Shadows and Deferred Shading

Department of Computer Science
Lund university

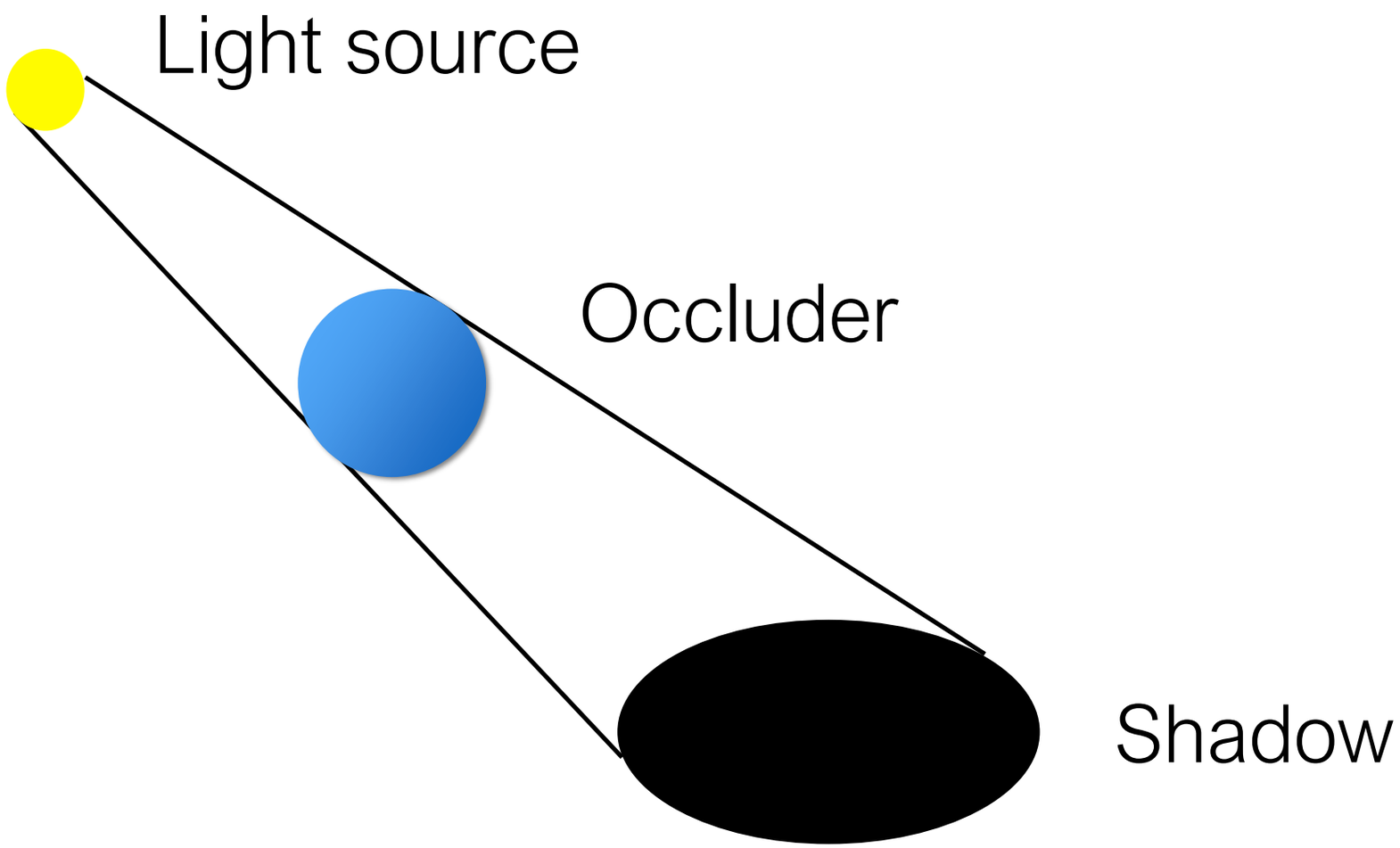
Shadows

- An important visual cue in images
- Indicates an important spatial relationship between objects

