

Bonus Lecture 2

Graphics Programming Intro

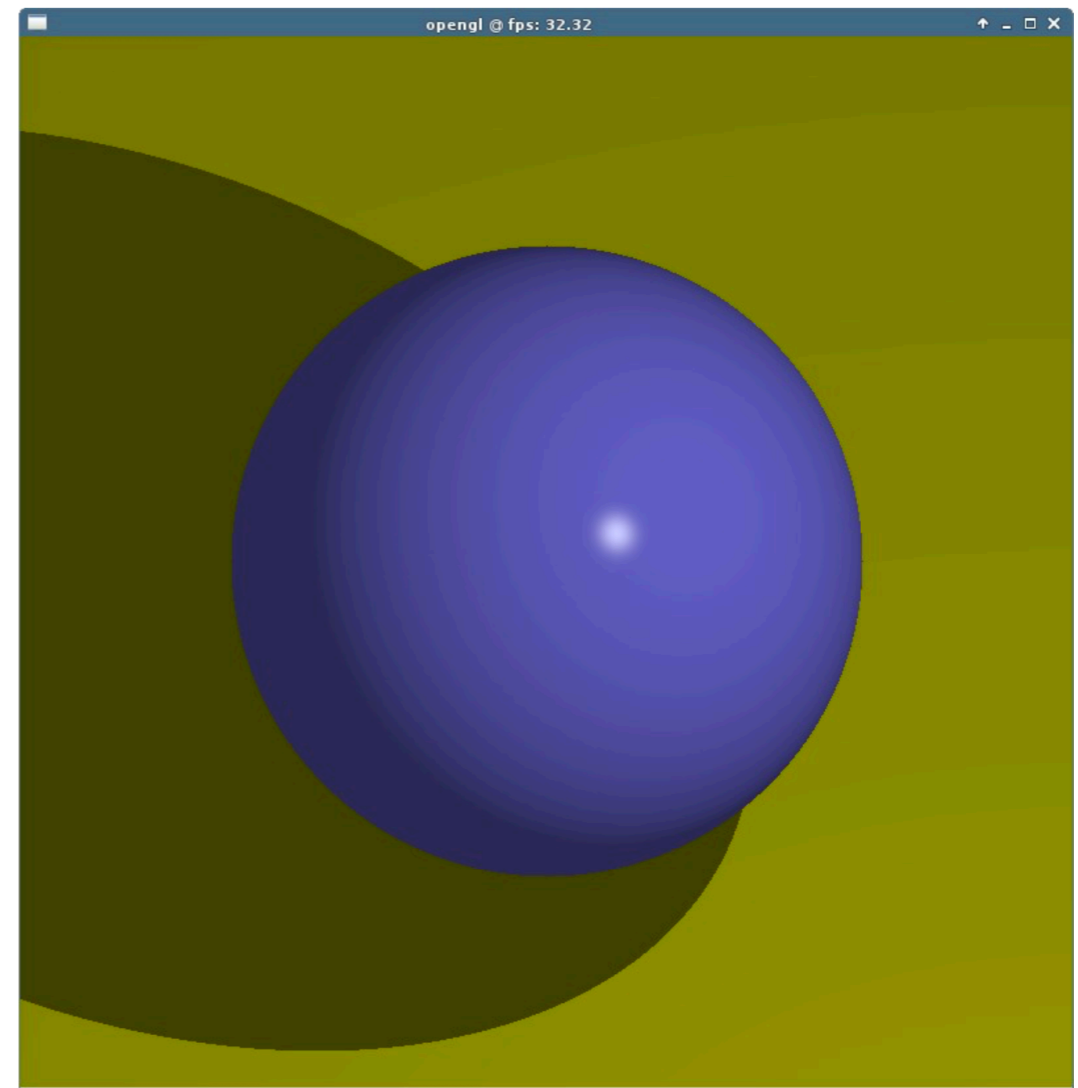
<gerdela@scss.tcd.ie>

3d Computer Graphics

- 2 major paradigms (others exist)
 - **ray tracing** / path tracing
 - mathematical model + rays of light
 - **high quality:** movies, animations, realistic stills
 - **rasterised**
 - flatten 3d triangles into pixels
 - **fast:** games, real-time or interactive simulations

