Intermediate Graphics & Animation Programming

GPR-300
Daniel S. Buckstein

Deferred Rendering
Week 6

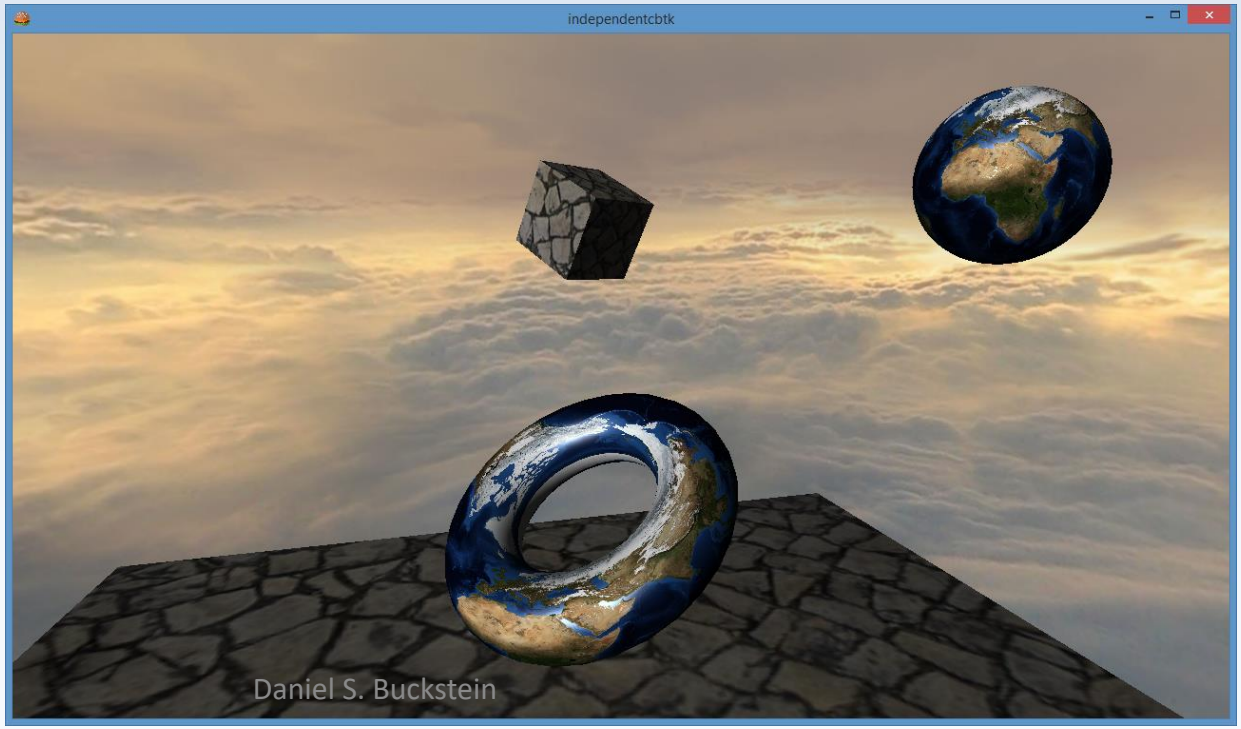
License

Deferred Rendering

- Forward rendering path
  - Review of forward shading
- Deferred shading
  - 2 passes
- Deferred lighting
  - 3 passes
- Intro to another deferred rendering topic

Forward Rendering

- “Forward shading/lighting” path:
- Simply put: perform lighting calculations ***as objects are drawn***.
- Example:



Forward Rendering

- General format of vertex shader:
  - Transform vertex (mandatory → `gl_Position`)
  - Pass attributes down pipeline to FS
- General format of fragment shader:
  - Receive attributes from VS (interpolated)
  - Prepare them for lighting (norm., etc.)
  - Sample relevant textures (diffuse, specular…)
  - Perform shading and lighting calculations (Phong)

Forward Rendering

- PROBLEM???
- Which gets drawn first?
- Why is this inconvenient?

