Satwiko Wirawan Indrawanto - 6201539
Basar Oguz - 6084990

Assignment 3 - Global Illumination: Path Tracer

Introduction

This project implements a physically based renderer, using the method known as Path Tracing. We use Monte Carlo sampling to estimate Kajiya's Rendering Equation for shading, which converges to an unbiased light transport solution. The path tracer supports triangular and spherical area lights and is capable of producing a depth of field effect by interactive focusing.

In order to achieve interactive frame-rates, we use some well-known variance reduction techniques such as importance sampling, Russian roulette, and next event estimation. In addition to these, Multiple Importance Sampling scheme is implemented, and supports multiple area lights. The renderer is also capable of rendering different types of materials, depending on their BRDF, using Microfacet BRDF calculations.

Scene with only HDR skydome as lighting.

Path Tracer Architecture

Material Interaction:

- Reflective surfaces (also set as mirrors in our previous raytracer) is calculated by a random path probability based on the shininess factor. It can range from 0 to 1000, based on mtl file from an obj file. A shininess factor of 1000 is fully specular while 0 is fully diffuse.
- Dielectric surfaces (also set as glass in our previous raytracer) is calculated by a random path probability based on the fresnel term. It either samples reflection or refraction.

We have several path tracing *Sample* function that is implemented on the *render* function in *raytracer.cpp*.

- *Sample* is our main sampling algorithm with all the added variance reduction.
- *SampleEX* is our experimental sampling algorithm with microfacets enabled.
- *SampleSimple* is a basic ray tracing algorithm without any additional variance reduction.
- *SampleMIS* is the multiple importance sampling algorithm, but it is missing the material interaction.
- *SampleWhitted* is Whitted style ray tracing.

Marsaglia's Xorshift is used as the random number generator.

Variance Reduction

Stratification: Implemented using the ray generator. See depth of field for more information.

Next Event Estimation: Implemented by taking direct and indirect illumination. Direct illumination is sampled by choosing a random light.

Importance Sampling: Lights are sampled with the probability proportional to their solid angle.

Russian roulette: Implemented, we had to reduce the chance of survivability when hitting some specific materials as it will result in a crash (dissolve type from obj files that relates to transparency).

Gamma correction is implemented in the *Render* function. Currently gamma is set at 1.0 as it is bright enough in our screen.

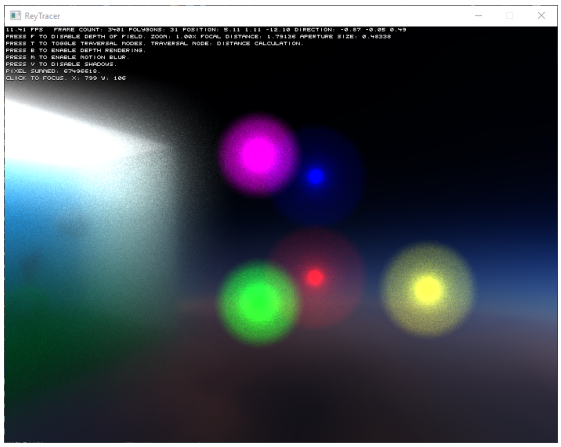
Additional Points

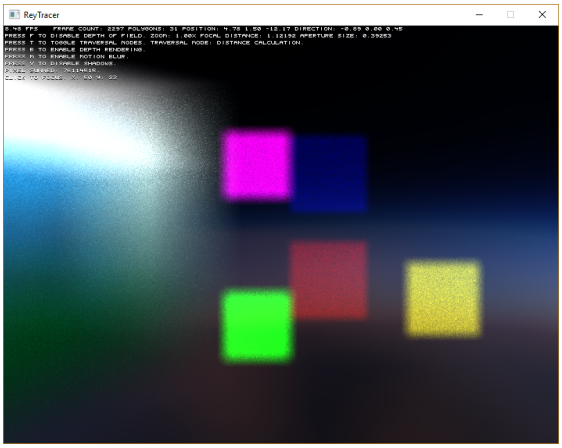
Multiple Importance Sampling:

A MIS sampling function is implemented in our raytracer, MIS works but it hasn't been implemented to handle material interaction. Currently it renders all materials as diffuse.

