Thesis Defense

LightWave: An Interactive Estimation of Indirect Illumination Using Waves of Light

By:
Michael Robertson
Friday, December 7, 2012

General Goal

- We want to render indirect illumination with indirect shadowing in **real-time**.
- But, until recent developments in technology, realistic lighting of scene has only been achievable **offline**.
- Now, real-time rendering is possible but we must make tradeoffs between scientific accuracy and performance.

Why?

- The desire to accurately recreate our world using computer graphics for:
  - Entertainment: Games, Animated Movies
  - Science: Weather Forecasts, Simulations
  - Medicine: Body Scans
  - Architecture, Design, and more...

Terminology

- Direct Illumination
  - Direct Shadows
- Indirect Illumination
  - Indirect Shadows
- Global Illumination

The Problem

- Direct Illumination =
  - Easy, stops at first surface it hits
- Indirect Illumination =
  - Hard, bounces infinitely
  - Indirect Illumination can be expressed accurately using a Neumann Series

Indirect Illumination

- Neumann Series = $\sum_{k=0}^{\infty} T^k$
- Global Illumination (The Rendering Equation, Kajiya 1986)=

$$I = g\epsilon + gMg\epsilon + gMgMg\epsilon + g(Mg)^3\epsilon + \ldots$$

- Neumann Series of form = $I = g\epsilon \sum_{k=0}^{\infty} (Mg)^k$
  - gε = direct term
  - gMgε = once scattered
  - gMgMgε = twice scattered, and so on…

Historical Inspirations

- Prior to the 19th century
  - Light was regarded as a stream of particles
  - Explained reflection and refraction
- 1801: Light rays interfere (wave-like)
- 1873: Light was compared to a form of high-frequency electromagnetic wave
- 1905: Einstein proposed that the energy of light waves is quantized in particles called photons.

Goals

- Indirect illumination with indirect shadows
- Fully dynamic light and objects
- Scalable
  - Scene Complexity
  - Support variety of hardware capabilities
- Implement using the GPU through the use of OpenGL and GLSL

Related Work

- Instant Radiosity (Keller 1997)
  - Virtual Point Lights (VPL's)
  - Each VPL acts independently as a producer of indirect light
  - VPL's are generated at the hit points of the light rays as they are traced into the scene from the primary light source
  - The contributions of each VPL is summed through the use of multiple rendering passes.

Related Work

- Instant Radiosity (Keller 1997)
  - Limitations: multiple rendering passes were required for dynamic objects or lights and limited to simple scenes
  - VPL's are used for other purposes: area lights, high dynamic range (HDR) environment maps, subsurface light scattering

Related Work

- Shadow Maps (Williams 1978), (Reeves, Salesin, and Cook 1987)
  - Method of rendering shadows
  - Shadow Maps are created in a render pass where the view of the scene is computed from the light's perspective.
  - Calculate the distance to the nearest surface for each pixel and store in a texture or buffer.
  - Use these distances for comparison when rendering the scene from the camera's perspective to determine whether a surface point is in shadow.

Scientifically Correct Versus Visually Appealing

- All techniques are approximations, since an infinite number of bounces of light would need to be calculated to get correct lighting.
- For real-time applications, we must err on the side of performance, but indirect lighting has been shown to be perceptually important so we can't ignore it.
- Realism is determined partially on whether the scene is visually pleasing to the eye.

Related Work

- Perceptual Influence of Approximate Visibility in Indirect Illumination (Yu 2009)
  - The most obvious lighting is direct lighting (high-frequency)
  - Indirect lighting usually has smooth changes in intensity allowing for interpolations.
  - Visibility determination for indirect shadows is the most expensive calculation.
  - Conclusion: “Accurate visibility is not required and certain approximations may be introduced”

Perceptual Influence...

- Conducted study using mostly computer scientists with imaging backgrounds.
- Involved looking at different renderings of the same scene using an offline reference rendering and different real-time approximated renderings.
- Approximated renderings of indirect shadows ranged from imperfect visibility to no visibility.
- 2 studies:
  - Blind ranking of all renderings
  - Estimate difference from reference based on a number scale

Perceptual Influence...

- Results:
  - The reference and all approximated renderings were found to be equally realistic with the no visibility rendering slightly less realistic.
  - All of the approximated renderings were found to be “very much similar” to the reference except for the no visibility rendering which was found to be “moderately similar.”

General Implementation

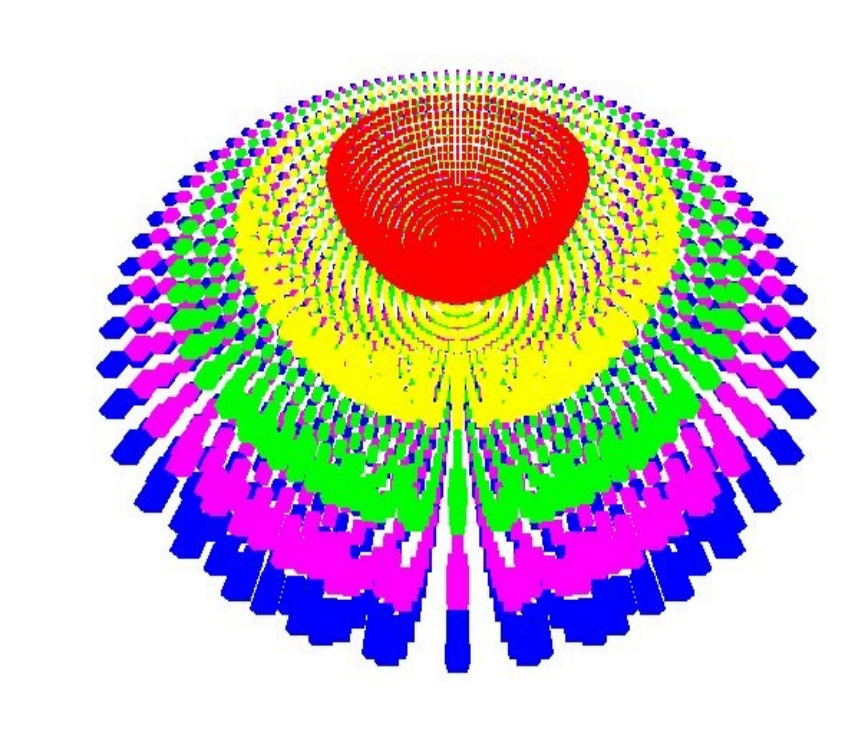
- Render indirect illumination using VPL's
  - Ignore the bouncing of indirect light
  - Simplify the Neumann series to just one iteration
- Use shadow maps for direct shadows and indirect shadows
- Provide 2 different techniques of rendering indirect shadows: accurate visibility and “integrated” visibility