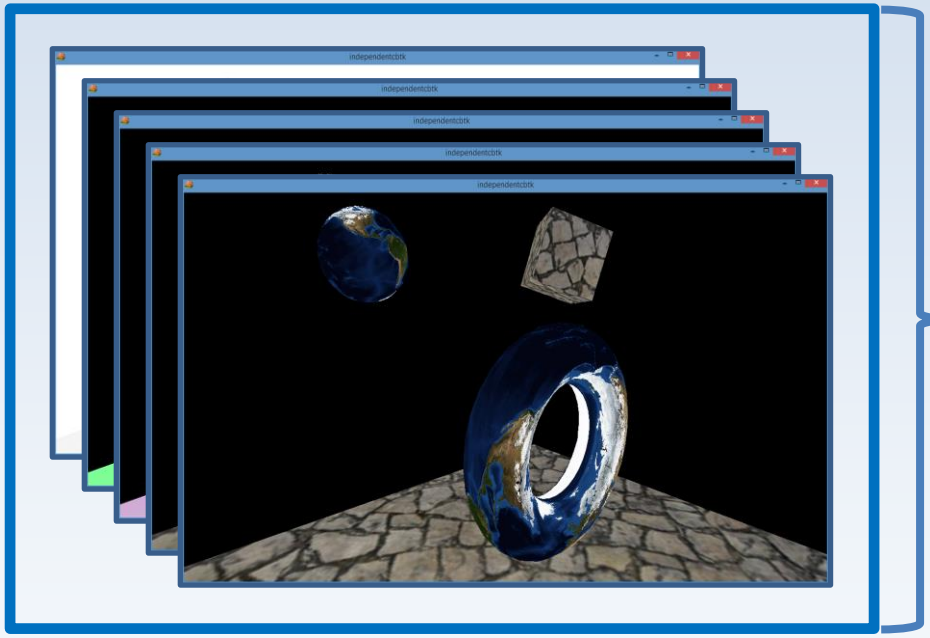
- Depth sorting ***<u>may</u>*** reduce rendering time…
  - No guarantees ☹

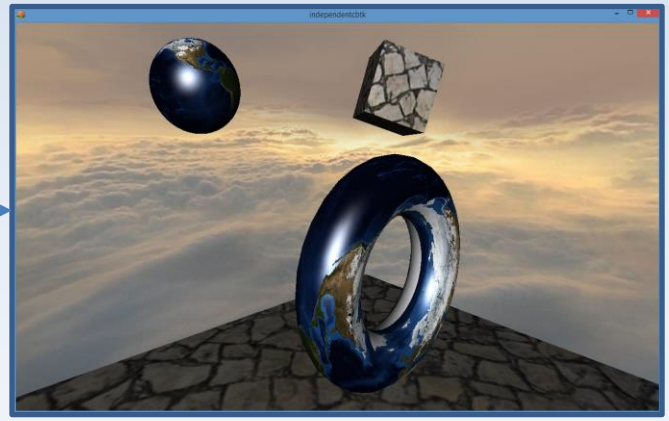
Deferred Shading

- "***Deferred*** rendering" path: as name suggests, we are waiting to do something…
- *Deferred shading*: do not perform shading operations while geometry is drawn!!!
- We split rendering into 2 passes:
- 1) Geometry pass
- 2) Shading pass ← comes after geometry… hence, deferred ☺

Deferred Shading

- Deferred shading pipeline:



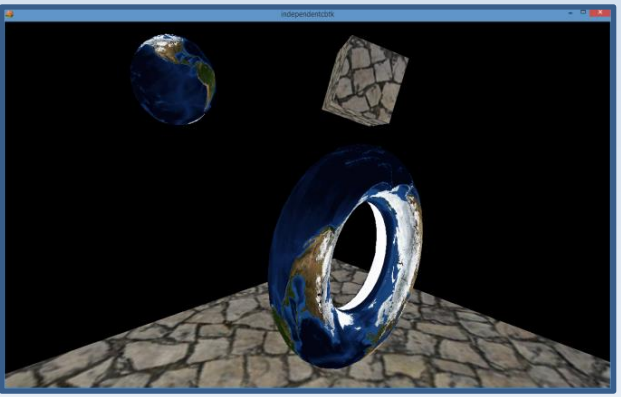
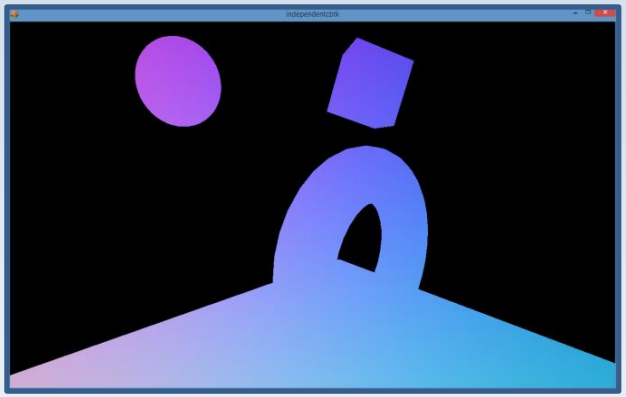