- Easy to do in ray tracing, harder in rasterization pipelines

© mmxvi mcd

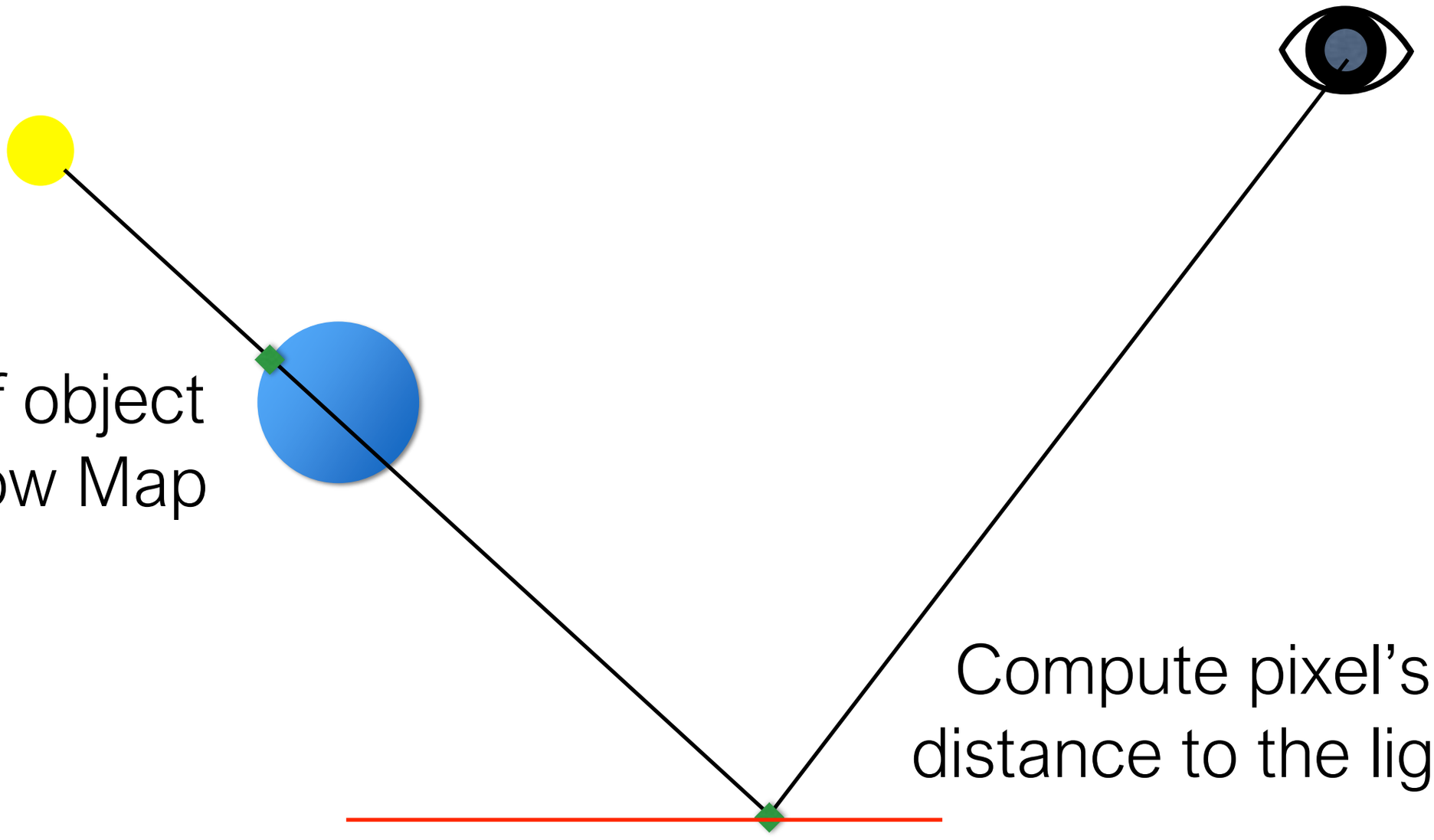
Shadows



Shadow Mapping

- Render the scene from the lights point of view
  - Use the depth buffer as the **Shadow Map**
- Render from the camera’s POV, then at each pixel
  - Transform the pixel’s position into the light’s coordinate space
  - Compare the distance to the light, to the value in the Shadow Map
  - If the pixel is further from the light, it is in shadow