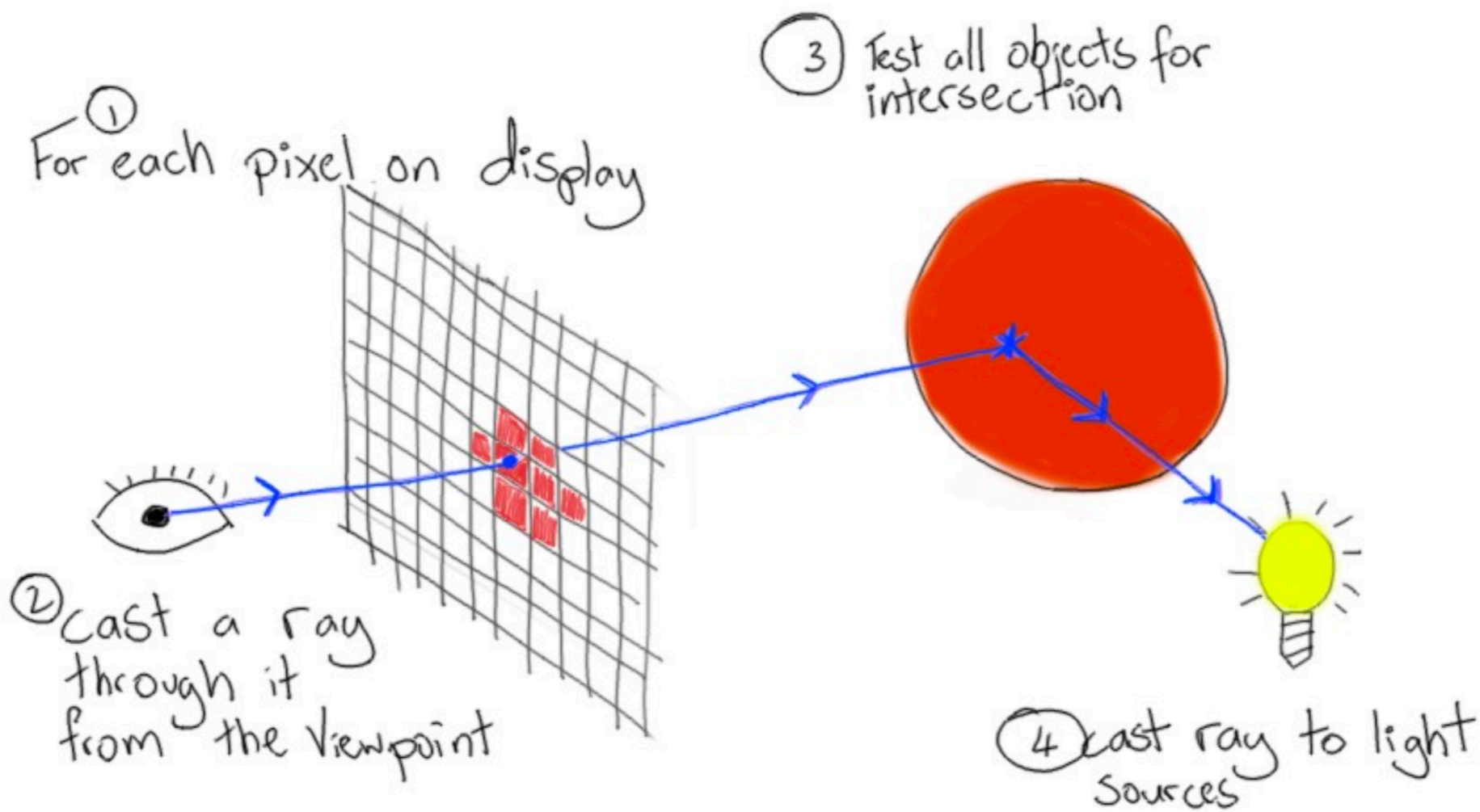
Ray Tracing

- high-quality renders
- basic on physics of light:
optics
- typically uses CPU
 - easy to parallelise - 1 thread per pixel
- curves, spheres, transparency
- may take a while to calculate