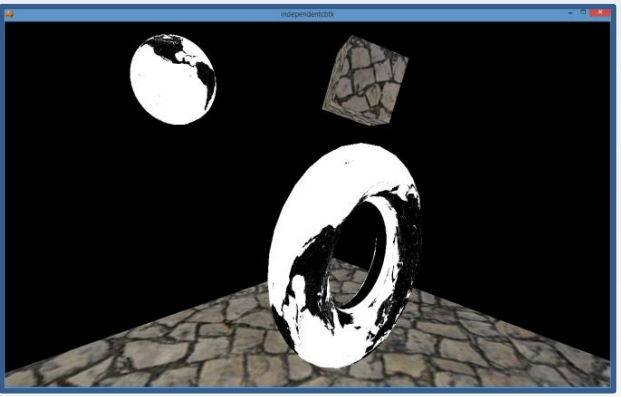
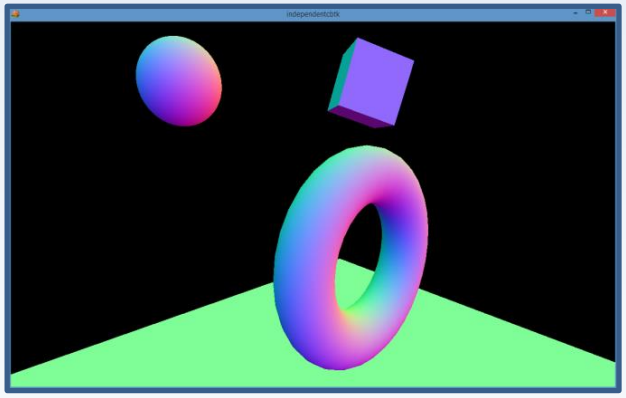
Pass 1: *Geometry*
→ MRT + depth target
→ ***One color target per attribute of interest***

Pass 2: *Deferred Shading*
→ Render FSQ to trigger full-screen post-proc
→ Perform shading operations!

Deferred Shading

- Example output from geometry pass:

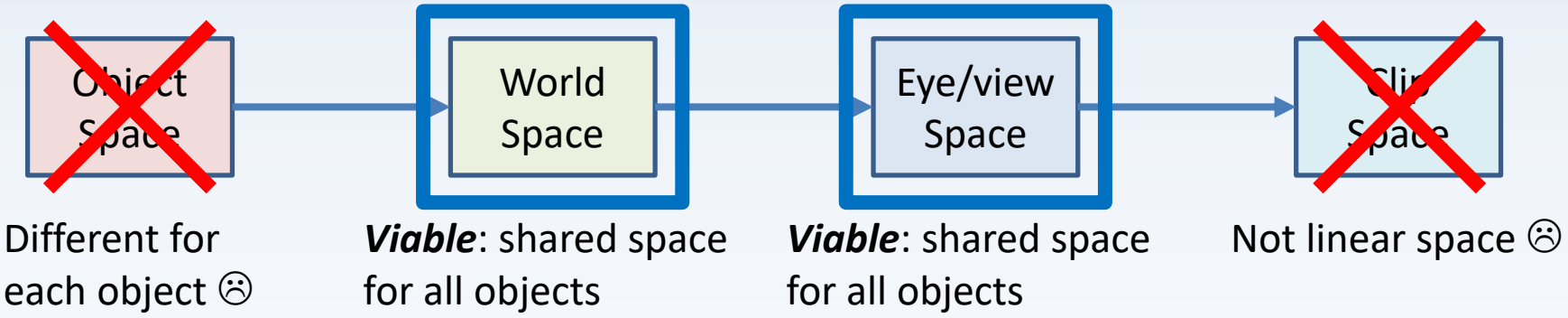



Deferred Shading

- Pass 1: Geometry pass
- Draw geometry to FBO, but do not perform lighting… what do we do instead???
- ***Output raw attributes as color*** ☺
- Each of these color targets is called a "***g-buffer***" (geometry buffer)
- Relevant things to output???

