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

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

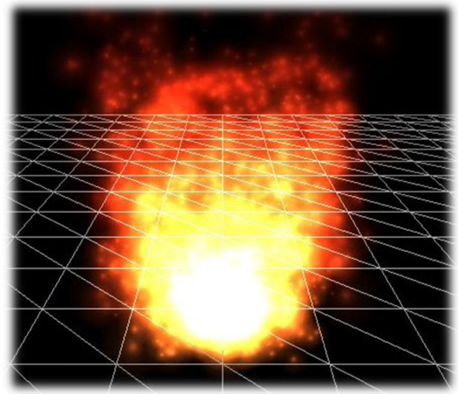
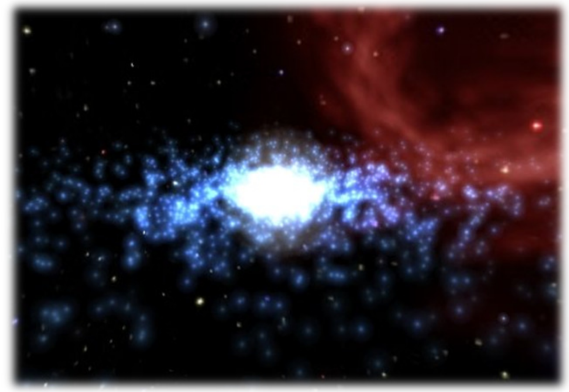
Announcements

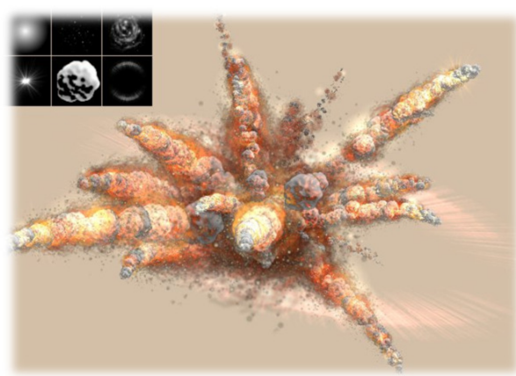
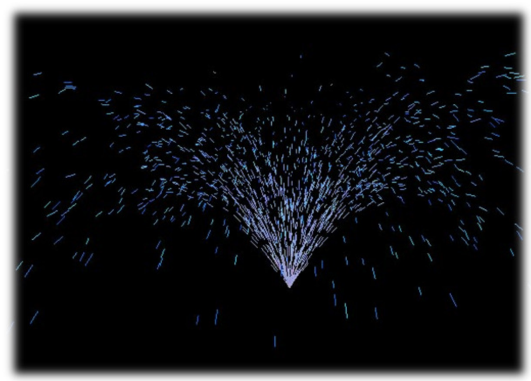
Lecture Overview

- Particle Systems
- Collision Detection
- Deferred Rendering

Particle Systems

- Used for:
  - Fire/sparks
  - Rain/snow
  - Water spray
  - Explosions
  - Galaxies

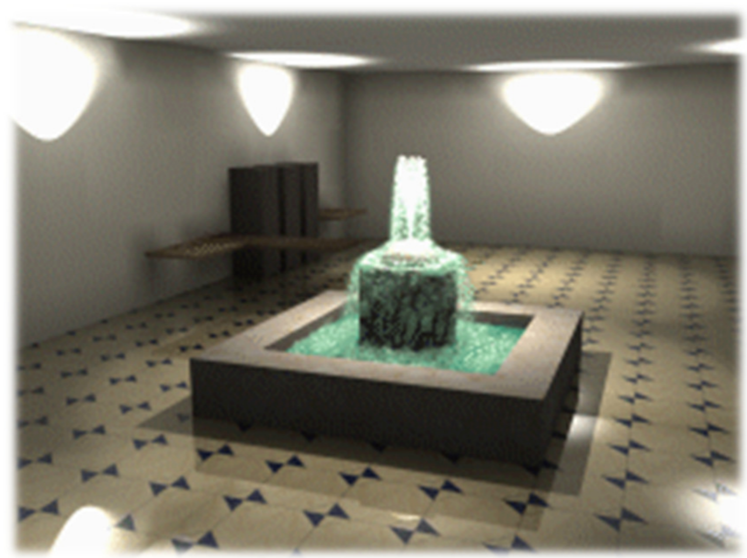
UCSD

Internal Representation

- Particle system is collection of a number of individual elements (particles)
  - Controls a set of particles which act autonomously but share some common attributes
- Particle Emitter: Source of all new particles
  - 3D point
  - Polygon mesh: particles' initial velocity vector is normal to surface
- Particle attributes:
  - position (3D)
  - velocity (vector: speed and direction)
  - color + opacity
  - lifetime
  - size
  - shape
  - weight

