# Abstract

# Introduction

## Aim

To test advanced screen-space techniques

NEW

To test advanced screen space techniques

(simulating real scenes with shadow casting)

## Objectives

## Technical Information

## Difficulties encountered? (move to implementation section?)

Large loops

A lot of memory bandwidth

Difficulties using the depth map to reconstruct positions

Difficulties adapting to different texture targets for different steps?

A lot of tweakable areas

## Outline?

# Background

## Global Illumination

## Ambient Occlusion

randomness

## Shadows

randomness

## Reflections?

# Implementation

## Shadows (mention alternatives)

## Deferred rendering

## SSAO

# Evaluation

* Test SSAO starting with one sample, going up
* Test SSAO with fixed uniformly distributed points
* Test impact of changing texture sizes
* Test impact of changing loop size in the algorithms or adding early break
* Test performance having an extra one channel target for blurring rather than reusing a large texture.
* Test normals being stored as 3x8bit ubytes rather than 3x16bit floats
* Test performance of reducing opengl context calls

# Conclusion

TODO:

* Fix shadow artefacts
* Start shrinking some textures like shadows and/or SSAO
* Increase the screen texture for antialiasing
* Allow resizing of window?????

Shadows:

\*traditional shadow problems / artifacts

\*cube shadows technique evaluation

Deferred rendering:

\*recreating position from depth

\*storing light (using tiles?)

SSAO:

\*Problems

\*Randomness

Investigation in Screen space techniques.

Aim:

Investigate if more advanced graphical techniques like deferred rendering, SSAO and point shadow casting are applicable on less powerful devices for real time graphics and how they can be controlled.

Investigate how application performance is effected by simple graphical techniques and how they can be controlled.

Objectives:

\*create a scene with some well knows graphical techniques

\*measure performance impact of individual elements

\*analyse performance and image quality changes that can be adjusted

Shadow Tests:

(differed rendering)

Screen image 1280x720 (all buffers the same size)

No Shadows:

0.0037s (270FPS)

\*Teapot: (1291 vertices, 2464 indices)

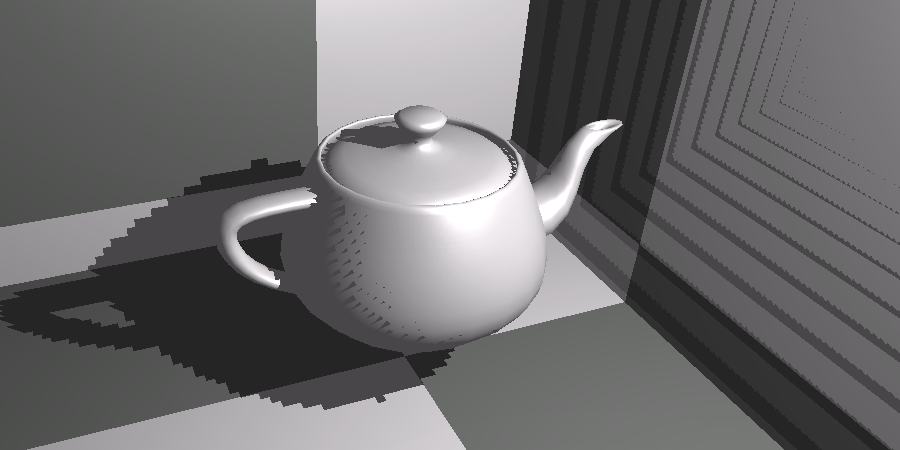
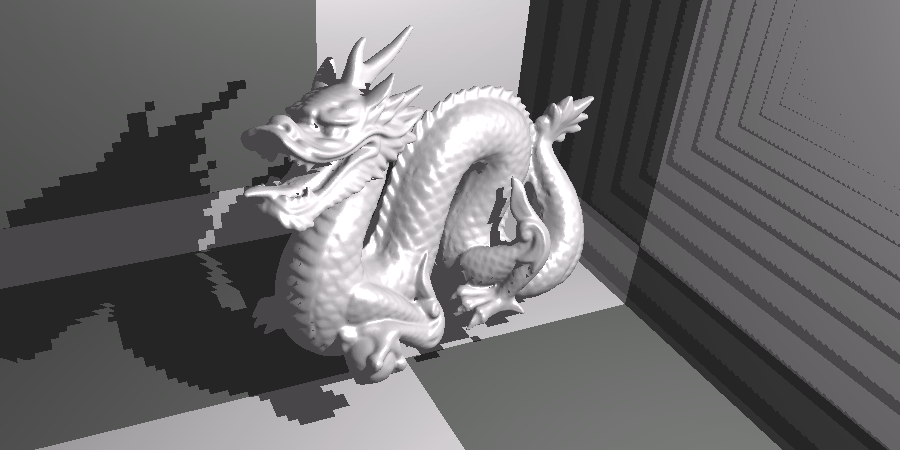
\*Dragon: (435544 vertices, 871305 indices)

