Group 16

Members:

- Nathan Huisman (5705320)
- Ísak Bieltvedt Jónsson (5799813)
- Maria Ruxanda Tudor (5742722)

Basic features

BVH Performance test:

	Cornell Box	Monkey	Dragon
Num triangles	32	967	87130
Time to create	0.05ms	3.25ms	719ms
Time to create (SAH+b)	0.10ms	4.48ms	521ms
Time to render	770ms	839ms	52588ms
Time to render (SAH+B)	657ms	802ms	48808ms
BVH levels	4	9	16
BVH levels (SAH+B)	9	15	29
BVH leaves	8	256	32768
BVH leaves (SAH+B)	11	361	30792

Lights and Shadows:

The blending formula is flawed; alpha represents how opaque an object is and should affect object visibility, not light intensity.

Example 1: With alpha = 0, it ignores possible specular reflections.

Example 2: With alpha = 1, it erroneously cancels light, though transparent objects still reflect/refract light.

Extra Features

Environment maps:

Description :

Environment mapping simulates distant reflections and illumination by mapping a spherical texture to the background of scenes.
When the camera ray does not intersect any objects sampleEnviromentMap() converts the rays 3D direction to 2D UV coordinates and then samples the environment map using sampleTexture(), creating a realistic backdrop!

Sources :

Lecture ray tracing
Fundamentals of Computer graphics 4th edition : chapters 4, 11, 13
https://mechref.engr.illinois.edu/dyn/rvs.html

Location :

extra.cpp & extra.h
Added implementation To extra.cpp and definitions to extra.h

- sampleEnvironmentMap(RenderState& state, Ray ray) : samples env map using sampleTexture()
- sampleTexture() : samples spherical environment map texture using UV coords.

Scene.cpp & scene.h
Added implementation to scene.cpp and changed struct in scene.h

- std::optional<Image> environmentMap : added env map to scene struct
- Case singleTriangle : loaded env map in the single triangle scene

Rendered Images :


Bloom filter:

Description :

The implementation uses a filter to extract the bright parts, and then applies horizontal and vertical blurring to these areas (of a specified size) to create the bloom effect. The final result is added to the original image with a specified scaling factor to control the intensity of the bloom effect.

Sources :

Bloom filter and box filter lecture slides

Location :

<u>extra.cpp & main.cpp & common.h</u>

Added implementation to extra.cpp, sliders in main.cpp and the variables “scaleSize” and “filterSize” in common.h

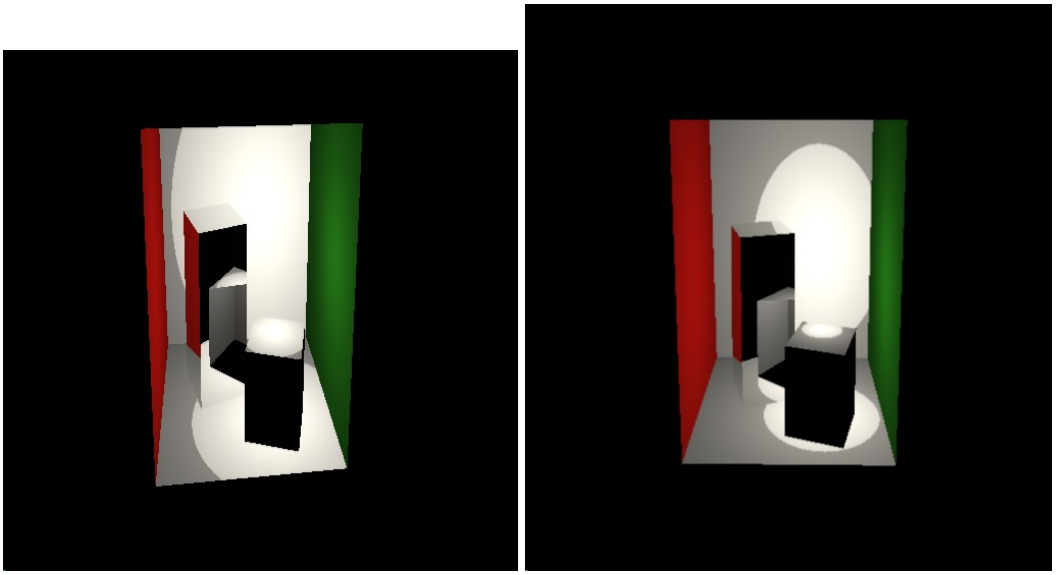
- postprocessImageWithBloom(...) : applies the bloom filter on a given image.
- factorial(...) : calculates the factorial of a given number

<u>main.cpp</u>

- Scale slider : in the UI you can set your own scale
- Filter size (pixels) : in the UI you can set the number of pixels of the filter

Rendered Images :

(different scale and filter size)



Glossy Reflection:

Description :

The method calculates multiple glossy reflection directions by generating random uniform samples on a disk centered around the reflected ray. The code uses the "plane_basis" function to obtain an orthogonal basis for the reflection plane. The function then evaluates these rays and accumulates their contributions by applying the material's specular reflectance (ks) to the hit color. The result is averaged over multiple samples to create a realistic glossy reflection effect.

