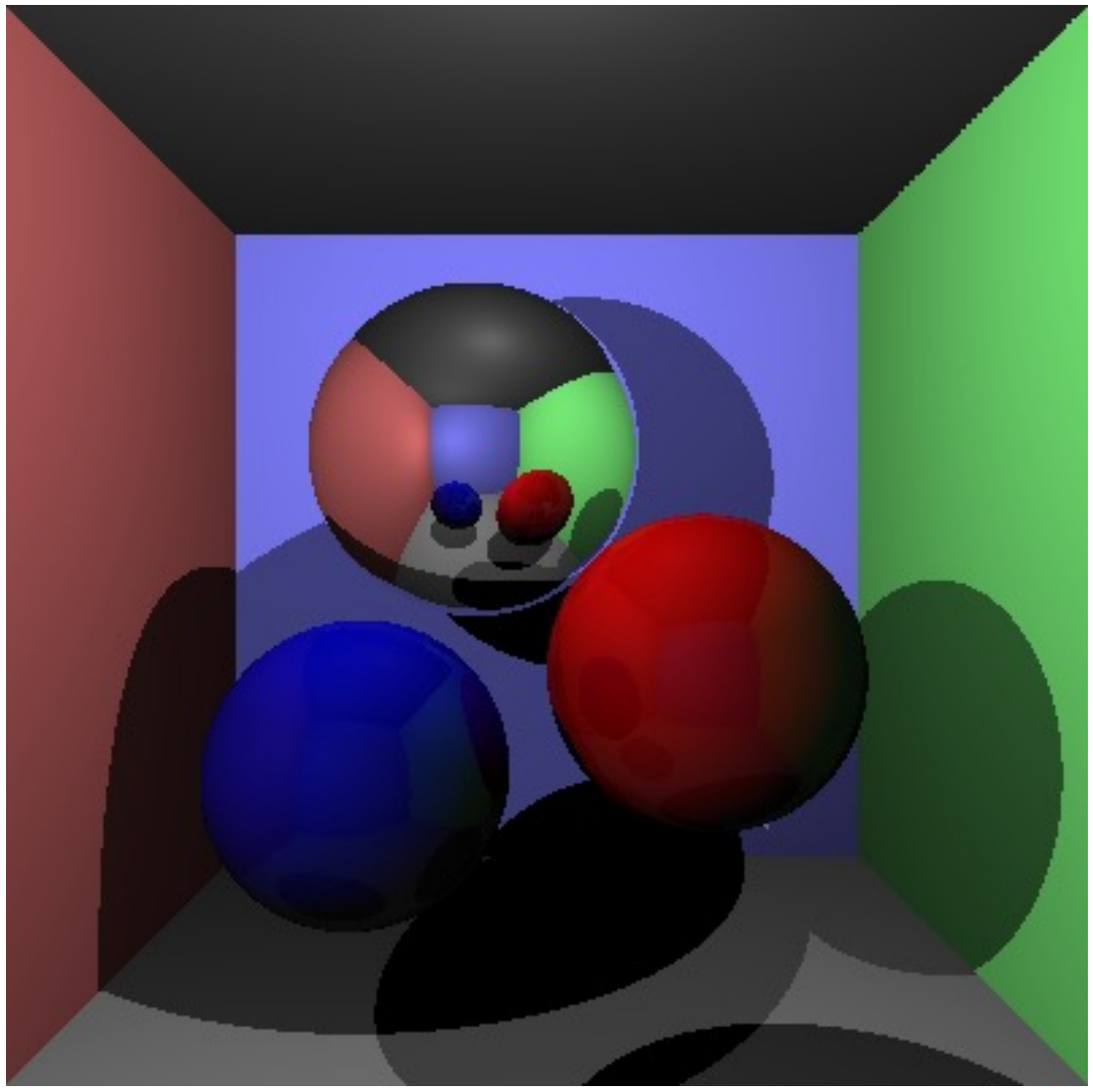


Stage 3D – Glossy materials

- ‘Glossy’ materials
- What do we mean by glossy?
  - Some reflection
  - Some diffuse reflection
  - Specularity
  - (Perhaps) Light ‘scattering’, depending on how ‘rough’
- Combining such components in a **realistic** way is important