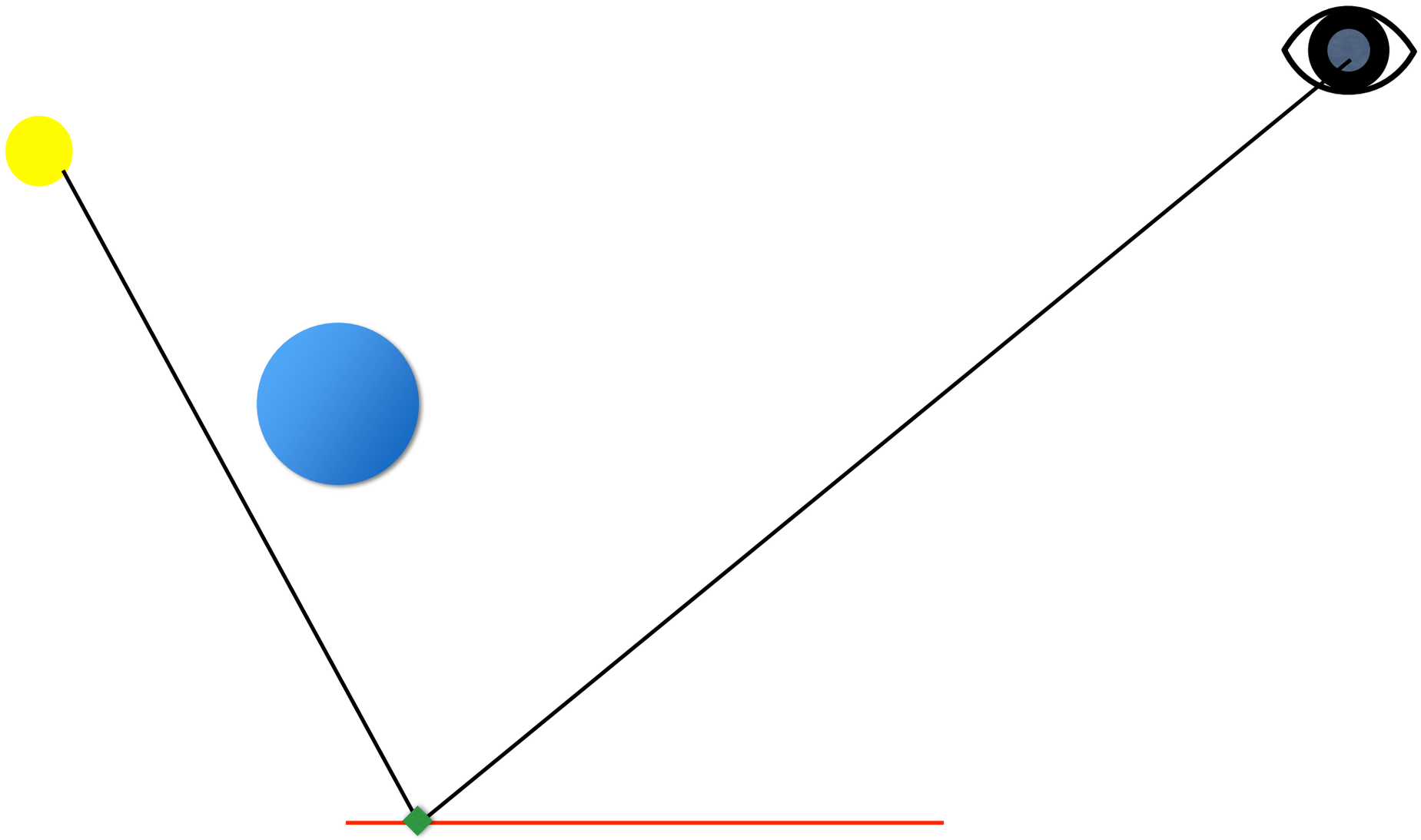
Shadows



Compare distance of pixel to depth in Shadow Map (SM)

Pixel_LightSpace.z > SM(x,y).z

Shadows



Lit

Compare distance of pixel to depth in Shadow Map (SM)

Pixel_LightSpace.z < SM(x,y).z

Render to Texture

- **Render/Draw Pass** - Every time we tell the GPU to draw, everything needs to be set up including :
  - Framebuffer with a Color buffer and Depth buffer
- We can reuse these buffers as textures
  - Called Render to Texture
- We use the same concept for Shadow Maps and Deferred Shading

Shadow Map Problems

- Self-shadowing
  - Imprecision causes errors, resulting in artifacts
  - Solution is to add a bias to Shadow Map value
- Aliasing
  - Increase resolution
  - Lots of other solutions exist
- Further reading
  - Efficient Real Time Shadows, SIGGRAPH 2012 course

Deferred Shading

- Separating geometric and light complexity
- Avoids shading occluded geometry and lights

Shiny PC Graphics in Battlefield 3, Part 2/5

Johan Andersson, DICE

http://youtu.be/UAgWi6hQ0Mk

How?