Implementation Specifics - Working Environment

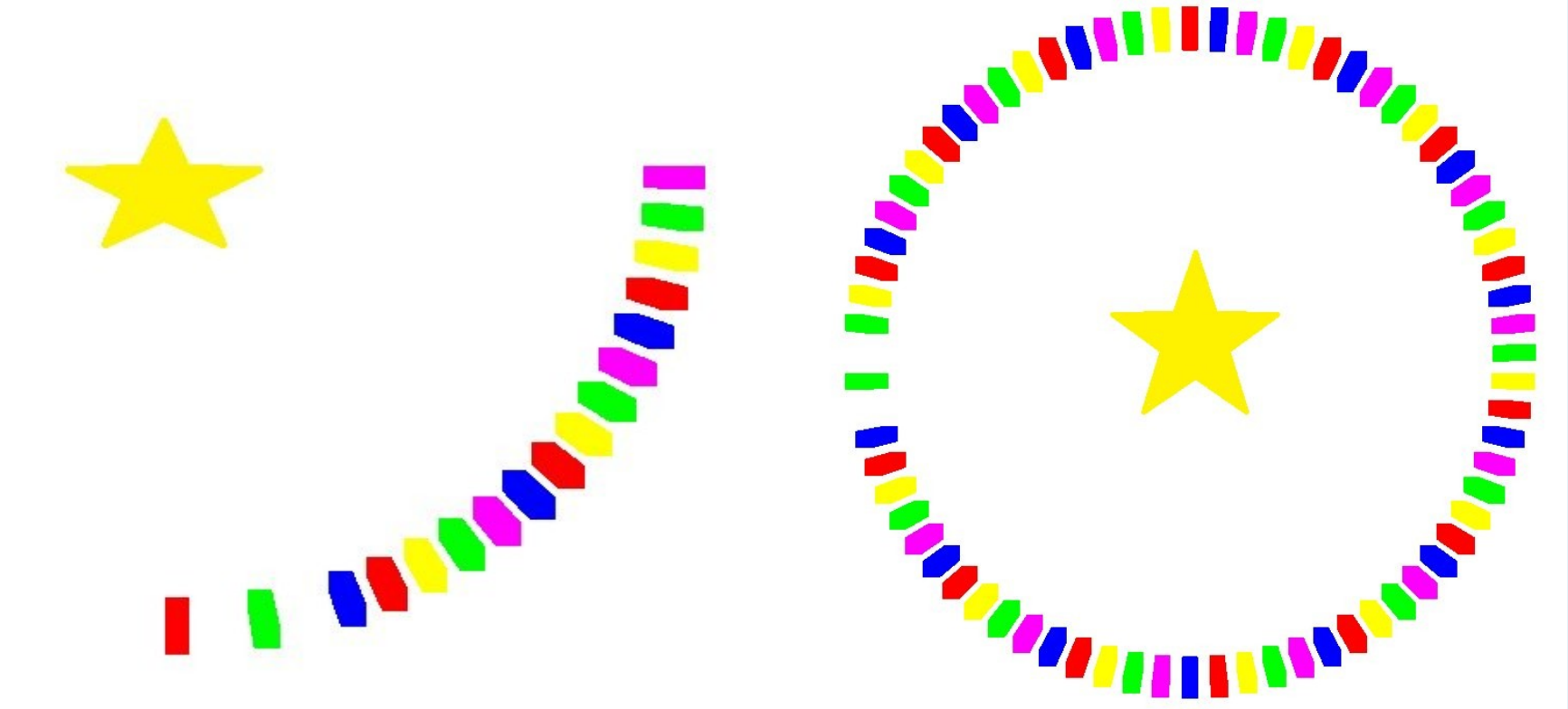
- CPU code is written in C++ using OpenGL.
- GPU code is written in GLSL.
  - This allows us to write our own lighting calculations rather than use the default settings.
- Hardware:
  - CPU: AMD Athlon 64 X2 Dual Core 5200+ 2.61GHz
  - GPU: Nvidia GeForce GTX 465 **(1GB mem)**

Implementation Specifics - VPL Organization

- VPL's will structured in such a way as to simulate the flowing of waves from the primary light source.
- This allows light to reach areas not visible to the primary light source.

