CSE 167:
Introduction to Computer Graphics
Lecture #18: Deferred Rendering

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Spring Quarter 2015

Announcements

- 3rd final project blog due Sunday
- Final Project due Tuesday
  - Presentations start at 9am in CSE 1202
- TA Evaluation
- CAPE

Lecture Overview

- Deferred Rendering Techniques
  - Deferred Shading
  - Screen Space Ambient Occlusion
  - Bloom
  - Glow

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

Lecture Overview

- Deferred Rendering Techniques
  - Deferred Shading
  - Screen Space Ambient Occlusion
  - Bloom
  - Glow
- The Future of Computer Graphics

Deferred Shading

- Postpones shading calculations for a fragment until its visibility is completely determined
  - Only fragments that really contribute to the image are shaded
- Algorithm:
  - Fill a set of buffers with common data, such as diffuse texture, normals, material properties
  - For the lighting just render the light extents and fetch data from these buffers for the lighting computation
- Advantages:
  - Decouples lighting from geometry
  - Several lights can be applied with a single draw call: more than 1000 light sources can be rendered at 60 fps
- Disadvantages:
  - Consumes more memory, bandwidth and shader instructions than traditional rendering

Particle system with glowing particles. Source: Humus 3D

Reference

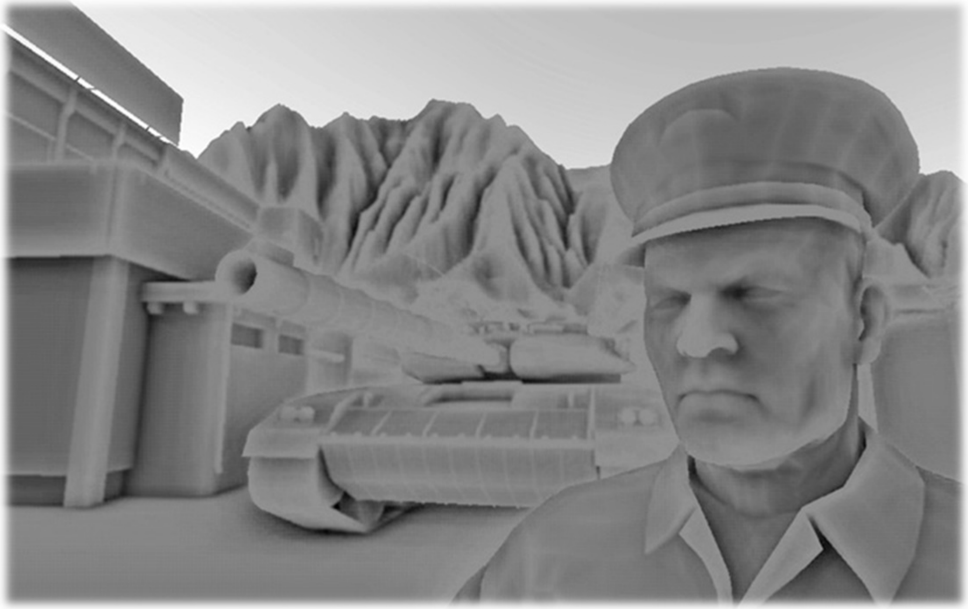
- Deferred Shading Tutorial:
  - http://gamedevs.org/uploads/deferred-shading-tutorial.pdf

Lecture Overview

- Deferred Rendering Techniques
  - Deferred Shading
  - Screen Space Ambient Occlusion
  - Bloom
  - Glow
- The Future of Computer Graphics

Screen Space Ambient Occlusion

- Screen Space Ambient Occlusion is abbreviated as SSAO
- "Screen Space" refers to this being a deferred rendering approach
- Rendering technique for approximating ambient occlusion in real time
- Developed by Vladimir Kajalin while working at Crytek
- First use in 2007 PC game Crysis