Stratification / Depth of field:

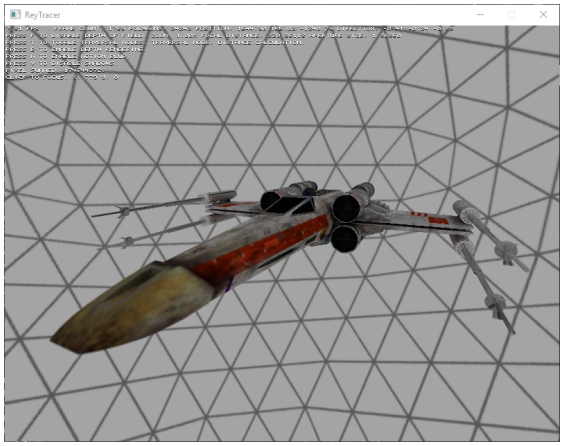
Bokeh is gained using generated ray origin positions focused to a virtual plane in space. We used a fast sine approximation to draw a circular motion which is then scaled by the aperture size. Using standard stratification scaled method scaled by the aperture size will result in a rectangle blur. A mouse click is implemented to focus, a ray will be generated and if it hits something, it will become the new focus point.





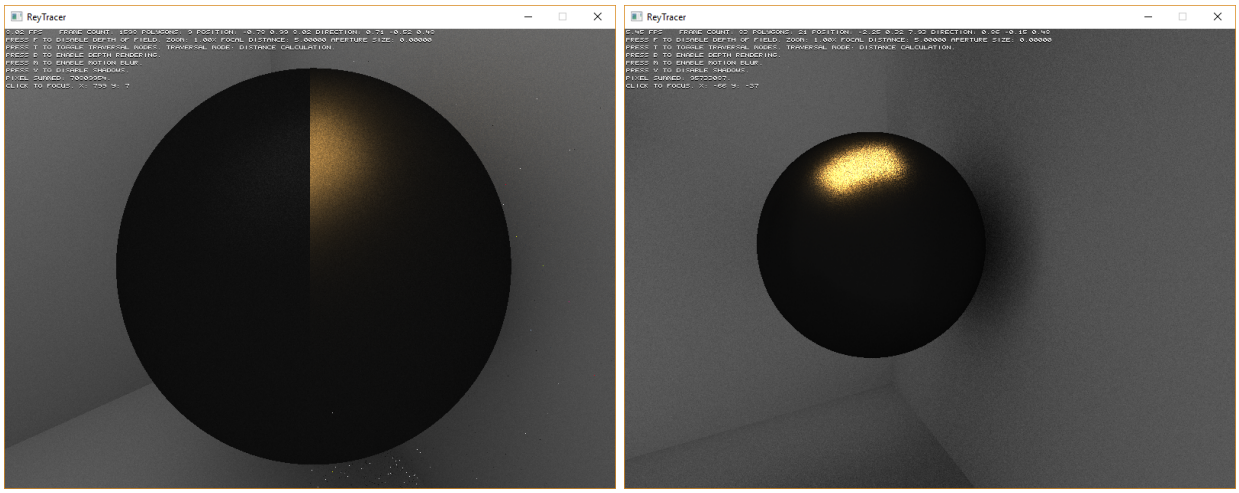
HDR Skydome & bilinear textures

Sample image drawn with a placeholder skydome and bilinear texture filtering but without any specific light source. Even if the scene doesn't have any assigned light sources, the HDR skydome would lit the scene.



Microfacets

A microfacet BRDF is implemented. Using GGX distribution term (as it worked for us), with Cook Torrance's geometry factor, and Schlick's Fresnel approximation. This is a sample image drawn with a low number for roughness and gold as the specular color on a black sphere. Half left of the image is without microfacet, half right is with microfacet. This is scene(9) in our path tracer.



