Computer Graphics

COMP3421/9415
2021 Term 3 Lecture 15

What did we learn last week?

Advanced Lighting and Maps

- Multiple addons/corrections for Phong Lighting
- Maps
- Diffuse, Specular and Normal Maps
- Extra surface details means less geometry needed

What are we covering today?

Image Reflections

- A step beyond specular lighting
- An actual image instead of just the light source
- Rendering objects with a material like chrome
- Cube Maps, Skyboxes
- Environment Mapping
- Realtime Reflections

First Reflections in Games?

Duke Nukem 3D's Bathroom Mirror

There's a bathroom mirror in a game in 1996

- Some context . . .
- Phong shading appears in games in 1999
- Quake is also released in 1996, with a very limited lighting model
- How did this game manage to handle reflections?

Image credit: 3DRealms and Gearbox Software

What's required for a believable reflection?

What are we expecting from a virtual mirror?

- Inverted scene
- Changes angle of view in realtime
- Perspective shift based on distance to (and through) mirror
- Character(s) moving relative to their current position

1996

- We don't even have a complete lighting model yet!
- How was this achieved?

Keep Tricking Those Humans

Never assume technology when dirty tricks will work

- There is no complicated inverted rendering
- There's no calculated perspective
- No ray tracing of vision "through" the mirror

Trick the humans!

- There's a whole other room on the other side
- It's 3D so its perspective is automatic
- There's a replica of the player that copies your movements

The tricks revealed

If you want to look into it

- Turning off clipping in the game allows you to walk through walls
- Turns out, walking "through" a mirror just places you in an inverted scene
- https://youtu.be/1NG1djv-mA?t=33

Is this how it's done?

No, we're going to reflect images without tricks

- It's interesting to look at though
- Genuine reflection took a few more years to appear in games
- But the techniques were already decades old!
- "Flight of the Navigator" (1986)

Image credit: Disney

Rendering Reflections

Reflections in Polygon Rendering

What's the basic idea?

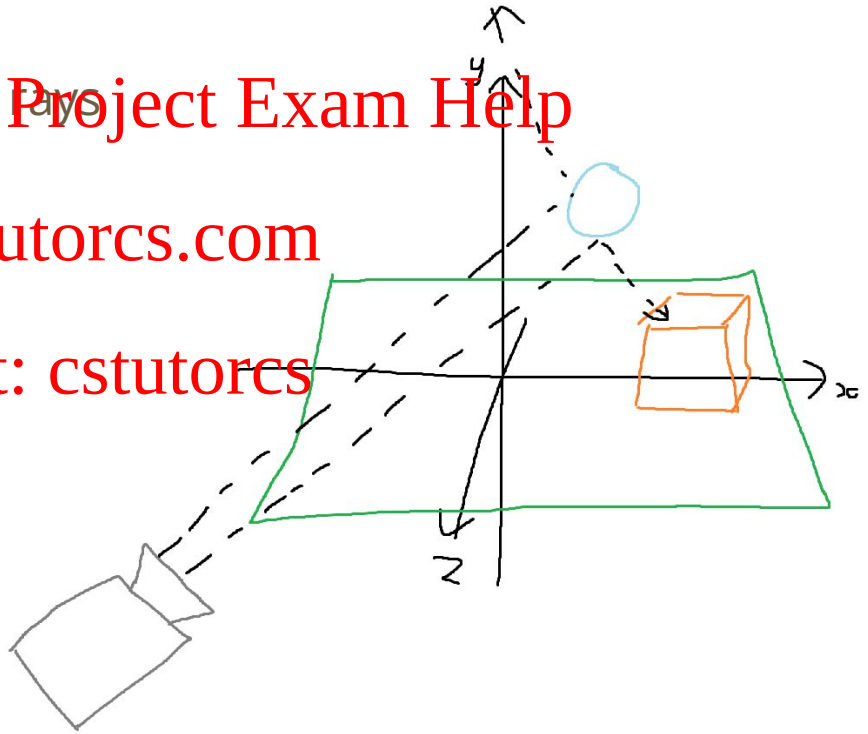
- We could do Ray Tracing!
  - (the expensive answer to all graphics questions)
- Cast a ray from a viewer to a reflective object

- Reflect using surface normals
- Take colour from where it hits and mix it with the surface

- This makes sense!