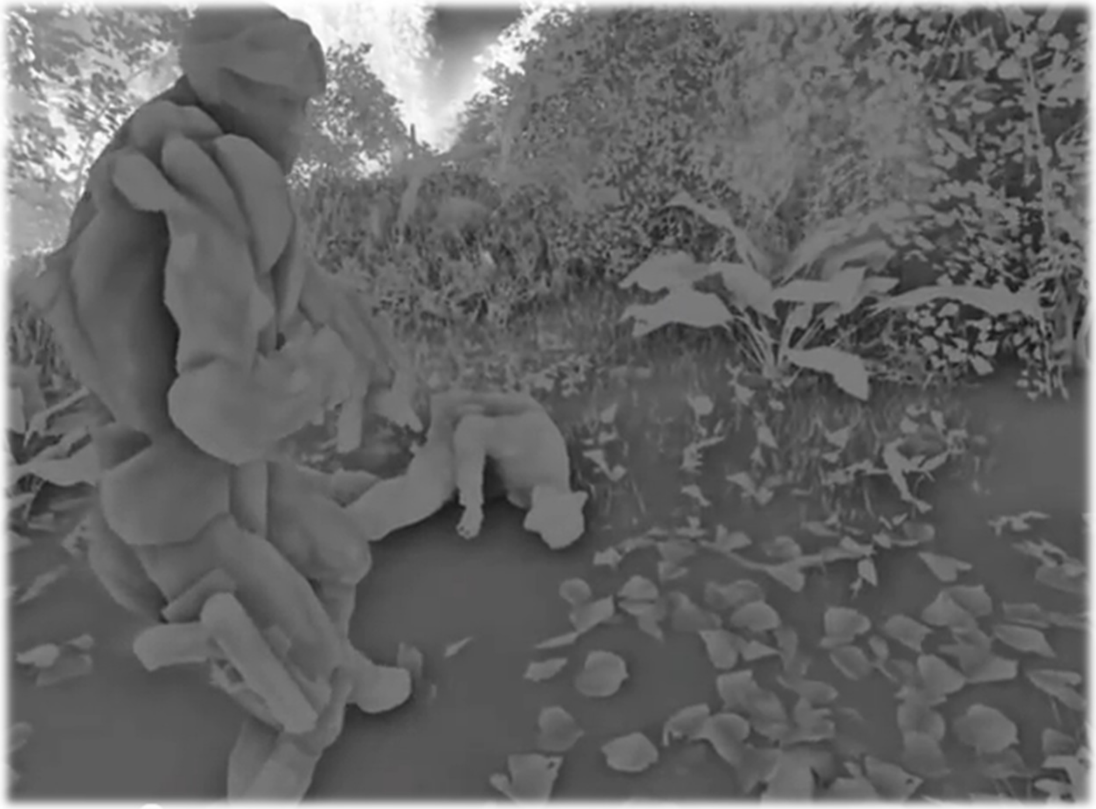
SSAO component

Ambient Occlusion

- Attempts to approximate global illumination
  - Very crude approximation
- Unlike local methods like Phong shading, ambient occlusion is a global method
  - Illumination at each point is a function of other geometry in the scene
- Appearance achieved by ambient occlusion is similar to the way an object appears on an overcast day
  - Example: arm pit is hit by a lot less light than top of head
- In the industry, ambient occlusion is often referred to as "sky light"

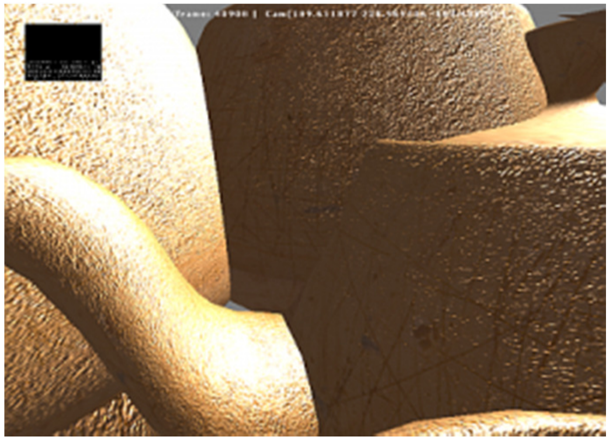
SSAO Demo

- Screen Space Ambient Occlusion (SSAO) in Crysis
  - http://www.youtube.com/watch?v=ifdAILHTcZk

