Computer Graphics

Dr Anton Gerdelan
gerdela@scss.tcd.ie
Room FR.35 (top floor O'Reilly)
http://antongerdelan.net

22 Sep 2014

Module Objectives

- Overview of major computer rendering approaches
- Implementation of several rendering algorithms
- Understanding of modern **graphics pipeline**
- **Shader** programming
- 3d transformations
- Larger scene management (**game project**)

Major Challenges

- Learning a rendering API (OpenGL)
- 3d transformation maths using linear algebra
- Understanding graphics hardware
- Understanding rendering algorithms
- Time management

- *“Where the #### is my triangle?”*
  (black screen of death)

Show of Hands

- BA CompSci / BAI Eng / Maths ?
- CS4052 / CS4D3A

Transfer Value

- 5 ECTS pts. (Euro. Credit Transfer & Accum. Sys.)

Timetable

- Michaelmas Term: Weeks 05 to 16
  (22 September 2014 - 12 December 2014)
- Study week is Week 11: 3 Nov - 7 Nov
- Bank Holiday Monday 27 Oct

- **Mondays 3-4pm** M21 (Museum Building) - Lecture
- **Tuesdays 11-12pm** LB107 (Lloyd Building) - Lecture
- **Tuesdays 12-1pm** ICT Lab 1 (Prefabs) - Lab

Week #	Mon-Tue Lecture Topic	Tue Lab
05	Introduction	OpenGL setup
06	Graphics Programming I	OpenGL setup
07	Maths & Transformations	Transformations & **Lab0 Due**
08	Graphics Programming II	Viewing & **Lab1 Due**
09	Virtual Cameras	Hierarchy &, **Lab 2 Due**
10	Illumination I	Walk-through & **Lab 3 Due**
11	*reading week*	-
12	Illumination 2	Game Project & **Lab 4 Due**
13	Game Project	Game Project
14	Animation Intro	Game Project
15	Curves	Game Project
16	Modelling	**Game Project Presentations**

- Lecture pace and lab topics may change a little

Organisation

- Presenting course for Assis. Prof. Rachel McDonnell
- Close as possible to existing format
- Course organised through blackboard on-line system
- Mistakes will be made! Let me know if something is missing!
- Lab time very limited - use **discussion forum** to get technical help
- Make sure that you can get onto the blackboard system now

Labs

- Demonstrators are:
  - Rowan Hughes (hughesrt@tcd.ie)
  - Tim Costigan (costigt@tcd.ie)
- OpenGL 3 or 4 - no deprecated stuff!
  - quickly reference the **Quick Reference Card** to check http://www.opengl.org/sdk/docs/
- C or C++ and **GLSL** (OpenGL Shader Language)

Labs

- Problems get increasingly complex, and each lab teaches you how to use several new features
- Viewing, animation, lighting and materials, textures etc., creating a game
- Mark awarded for final project

Labs

- Demo program to demonstrators who will grade it
- Feed-back from grades not available in-class
- Submit **report+pictures+source code** *via* Blackboard
- Demonstrator time to help very limited
- Work is individual - do not submit the same assignments

Assessment

- Project - 20% - developed in increments
- End of year exam - 80%

Communication

- Attend lectures and labs
- Interactive - bring pen and paper
- Check blackboard
- Check email
- Email me at gerdela@scss.tcd.ie

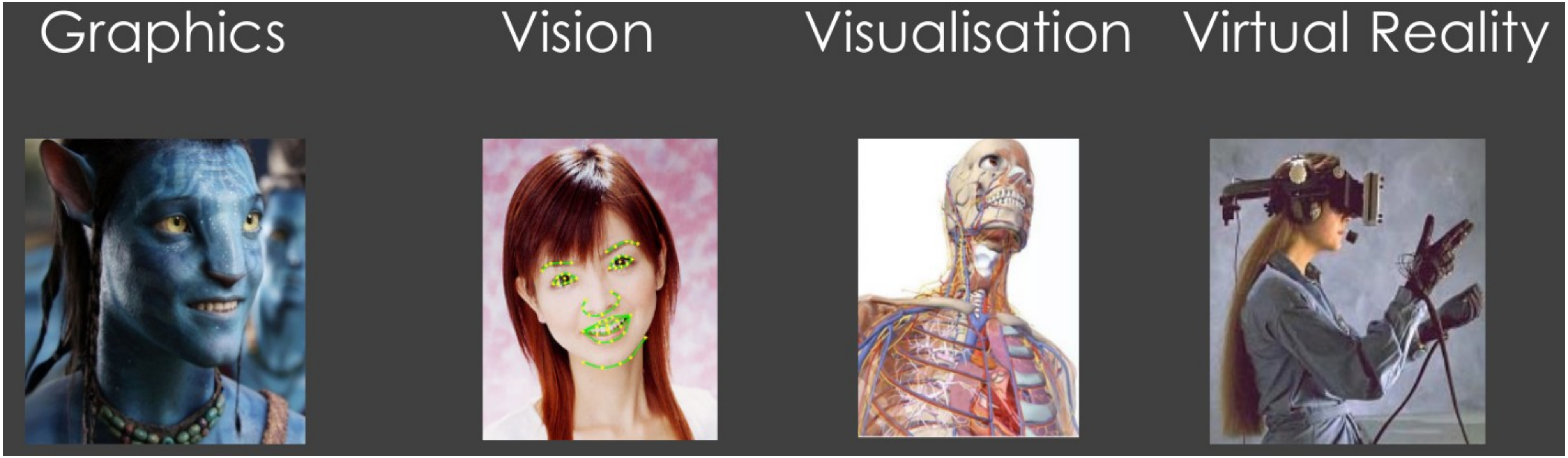
Recommended Texts

- Shirley, Marschner "*Fundamentals of Computer Graphics*" 3rd Ed.
- Hughes, van Dam, et al. "*Computer Graphics Principles and Practice*" 3rd Ed.
- Angel "*Interactive Computer Graphics: A top-down approach with shader-based OpenGL*" 6th Ed.
- Watt "*Three Dimensional Computer Graphics*"
- Shreiner "*OpenlGL Programming Guide*" 8th Ed.
- Howard Anton "*Elementary Linear Algebra*"

Questions?

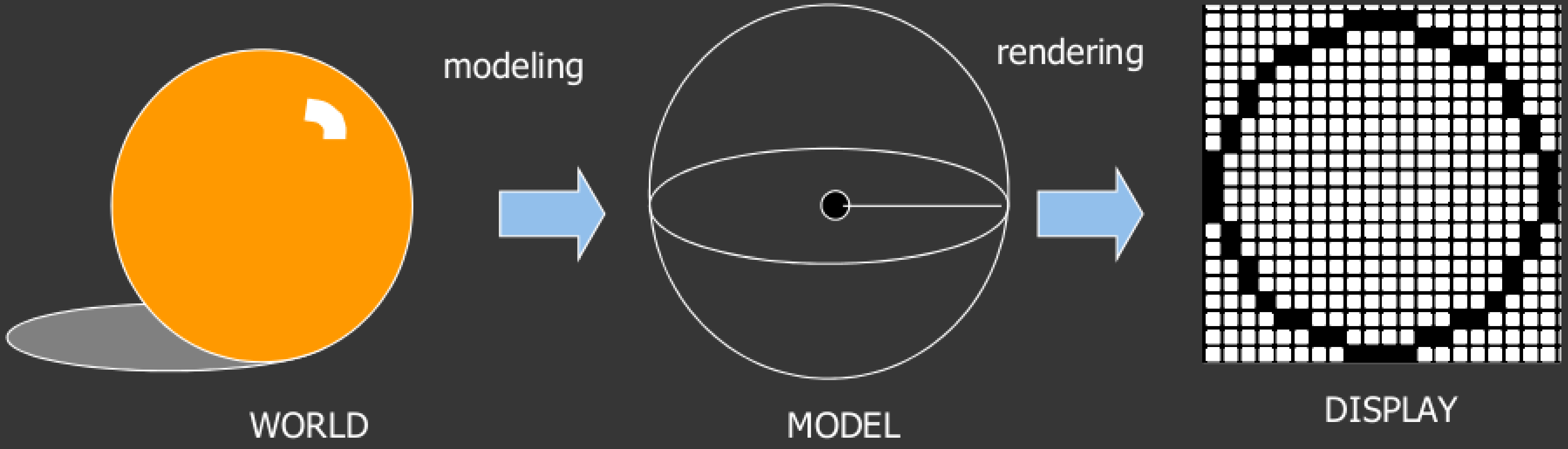
- this would be a suitable occasion for a short pause/discussion

Visual Computing



Using computers both to generate images synthetically and to integrate or alter visual and spatial information sampled from the real world

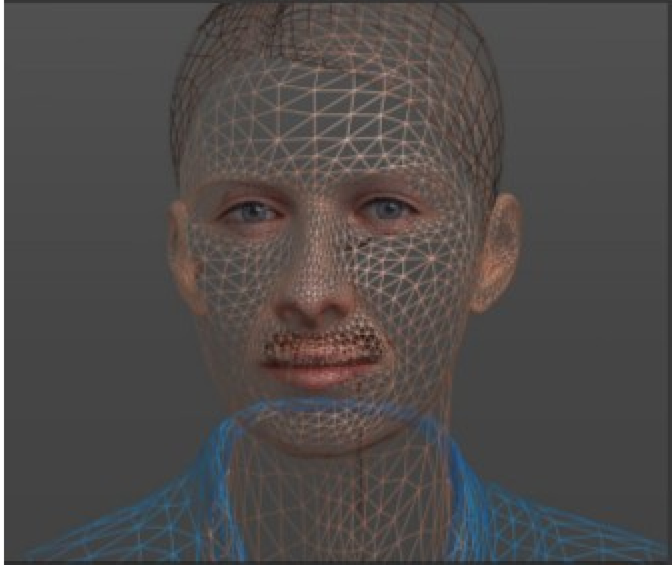
Computer Graphics



"Computer Graphics is concerned with producing images (or animations) using a computer."