Ray Tracing

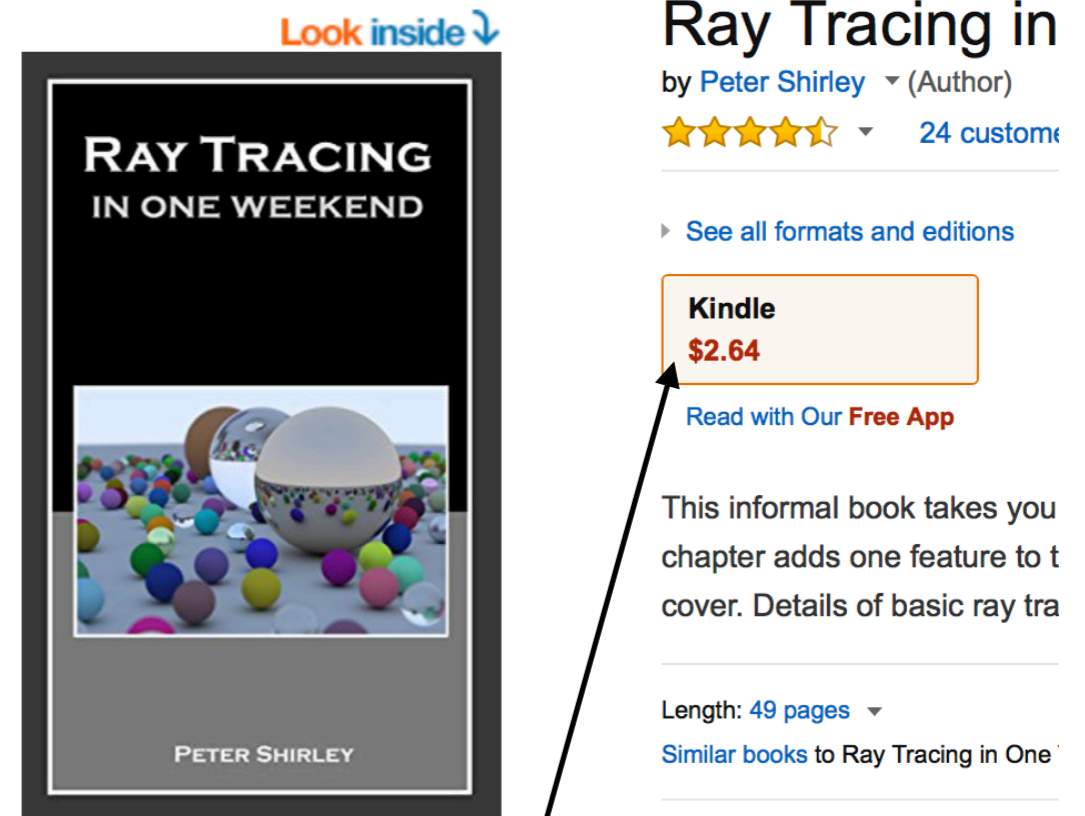
- **For each pixel** on the screen
 - **cast a mathematical ray** forward in direction of view
 - if it **intersects** with an object in the scene
 - **colour the pixel** with that colour...
- Can keep bouncing/bending rays **until they hit a light**
 - accurate shading, refraction, reflection, shadows
- Note that real light rays bounce from light into the eye
 - **reversed** from eye to light = **fewer calculations**