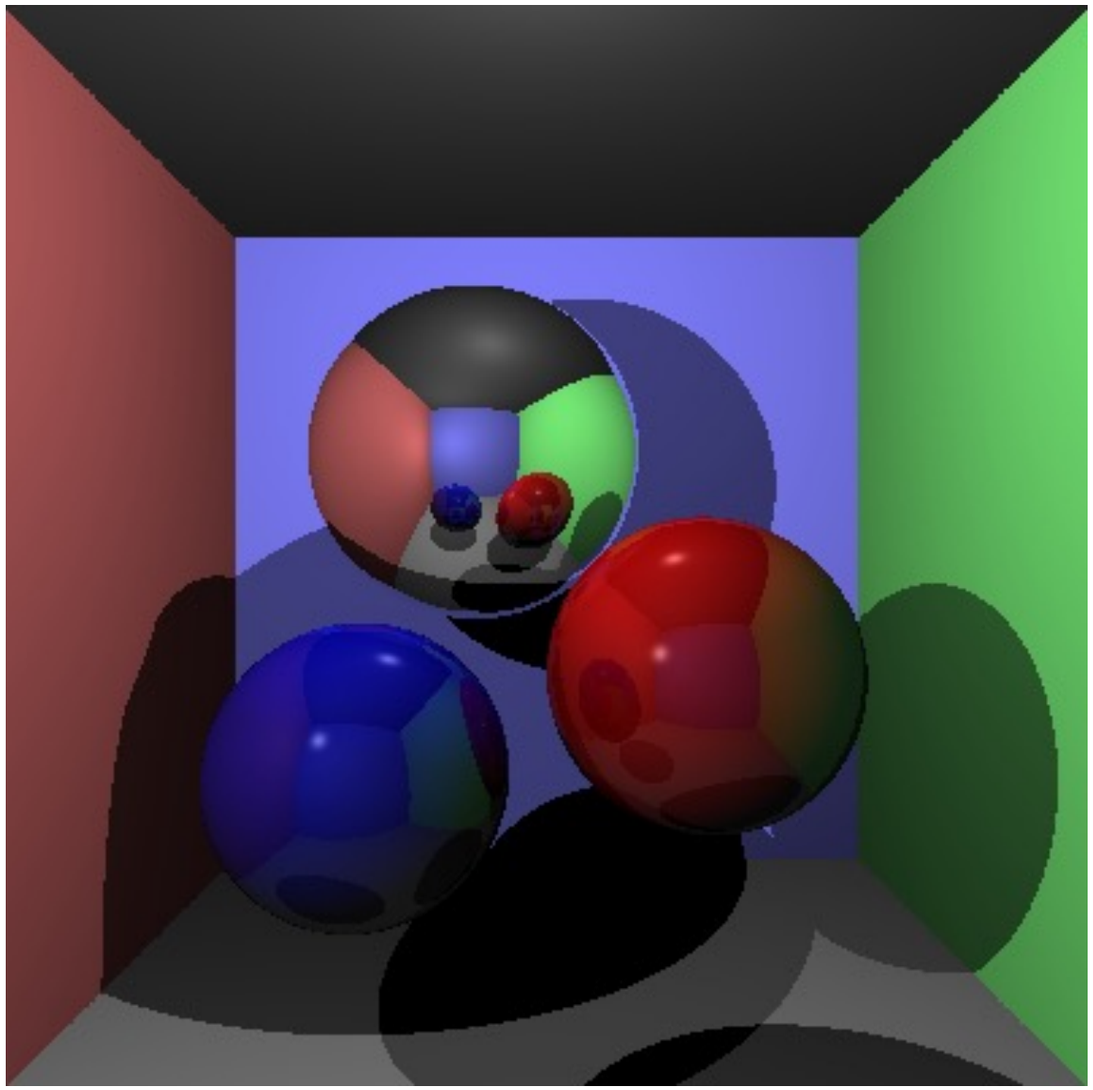
Stage 3D – Glossy materials

- Good start…
- Mix diffuse + reflection



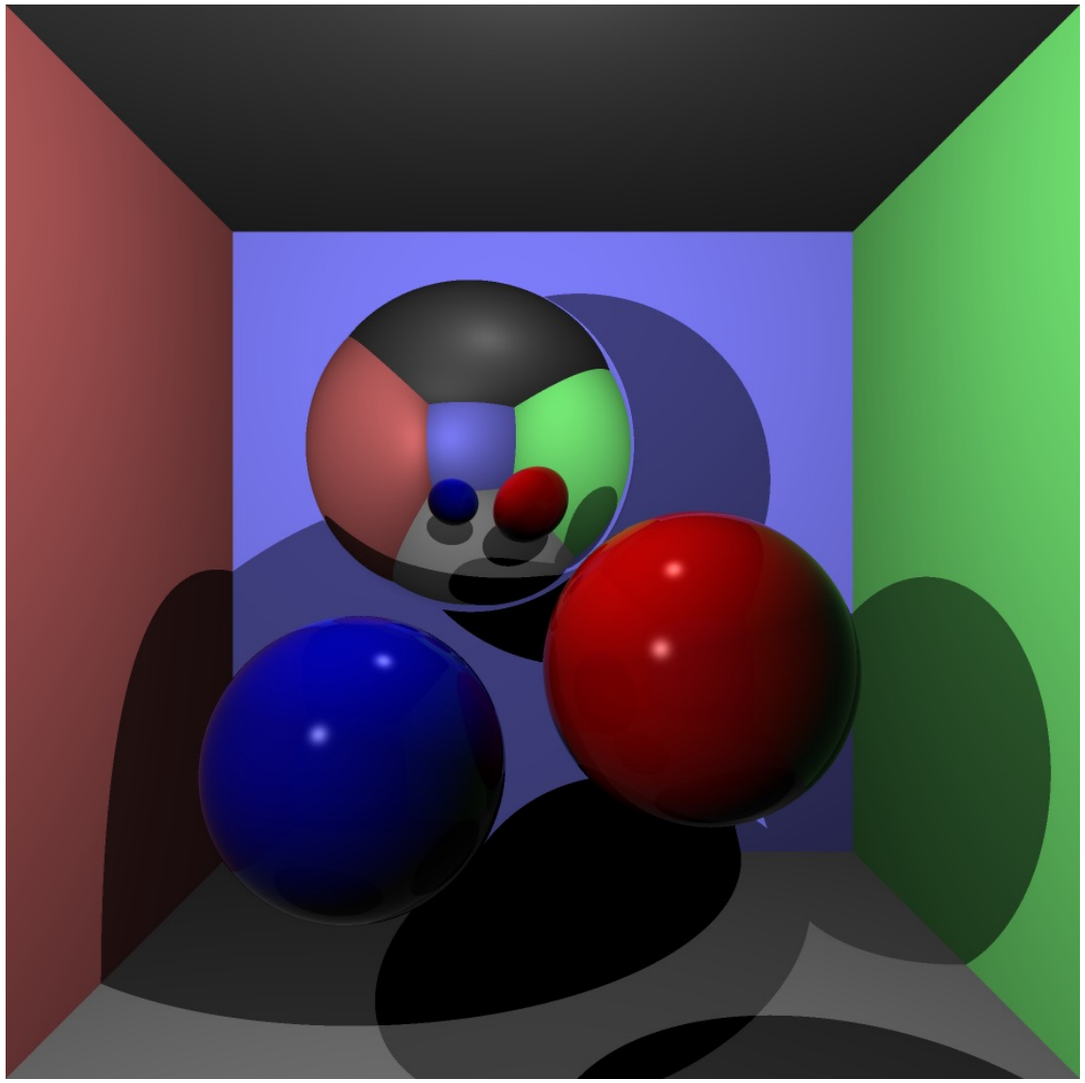
Stage 3D – Glossy materials