Computer Graphics

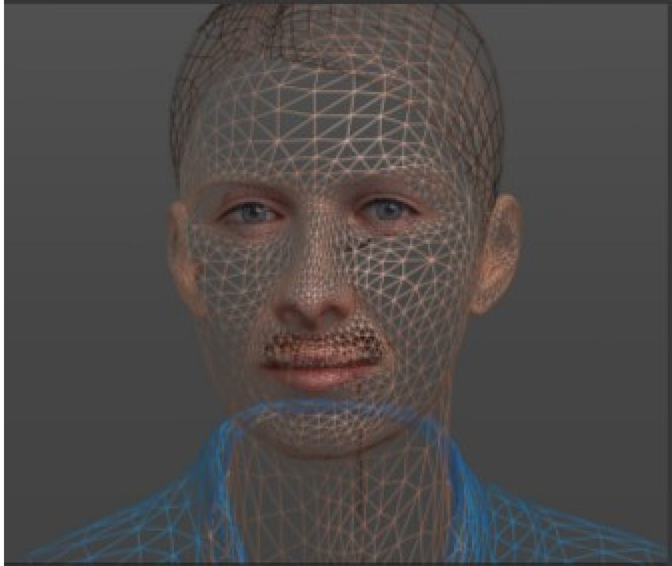
Modelling



Creating or capturing the representation of objects - motion often geometrical

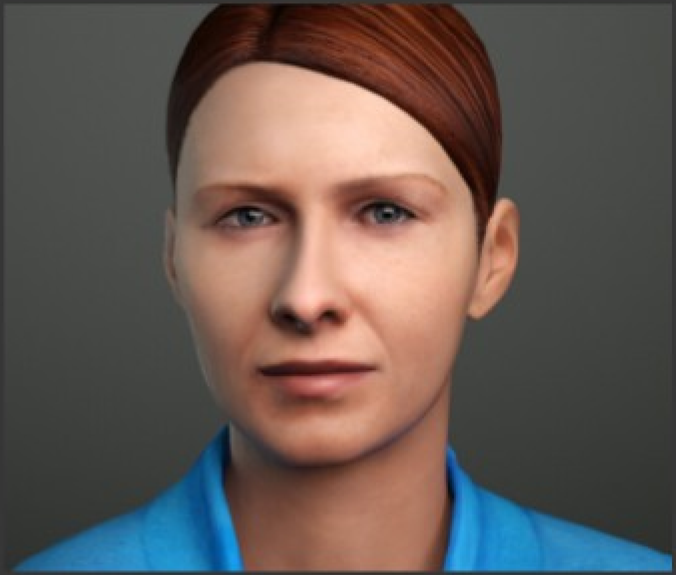
Computer Graphics

Modelling



Creating or capturing the representation of objects - motion often geometrical

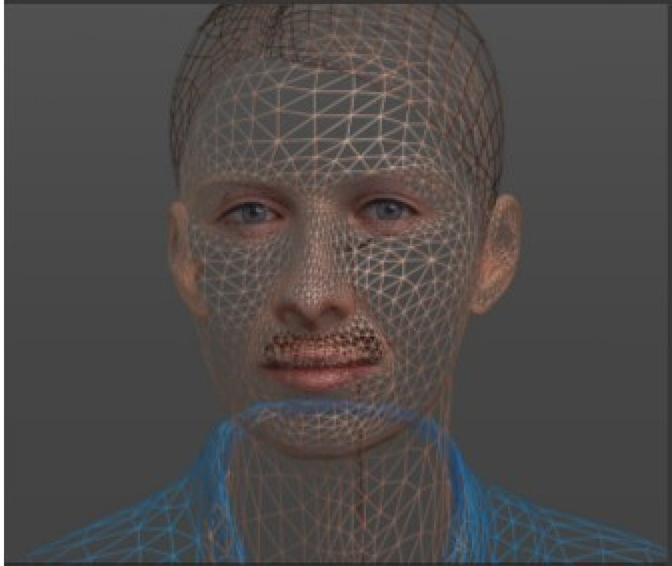
Rendering



Creating an image of these objects on a display device

Computer Graphics

Modelling



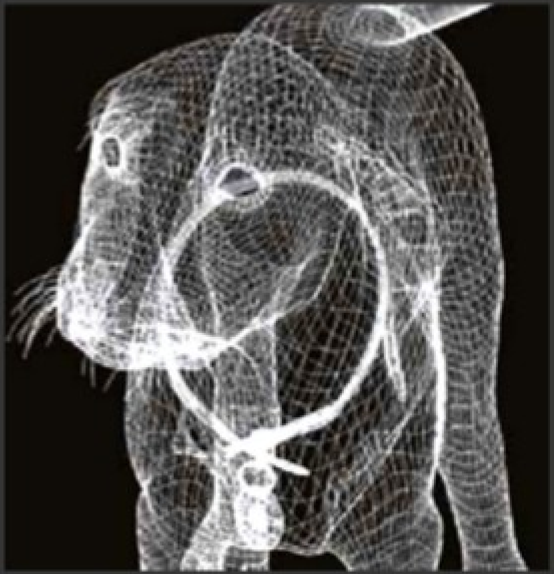
Creating or capturing the representation of objects - motion often geometrical

Rendering

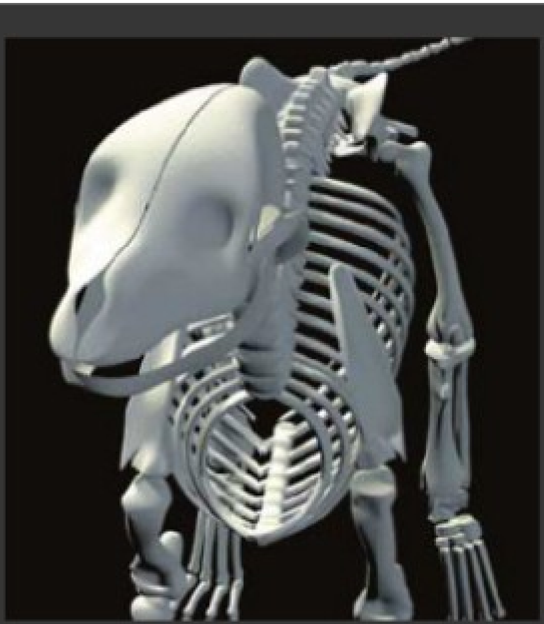
Creating an image of these objects on a display device

Animating

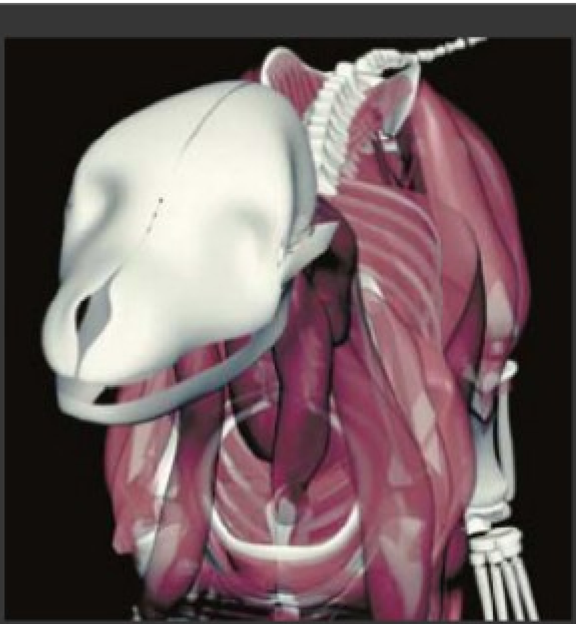
Making objects move by describing how they change over time