Ray Traced Rendering

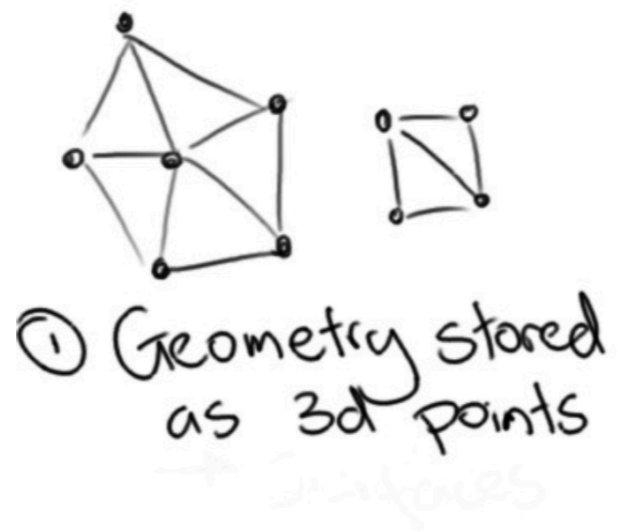
Weekend Project?

- You have all the skills to do this now
 - linear algebra (e.g. ray + sphere intersection)
 - programming
 - write to an image file
- Peter Shirley's mini-books are a great way to start
 - good coding practice
 - fun output
 - no libraries



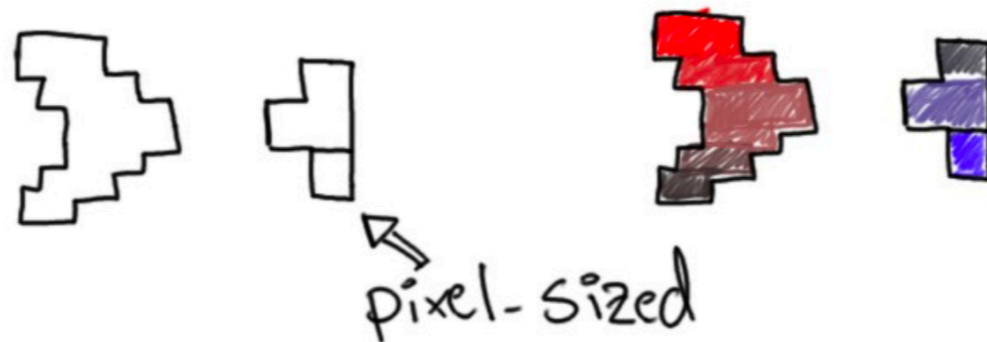
Rasterised Rendering

- typically uses specialised Graphics Processing Unit (GPU)
 - highly parallel
 - fast floating point computation
- everything is triangles and floating point numbers
 - CPU is better at integers, and branching code



③ Geometry flattened to 2D surfaces

④ Colour, etc.



Rasterised Rendering

How to talk to the GPU

- make some buffers of 3d points

- usually arrays of floats

```
float vertex_buffer[] = {x, y, z, x, y, z, x, y, z};
```

- write a shader program (define the **style** of drawing)