Deferred Shading

- Pass 1: Geometry pass
- ***One key restriction on this pass***
- All ***attributes*** must be output in the ***same space for lighting to work properly later!!!***



Deferred Shading

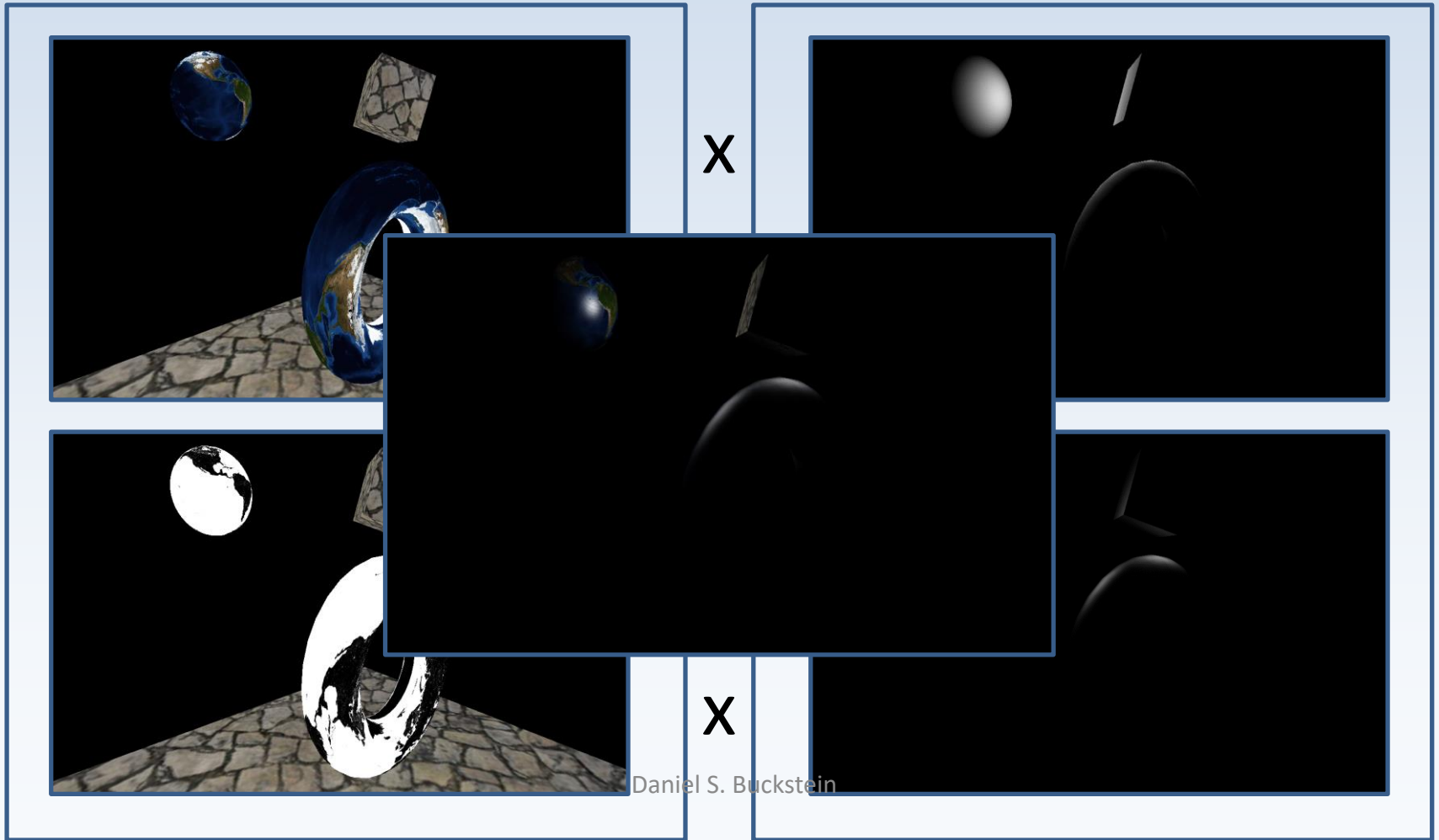
- Pass 2: Shading pass
- We have the scene drawn as a series of g-buffers…
- …now we use it to reconstruct the scene
- Pass in a uniform array of light positions…
- Deserialize normal sample and we’re done, right???