Findings:

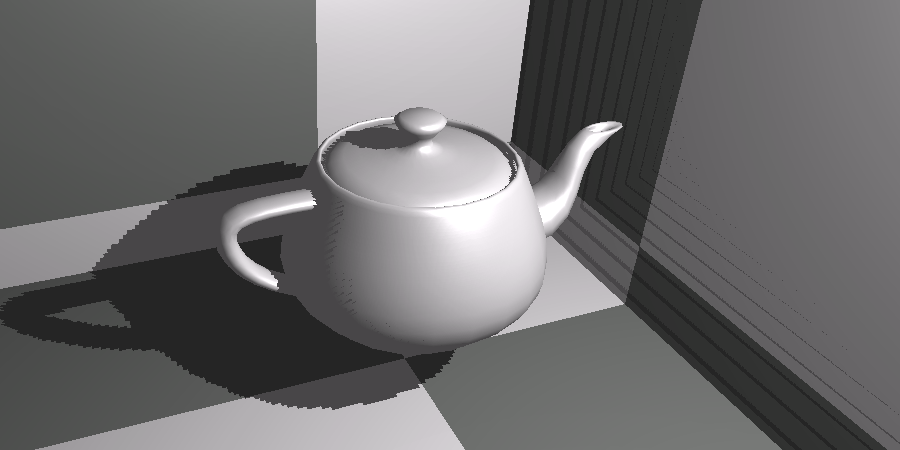
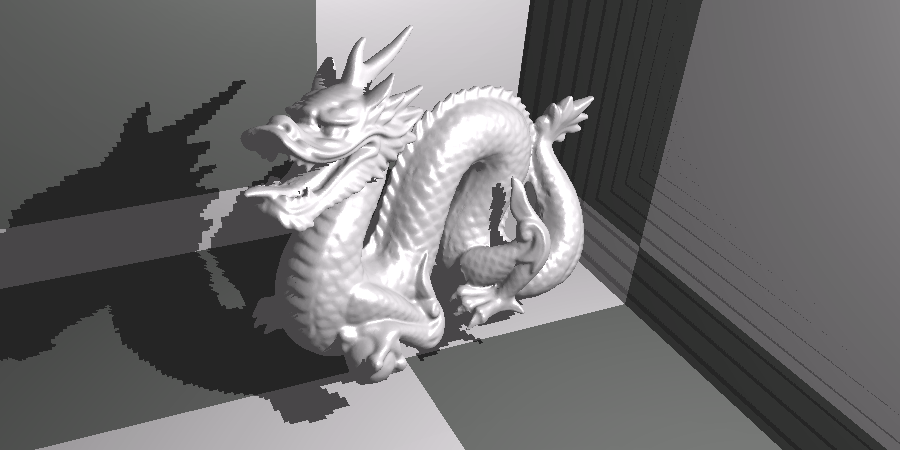
* Almost linear decrease in performance with scene complexity
* Gradual decrease in performance with increase in memory resources
  + Considering that it’s a cubemap (6\*(x^2) change in memory for depth texture multiplier x)