- Render geometry to geometry buffer
  - **G-Buffer**
  - Render lights to accumulative lightbuffer
  - Resolve by blending lightbuffer and parts of the geometry buffer

Off screen buffers

- Depth Buffer - *d(x,y)*
- Color Buffer - *RGB(x,y)*
  - Textures are color buffers too!
- Pixel shader can write to more than just the color buffer

© mmxvi mcd

Geometry buffer

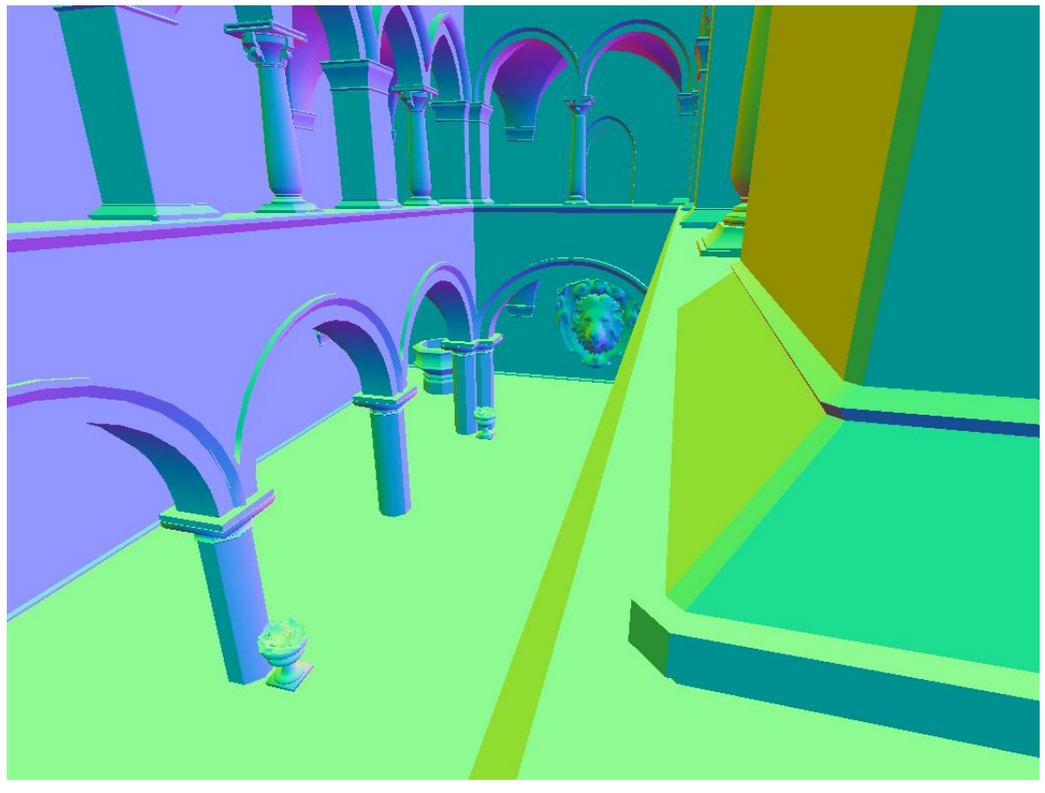
- Several fullscreen textures containing geometric information
  - Depth
  - Normal
  - Diffuse texture colour
  - Specular power
  - …