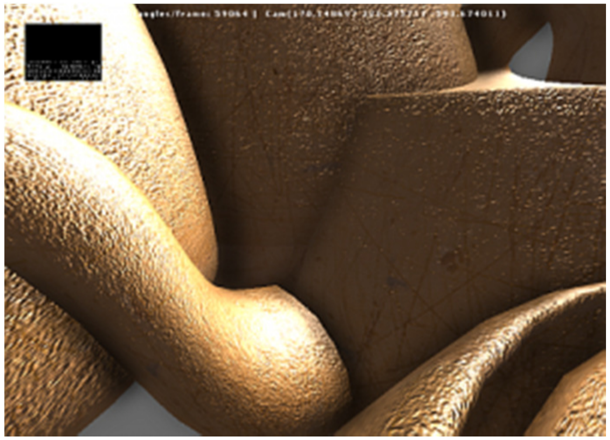


SSAO With Normals

- First pass:
  - Render scene normally and copy z values to g-buffer's alpha channel and scene normals to g-buffer's 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

UCSD

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

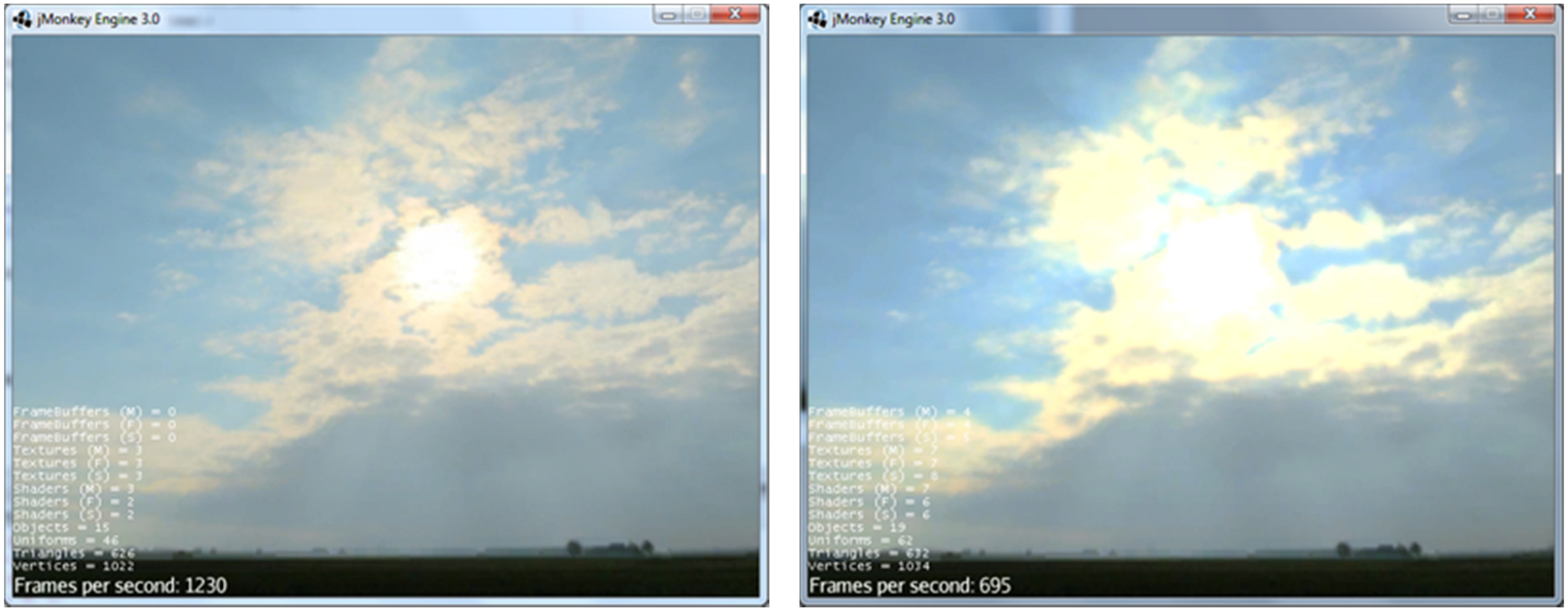
References

- Nvidia’s documentation:
  - http://developer.download.nvidia.com/SDK/10.5/direct3d/Source/ScreenSpaceAO/doc/ScreenSpaceAO.pdf
- SSAO shader code from Crysis:
  - http://69.163.227.177/forum.php?mod=viewthread&tid=772
- Another implementation:
  - http://www.gamerendering.com/2009/01/14/ssao/

