Visible Space with Shadow

# Introduction

The goal of the project was to find the visible space from a point of view using shadows. Finding the visible space can aid in many different techniques from path finding to collision detection. For this to be possible the visible space must be in a format, like a 2D array, that can be used. In addition, to aid in collision detection, the visible space must be generated in real time.

There are a couple of solutions to find the visible space. One is an adaptation of back face culling and the other is to use shadows. Using shadows provides an interesting avenue for solving the visible space problem. The dark part of a shadow on a surface can be a kin to the lights non-visibility of the surface. The following sections will introduce different shadow generation techniques and explain how to convert a scene with shadows to a visible space format.

# Generating Shadows

There are three common techniques for generating shadow. They are shadow mapping, shadow volume and ray tracing. Shadow mapping is the easiest to implement and requires the use of the Z-buffer. Shadow volume is more complicated to implement then shadow mapping and requires the stencil buffer in addition to the Z-buffer to create shadows. Lastly, ray tracing is the hardest to implement because it requires generating multiple rays from the camera to the object.

In this project the shadow mapping and the shadow volume techniques were looked at and explored as possible solutions to view visible space.

# Shadow Mapping

The shadow mapping is technique for creating shadows. The technique was introduced in 1978 by Lance William . When the algorithm was first introduced, the key advantages were that it could handle indefinitely large environment and it had a linear cost growth. The advantages are the result using the Z-buffer / depth buffer. As a result of using the Z-buffer, objects being rendered do not need to be sorted beforehand and are implicitly sorted by the Z-buffer.

Shadow mapping is a two step process that uses the Z-buffer. The first step renders the scene from the lights point of view and store only the depth in the Z-buffer. The Z-buffer holds the depth information for the rendered scene. The closer the object, the lower the depth value and the farther the object, the higher the depth value. A visual representation of the Z-buffer is shown in Figure 1. In the visual representation, the closer objects are darker and thus have a lower depth value while the farther objects and the background have a higher value and are lighter.



Figure : Visual representation of the Z-buffer

The second step renders the same scene from the camera's point of view. In the second step the Z-buffer

# Shadow Volume

# Conclusion

# Bibliography

|  |  |
| --- | --- |
| [1] | L. Williams, "CASTING CURVED SHADOWS ON CURVED SURFACES," *SIGGRAPH Comput. Graph.,* vol. 12, no. 3, pp. 270--274, Aug 1978. |