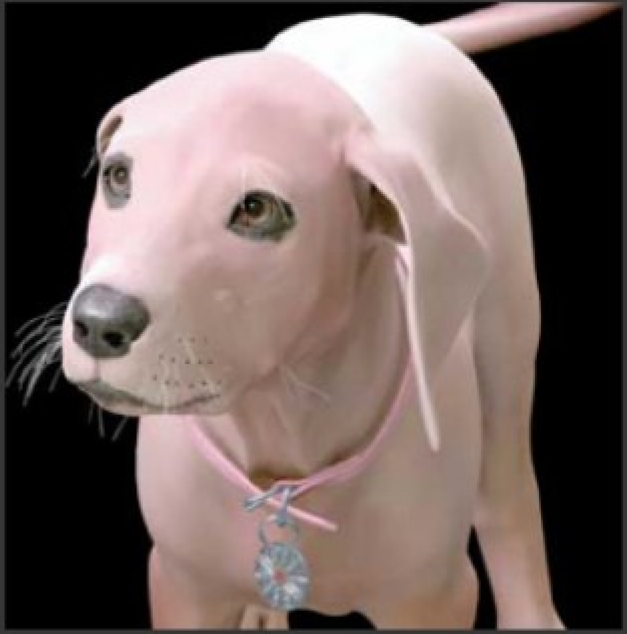
Wireframe Model



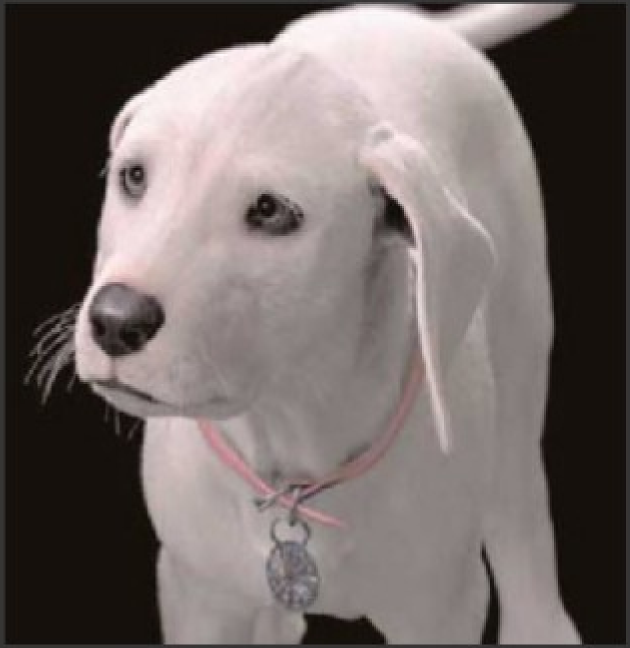
Skeletal Model



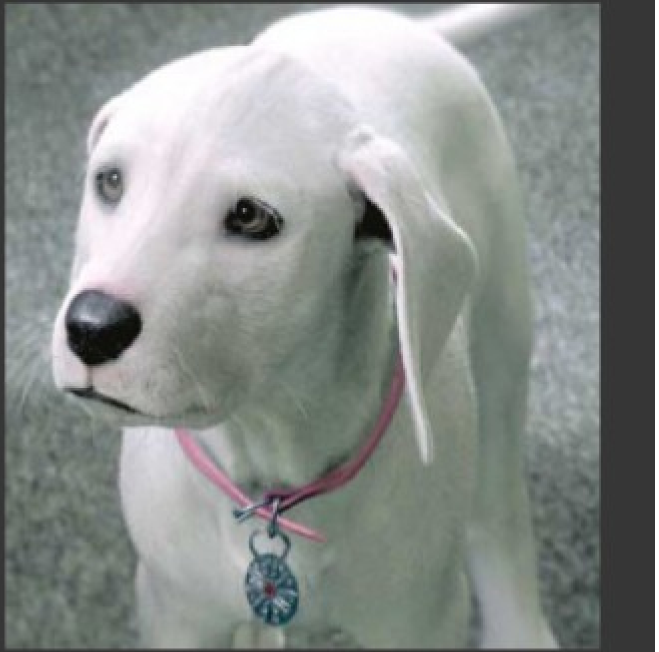
Muscle Model



Skin



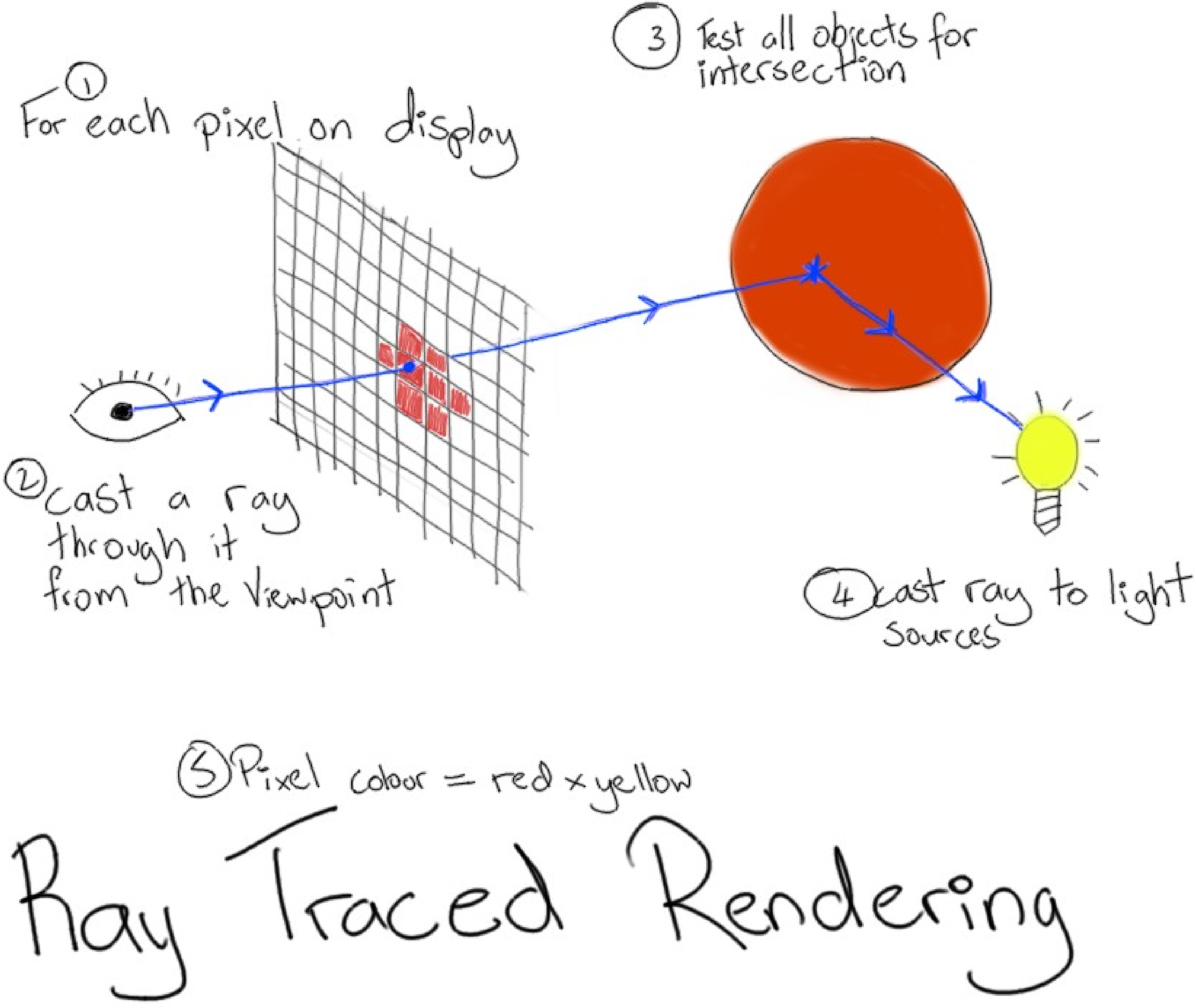
Hair



Render and Touch up

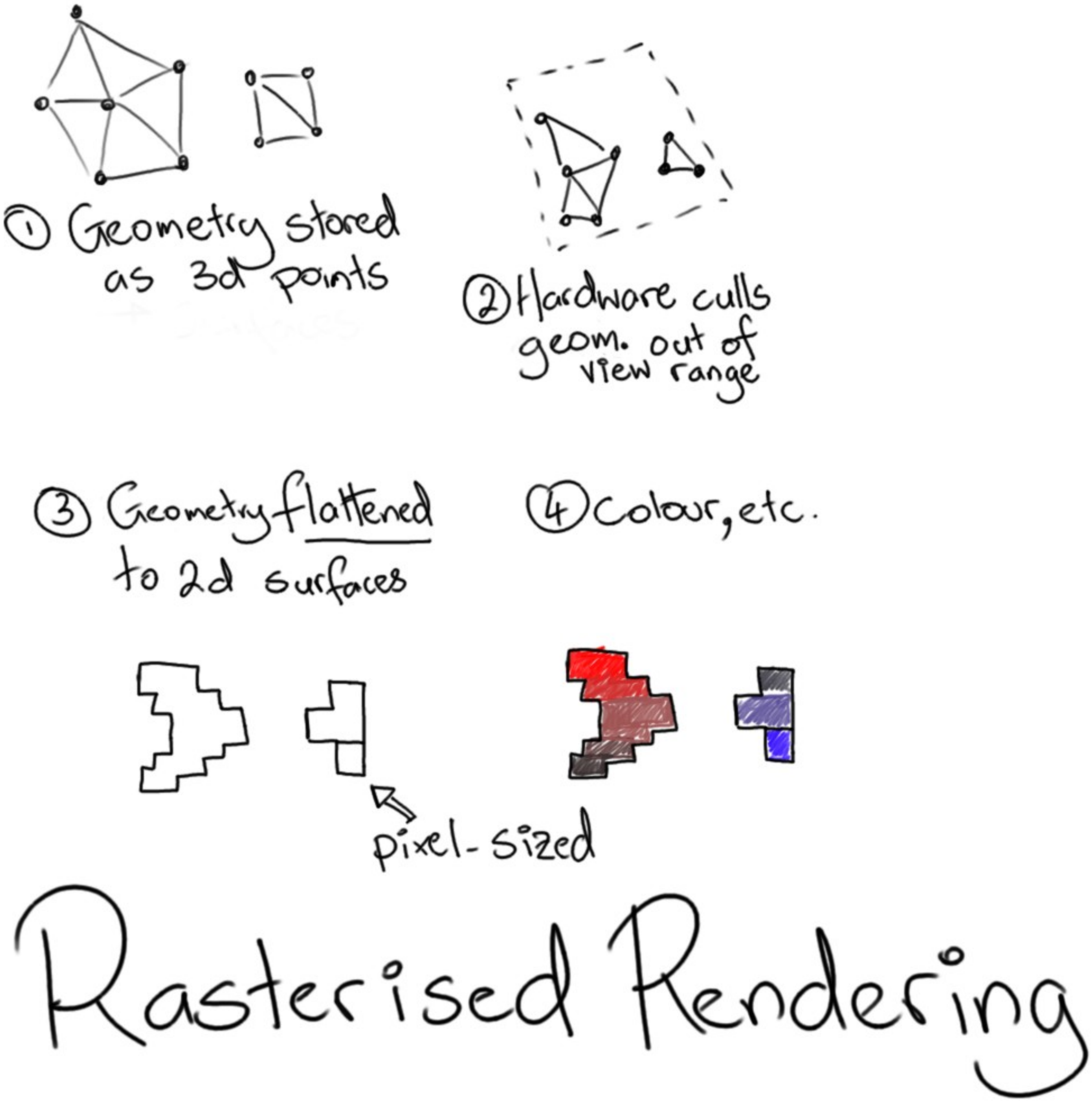
© Walt Disney and TSL

Ray-Traced vs. Per-Primitive

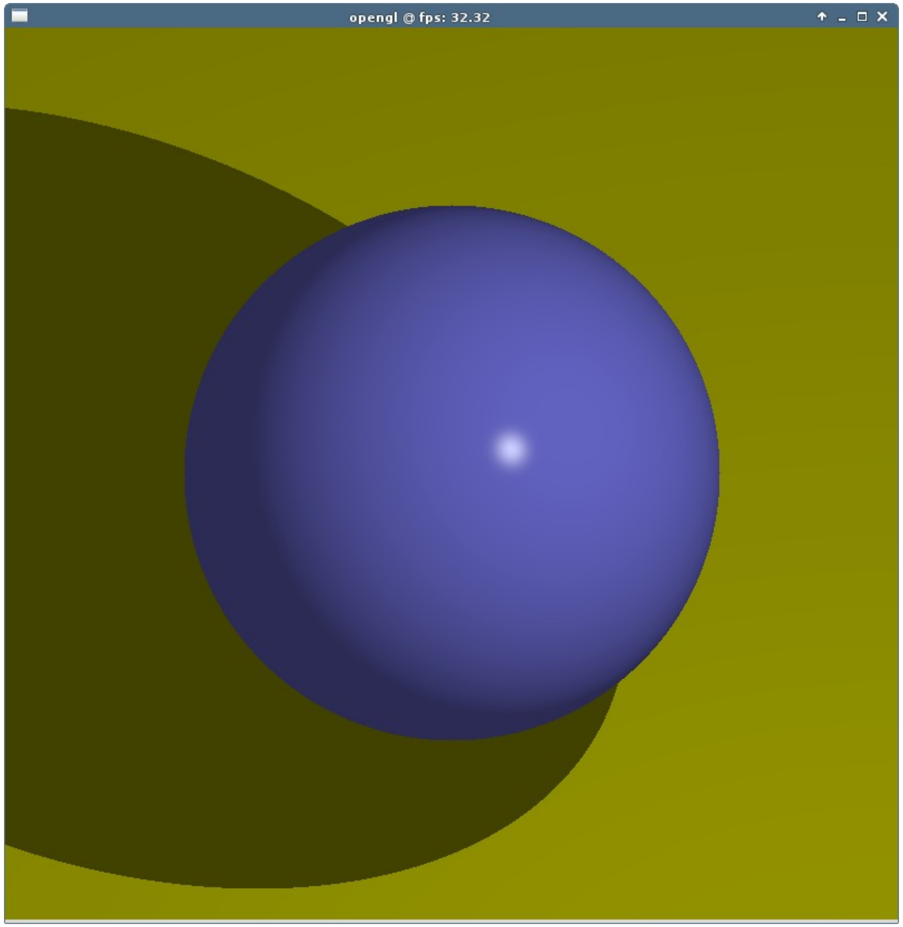