Deferred Shading

- The rest of the process is the same: use attributes to compute everything:
- Lambertian coefficient: ***diffuse shading***
- Phong coefficient: ***specular shading***
- Composite with original *diffuse* and *specular texture samples* respectively
- Phong lighting = sum ☺

Deferred Shading

- Final image = *diffuseCol(kd) + specularCol(ks)*



Deferred Shading

- Summary of deferred shading:
- ***2 passes***:
- 1) Render geometry, store key attributes as textures
- 2) Render ***full-screen quad*** to perform deferred shading
  - Apply result by multiplying by original colours

Deferred Lighting

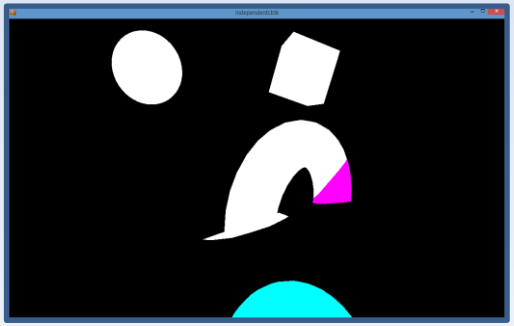
- ***Deferred shading***: one problem…
- ***Full-screen quad***: what are the implications?

- Enter ***deferred lighting***: the *light itself* defines where lighting happens!!!
- ***3 pass algorithm***

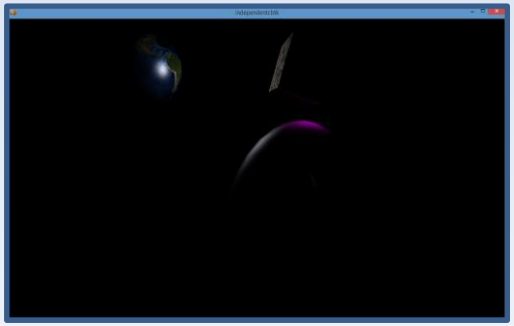
Deferred Lighting

- Deferred lighting pipeline:

Pass 1: *Geometry*
→ MRT + depth target
→ …exactly the same as deferred shading ☺



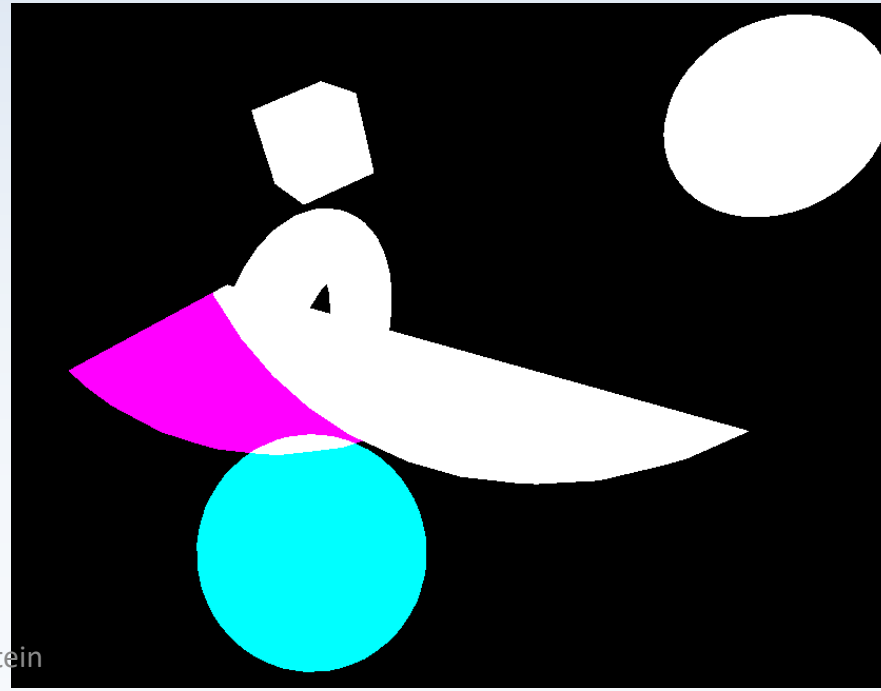
Pass 2: *"Light Pre-pass"*
→ Render lights as actual volumes!