Gimp cropping: 300:100:900:450

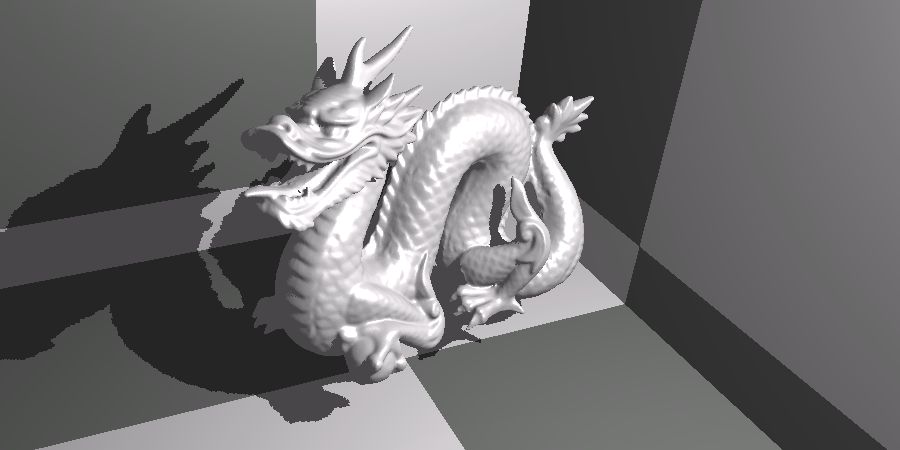
128:128



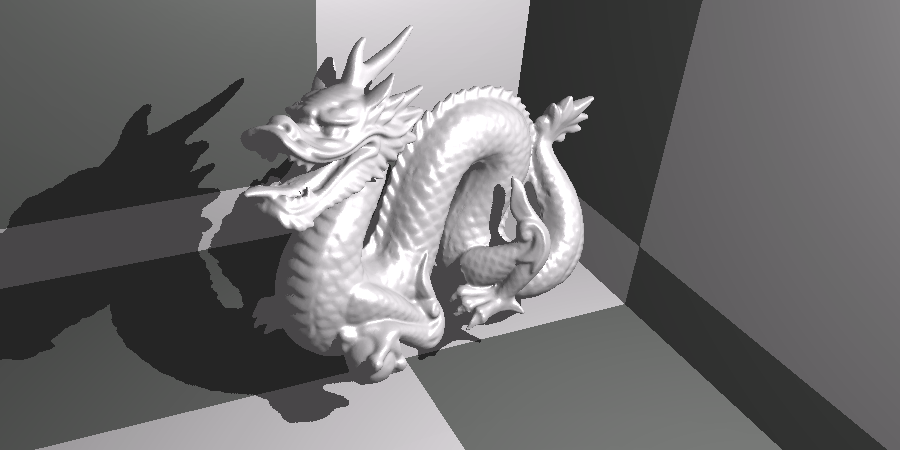
256x256



640x640



