CSE 167:
Introduction to Computer Graphics
Lecture #20: Deferred Rendering

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2020

Announcements

- Tomorrow, December 9th at 1pm:
  - Discussion Project 4 and Final Exam
- Sunday, December 13th at 11:59pm:
  - Homework Project 4 due
- Thursday, December 17th 2:30pm until Dec 18th 2:30pm
  - Final Exam
  - Timed 3-hour Canvas quiz, to be taken within 24h
- Sunday, December 20thth at 11:59pm:
  - Homework Project 4 late deadline
- Jon Paden (jpaden@ucsd.edu) looking for interns for ARNOC project

Deferred Rendering

- Opposite to Forward Rendering, which is the way we have rendered with OpenGL so far
- Deferred rendering describes post-processing algorithms
  - Requires two-pass rendering
  - First pass:
    - Scene is rendered as usual by projecting 3D primitives to 2D screen space.
    - Additionally, an off-screen buffer (G-buffer) is populated with additional information about the geometry elements at every pixel
      - Examples: normals, diffuse shading color, position, texture coordinates
  - Second pass:
    - An algorithm, typically implemented as a shader, processes the G-buffer to generate the final image in the back buffer

Deferred Shading

- Postpones shading calculations for a fragment until its visibility is completely determined
  - Only visible fragments are shaded
- Algorithm:
  - Fill a set of buffers with common data, such as diffuse texture, normals, material properties
  - Render lights with limited extent and use data from the buffers for the lighting computation
- Advantages:
  - Decouples lighting from geometry rendering
  - Several lights can be applied with a single draw call. E.g., >1000 lights can be rendered at 60 fps
- Disadvantages:
  - More expensive (memory, bandwidth, shader instructions)
- Tutorial:
  - http://gamedevs.org/uploads/deferred-shading-tutorial.pdf

Particle system with glowing particles. Source: Humus 3D

Deferred Lighting

- Video:
  - https://www.youtube.com/watch?v=zOVsxIdANcg

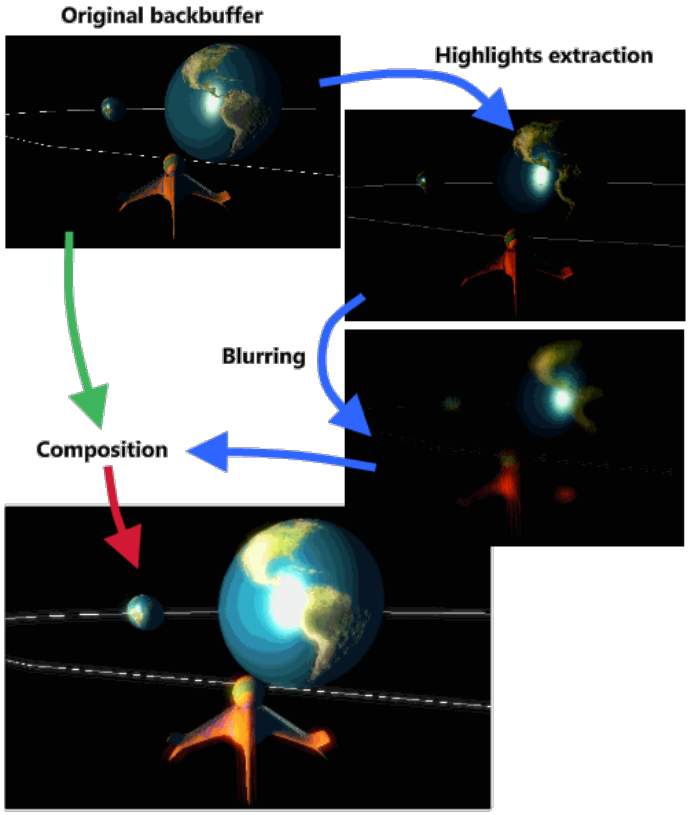
Bloom Effect

Left: no bloom, right: bloom. Source: http://jmonkeyengine.org

- Computer displays have limited dynamic range
- Bloom gives a scene a look of bright lighting and overexposure
- Provides visual cues about brightness and atmosphere
  - Caused by light scattering in atmosphere, or within our eyes

Bloom Shader

- Step 1: Extract all highlights of the rendered scene, superimpose them and make them more intense
  - Operates on G-buffer
  - Often done with G-buffer smaller (lower resolution) than frame buffer
  - Highlights found by thresholding luminance
- Step 2: Blur off-screen buffer, e.g., using Gaussian blur
- Step 3: Composite off-screen buffer with back buffer



Bloom shader render steps.
Source: http://www.klopfenstein.net

Glow vs. Bloom

- Bloom filter looks for highlights automatically, based on a threshold value
- If you want to have more control over what glows and does not glow, a glow filter is needed
- Glow filter adds an additional step to Bloom filter: instead of thresholding, only the glowing objects are rendered
- Render passes:
  - Render entire scene back buffer
  - Render only glowing objects to a smaller off-screen glow buffer
  - Apply a bloom pixel shader to glow buffer
  - Compose back buffer and glow buffer together