Dynamic Updates

- Particles change position and/or attributes with time
- Initial particle attributes often created with random numbers
- Frame update:
  - Parameters: simulation of particles, can include collisions with geometry
    - Forces (gravity, wind, etc) accelerate a particle
    - Acceleration changes velocity
    - Velocity changes position
  - Rendering: display as
    - OpenGL points
    - (Textured) billboarded quads
    - Point sprites



Source: http://www.particlesystems.org/

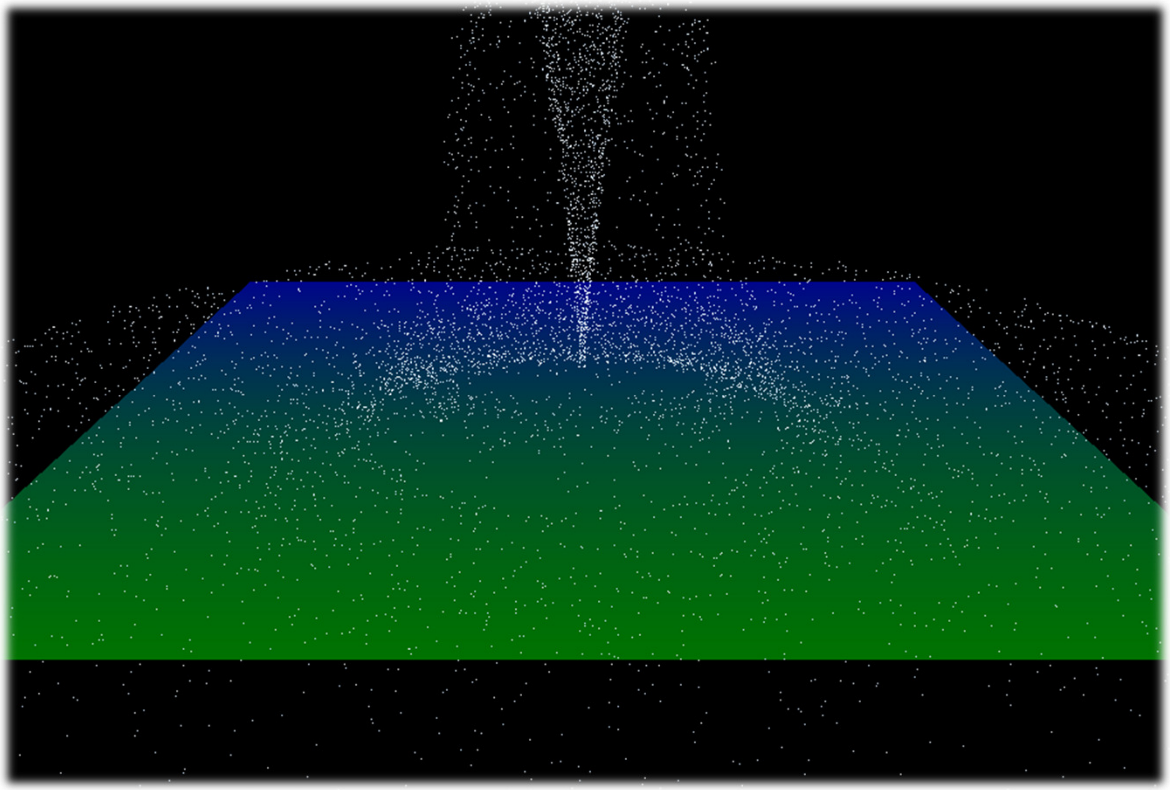
Point Sprite

- **Screen-aligned element of variable size**
- **Defined by single point**
- **Sample code:**

```
glTexEnvf(GL_POINT_SPRITE, GL_COORD_REPLACE, GL_TRUE);
glEnable(GL_POINT_SPRITE);
glBegin(GL_POINTS);
  glVertex3f(position.x, position.y, position.z);
glEnd();
glDisable(GL_POINT_SPRITE);
```