- **vertex shader** - how to **position** each points

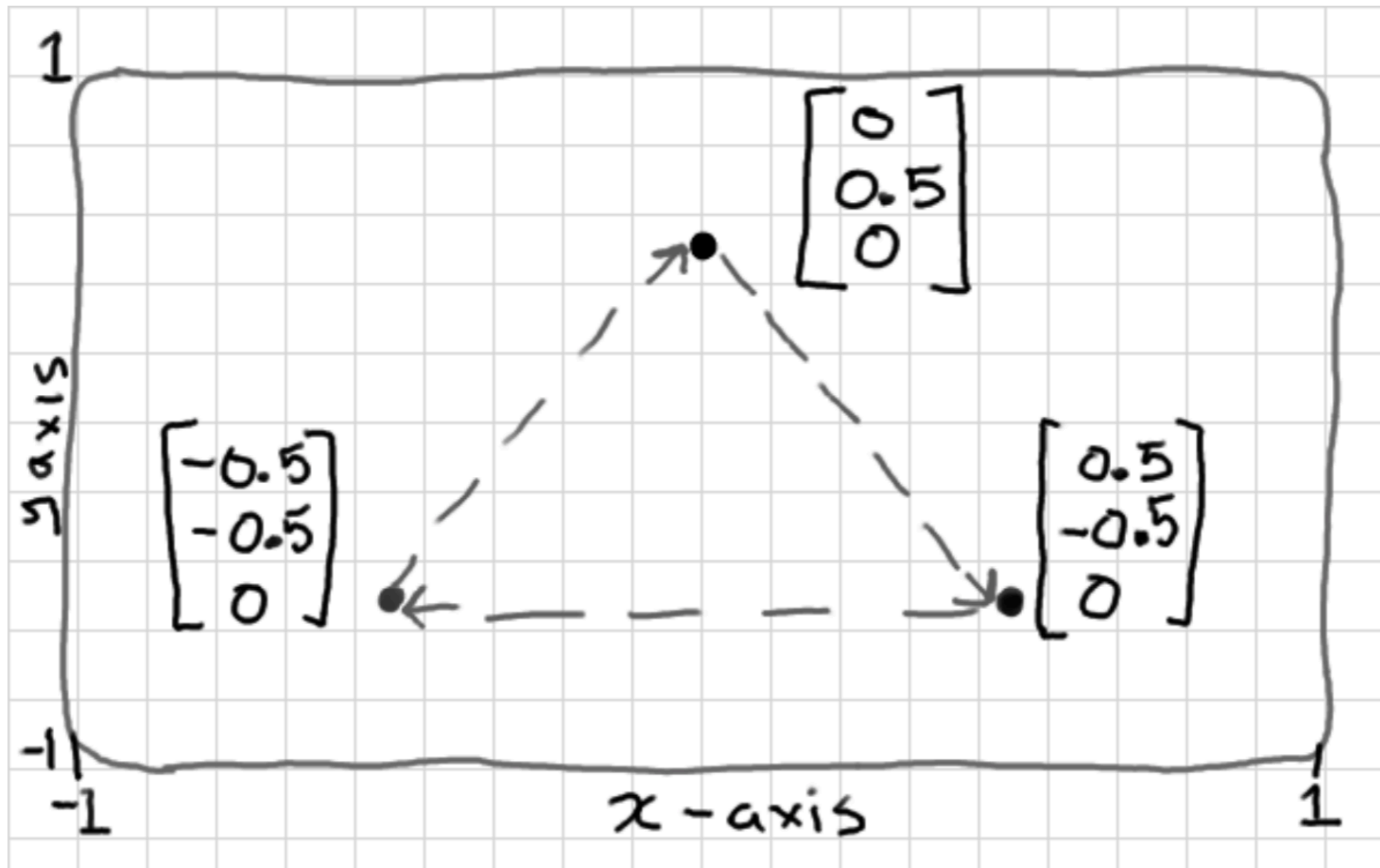
- **fragment shader** - how to **colour** each pixels

- these look almost exactly like C programs

- copy these to the GPU using the API of your choice...

- draw(my buffer, with my shader program);

Vertex Buffer



It always helps to draw your problem on paper first. Here I want to define a triangle, with the points given in clock-wise order, that fits into the screen area of -1:1 on x and y axes.


Vertex Shader String

```
const char* vertex_shader =
"#version 400\n"
"in vec3 vp;"
"void main() {"
"  gl_Position = vec4(vp, 1.0);"
"}";
```

- Runs one shader for each of your 3d points in the buffer
- **Input** (3d point position) is: `in vec3 vp;`
- **Output** is built-in 4d point `gl_Position = ...`
- Add/subtract/modify values (or a time variable) to `vp` to make it animate
 - `vec3 offset = vec3(0.0, 1.0, 0.0);`
 - `gl_Position = vec4(vp + offset, 1.0);`
- Usually use **linear algebra** to create camera viewpoint and angle
 - Change the 1.0 at the end to get the GPU to simulate a perspective camera; **`xyz = xyz / w`**

Fragment Shader String

```
const char* fragment_shader =  
"#version 400\n"  
"out vec4 frag_colour;"  
"void main() {"  
"  frag_colour = vec4(0.5, 0.0, 0.5, 1.0);"  
"}";
```