① For each pixel on display
② cast a ray through it from the viewpoint
③ Test all objects for intersection
④ cast ray to light sources
⑤ Pixel colour = red × yellow
Ray Traced Rendering

Ray-Traced vs. Per-Primitive

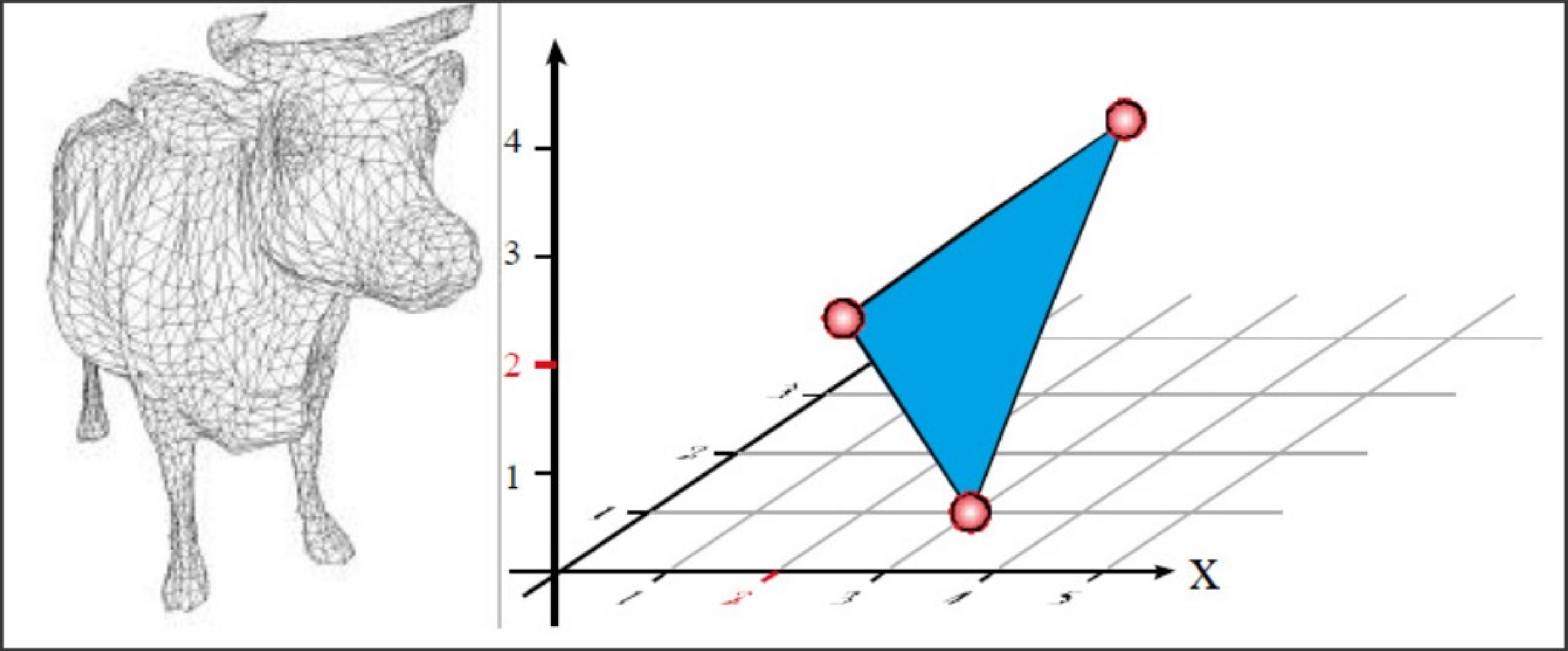

① Geometry stored as 3d points
② Hardware culls geom. out of view range
③ Geometry flattened to 2d surfaces
④ colour, etc.
pixel-sized
Rasterised Rendering