Lecture Overview

- Deferred Rendering Techniques
  - Deferred Shading
  - Screen Space Ambient Occlusion
  - Bloom
  - Glow
- The Future of Computer Graphics

Bloom Effect



```
FrameBuffers (M) = 0
FrameBuffers (F) = 0
FrameBuffers (S) = 0
Textures (M) = 3
Textures (F) = 3
Textures (S) = 3
Shaders (M) = 3
Shaders (F) = 2
Shaders (S) = 2
Objects = 15
Uniforms = 46
Triangles = 626
Vertices = 1022
Frames per second: 1230
```

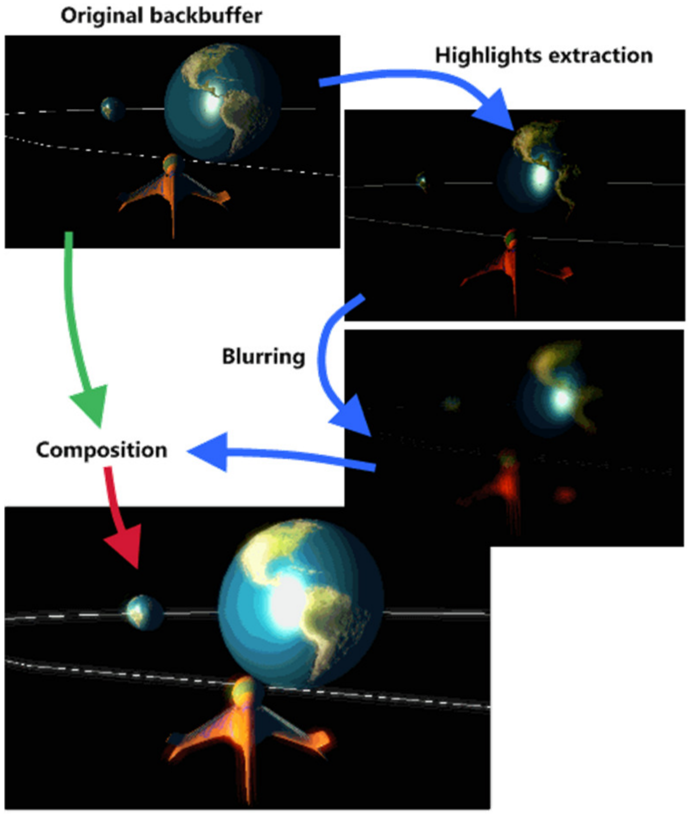
```
FrameBuffers (M) = 4
FrameBuffers (F) = 4
FrameBuffers (S) = 5
Textures (M) = 7
Textures (F) = 7
Textures (S) = 8
Shaders (M) = 7
Shaders (F) = 6
Shaders (S) = 6
Objects = 19
Uniforms = 62
Triangles = 632
Vertices = 1034
Frames per second: 695
```

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

- Bloom gives a scene a look of bright lighting and overexposure

Bloom Shader

- Post-processing filter: applied after scene is rendered normally
- Step 1: Extract all highlights of the rendered scene, superimpose them and make them more intense
  - Operates on back buffer
  - Often done with off-screen buffer smaller than frame buffer
  - Highlights found by thresholding luminance
- Step 2: Blur off-screen buffer, e.g., with Gaussian blurring
- Step 3: Composite off-screen buffer with back buffer