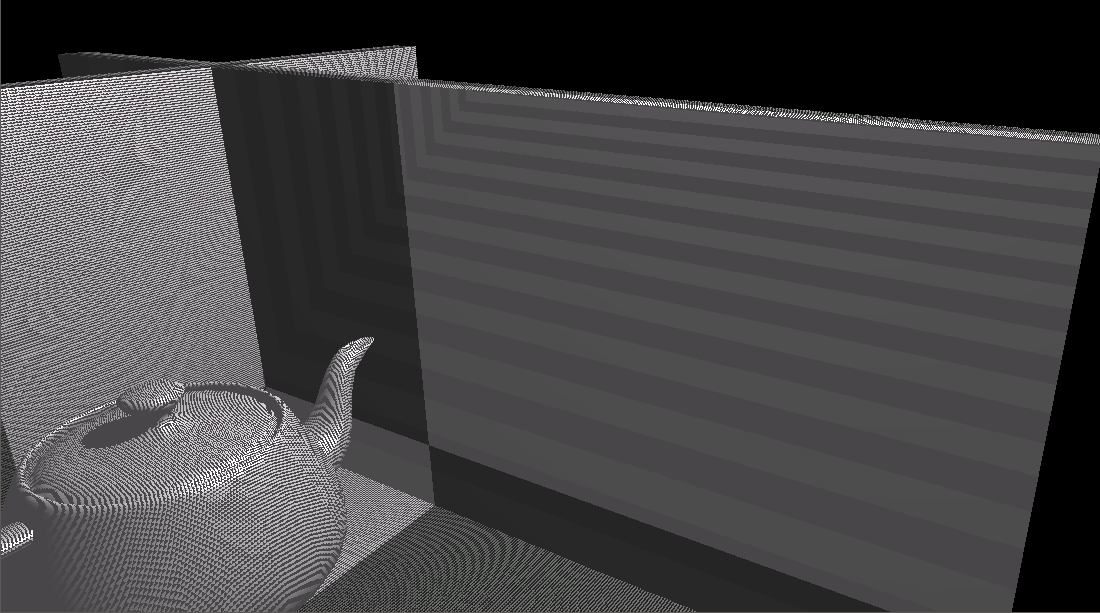
1280x1280



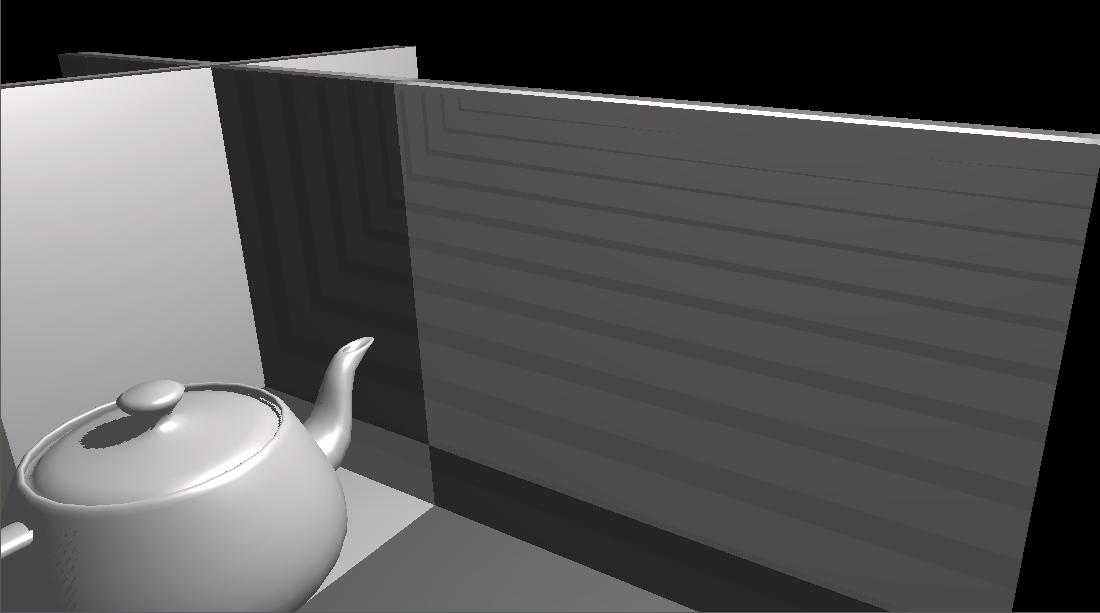
Shadow acne:

Images above were made with a 0.05 bias!