Ray-Traced

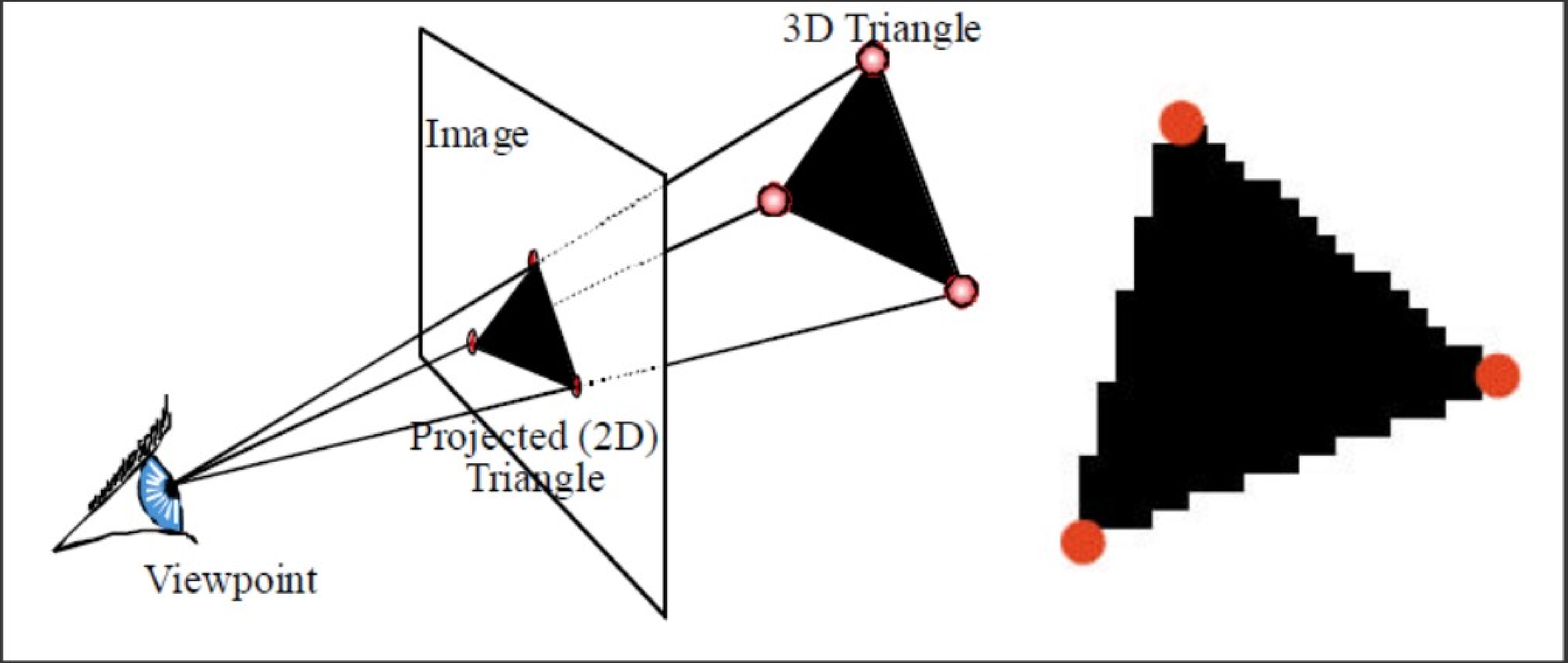


- Per-pixel rendering
- Geometry is represented as a center (0,10,0) and radius 3.0
- Exact to-pixel shapes
- No polygons
- Complex optics