Pass 3: *Deferred lighting*
→ Composite lighting with original colours ☺

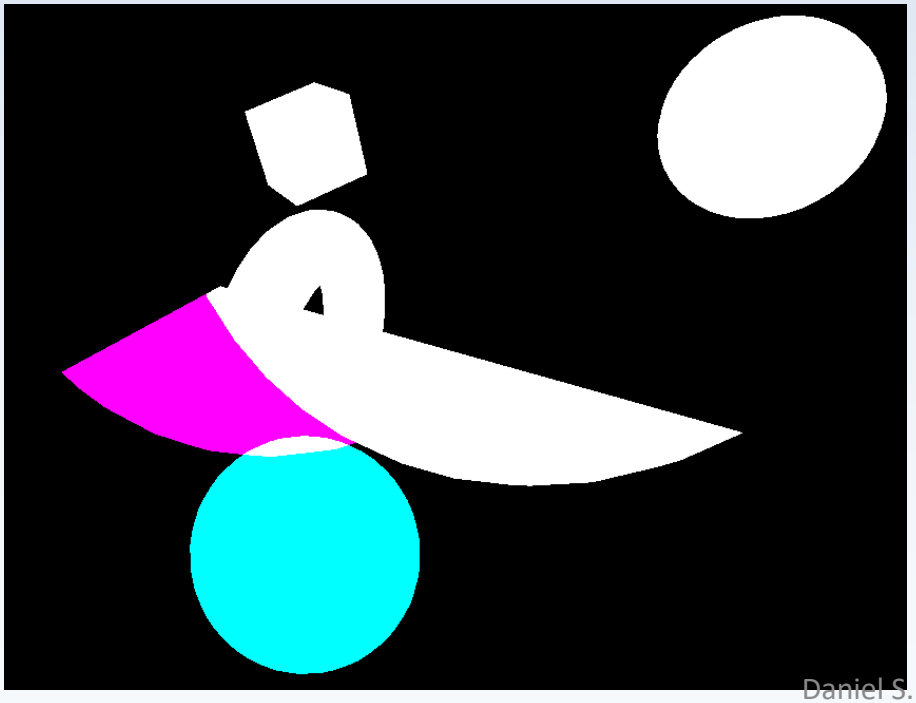
Deferred Lighting

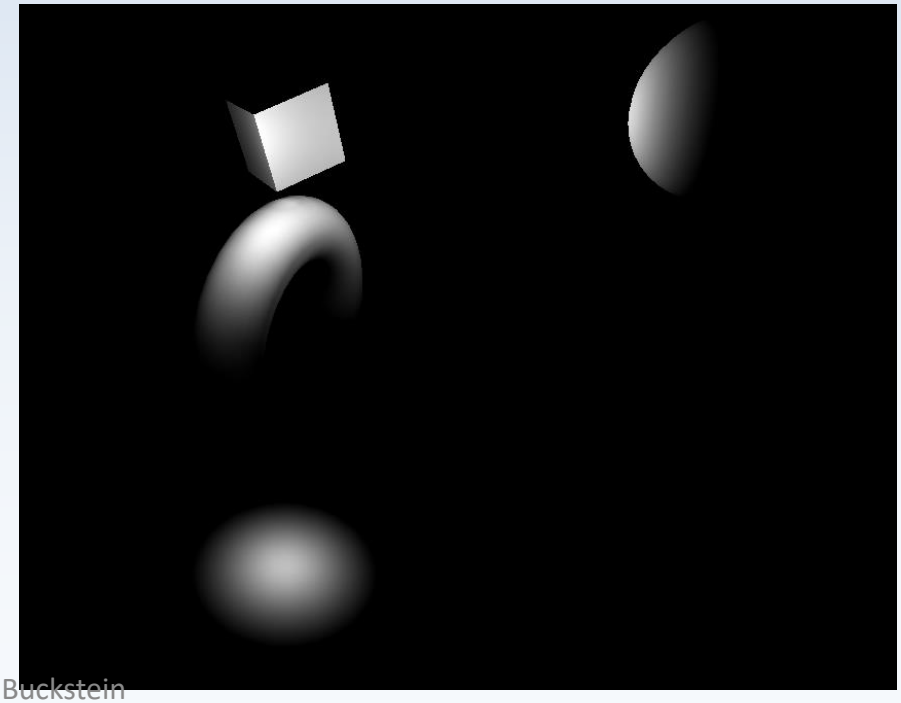
- Pass 1: ***exactly*** the same as deferred shading
- Pass 2: “Light pre-pass”
- Draw light spheres and perform lighting on affected fragments ***only***
- Potentially-shaded fragments are processed