Demo

- Demo software by Prof. David McAllister:
  - http://www.calit2.net/~jschulze/tmp/Particle221Demos.zip



References

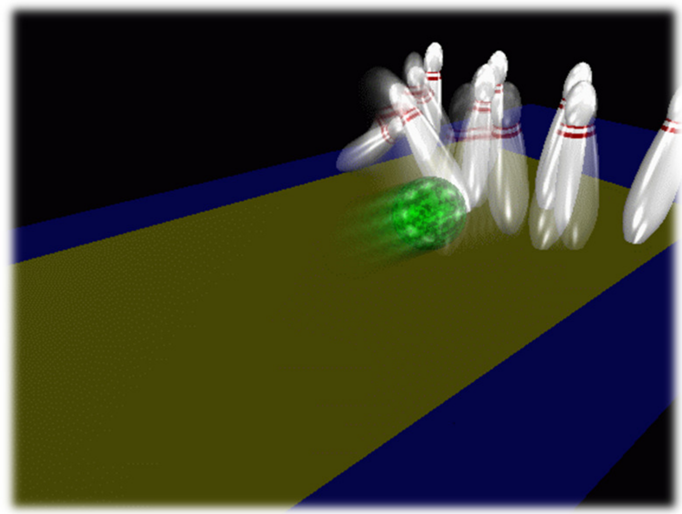
- Tutorial with source code by Bartlomiej Filipek, 2014:
  - http://www.codeproject.com/Articles/795065/Flexible-particle-system-OpenGL-Renderer
- Articles with source code:
  - Jeff Lander: "The Ocean Spray in Your Face", Game Developer, July 1998
    - http://www.darwin3d.com/gamedev/articles/col0798.pdf
  - John Van Der Burg: "Building an Advanced Particle System", Gamasutra, June 2000
    - http://www.gamasutra.com/view/feature/3157/building_an_advanced_particle_.php
- Founding scientific paper:
  - Reeves: "Particle Systems - A Technique for Modeling a Class of Fuzzy Objects", ACM Transactions on Graphics (TOG) Volume 2 Issue 2, April 1983
    - http://zach.in.tu-clausthal.de/teaching/vr_literatur/Reeves%20-%20Particle%20Systems.pdf

Lecture Overview

- Particle Systems
- Collision Detection
- Deferred Rendering

Collision Detection

- Goals:
  - Physically correct simulation of collision of objects
    - Not covered here
  - Determine if two objects intersect
- Slow calculation because of exponential growth $O(n^2)$:
  - # collision tests = n*(n-1)/2



UCSD

Intersection Testing

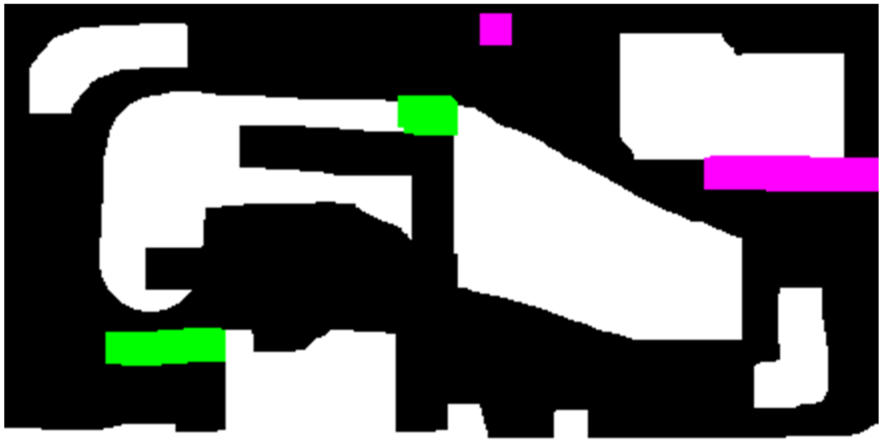
- Purpose:
  - Keep moving objects on the ground
  - Keep moving objects from going through walls, each other, etc.
- Goal:
  - Believable system, does not have to be physically correct
- Priority:
  - Computationally inexpensive
- Typical approach:
  - Spatial partitioning
  - Object simplified for collision detection by one or a few
    - Points
    - Spheres
    - Axis aligned bounding box (AABB)
  - Pairwise checks between points/spheres/AABBs and static geometry