- Executes *after* the triangle has been flattened onto 2d display (rasterised)
- Runs one shader for each pixel-sized area of your triangle on screen
- **No inputs** here - but we could add some outputs from vertex shader as inputs
- **Output** is an **RGBA** (red,blue,green,alpha) colour which I called **frag_colour**
- Modify the R G B A values here to do interesting stuff
 - Simulate lighting
 - Crazy patterns

GPU APIs

- OpenGL - available on most platforms incl. web
- Direct3D - Microsoft
- Metal - Apple
- Vulkan - new. from the OpenGL people (Khronos Group)
- Use as programming libraries
- But are more like hardware drivers really
- All are very time-consuming to learn (and ∴ poorly designed IMO)

Learning OpenGL

- Current version is OpenGL 4.5
 - **Beware** old stuff (pre ~3.2) is very different and up to **15 years out of date**
(including some lecture slides... ..)
- I have loads of stuff on
 - my website <http://antongerdelan.net/opengl/>
 - GitHub https://github.com/capnramses/antons_opengl_tutorials_book
 - ebook on Amazon/Itch
- Ask me any time
- There are some pretty good tutorial websites around now
 - <https://learnopengl.com> - Joey de Vries
 - open.gl - Alexander Overvoorde

OpenGL Overview

- OpenGL only draws triangles
- Download a couple of helper libraries (GLFW and GLEW) - open a Window, keyboard input, etc
- Update video drivers to install latest OpenGL libs
- Write a C program
 - Link against OpenGL and the helper libs
 - Include headers for helper libs
- Use functions from OpenGL - docs.gl is a great unofficial API doc
- Have fun drawing stuff and playing with shaders

Blooper Reel

HZ 60.01
GLDRAWS 120
VERTICES 65740

AS STEALTHILY AS A PANTHER
CRONGDOR CREPT ACROSS THE ROOFTOPS



Anton of Dublin @capnramses · 23 Feb 2014

switching lights from uniforms to UBOs. things got weird. #blooperreel





Anton of Dublin @capnramses · Mar 1

game's **bug** list is cleared!
from the changelog: "BUGFIX: **decapitated** heads can only bounce 32 times"



Anton of Dublin @capnramses · Feb 14

dungeon security precautions are at an all time low - had a **bug** where portcullises were not locked in the sewers



unexpected, but humorous porting bug surprise - giant decapitated heads, bouncing around!

49 1594 days ago



Anton of Dublin

@capnramses

well i sort of got it working with one light so i tried 3 at once. hmmm i like the green. bug or feature?