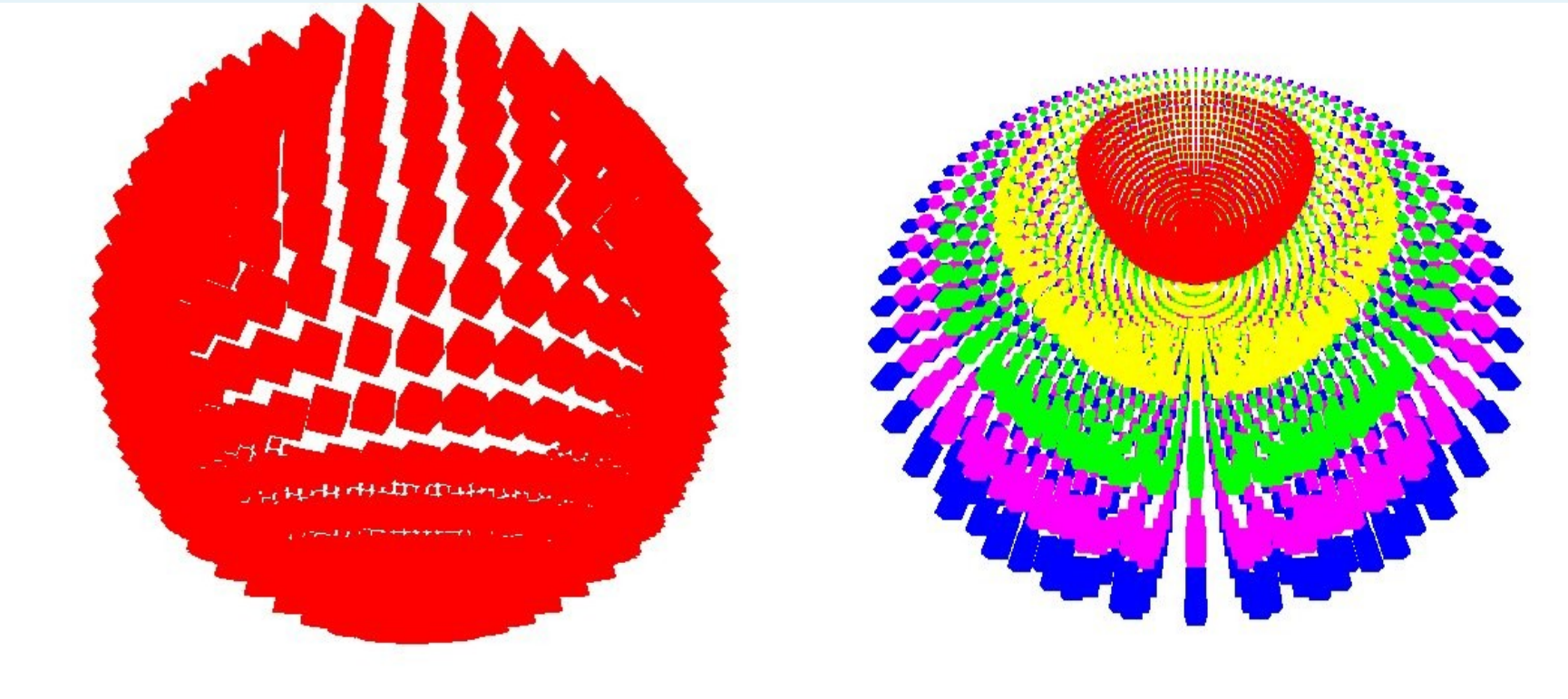


Implementation Specifics - VPL Organization



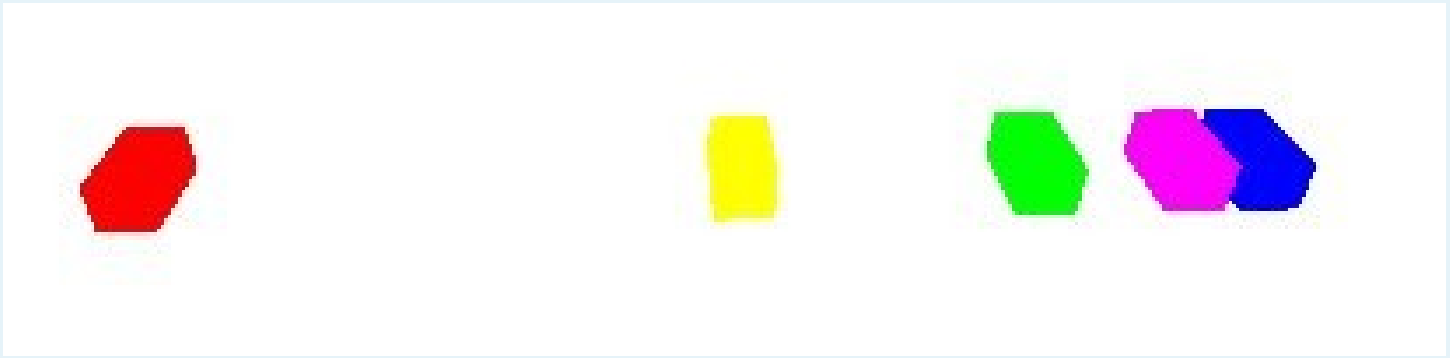
- Default: 6485 total VPL's. Every 5 degrees around the horizontal axis of the light (left) and every 5 degrees around the vertical axis (right).

Implementation Specifics - VPL Organization



- This forms a hemisphere around the primary light. (left) Default: 5 stacked hemispheres (right)

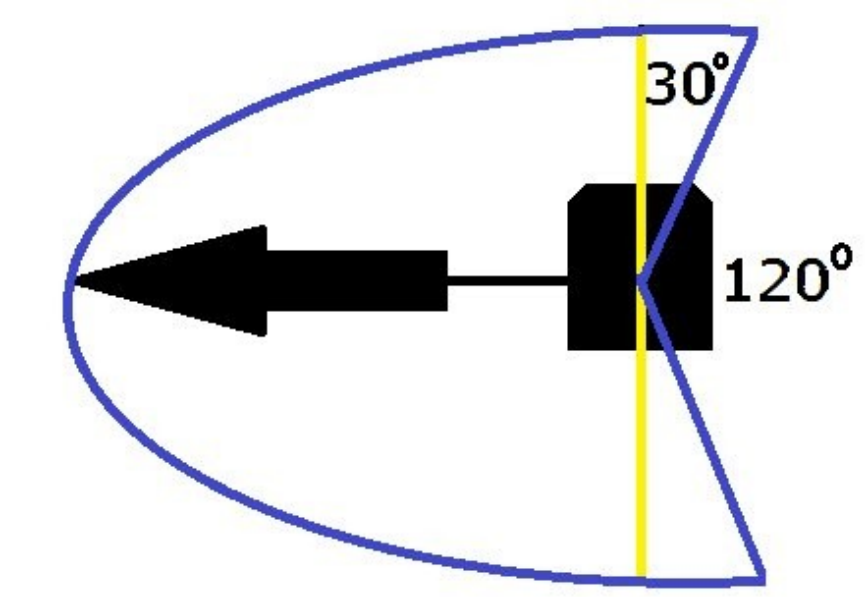
Implementation Specifics - VPL Organization



- The distance between each “shell” of VPL's will be logarithmic.
- The attenuation of the light produced by a VPL in each shell will be exponential.
  - The amount of drop off in the intensity of the light will increase exponentially per shell.

Implementation Specifics - VPL Contributions

- In keeping with the wave theme: VPL's viewable range for contributing light will be extended.