- Getting better…
- Mix diffuse + reflection
- Specular highlights



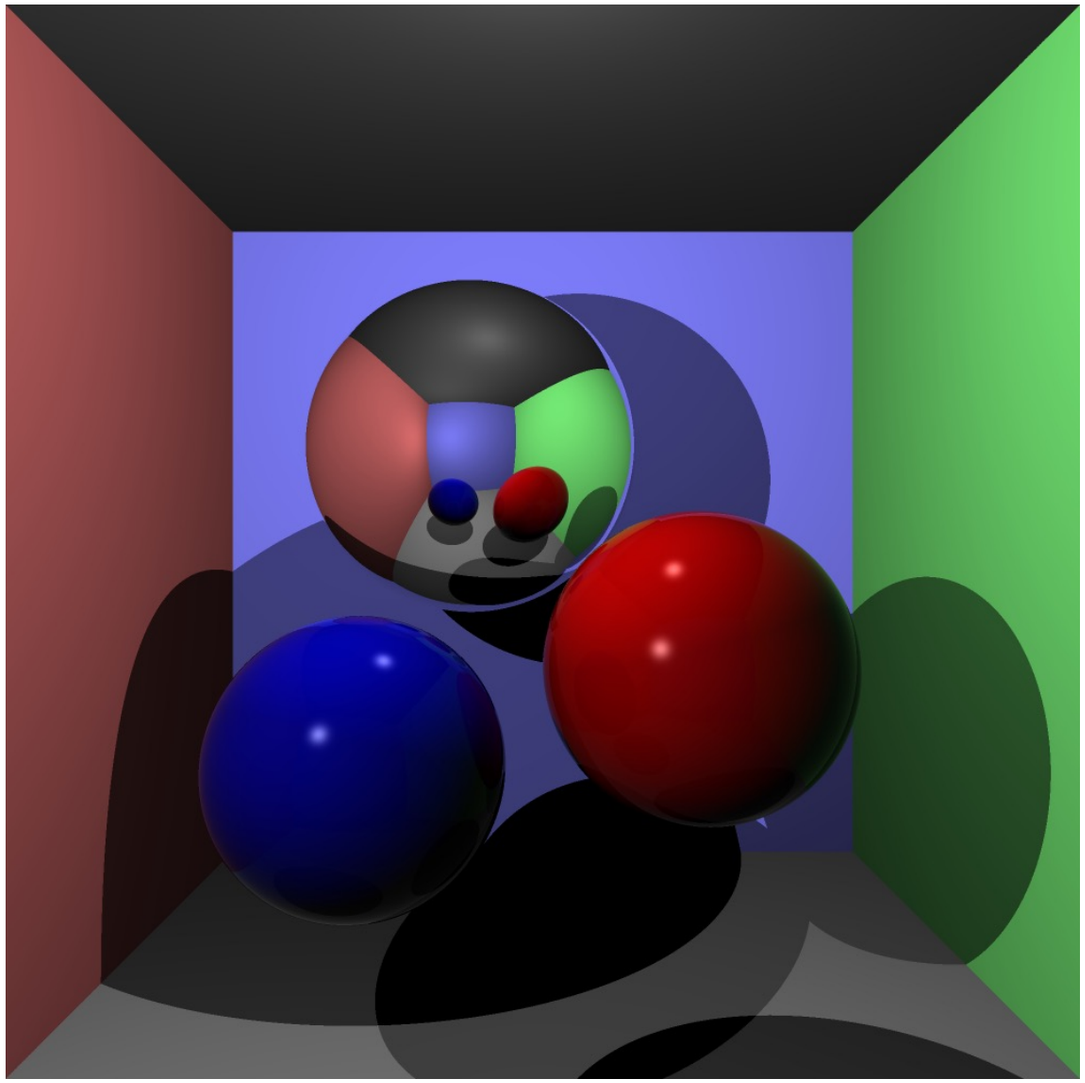
Stage 3D – Glossy materials

- Better still!
- Mix diffuse + reflection