Ray Tracing for Reflections

- A chrome sphere in a scene
- Some ideas of reflected vision Rays



Analysis of Ray Tracing Reflections

Pros

- 100% accurate in theory
- Deals with any kind of curved or warped reflective surface (including normal mapped etc)

- Can sample the colour of the reflecting surface as well as the final point

Cons

- How expensive is it to find out what object you hit with the reflected ray?
  - The answer is the current retail price of a RTX graphics card
  - The real answer is close to O(n) where n is the number of triangles in the scene!

A Compromise

Can we get the pros without the cons?

- The ray bouncing off the shiny surface feels right
- The collision detection afterwards doesn't
- Can we prepare the scene so we're not doing collision detection per ray every frame?

Cube Maps

What is a Cube Map?

A particular form of texture

- Acts like a cube in world space
- It's like we're in the centre of a box looking at its inner surfaces
- Colour data is on the 6 faces of the cube facing inwards
- We can sample the texture using a direction vector

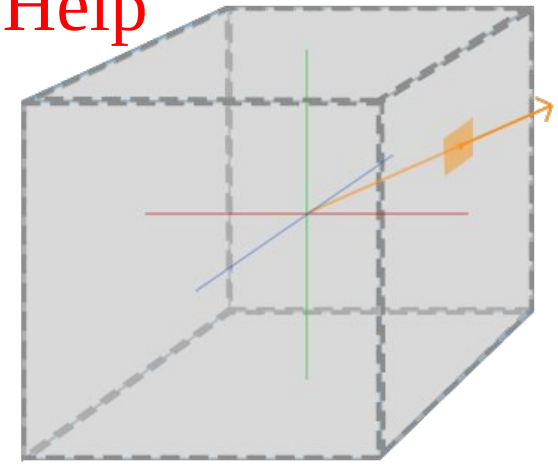


Image credit: learnopengl.com

Skybox

A very common use of a cube map

- Represents everything in the distance
- Things that can be seen, but don't need geometry
- Too far away to need real detail
- Not just used for reflections!
- These will usually be our scene backgrounds

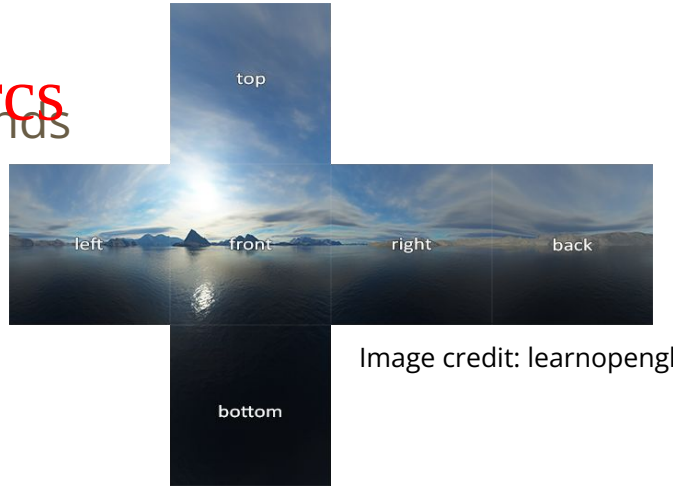


Image credit: learnopengl.com

Sampling a Cube Map

We sample a cube map with a direction vector

- Assumes the cube map is very distant
  - Like a directional light
- Place the centre of the cube at (0,0,0)
- Interpolate the direction vector until it reaches the cube's surface
- Sample the colour from that position

Creation of Cube Maps

The example looks like a photo

- Photography techniques with chrome spheres
- Or multiple photos
- A cube map is a view of a scene looking in every direction from a single point

Image credit: Gene Miller (% https://www.pauldebevec.com/)

Break Time

What's the most important thing in Graphics?