Deferred Lighting

- Allows for easy attenuation and does not assume all fragments are affected by all lights!





Deferred Lighting

- ***ALL of the lights are drawn in the same pass***
- Lighting is processed here
- How do we accumulate the result of multiple lights???
- Additive blending:

```
glBlendFunc (GL_ONE, GL_ONE);
```

1 x new pixel + 1 x old pixel

Deferred Lighting

- Pass 3: Full-screen quad, simple composite!!!
- Multiply ***original diffuse*** and ***specular colour*** by ***respective shading computed in pass 2***
- This is the exact same final step as deferred shading…
- …but lighting has been computed ***only where there is light***

Deferred Lighting

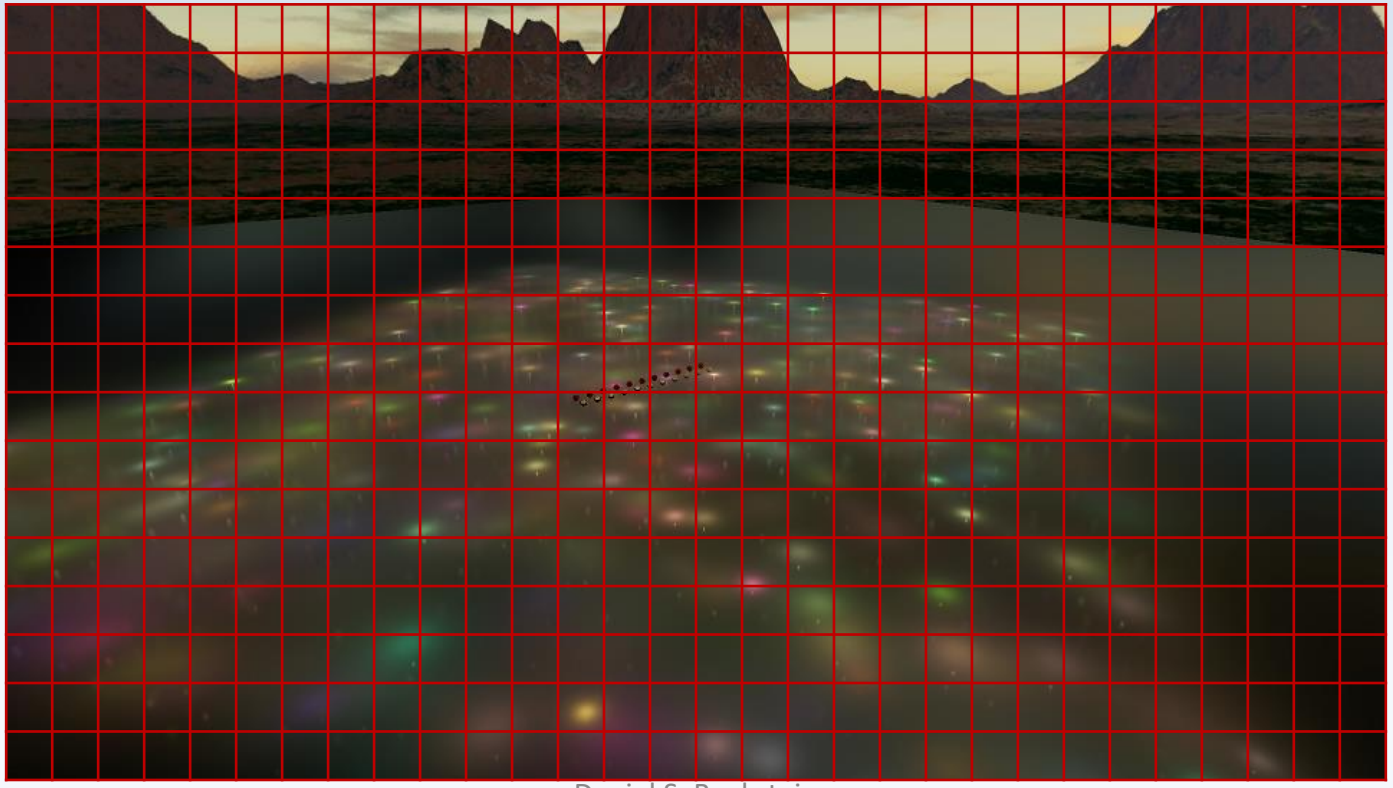
- Summary of deferred lighting:
- ***3 passes***:
- 1) Geometry (same as deferred shading)
- 2) Light pre-pass
- 3) Composite