- Specular highlights
- Fresnel effect



Stage 3D – Glossy materials

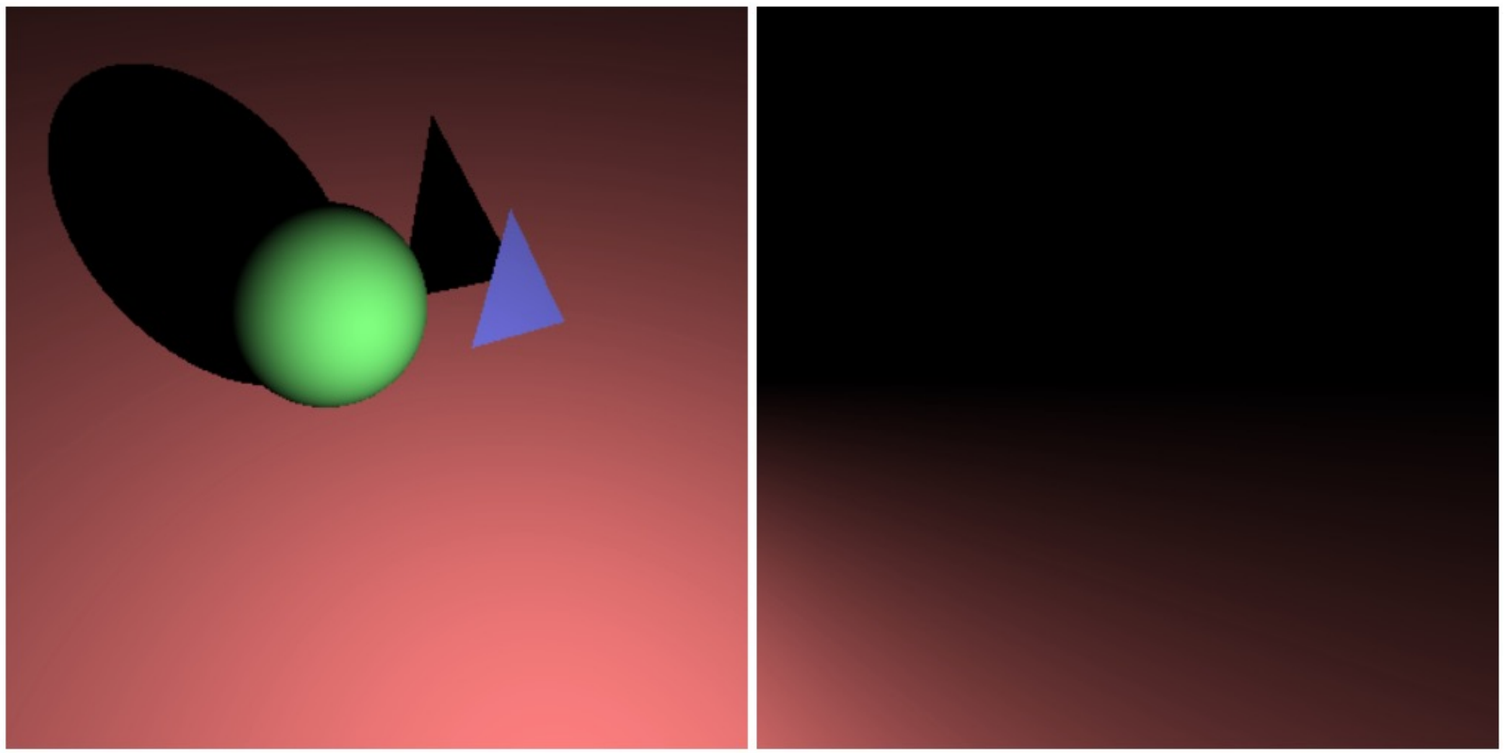
- Better still!
- Mix diffuse + reflection
- Specular highlights
- Fresnel effect