- Does the human observer believe it?
- Is it realistic enough for them?
- Or are they engaged with the art style?

- This is *Need 4 Speed: Most Wanted (2012)*

- Note how high quality the car is
- vs how plain and repetitive the environment is
- The effort is where the human's eyes are!

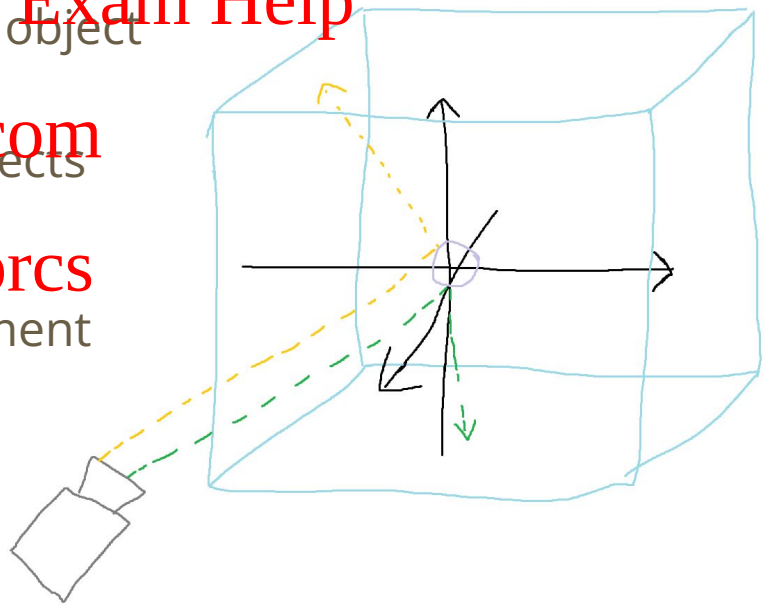
Image credit: EA

Environment Mapping

A Cube Map of the Environment

If we have a cube map

- Represents the environment around an object
- Can be sampled with a direction vector
- We can now reflect vectors off shiny objects
  - We'll use information per fragment
  - Direction to the Viewer and Surface Normal
- And see where they reflect the environment



In OpenGL

OpenGL has "`GL_TEXTURE_CUBE_MAP`"

- A convenient set of 6 textures
- We'll still have to generate 6 individual textures and bind them to the cube map
- Sampling from the cube map is done with a direction vector
- OpenGL handles sampling the actual colour

Static Environment Reflections

Pre-prepared vs dynamically created

- Environment Mapping gives us the static background of the scene
- For a lot of reflections this is enough

- But we want foreground objects and movement to also be reflected

- Can we create a cubemap dynamically?
- We're going to need some more tech to do this in realtime . . .

Metal Mario uses Environment Mapping in Mario 64 (1996)
Image credit: Nintendo

Frame Buffers

Framebuffer Revision

A target to render to

- The default framebuffer is the last buffer we write colours to
- Its RGB values are sent to the monitor
- We can create other framebuffers

- We can also render to those framebuffers

- They can store textures (colour) and other data
- In HDR, we created a framebuffer the same size as our monitor/window
- But we're not limited to that!

Framebuffers and Cube Maps

We can use the same concept!

- Cube maps are textures
- Frame buffers can store textures
- Frame buffers can be written to in realtime
- We can write to the 6 faces of the cube map while rendering our frame!