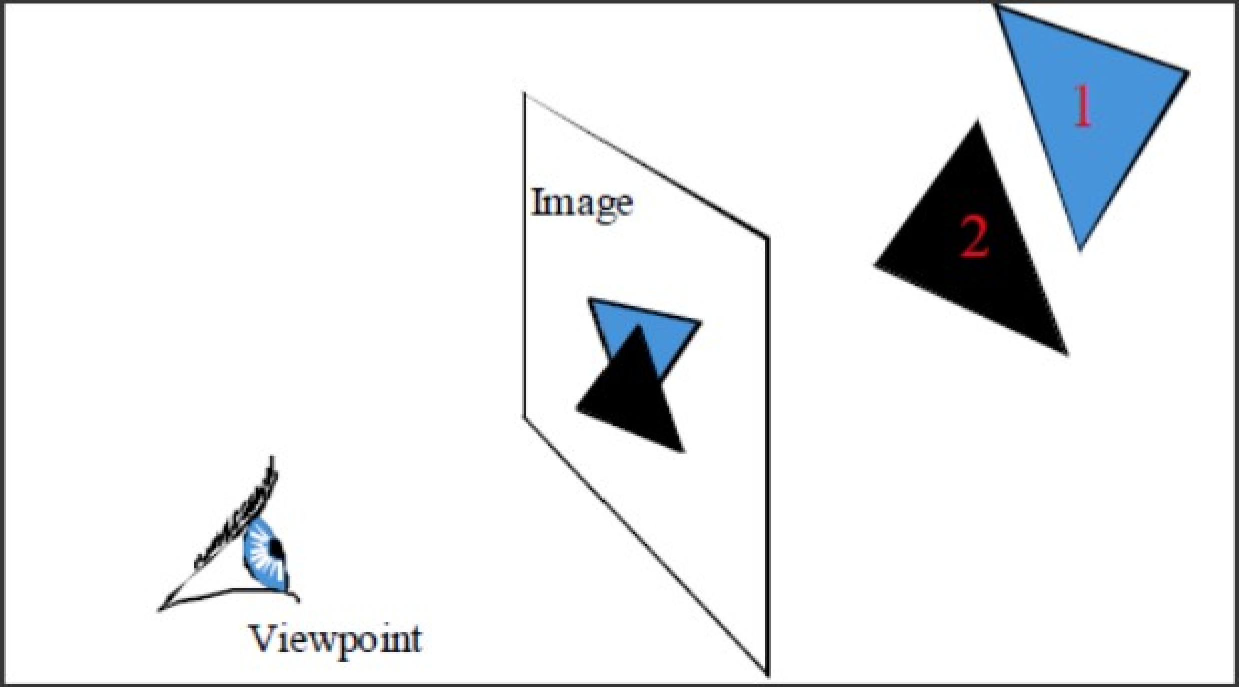
Polygons



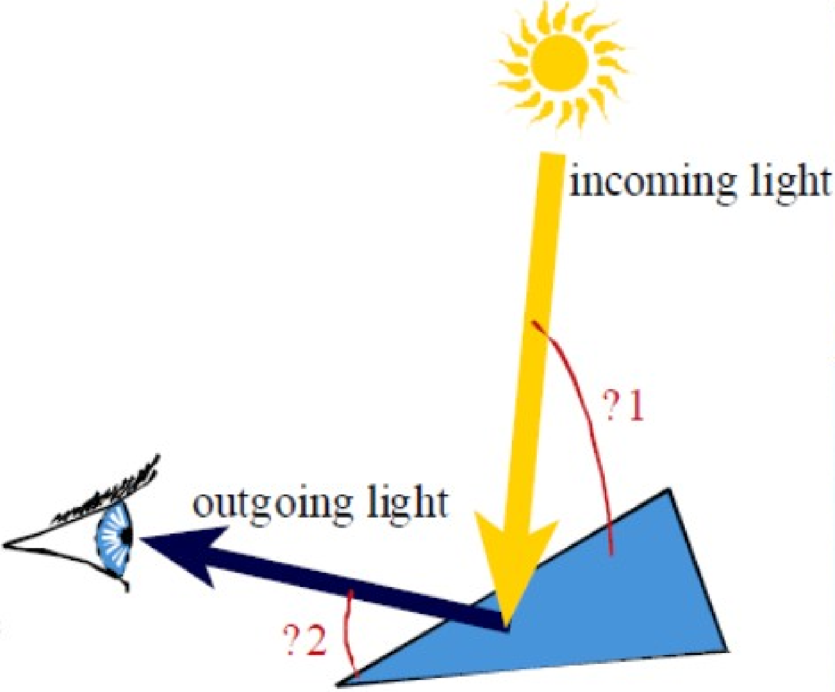
Rendering

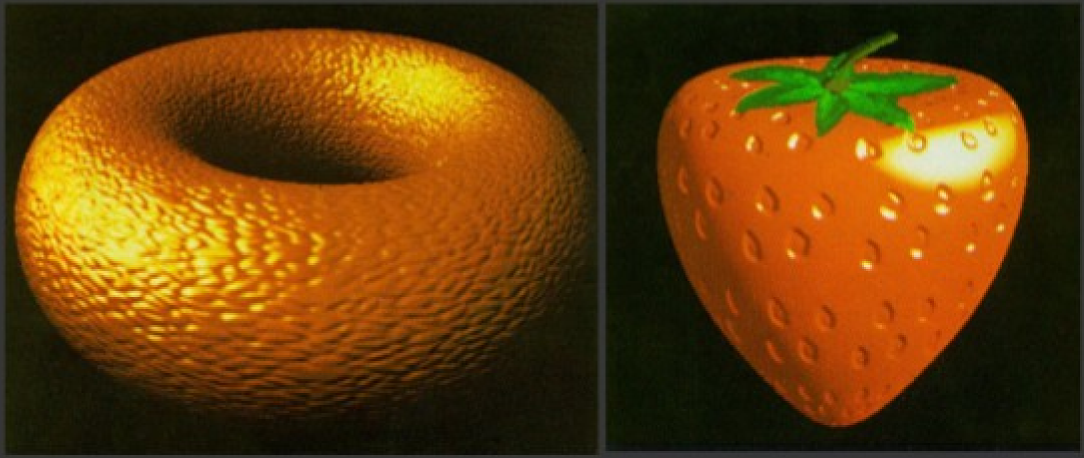


Visibility

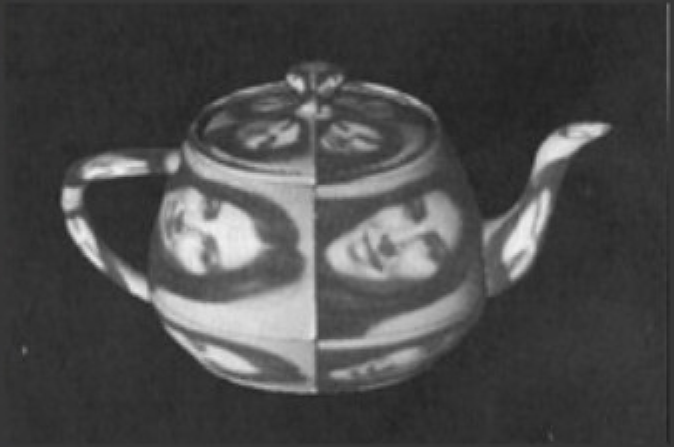


Shading and Materials

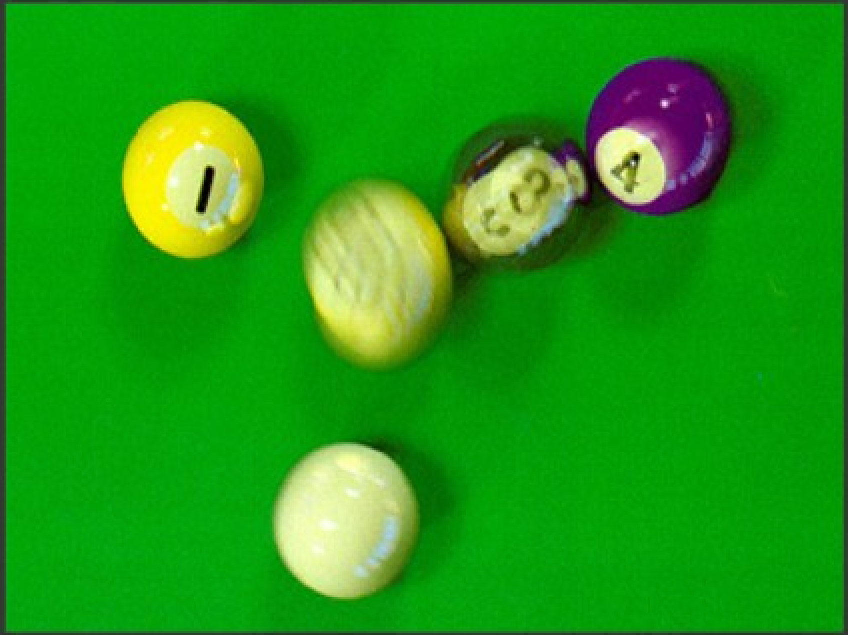




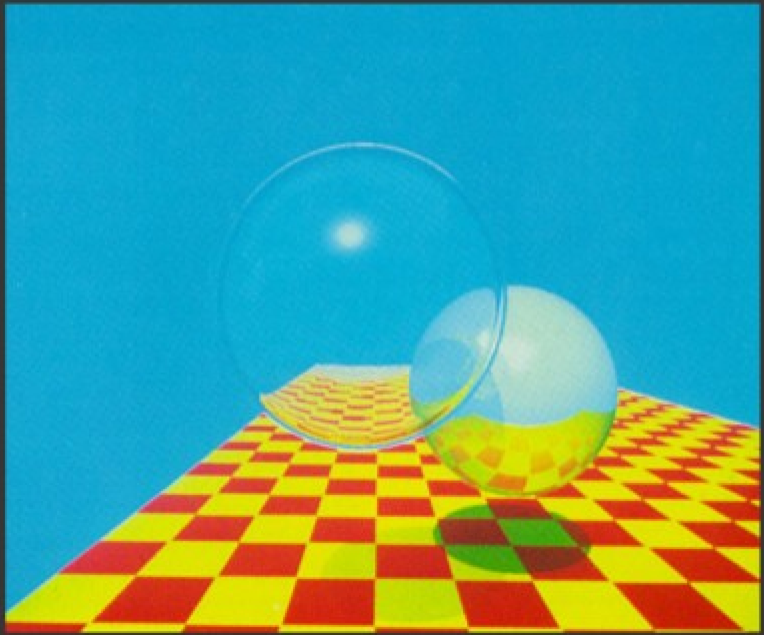
First bump-mapped images (Blinn 1978)



Early texture-mapped image (Catmull 1974)

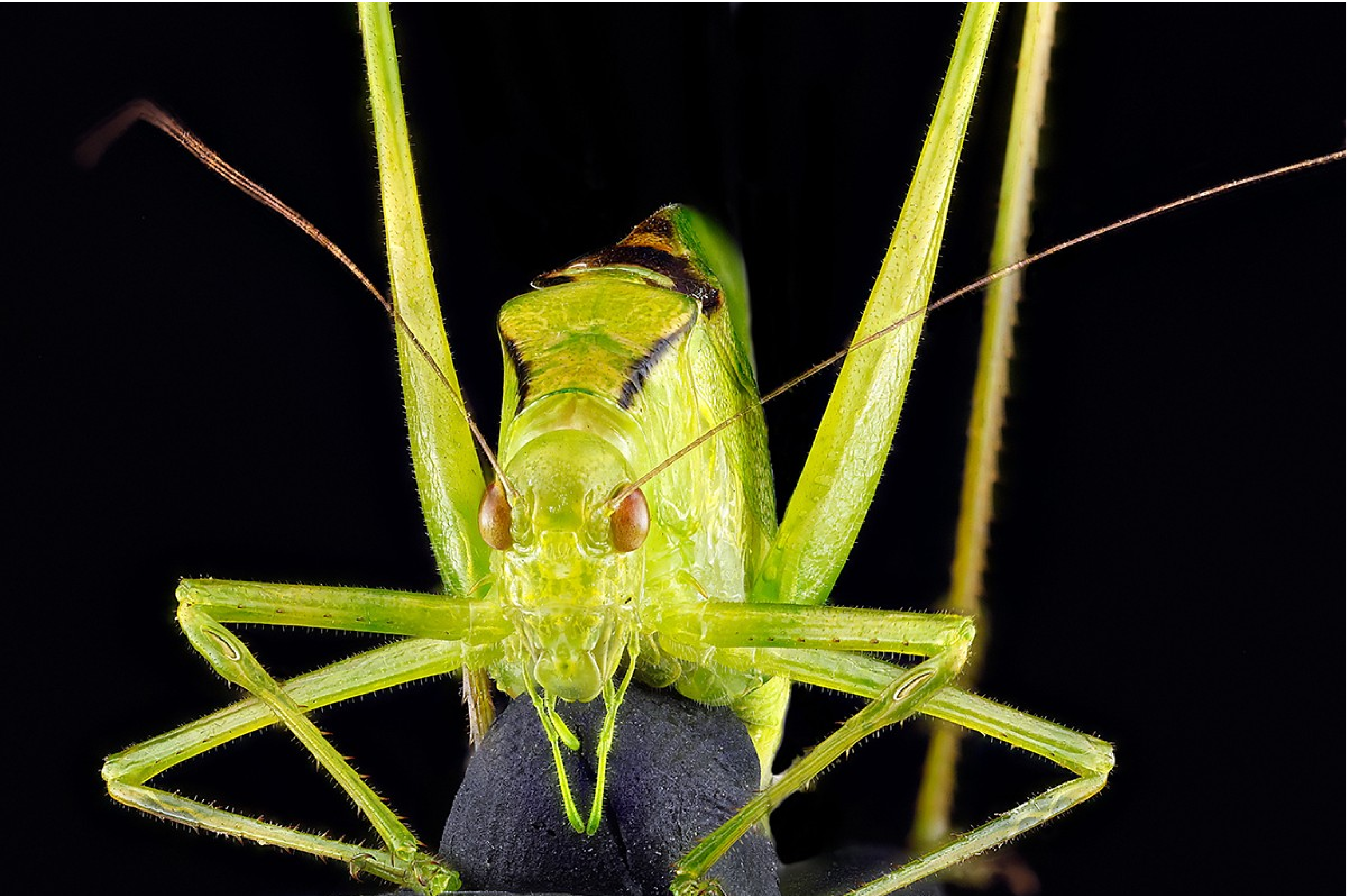


First distributed ray traced image (Cook 1984)



First ray traced image (Whitted 1980)