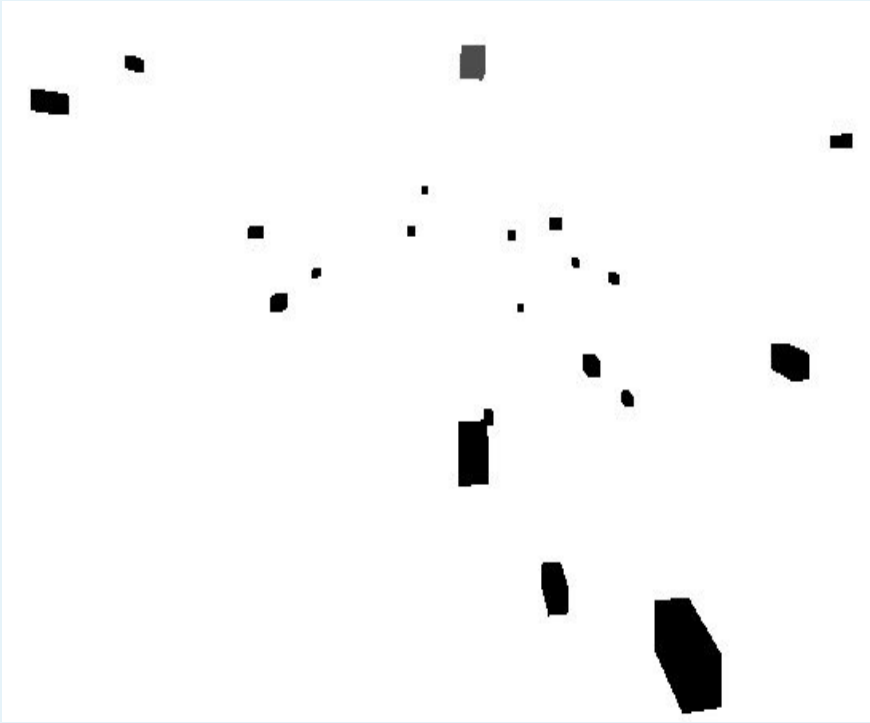


Implementation Specifics - Shadows

- Direct shadows will be computed using a single shadow map.
- Indirect shadows will be computed using 2 different methods:
  - Accurate Visibility: Use 20 shadow maps each computed using the perspective of a randomly chosen VPL to cast shadows.
  - Integrated Visibility: Use 5 shadow maps each computed using the perspective of a specific VPL to cast shadows.

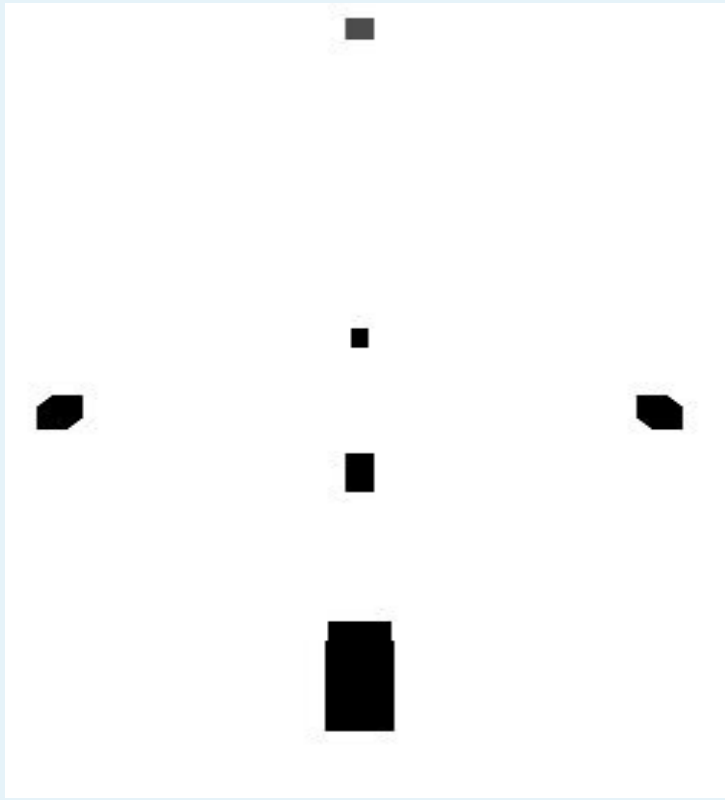
Implementation Specifics - Accurate Shadows

- Compute 20 shadow maps using randomly chosen VPL's (right)
- This will result in 20 indirect shadows using accurate visibility.
- 1 shadow per shadow map



Implementation Specifics - Integrated Shadows

- Compute 5 shadow maps using specific VPL's (right)
- We then interpolate between shadow maps in order to create additional shadows based on the number of steps taken.
- 8-24+ shadows per shadow map



Implementation Specifics - Integrated Shadows

- This is done by calculating the coordinate corresponding to our given vertex in 2 given shadow maps.
- We then calculate a vector connecting these 2 coordinates.
- We step across the vector using the user-specified number of steps resulting in in-between coordinates.
- At each step, we determine whether the vertex would be in shadow using this in-between coordinate and 1 of the 2 shadow maps.