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

References

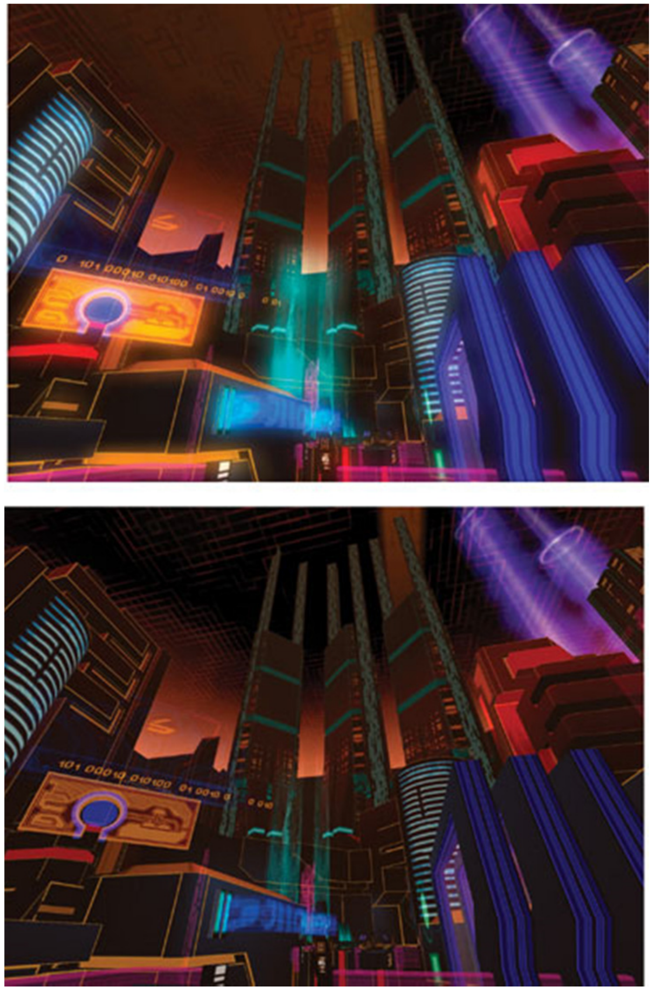
- Bloom Shader
  - http://www.klopfenstein.net/lorenz.aspx/gamecomponents-the-bloom-post-processing-filter
- GLSL Shader for Gaussian Blur
  - http://www.ozone3d.net/tutorials/image_filtering_p2.php

Lecture Overview

- Deferred Rendering Techniques
  - Deferred Shading
  - Screen Space Ambient Occlusion
  - Bloom
  - Glow
- The Future of Computer Graphics

Glow Effects

- Glows and halos of light appear everywhere in the world
- They provide powerful visual cues about brightness and atmosphere
- In computer graphics, the intensity of light reaching the eye is limited, so the only way to distinguish intense sources of light is by their surrounding glow and halos
- In everyday life, glows and halos are caused by light scattering in the atmosphere or within our eyes



A cityscape with and without glow.
Source: GPU Gems

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 modifies the thresholding step of the Bloom filter: only the glowing objects are rendered
- Render passes:
  - Render entire scene to the 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
- Glow example:
  - https://www.youtube.com/watch?v=kDOFM9Rj5dY

References

- GPU Gems Chapter on Glow
  - http://http.developer.nvidia.com/GPUGems/gpugems_ch21.html
- Bloom and Glow
  - http://jmonkeyengine.org/wiki/doku.php/jme3:advanced:bloom_and_glow

The Future of Computer Graphics

- ACM SIGGRAPH 2015 Technical Papers (3:20)
  - https://www.youtube.com/watch?v=XrYkEhs2FdA
- SIGGRAPH 2015 - Computer Animation Festival Trailer (3:28)
  - https://www.youtube.com/watch?v=UH-mdAdT1BI
- Cryengine on Steam (2:41)
  - http://store.steampowered.com/app/220980/
- Top 5 Best Next Gen Game Engines Of The Future (12:16)
  - https://www.youtube.com/watch?v=vTF7wz0-AQs
- The Centrifuge Brain Project, 2013 (6:35)
  - https://www.youtube.com/watch?v=RVeHxUVkW4w

Good luck with your final projects!