Sweep and Prune Algorithm

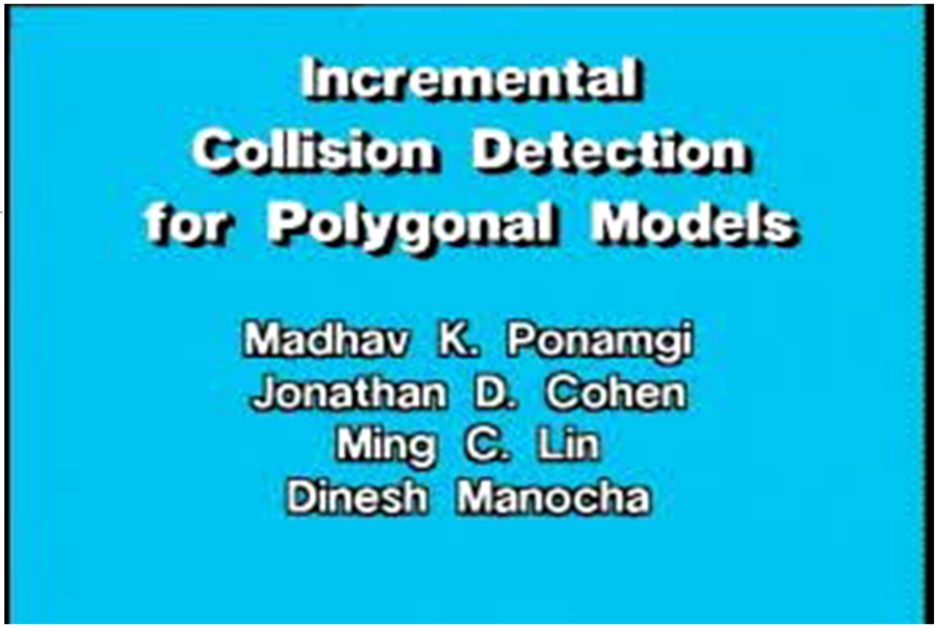
- Sorts bounding boxes
- Not intuitively obvious how to sort bounding boxes in 3-space
- Dimension reduction approach:
  - Project each 3-dimensional bounding box onto the x,y and z axes
  - Find overlaps in 1D: a pair of bounding boxes can overlap if and only if their intervals overlap in all three dimensions
    - Construct 3 lists, one for each dimension
    - Each list contains start/end point of intervals corresponding to that dimension
    - By sorting these lists, we can determine which intervals overlap
    - Reduce sorting time by keeping sorted lists from previous frame, changing only the interval endpoints
- Alternative: project bounding boxes onto coordinate axis planes and look for overlaps in 2D

Collision Map (CM)

- 2D map with information about where objects can go and what happens when they go there
- Colors indicate different types of locations
- Map can be computed from 3D model, or hand drawn with paint program
- Granularity: defines how much area (in object space) one CM pixel represents



References



- I-Collide:
  - Interactive and exact collision detection library for large environments composed of convex polyhedra
    - http://gamma.cs.unc.edu/I-COLLIDE/
- OZ Collide:
  - Fast, complete and free collision detection library in C++
  - Based on AABB tree
    - http://www.tsarevitch.org/ozcollide/