Implementation Specifics - Integrated Shadows

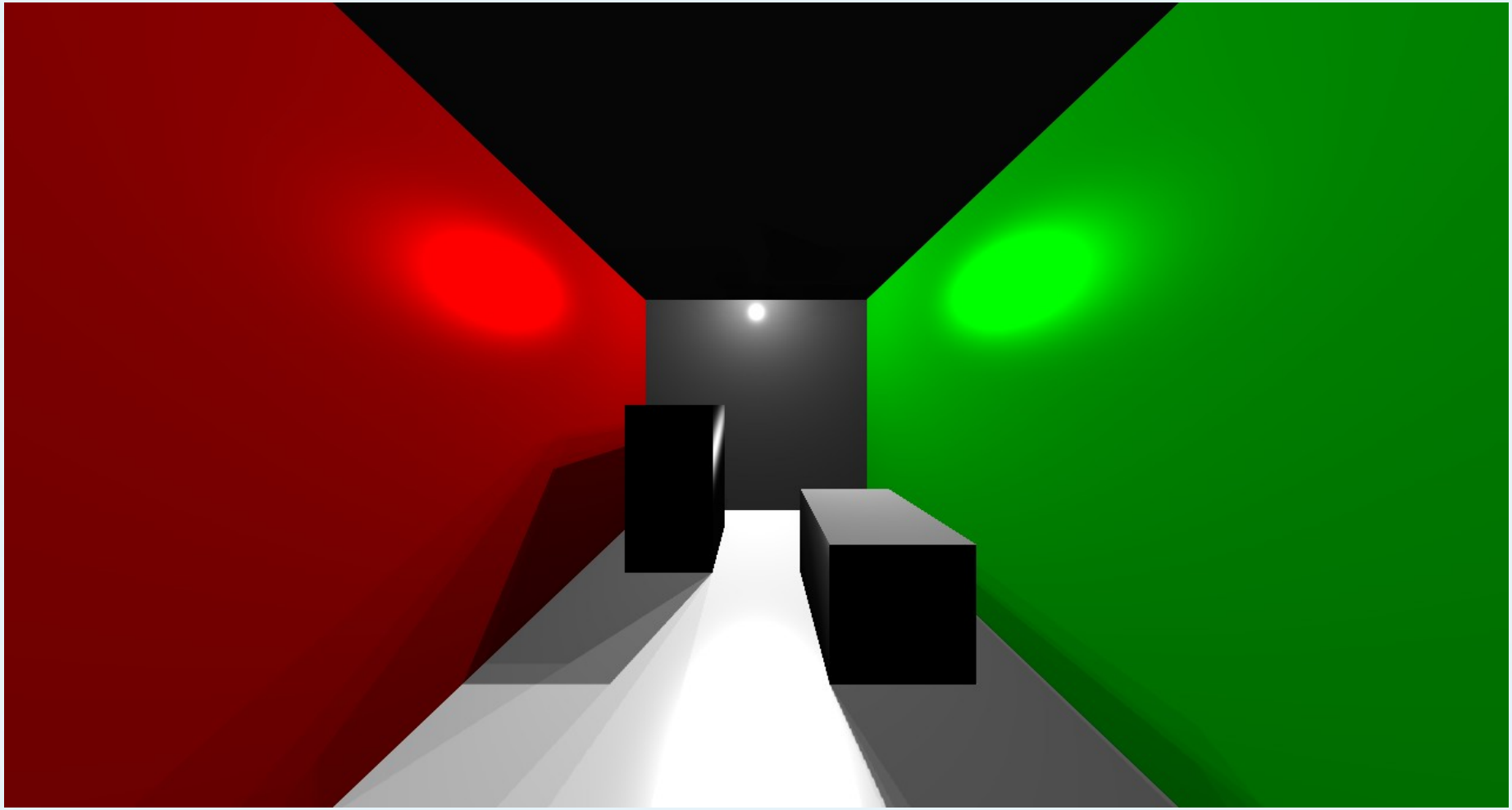
- The in-between coordinates in the 1st half of the vector will use the 1st shadow map and the 2nd half will use the 2nd shadow map.
- We do this for all possible combinations of the 5 shadow maps:

$$\binom{5}{2} = \frac{120}{2 * 6} = 10$$

- Then the total number of indirect shadows shadows will be:

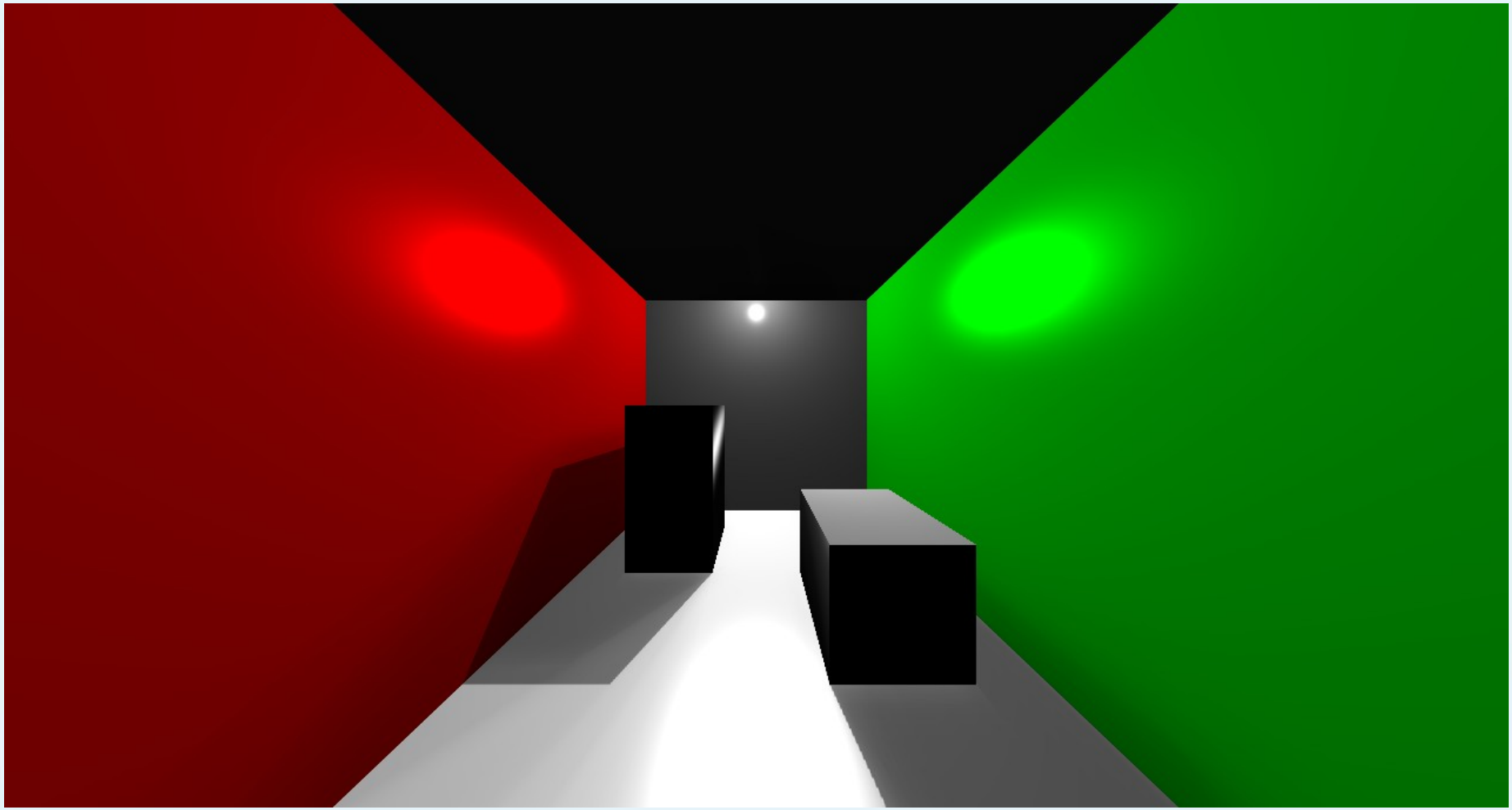
$$numShadows = (numSteps + 2) * 10$$

Implementation Specifics - Integrated Shadows



- Why interpolate? Why not just use 5 shadows?