- Other methods valid too…

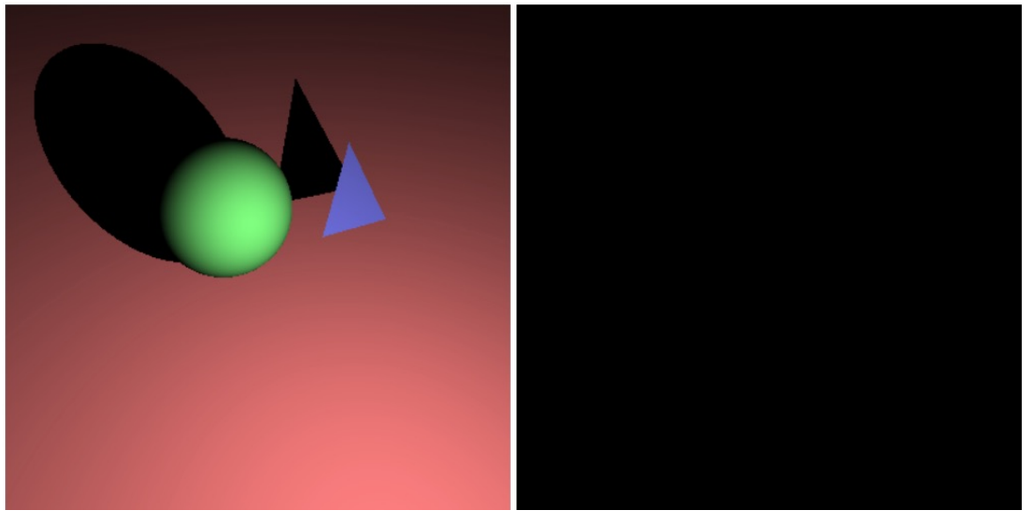
Stage 3E – Custom camera

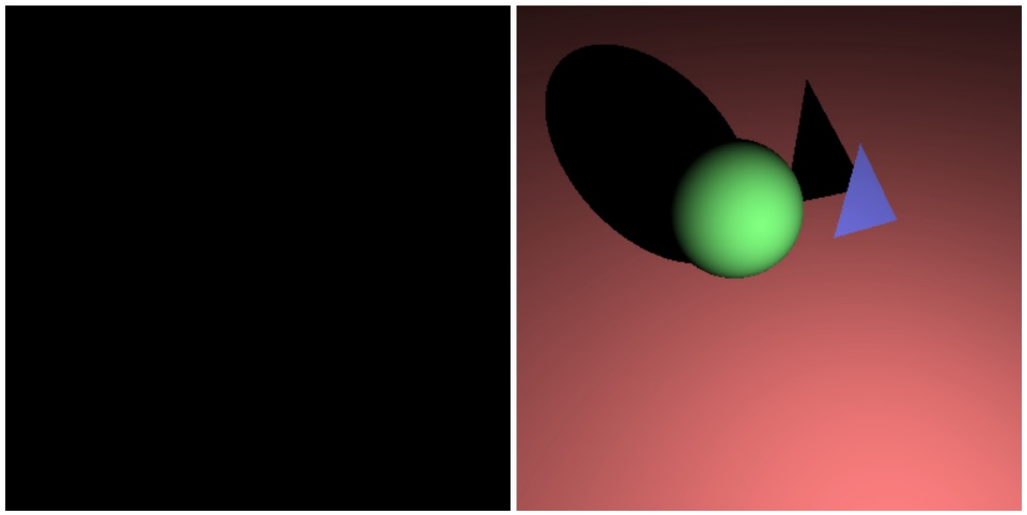
- Custom camera position/orientation
- Using **angle** and **axis** to represent rotation
- Common issue: Interpeting **axis** as camera local z ‘direction’