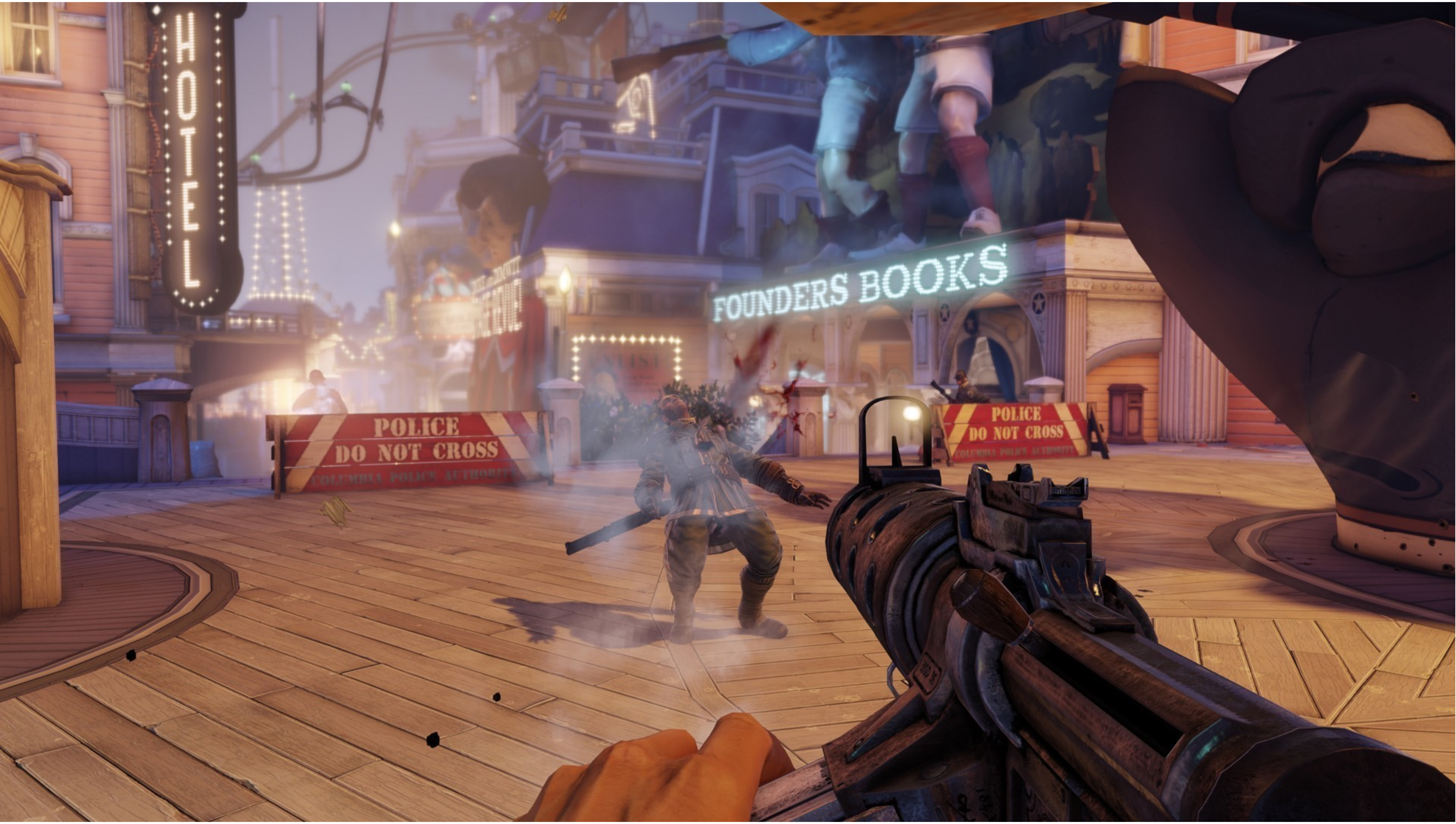
Photograph or CG?



Autodesk “Fake or Foto”

http://area.autodesk.com/fakeorfoto/

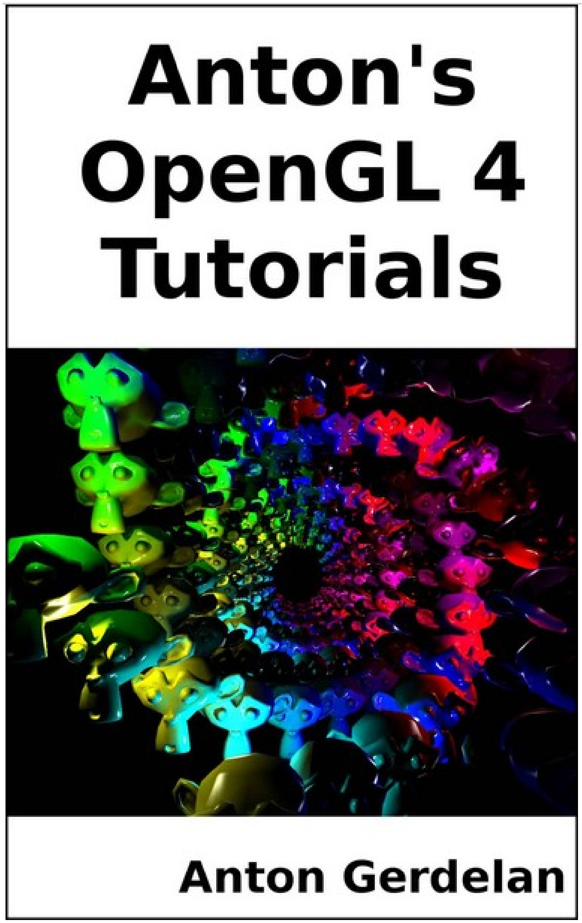
How Many Rendering Techniques Can You Spot?



GV2 - Graphics / Vision / Visualisation Group

- http://gv2.scss.tcd.ie/
- Image, Video and Audio Processing & Analysis
- Perception & Graphics
- Real-time Rendering & Animation
- Custom and Multi-core Hardware Architectures
- Computer Vision & Augmented Reality