If bias is not applied, you get acne:

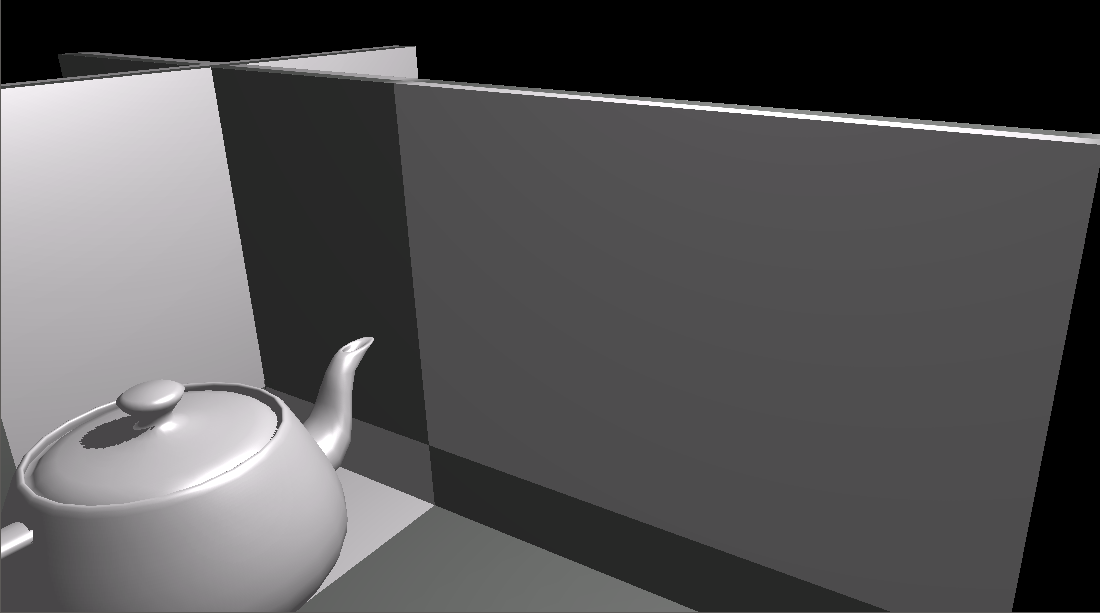
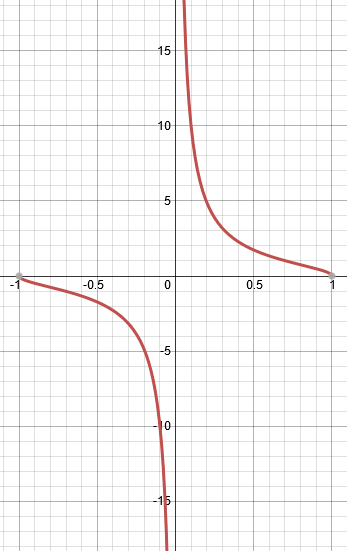


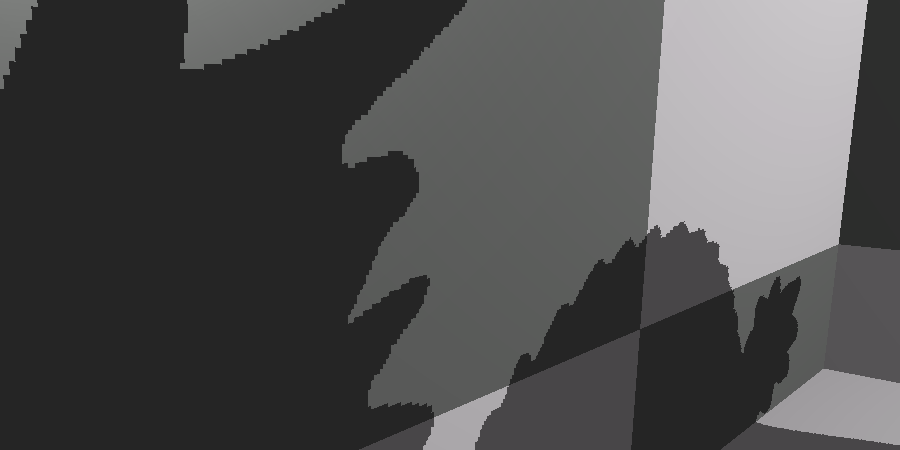
But even with a bias you get problematic areas when the surface is in a steep slope with the light source.



A further multiplier for the bias can be applied:

tan(acos(slope));



Closeup to the problem of shadows even in 1280x1280:

Using 512x512 from now on!!!

From now on it’s just screen space techneques