10:57 PM - 23 Feb 2014



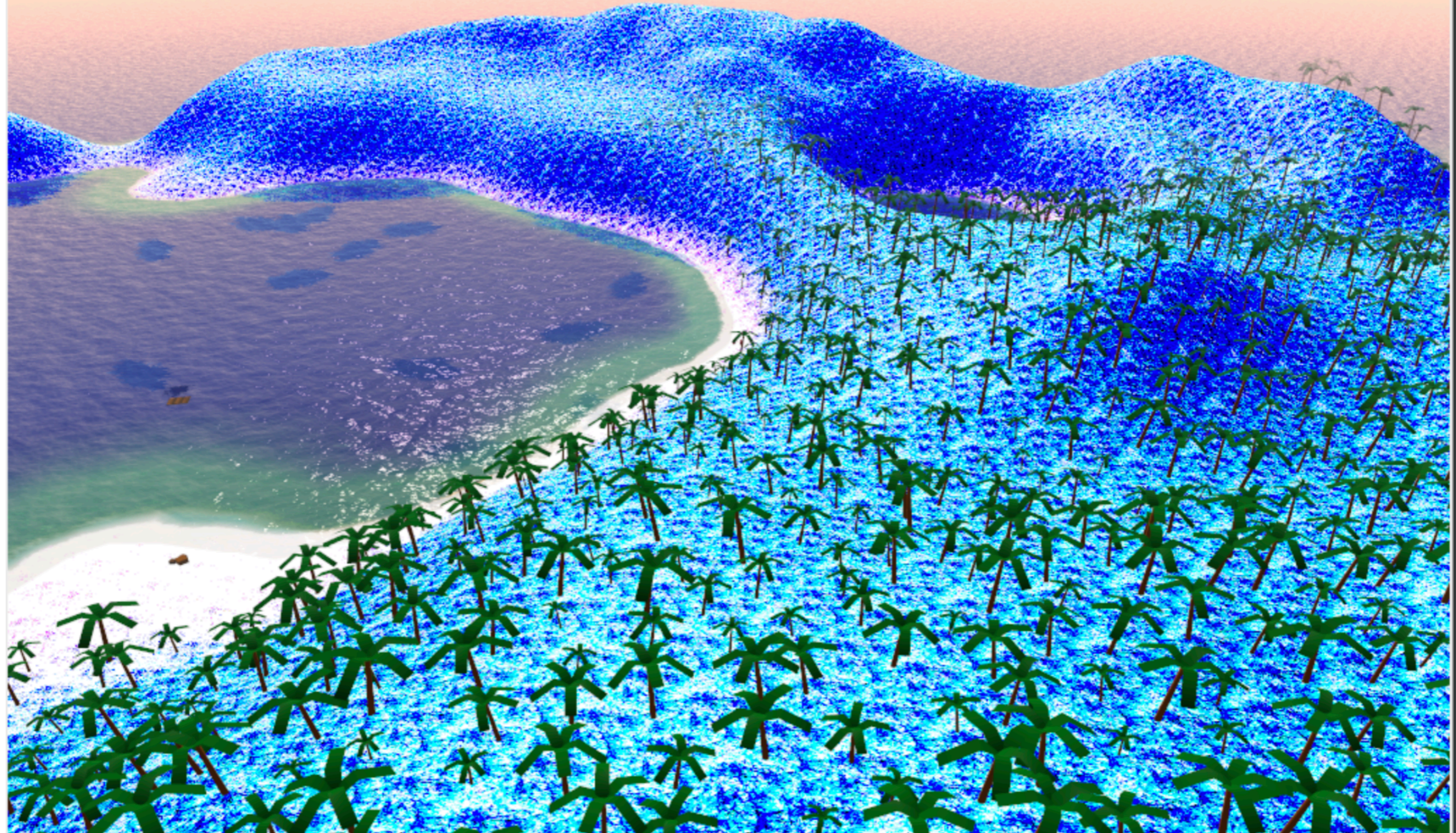
Anton of Dublin @capnramses · 30 Mar 2015

just fixed a **bug** where baddies would leap, lemming-like, over cliffs to their doom in their zeal to attack the hero