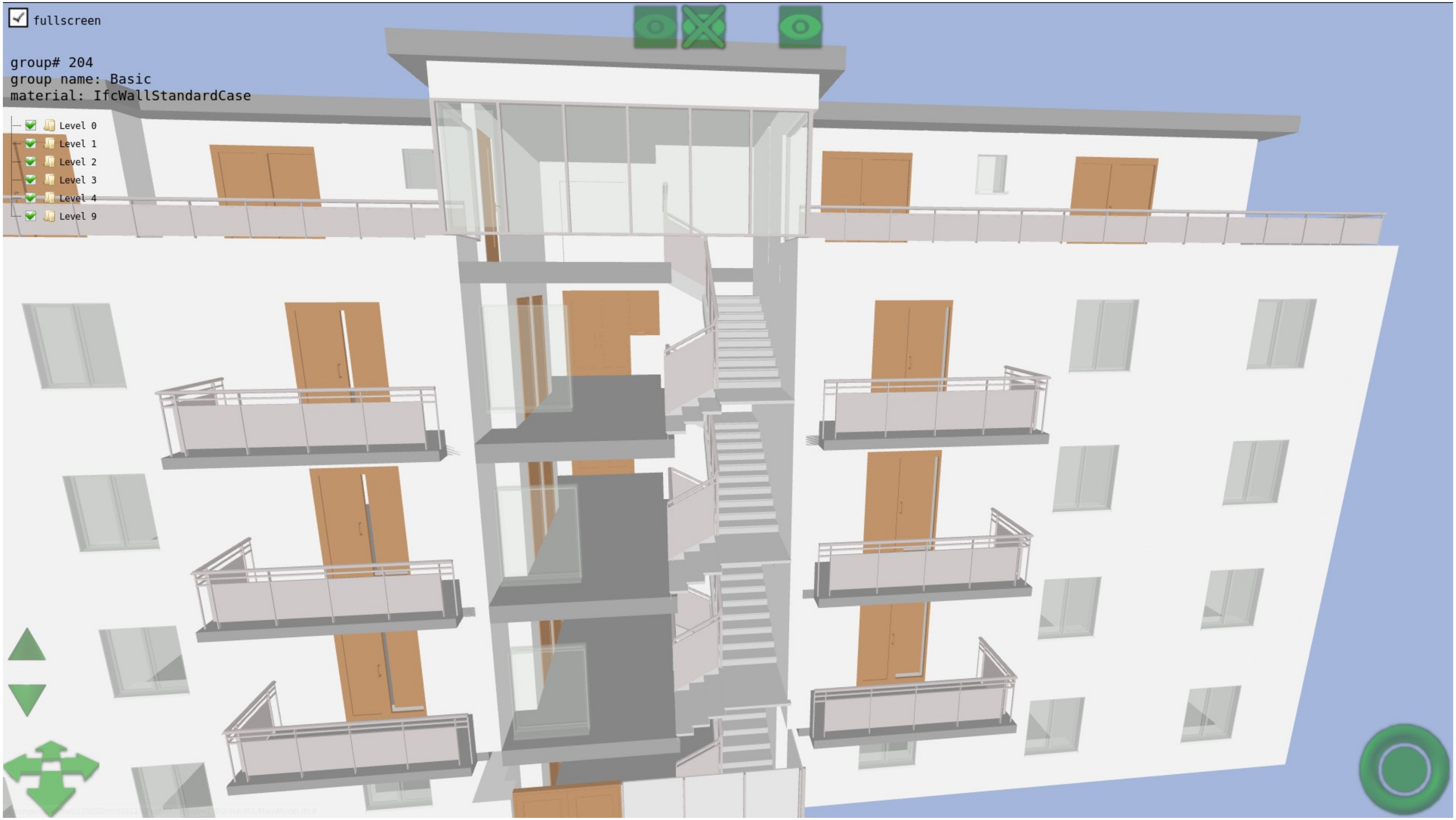
Me



- Search "opengl" on Amazon
- Kindle / cloud
- lots of demos on GitHub

https://github.com/capnramses/antons_opengl_tutorials_book

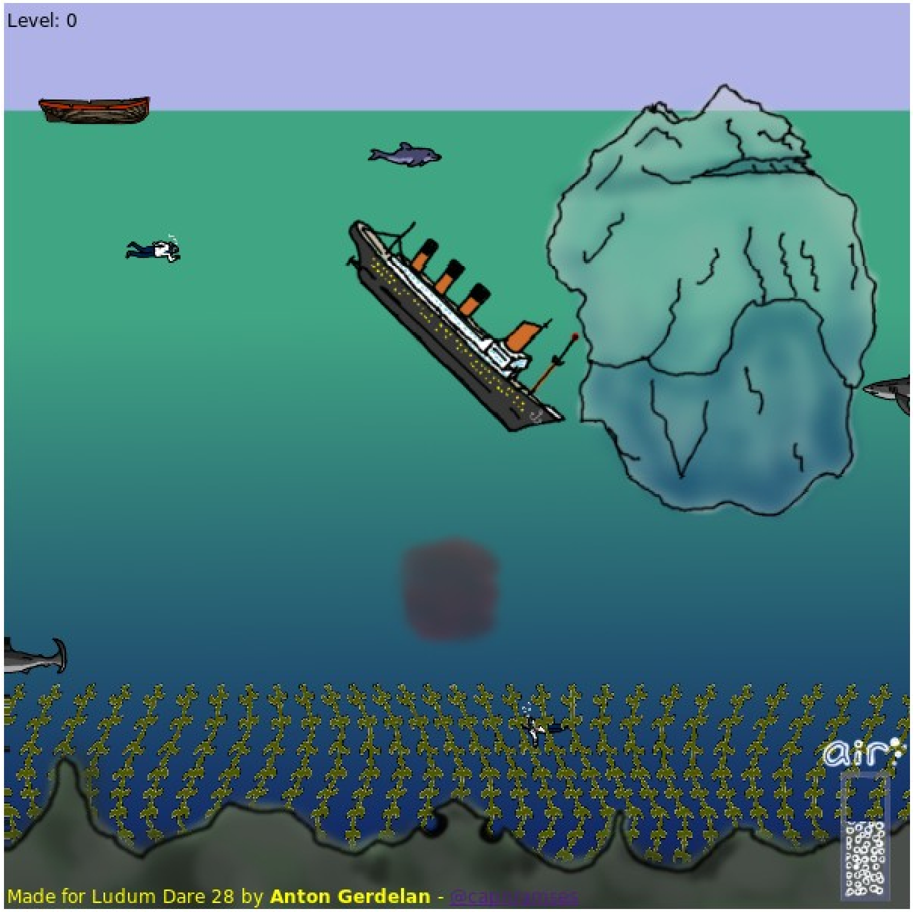
WebGL



☑ fullscreen

```
group# 204
group name: Basic
material: IfcWallStandardCase
```

Ludum Dare



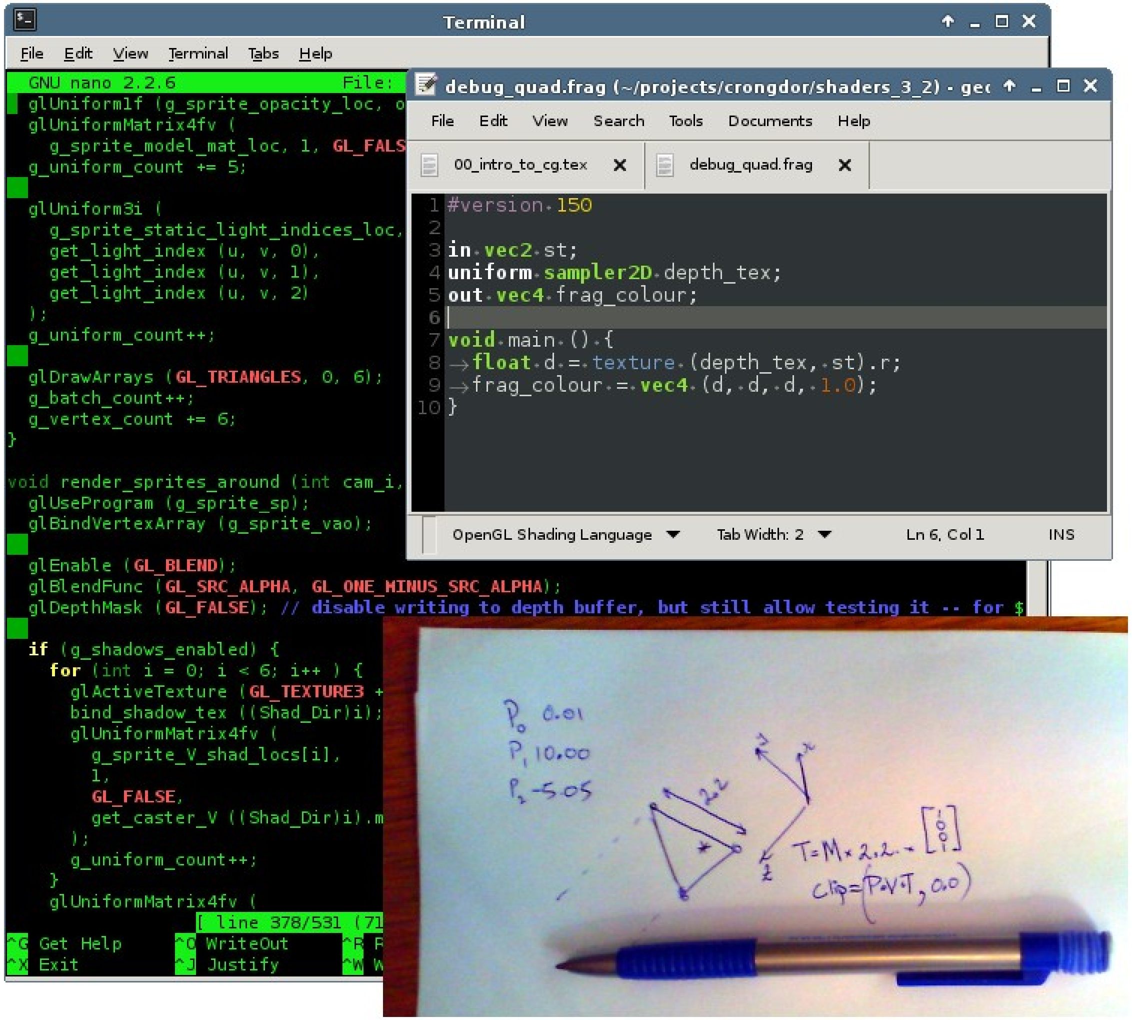
Made for Ludum Dare 28 by **Anton Gerdelan** - @capnramses

antongerdelan.net

Summary

- Outline of course
- What computer graphics is used for
- Research in TCD

- Next Lecture - **Graphics Programming!**



```
  glUniform1f (g_sprite_opacity_loc, o
  glUniformMatrix4fv (
    g_sprite_model_mat_loc, 1, GL_FALS
  g_uniform_count += 5;

  glUniform3i (
    g_sprite_static_light_indices_loc,
    get_light_index (u, v, 0),
    get_light_index (u, v, 1),
    get_light_index (u, v, 2)
  );
  g_uniform_count++;

  glDrawArrays (GL_TRIANGLES, 0, 6);
  g_batch_count++;
  g_vertex_count += 6;
}

void render_sprites_around (int cam_i,
  glUseProgram (g_sprite_sp);
  glBindVertexArray (g_sprite_vao);

  glEnable (GL_BLEND);
  glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
  glDepthMask (GL_FALSE); // disable writing to depth buffer, but still allow testing it -- for $

  if (g_shadows_enabled) {
    for (int i = 0; i < 6; i++ ) {
      glActiveTexture (GL_TEXTURE3 +
      bind_shadow_tex ((Shad_Dir)i);
      glUniformMatrix4fv (
        g_sprite_V_shad_locs[i],
        1,
        GL_FALSE,
        get_caster_V ((Shad_Dir)i).m
      );
      g_uniform_count++;
    }
    glUniformMatrix4fv (
                    [ line 378/531 (71
```

```
#version 150

in vec2 st;
uniform sampler2D depth_tex;
out vec4 frag_colour;

void main () {
	float d = texture (depth_tex, st).r;
	frag_colour = vec4 (d, d, d, 1.0);
}
```

What to Do Now

- Get/share/borrow a graphics theory reference
  - we used “*Real Time Rendering*” in Sweden
- Read the introduction chapters on graphics theory
- Know what vector graphics, and rasterisation are
- Read about the graphics pipeline
- Identify gaps in your vector/matrix maths
- Brush up on C skills (memory, pointers, addresses)
- Start doing basic OpenGL 4 tutorials now