Afton Geil & Calina Copos
**Project #3**
EEC277 02/21/2012

# Project Proposal

## Voxel-Based Screen Space Ambient Occlusion

### 1. Background

One of the biggest challenges in photorealistic rendering is calculating realistic lighting effects. Real world lighting is created by the interaction of photons with objects. We perceive an object when photons emitted from a light source hit an object, bounce off, then reach our eyes. Ray-tracing methods most closely approximate this by tracing the path from each point in the scene back to the camera. This results in very realistic-looking scenes, but it is very slow. In real-time computer graphics, lights are generally modeled either as directional or ambient. Directional lighting travels in one direction and originates at some source that may be within the field of view or a a distant point outside of the field of view. Ambient light accounts for the many times that light bounces off different objects so that objects that do not directly face the light source are still illuminated. Ambient lighting is generally modeled as isotropic lighting that originates at every point in space. For directional lighting, shadows are used to darken the areas that are blocked from interaction with the light. When objects are close to each other, they can also block some of the ambient light. Ambient occlusion accounts for this effect, resulting in a much more realistic scene.

The basic principle behind ambient occlusion methods is to look at each point and determine what fraction of the surrounding volume is filled. The point is then darkened by an amount proportional to this value. Computing this volumetric integral can be rather complicated; however, we believe voxelization can simplify this calculation. Voxelization involves rendering a scene by breaking all of the objects up into volumetric elements, called voxels, rather the using triangle meshes to approximate the shape of an object. Approximating the integral as a sum of these elements should simplify the volume calculations, hopefully resulting in a faster overall rendering of the scene.

## 2. Related Work

Ambient occlusion techniques are grouped into two types: object-space methods, which operate over all points, and screen-space methods, which operate only over points that are visible in the final image. For complex scenes, object-space methods are very expensive and are not practical for real-time applications. The first implementation of screen-space ambient occlusion was in *Crysis*[1]. In this implementation, the Z-buffer was the only input needed to calculate the ambient occlusion. For each pixel in the scene, surrounding pixels are tested to determine whether they are in front of the pixel in question. The occlusion values are then weighted by their distance from the pixel in question, then added to produce the final ambient occlusion value.

McGuire [2] computes what he calls "ambient occlusion volumes" by forming a bounding volume of polygons around the pixel and combining the contributions of all fragments within this volume using an integral approximation developed by Baum, et al [3]. McGuire achieves visually appealing results with his algorithm, but the computation cannot be accomplished in real-time. The work most relevant to our proposed work is the work by Penmatsa and Wyman [4]. They compute ambient occlusion using voxels and sum up the contribution of each voxel face. They also measure the curvature of objects and use this for multiresolution effects, which they implement using mipmaps. Their solution runs at 40fps, but the quality of the images diverges considerably from ground truth. This is probably because the ambient occlusion integral approximation they implement is inaccurate, as shown in [3].

## 3. Goals

This project aims to introduce a new approximation algorithm for the voxel-based screen space ambient occlusion problem. It combines previous work to achieve substantially improved quality over fast methods and substantially improved performance compared to more accurate methods. Our **goals** for this project are to:

- implement known work in voxelization to obtain a voxelized representation of the scene;
- compute a bounding occlusion volume per screen pixel dependent on user input
- analytically determine the amount of ambient occlusion per screen pixel due to neighboring voxels within bounding occlusion volume

The implementation of computing ambient occlusion with be done as a CUDA program while voxelization will be implemented using VoxelPipe [5]. The main contribution of this project is an efficient algorithm for estimating ambient occlusion using an analytic solution to the integral form of the full matrix radiosity method:

$$AO_P(\hat{N}) = \frac{1}{2\pi} \sum_{i=0}^{n-1} \hat{N} \cdot \arccos\left(\frac{p_i \cdot p_j}{\|p_i\| \cdot \|p_j\|}\right) \frac{p_i \times p_j}{\|p_i \times p_j\|}$$

where $j = (i+1) \mod n$. Here, $p_0, p_1, ..., p_{n-1}$ denote the three-dimensional vertices of a polygon $P$ and $\hat{N}$ is the normal associated with an infinitesimal smooth patch with center at the origin. The corrected integral approximation was first introduced by [3] and later applied in another ambient occlusion algorithm by [2].

**Preliminary ambient occlusion algorithm.**

(1) Set occlusion buffer to 1
(2) Render voxelized scene
(3) For each screen pixel:
- Unproject to world space coordinates and compute occlusion bounding volume
- Determine the position and normal of voxels within volume
- Use analytic solution to compute occlusion
- Decrement occlusion buffer accordingly

## 4. Milestones

- Due 2/24: Implement voxelization using VoxelPipe, build test scenes;
- Due 3/2: Compute for each screen pixel: object space occlusion bounding volume, position and distance to voxels within bounding volume;
- Due 3/9: Compute for each screen pixel: ambient occlusion due to neighboring voxels using analytically determined radiance transfer formula;
- Due 3/15: Implement ray tracer (povray) for comparison, prepare presentation slides and write-up.

## References

[1] Mittring, M.: "Finding next gen: Cryengine 2". In: ACM SIGGRAPH 2007 Courses, pp. 97–121 (2007).

[2] McGuire, M.: "Ambient Occlusion Volumes". In: Proceedings of High Performance Graphics, pp. 47-56 (2010).

[3] Baum, D., Rushmeier, H., Winget, J.: "Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors." Computer Graphics 23(3), pp. 325–334 (1989).

[4] Penmatsa, R., Wyman, C.: "Voxel-Space Ambient Occlusion". (UICS-12-01) Iowa City: The University of Iowa (2012).

[5] Pantaleoni, J.: "VoxelPipe: a programmable pipeline for 3D voxelization". In: Proceedings of High Performance Graphics, pp. 99–106 (2011).