Stage 3E – Custom camera

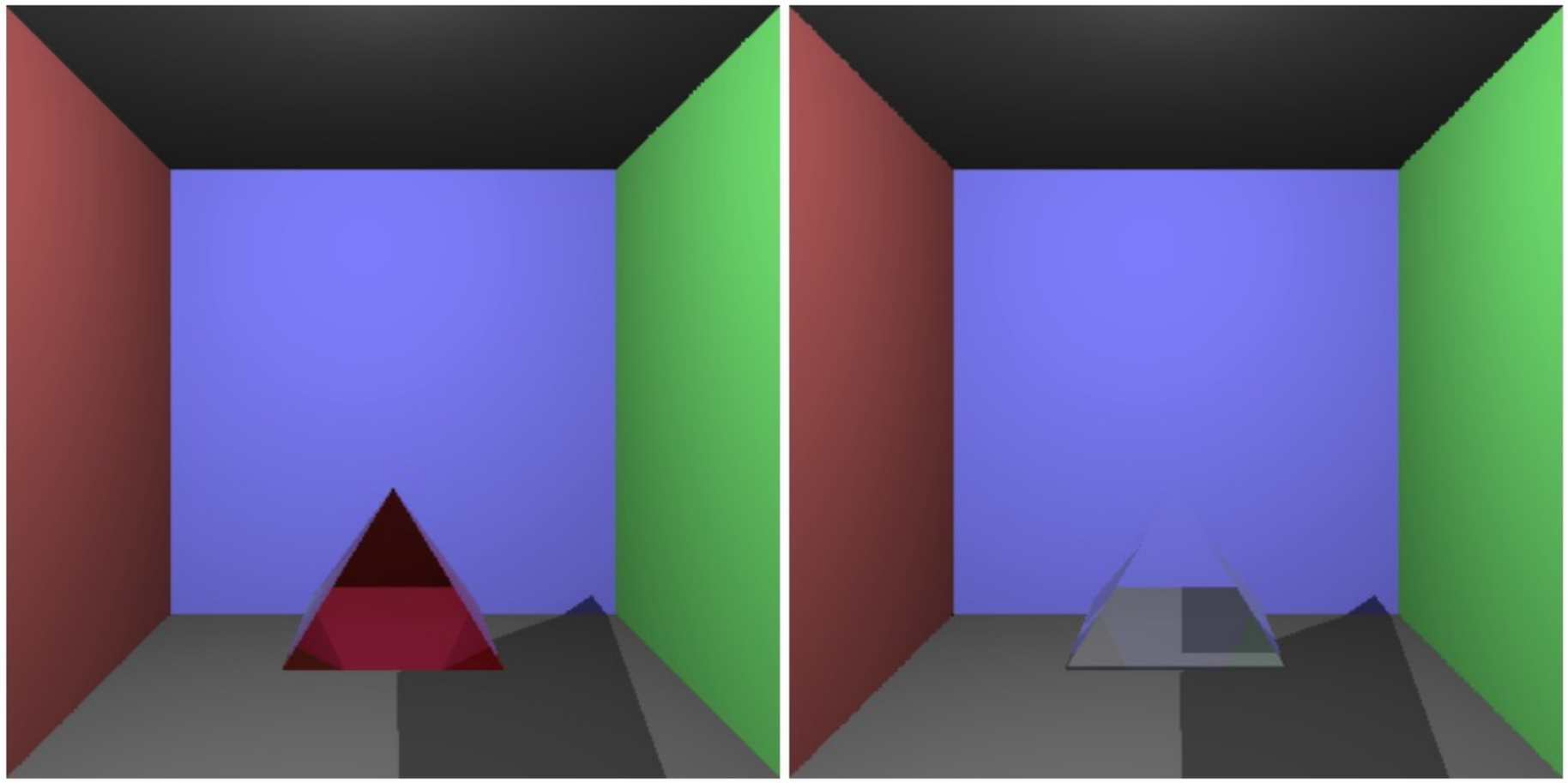
- Another common issue:
- Wrong angle direction
- Left-handed coordinates!