Deferred Rendering

- Advanced topic in deferred rendering:
- ***Tiled deferred rendering***: takes advantage of another type of shader…
- The ***compute shader***
- Not vertex, not fragment… runs on its own
- Its job is whatever you want it to be in 3 dimensions

Deferred Rendering

- Compute shader breaks image into “work groups”, easier to process many pixels at once!

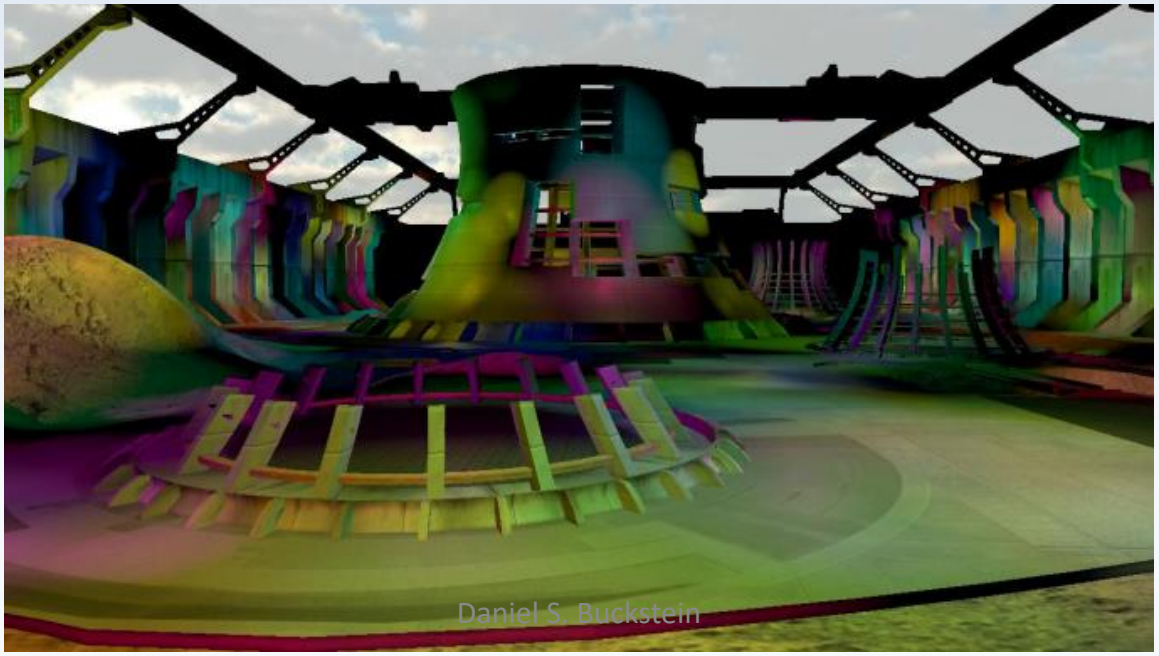


Daniel S. Buckstein

http://leifnode.com/wp-content/uploads/2015/05/TiledDeferredLights.png

Deferred Rendering

- Check out a presentation about this: https://software.intel.com/sites/default/files/m/d/4/1/d/8/lauritzen_deferred_shading_siggraph_2010.pdf



The end.

- Questions? Comments? Concerns?