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

UCSD

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

UCSD

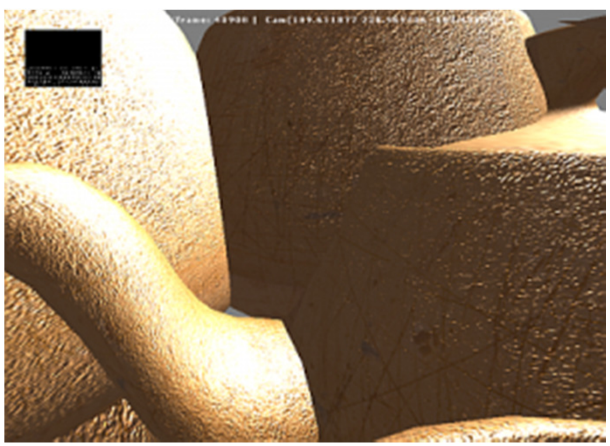
Basic SSAO Algorithm

- 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

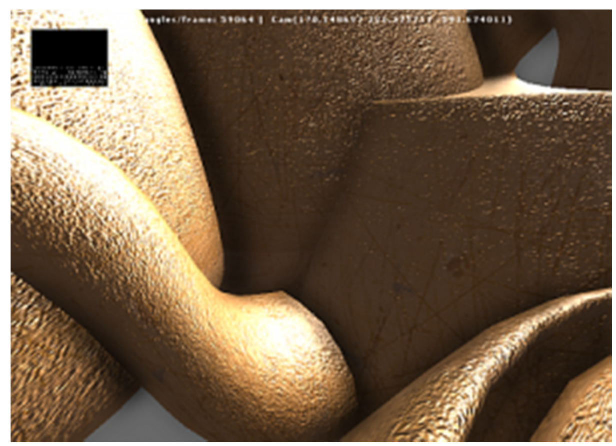
Ambient occlusion values in red color channel
Source: www.gamerendering.com