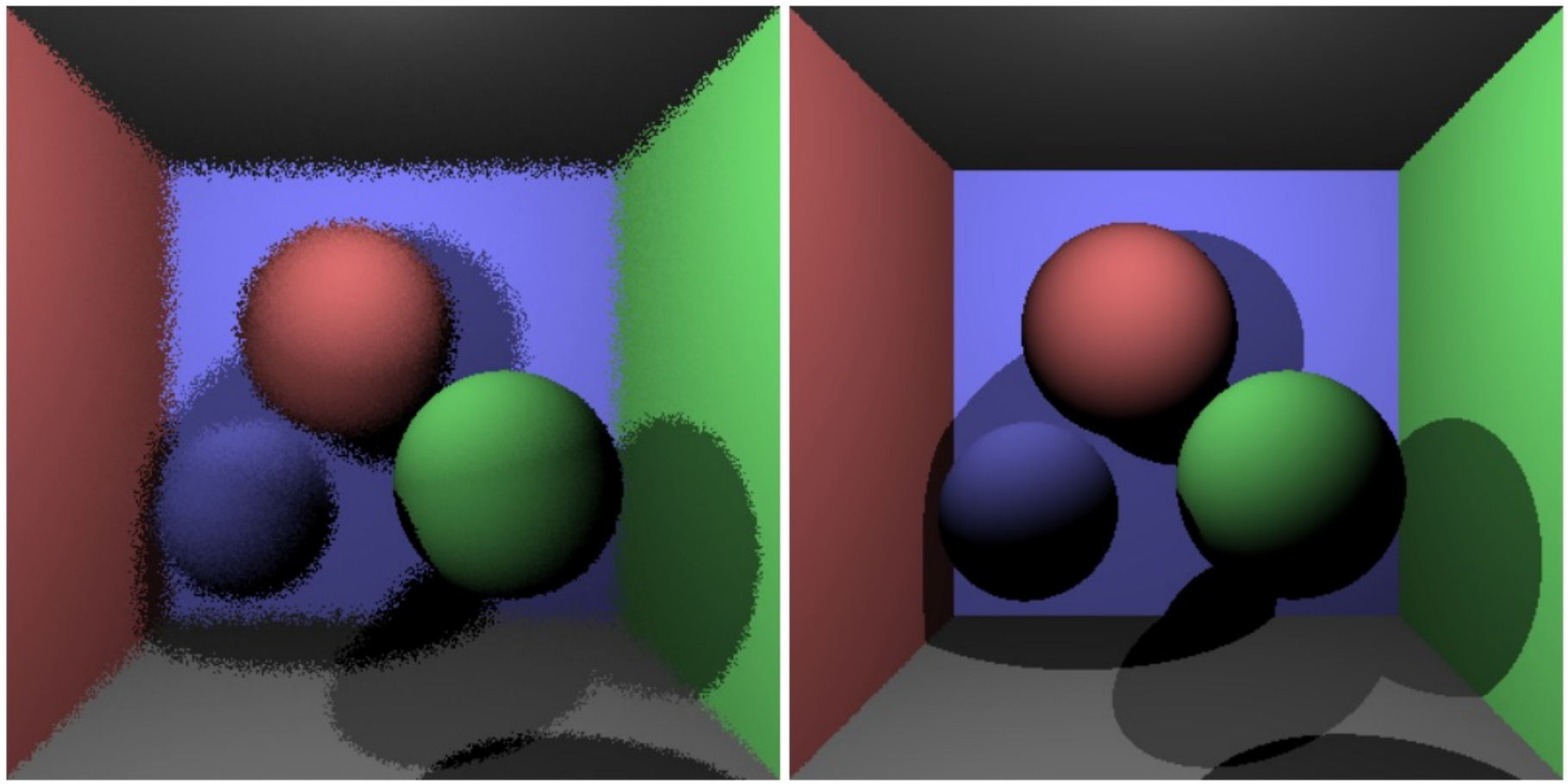
Stage 3F – Beer's law

- Generally well done, but often issues with the pyramid test



Stage 3G – Depth of Field

- Again, generally well done – sometimes nil output though!:

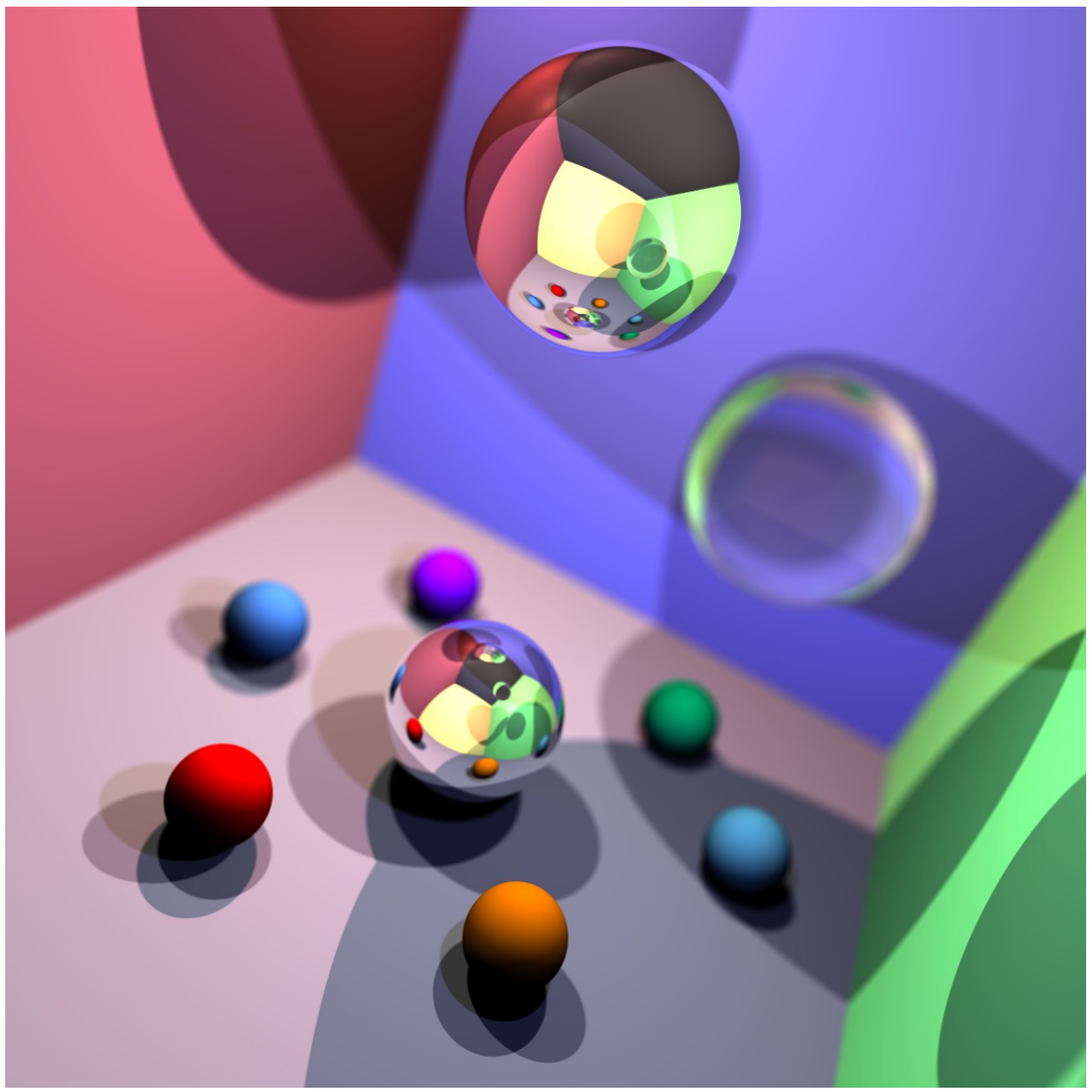


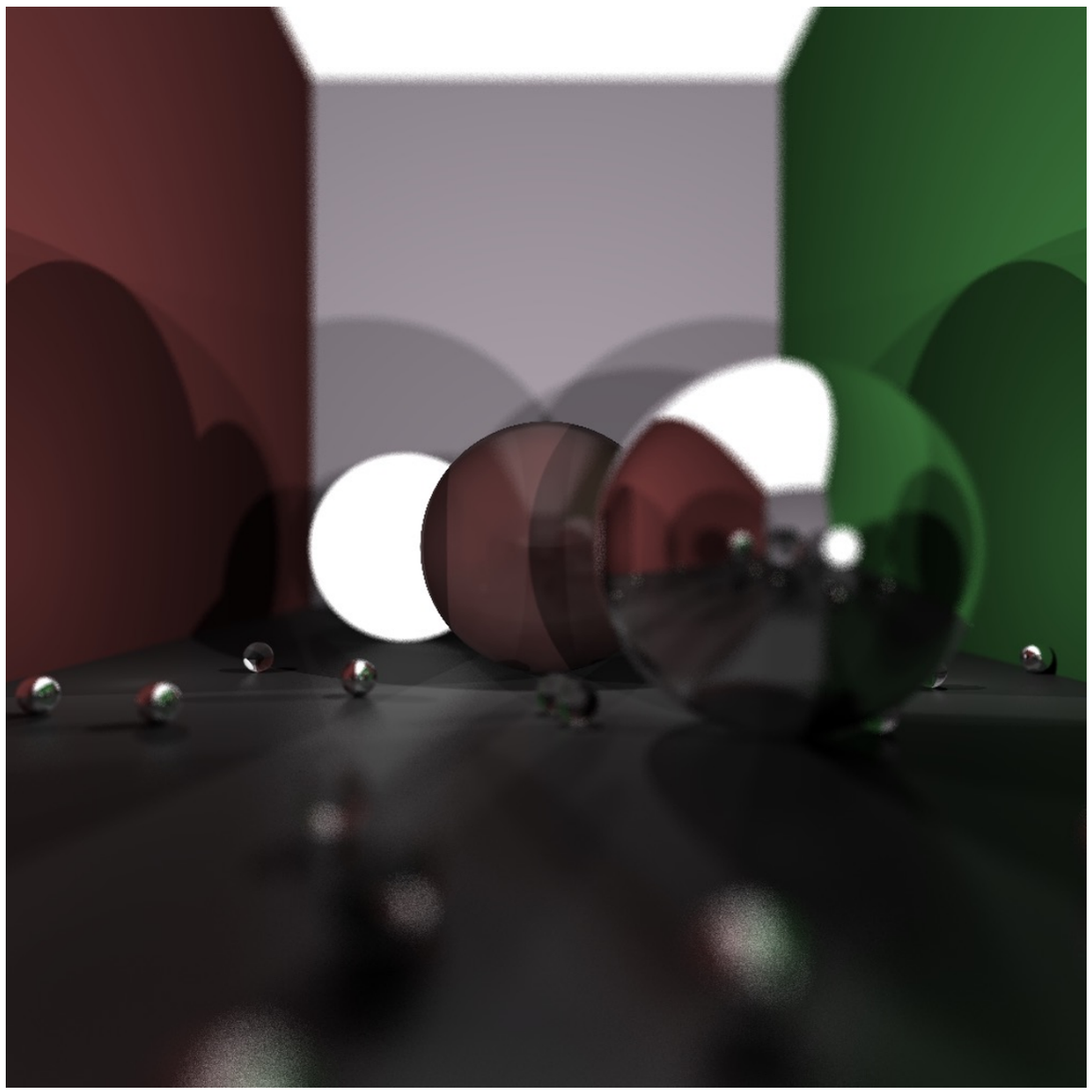
Final Render

- Criteria: **Coverage, Quality, Creativity**
- Interpret score as *additive* (even though subtractive rubric)
- High standard! Only the best images got 2.5-3 marks.

- Top scoring images:
  - Well implemented features (of course!)
  - Showed off all features, including ‘non-default’ resolution!
  - Quality renders – e.g., high AA/resolution, minimal noise
  - Very unique scene layout, *or* some artistic flair

Final Render





Final Render

Key Takeaways

- Read specifications closely – nitpick at everything!
- Graphics programming often requires research.
- Develop reliable debugging strategies.
- Always watch out for ‘edge cases’!
- Test, test, test

What's next…

- **Project 2...**
- Very different goals, but shared underlying theory
- Good shader programming requires a strong theoretical and mathematical understanding!
- Your 'graphics toolbox'…
  - Vector maths
  - Light theory
  - Ray-object intersections (very useful in game dev!)
  - Spatial data structures
  - And more!

What's next…

- Project 2…

- **FAQ: What can I use from the asset store?**
  - A: In short, purely *artistic* assets – **not** code or game logic.
  - Models, textures, images, sounds, music, etc…

- **FAQ: Can I use the Unity shader graph?**
  - A: Not for *assessed* shaders – they need to be your own **HLSL code**.
  - You are welcome to use workshop shaders as a starting point.

Questions

- Personal project questions: email your tutor!
- General questions – ask away!