Geometry buffer: Depth

- Screen depth of the geometry
- 32 bit floating point
- Range 0.0 - 1.0

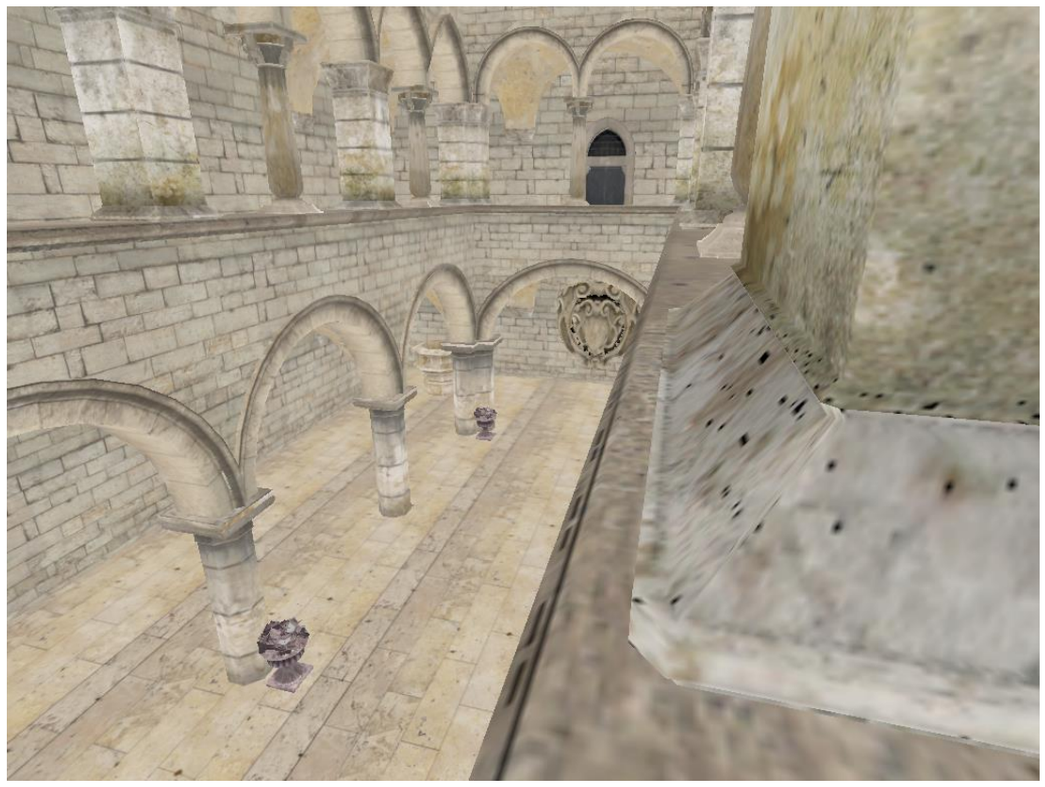
Geometry buffer: Normals

- World space normals
- Encoded in RGB
- 8 bits per dimension



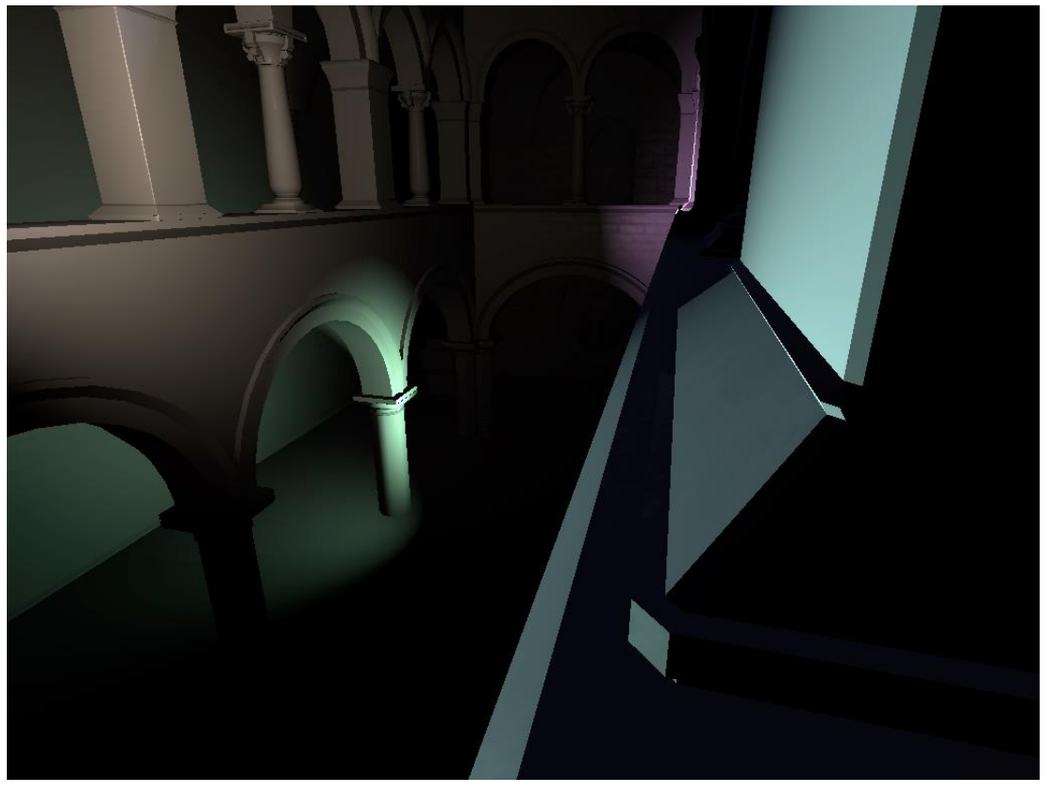
Geometry buffer: Diffuse and Specular texture