SSAO With Normals

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

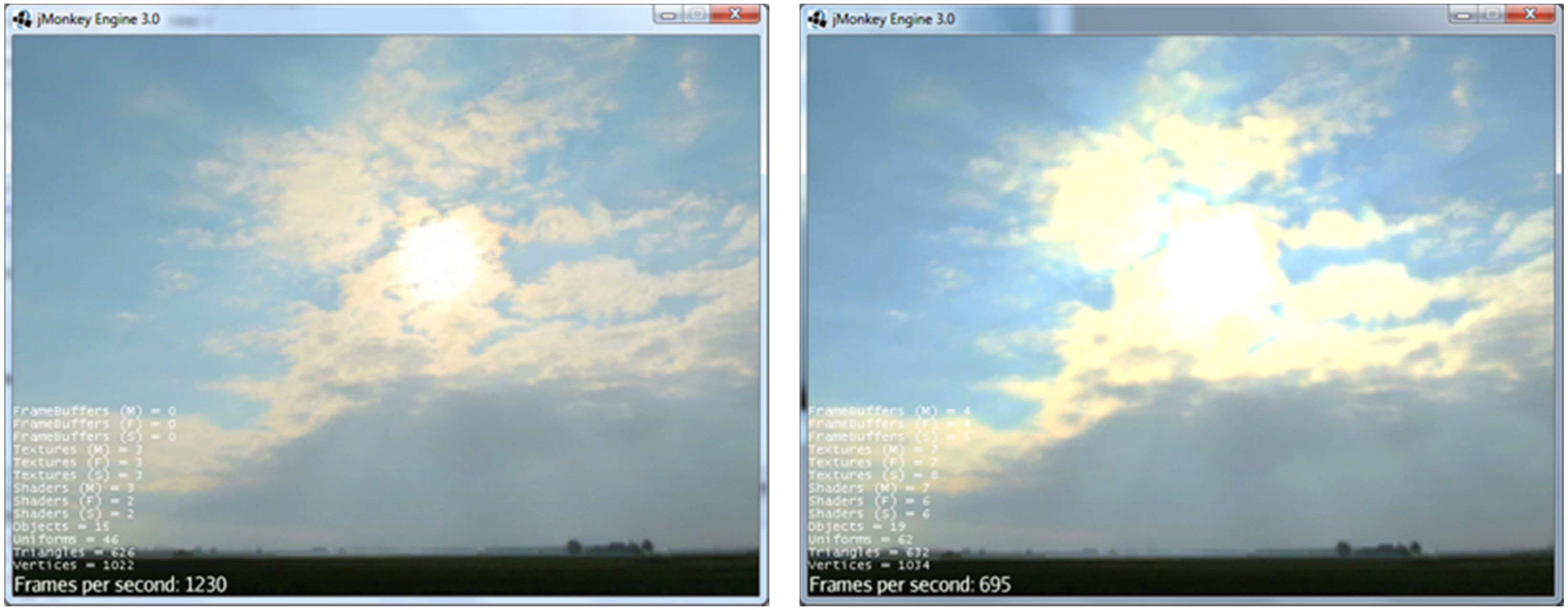
References

- 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



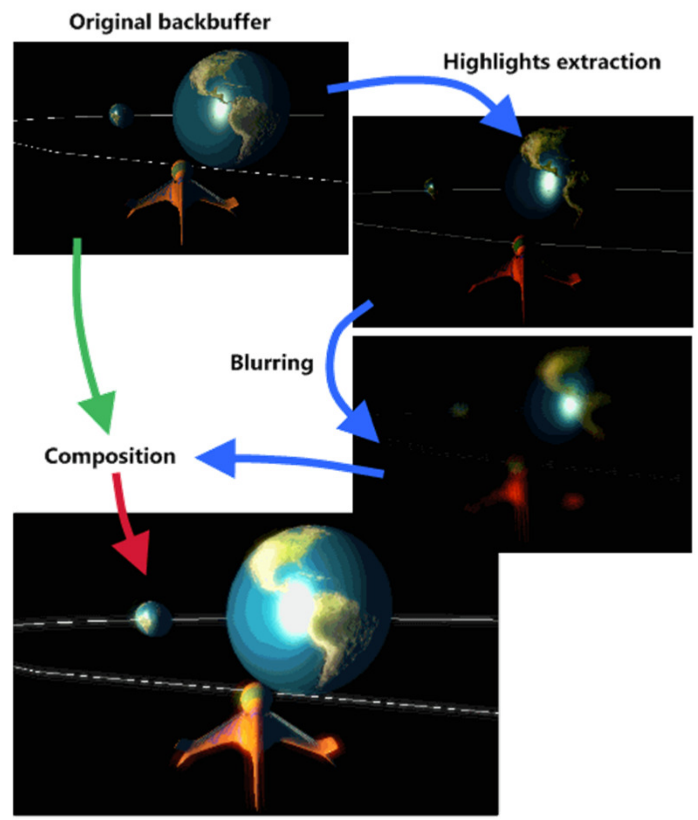
Left: no bloom, right: bloom.
Source: http://jmonkeyengine.org

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

UCSD

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

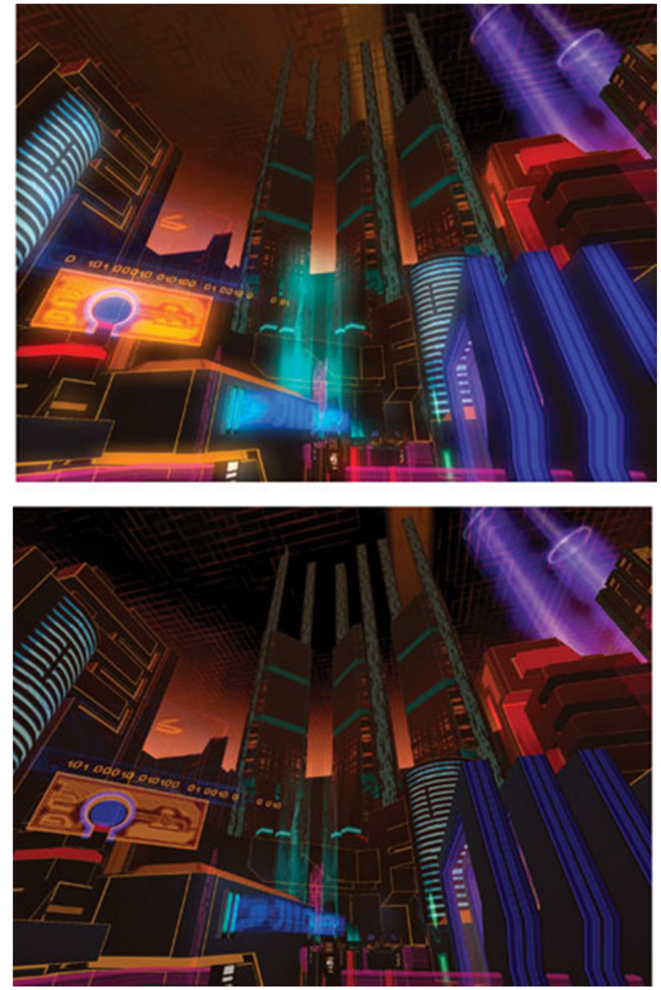
- Bloom Shader
  - http://www.klopfenstein.net/lorenz.aspx/gamecomponents-the-bloom-post-processing-filter
- GLSL Shader for Gaussian Blur
  - http://www.ozone3d.net/tutorials/image_filtering_p2.php

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 steop 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

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 Asia, Dec 3-6, 2014 in Shenzen/China (2:58)
  - https://www.youtube.com/watch?v=s8lzXMWMngU
- Cryengine 4 Trailer, 2013 (3:02)
  - https://www.youtube.com/watch?v=aseq4T81P7g

- The Centrifuge Brain Project, 2013 (6:35)
  - https://www.youtube.com/watch?v=RVeHxUVkW4w

Good luck with your final projects!