Sources :

https://blog.yiningkarlli.com/2012/12/blurred-glossy-reflections.html

https://stats.stackexchange.com/questions/481543/generating-random-points-uniformly-on-a-disk/481544#481544

page 374 - the book Computer Graphics Principles and Practice by John F. Hughes

Location :

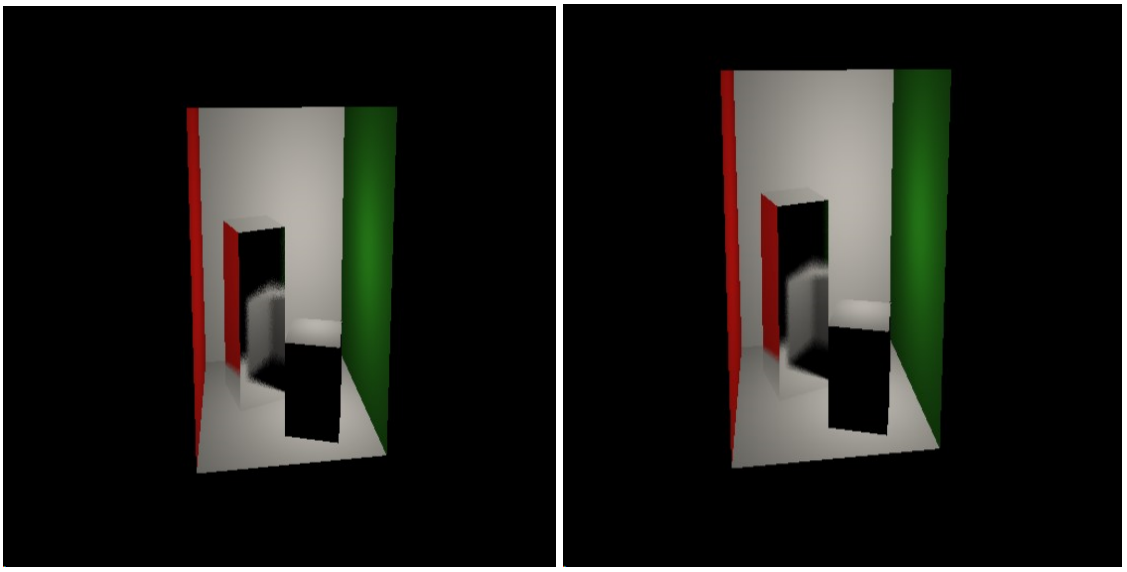
extra.cpp

Added implementation to extra.cpp

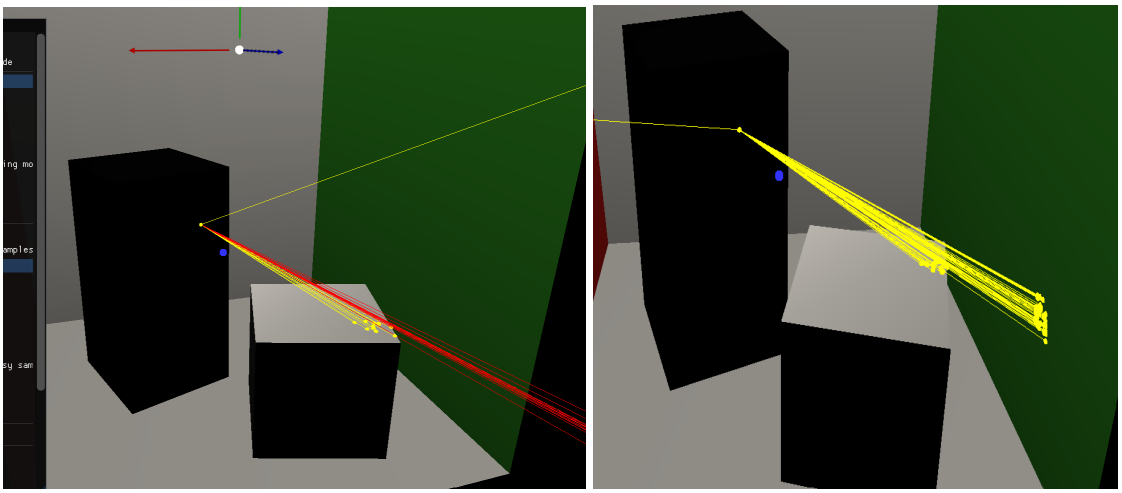
- renderRayGlossyComponent(...) : applies glossy reflections to a scene
- plane_basis(...) : returns an orthogonal basis for the reflection plane

Rendered Images :

(little vs. more samples)



Debug Images:



Motion Blur:

Description :

It simulates fast object movement over a specified number of time intervals (samples) using a cubic Bezier curve to displace vertices. The code iterates through samples, transforming the scene based on the curve and calculating ray contributions for each time interval. The results are averaged to achieve the final image with motion blur. The division by 15 smooths vertex displacement. The implementation also uses multi-threading to improve performance in the rendering process.

Sources :

Slides Ray Tracing:
https://brightspace.tudelft.nl/d2l/le/content/595314/viewContent/3475537/View

https://en.wikipedia.org/wiki/B%C3%A9zier_curve

page 980 - the book Computer Graphics Principles and Practice by John F. Hughes

Location :

extra.cpp

Added implementation to extra.cpp

- renderImageWithMotionBlur(...) : adds a motion blur effect to the rendered image

Rendered Images :
(ray traced screenshot vs. rendered image)