Video: Glowing Lava

- https://www.youtube.com/watch?v=hmsMk-skqul

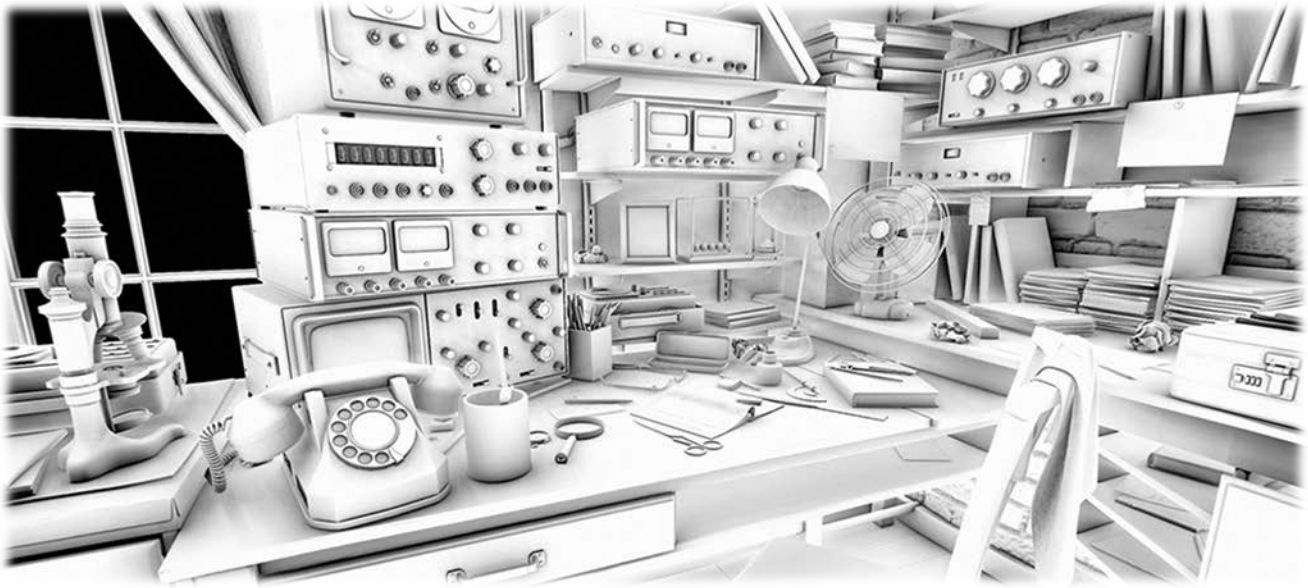
References

- Deferred Shading
  - https://learnopengl.com/Advanced-Lighting/Deferred-Shading
- Bloom Tutorial
  - http://prideout.net/archive/bloom/
- GPU Gems Chapter on Glow
  - http://developer.download.nvidia.com/books/HTML/gpugems/gpugems_ch21.html
- GLSL Shader for Gaussian Blur
  - http://www.ozone3d.net/tutorials/image_filtering_p2.php

Screen Space Ambient Occlusion

Screen Space Ambient Occlusion (SSAO)

- "Screen Space" → deferred rendering approach
- Approximates ambient occlusion in real time
- Developed by Vladimir Kajalin (Crytek)
- First use in PC game Crysis (2007)



SSAO component

Ambient Occlusion

- Crude approximation of global illumination
- Often referred to as "sky light"
- Global method (not local like Phong shading)
  - Illumination at each point is a function of other geometry in the scene
- Appearance is similar to what objects appear as on an overcast day
  - Assumption: concave objects are hit by less light than convex ones

Basic SSAO Algorithm

- First pass:
  - Render scene normally and write z values to G-buffer's alpha channel
- Second pass:
  - Pixel shader samples depth values around the processed fragment and computes amount of occlusion, stores result in red channel
  - Occlusion depends on depth difference between sampled fragment and currently processed fragment

Ambient occlusion values in red color channel
Source: www.gamerendering.com

SSAO With Normals

- First pass:
  - Render scene normally and copy z values to G-buffer's alpha channel and scene normals to RGB channels
- Second pass:
  - Use normals and z-values to compute occlusion between current pixel and several samples around that pixel

No SSAO

With SSAO

SSAO Discussion

- Advantages:
  - Deferred rendering algorithm: independent of scene complexity
  - No pre-processing, no memory allocation in RAM
  - Works with dynamic scenes
  - Works in the same way for every pixel
  - No CPU usage: executed completely on GPU
- Disadvantages:
  - Local and view-dependent (dependent on adjacent texel depths)
  - Hard to correctly smooth/blur out noise without interfering with depth discontinuities, such as object edges, which should not be smoothed out

SSAO References

- Nvidia's documentation
  - http://developer.download.nvidia.com/SDK/10.5/direct3d/Source/ScreenSpaceAO/doc/ScreenSpaceAO.pdf

Deferred Rendering in "Uncharted 4"

- Naughty Dog, Inc. built the game on a deferred rendering system. Using some extensions on the algorithm we described in this lecture, they supported materials like glass, fabric, wood, rock, water, and metal.
  - https://www.youtube.com/watch?v=hh5HV4iic1Y
- Source: "Advances in Real-Time Rendering in Games" as part of SIGGRAPH 2016
  - http://advances.realtimerendering.com