Rendering to Textures

Render to Texture

A technique to create texture data in realtime

- Previously our textures were images given to our application
- But a texture in OpenGL is a colour data buffer
- Buffers can be written to by our shaders

- eg: The Framebuffer colour is set by the final fragcolor in our shaders

- So our 6 framebuffers that make up our cube map could be dynamically created

Render Targets and Cube Maps

Working with Framebuffers

- By default we're rendering to the main framebuffer
- But at any time in our fragment shader, we can bind to a different framebuffer
- So we can actually render 6 different times
- Each time rendering to a face of the cube map

Realtime Reflections

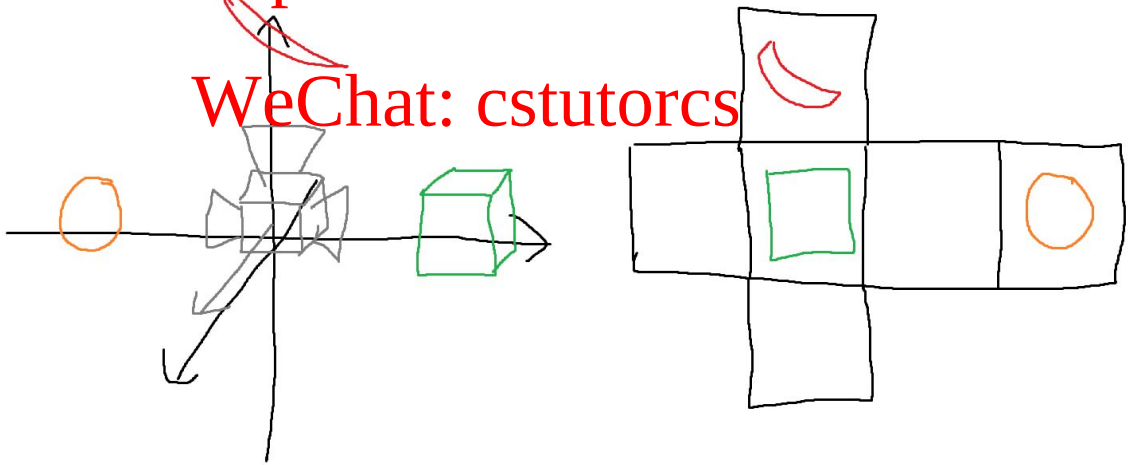
Making a cube map dynamically

A few renders in one

- Set up a temporary camera where your shiny object is
- Set up a cube map with its 6 frame buffers
- Render 6 times, aiming the temporary camera in each of 6 directions
  - Positive and negative in each axis
  - Each render writes data to one face of the cube map
- Then use that cube map to sample from for reflections

Realtime Cube Map Creation

- Remove the shiny object and replace it with the new camera
- Take 6 renders in different directions and write the results to the cube map's framebuffer textures



Realtime Reflections

With a new Cube Map each frame

- This cube map is created at the location of the reflective object
- Reflections from the main camera's viewpoint can use that cube map
- Reflect the viewer direction vector using the surface normal for each fragment

Realtime Cube Maps

Pros

- Able to reflect dynamic objects
- Able to reflect more than just the Environment Map

Cons

- Wait, 7 renders per frame?
- Faster than ray tracing, but still not "fast"
- We have tricks to save processing time . . .

Shortcuts for efficiency

How do we reduce the cost of the 7 renders per frame?

- Leverage the fact that reflections are usually distorted and unclear
- Cube map is not created every frame!
  - The scene around the shiny object might not change that much
  - Only render the cube map every few frames

- Lower the resolution of the cube map
  - Reflections don't need the same resolution as the screen
  - The cube map can be very low resolution and still get close to one reflected texel per pixel

- Leave out the downwards render
  - The Need 4 Speed optimisation
  - If very few of the reflections are going to use the downward face, we won't need to render it

What did we learn today?

Reflecting on Reflections

- Ideas about how to handle reflections
- Cube Maps
  - Sampling using direction vectors
- Environment Mapping
- Frame Buffers and Rendering to Textures
- Realtime Cube Map creation