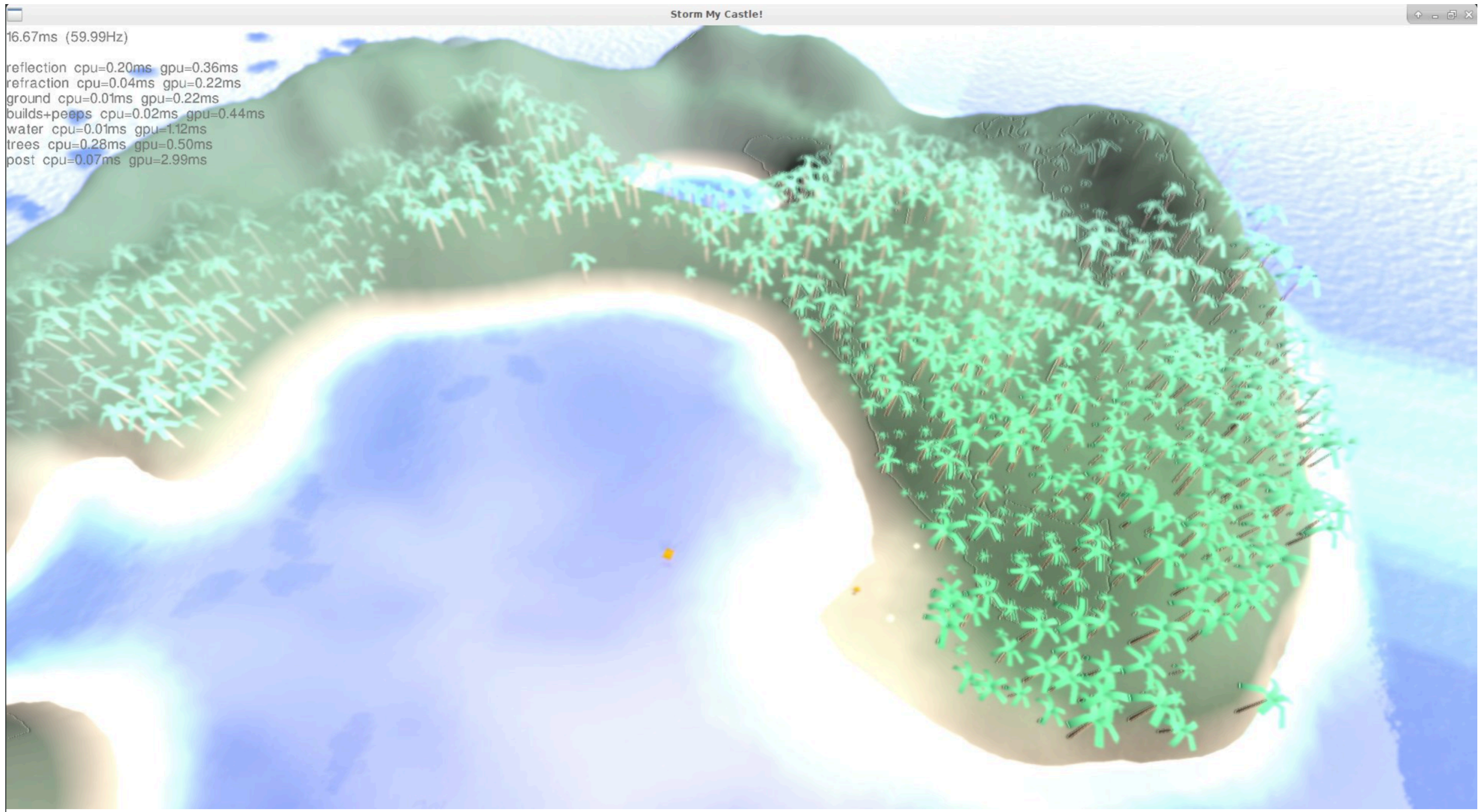
16.66ms (60.01Hz)

reflection cpu=0.15ms gpu=0.26ms
refraction cpu=0.06ms gpu=0.14ms
ground cpu=0.01ms gpu=0.14ms
water cpu=0.01ms gpu=0.72ms
trees cpu=0.29ms gpu=0.37ms



Anton of Dublin @capnramses · Aug 29
woops. #blooperreel

Retweet icon, Reply icon, Heart icon (3), Video icon, More options icon



Anton of Dublin @capnramses · Sep 13

"i feel sick but i'll try to fix that bloom shader". FAIL. #blooperreel

1 1 1

Rendering Engine

- Library or middleware or whole dev framework
- Sit on top of graphics APIs - often support many
- Do more than just draw triangles
 - load/save different types of files
 - render text
 - eg know how to animate rigged human characters
- I enjoy writing the rendering code more than using it...

Final Admin

- Mike Brady will have you for the **final lecture tomorrow**
 - probably will ask for course feedback
- We can cancel the tutorial + lab this week
- If you need me next year
 - personal email on website
 - ask Peter or Mike
- I'll upload a sample exam paper ASAP
- You've survived **over 50 hours of me talking about code.**
- Goal was to improve code skills/tools + theory because you'll need both
 - Hopefully that worked and it was interesting enough!