Implementation Specifics - Integrated Shadows

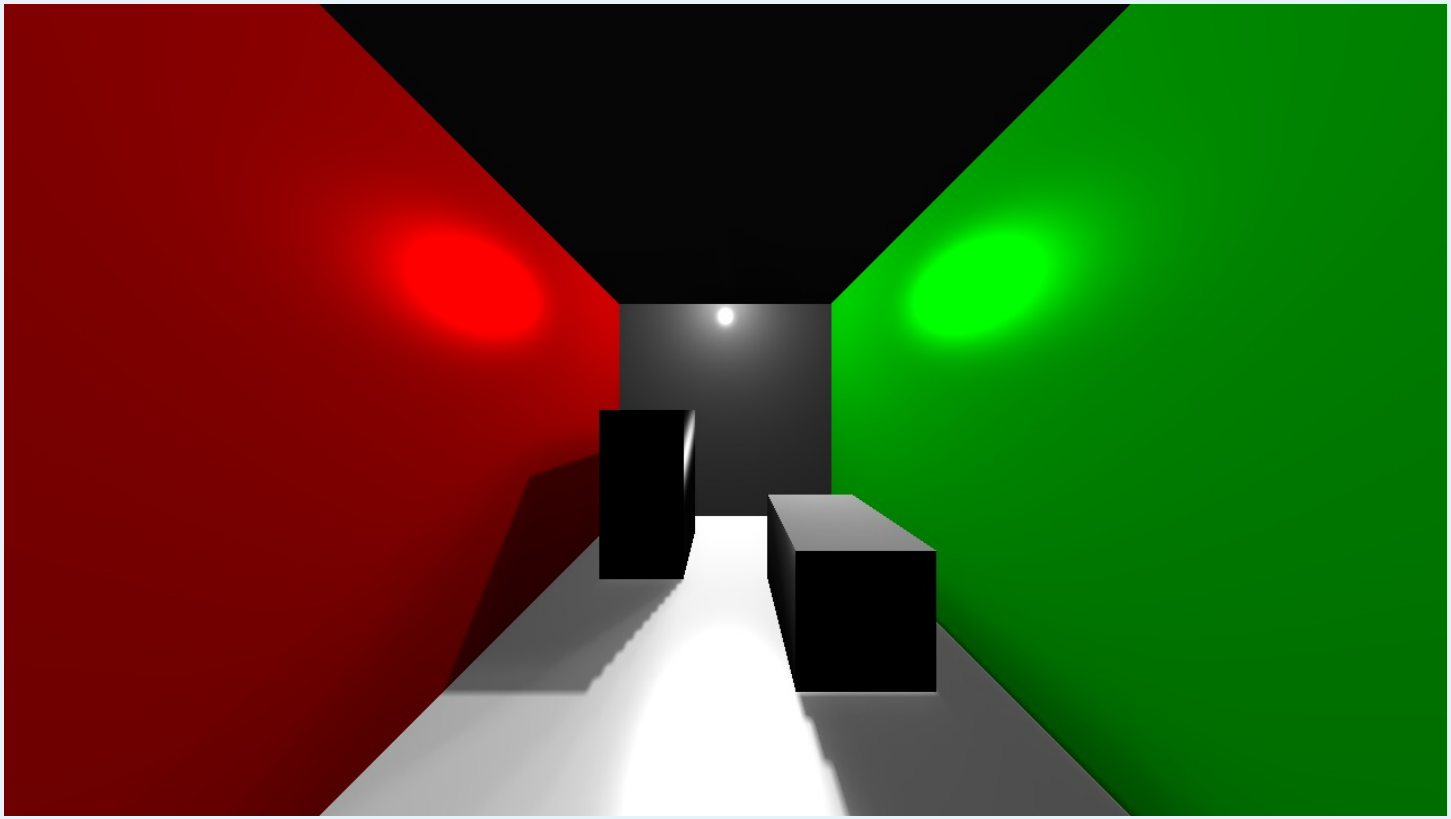


- Compared to integrated shadows: (smoother)