- Texture colour
- Encoded as RGB
- 8 bits per channel



Lightbuffer

- Accumulative buffer
- Render all lightsorces sequentially



Lights

- Render each light as a volume
  - Pointlights -> sphere
  - Spotlights -> cone
- Only shades visible pixels covered by light volume
  - Which light volume pixels will be visible?

Lights continued

- Two ways to render
  - Backface culling and depth-test less
  - Frontface culling and depth-test greater
- Both has advantages and disadvantages

Lights
Backface culled
depth-test less

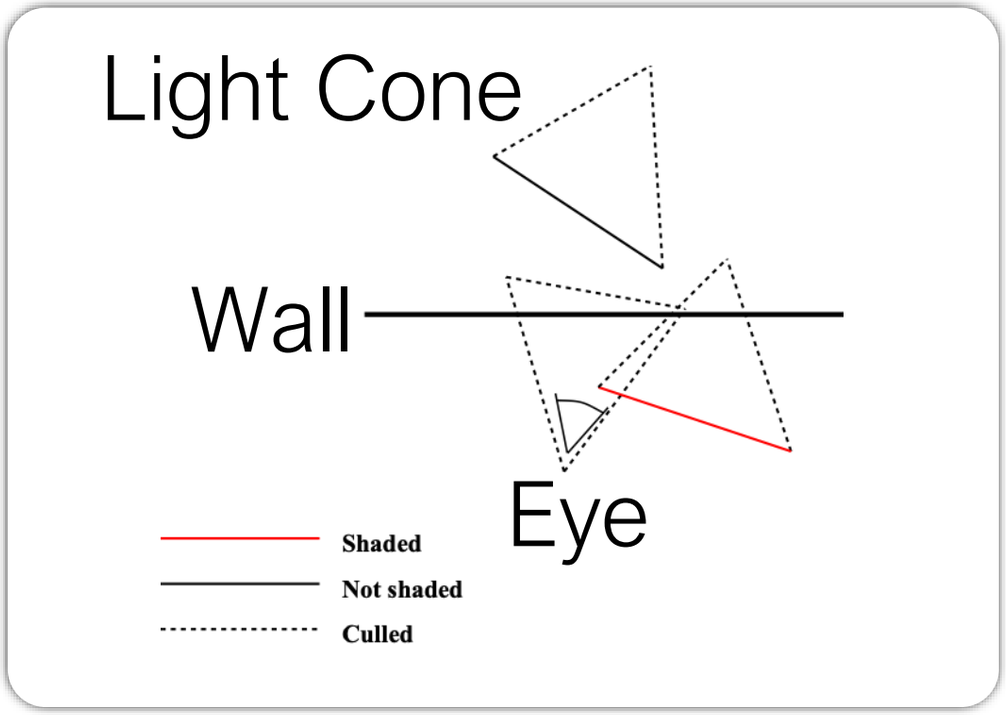
- Renders the outside of the light volume
- Shades parts that are in front of the rendered geometry

Lights

Backface culled cont.

depth-test less

- Advantages
  - Doesn’t shade occluded lights
- Disadvantages
  - Doesn’t work if the camera is inside the light volume