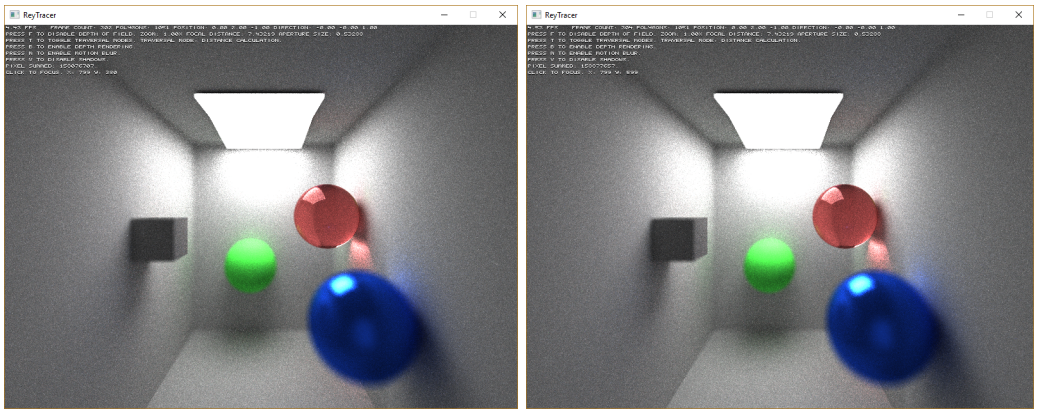
Results & Performance

Performance increased greatly with applying russian roulette. Cornell box scene consists of one diffuse sphere, one dielectric sphere, and one semi specular sphere. Depth of field is enabled on both tests.

- Without russian roulette: 1 - 2 frames per second.
- With russian roulette: 4 - 5 frames per second.

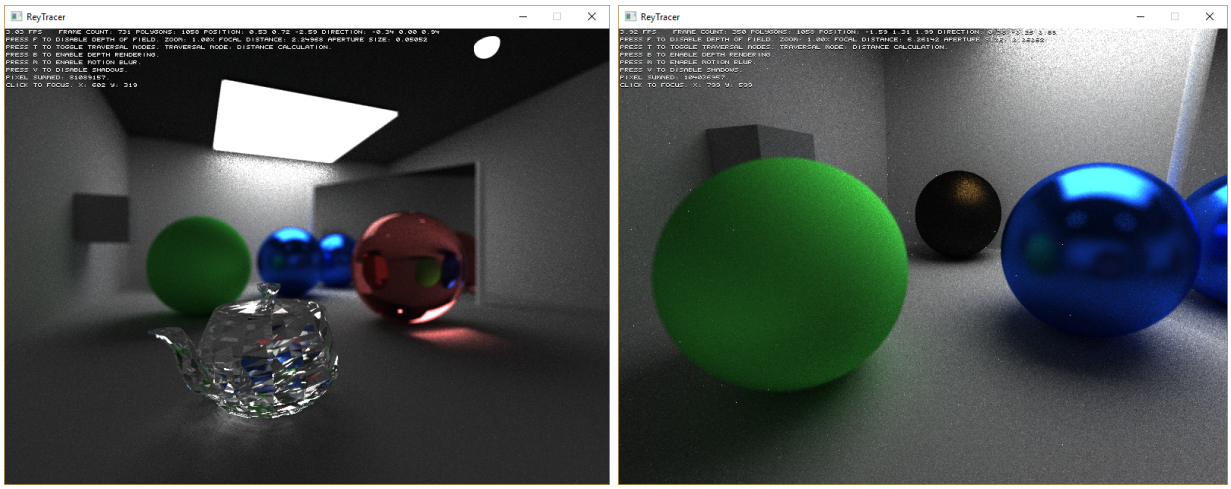
Marsaglia's Xorshift is used as the number generator, this is done to minimize the number of clock cycles needed to generate a random number as it required a very small code and state.

Fast sine approximation is used to improve the performance of our ray depth of field generation, we tried to apply it to our *CosineWeightedDiffuseReflection* function but turns out that it was not accurate enough to converge to the desired image.



Left screenshot is with the *cosf* and *sinf* function, while the right one is using the sine approximation function. There is a slight degrade in quality when using approximation.

Demo scene



If *SampleEX* is enabled, then behind the green diffuse ball there hides a black sphere with gold specularity.

- Use 0 and 9 to zoom in
- Use - and + to focus manually
- Use [and] to decrease or increase depth of field

References

Marsaglia, George (July 2003). "Xorshift RNGs". Journal of Statistical Software. 8 (14). doi:10.18637/jss.v008.i14.