Memory Analysis
Shadow Maps

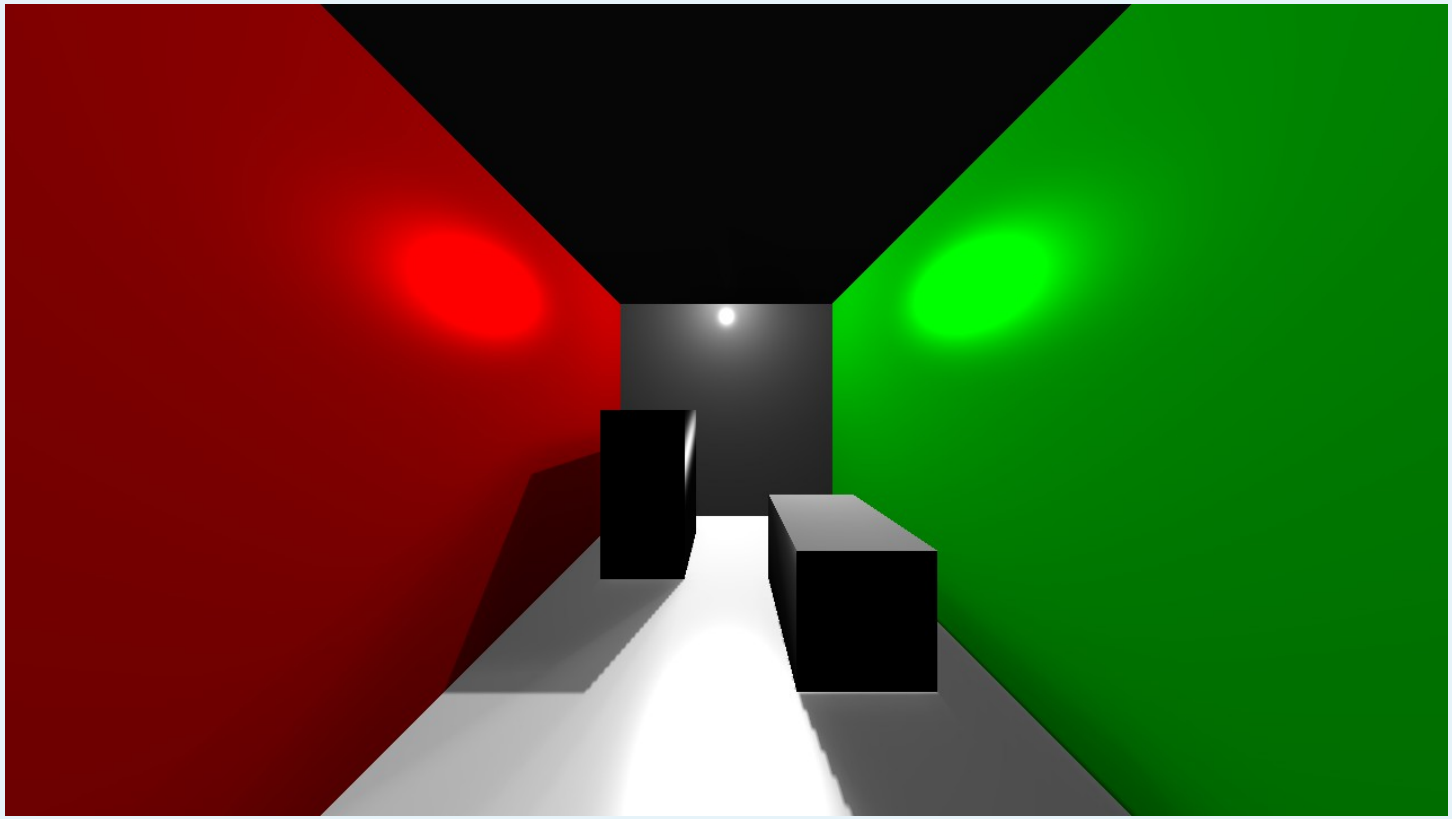
- The default resolution will be 1280x720.
- The shadow maps should be a ratio of this resolution.
- The larger they are, the less jaggies they show. (Examples next)
- For this implementation, I'll be using shadow maps 3X larger than the resolution.
  - 3840x2160

Memory Analysis
Shadow Maps



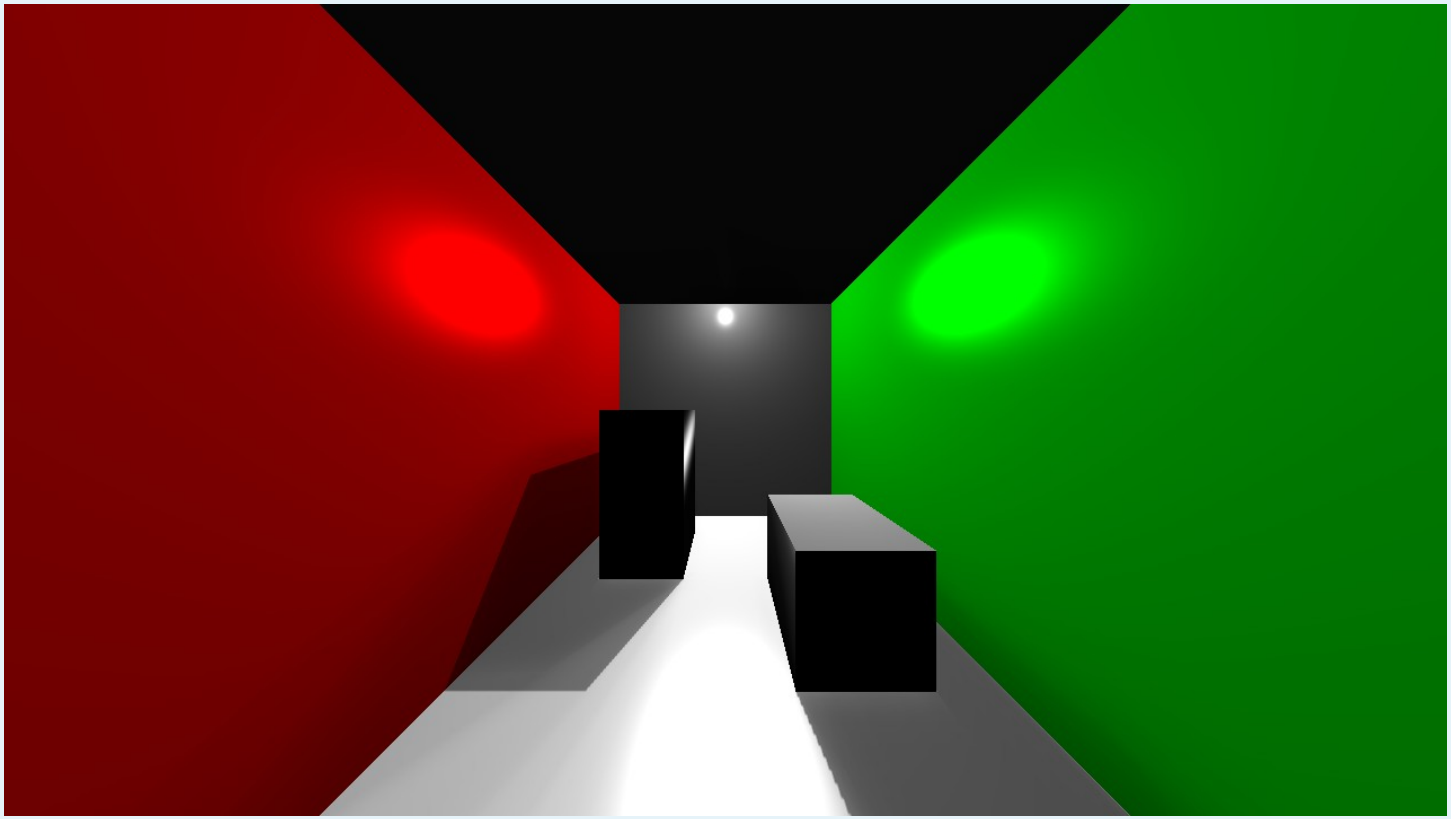
- Shadow Maps half of the screen resolution:
  - 640x360