Lights
Frontface culled
depth-test greater

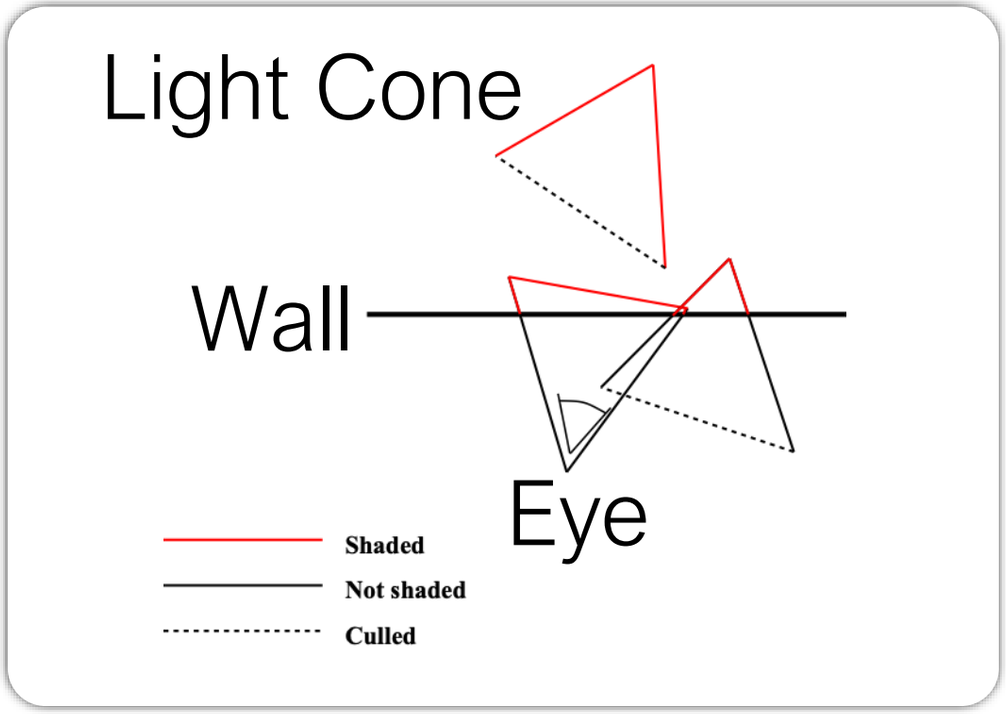
- Renders the inside of the light volume
- Shades lights that are behind the rendered geometry

Lights

Frontface culled cont.

depth-test greater

- Advantages
  - Works if the camera is within the light volume
- Disadvantages
  - Renders occluded light sources



Shading lights

- Render the light volume
- For each covered pixel
  - Read depth, screen space coordinates
  - Perform inverse projection to get world space coordinates
  - Read normal
  - Shade

Lights, shading

- Point lights
  - Phong
  - Falls off by distance squared
- Spot lights
  - Phong
  - Falls off by distance squared
  - Linear falloff from centre of spotlight to the edge
    - Calculate using Light_direction·Light_vector

Shadow maps

- Resulting shadow

- Blockiness depending on the resolution

- Can we make it better?

Percentage Closer Filtering

- Several shadow map comparisons
- Square filter kernel is enough, better ones exists
  - Example: square 4x4
- Weight the result together

Deferred shading with shadow maps

- Result
  - Realistic lighting in real time
  - Multiple light sources with shadows

Deferred Shading Lab

© mmxvi mcd

Deferred Shading Lab

- Rendering Passes
  - 1. Render scene - create G-Buffer
  - 2. Loop over all lights - create Light Buffer
    - Create shadow map
    - Render light’s contribution
  - 3. Compute final image using G-Buffer and Light Buffer - Resolve Shader

Deferred Shading Lab

- Framework using C++ OpenGL code
  - Uses Bonobo helper functions
  - Need to add clears
    - `glClearDepthf(1.0f);`
    - `glClearColor(0.5f, 0.6f, 0.7f, 1.0f);`
    - `glClear(GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT); // Clear both depth and color`

Hints

- gl_FragCoord is [0..XResolution, 0..YResolution, 0..1]
- Anything used with matrix multiplication is usually [-1,1]
- Store values in textures, or G-Buffer as 0-1
- Normals we have to convert to [0,1]
  - i.e. normal.x*0.5+0.5
- Textures are indexed with [0,1]
  - if you are using a transformed value it’s in [-1,1], you need to convert it first
  - i.e. xcoord*2.0 -1.0 (xcoord+1.0)/2.0
- After multiplication by a ViewProjection matrix, the position in the new space must be divided by w
- Follow instructions in Assignment 2 description on webpage
- Look into more ways of accessing texels: textureFetch(), textureOffset()

Other

- Start thinking about projects