Memory Analysis
Shadow Maps



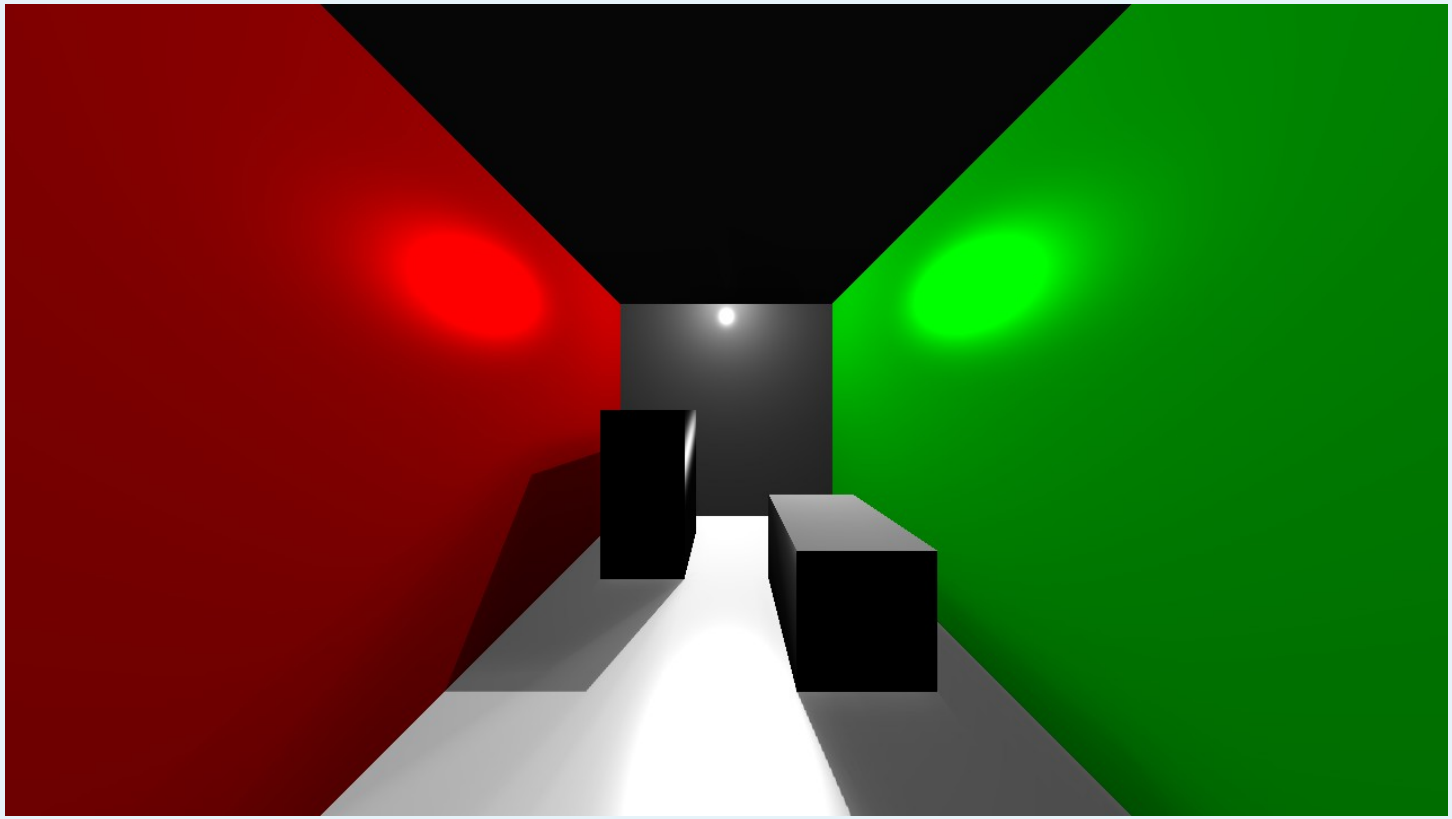
- Shadow Maps same size as screen resolution: 1280x720

Memory Analysis
Shadow Maps



- Shadow Maps twice the screen resolution:
  - 2560x1440

Memory Analysis
Shadow Maps



- Shadow Maps 3X the screen resolution:
  - 3840x2160

Memory Analysis
Shadow Maps

- Knowing the texture size of the shadow maps and knowing that they store floats, we can calculate the amount of memory required for the GPU:

  (3840*2160*4)/(1024*1024) = **31.64MB per SM**

- Accurate Shadows (21 total SM):

  (3840*2160*4*21)/(1024*1024) = **664.45MB**

- Integrated Shadows (6 total SM):

  (3840*2160*4*6)/(1024*1024) = **189.84MB**

Memory Analysis
Shadow Maps

- Recall the GPU I'm using was: Nvidia GeForce GTX 465 **(1GB mem)** allowing me to run either implementation.
- But many computers today have GPU's with 256MB or 512MB of available memory.
- A goal was to have the implementation be scalable, so integrated shadows looks more appealing in memory use.

Implementation Specifics - CPU Responsibilities

- Initializing the window, variables, and textures.
- Compute the VPL data (position, normals) and store in textures. This is updated every time the light is moved.
- Generate all of the shadow maps by rendering the scene from the perspective of each light (not seen by user). This is done every time an object or the light is moved.

Implementation Specifics - GPU Responsibilities

- Lighting calculations using the VPL data and shadow maps passed in from the CPU.
- Lighting calculations are done per-vertex.
- GPU is preferred in this area due to the ability to excel in parallel processing of data.
- Since the same calculations are done for every vertex, we can do this is parallel.

Results

- In order to show how scalable the implementation is, the results will show the differences in performance (frames per second) when changing different parameters including:
  - Screen Resolution (and Shadow Map size)
  - Number of VPL's (increase the angle from 5 degrees or reducing the number of shells)
  - Number of shadow maps (for accurate shadows)
  - Number of steps to interpolate (integrated shadows)

Results – Accurate Shadows

Table 5.1: Varying the Angle Between VPL Rays (Impact on FPS)

Resolution	Angle	VPL's Per Ray	Total #VPL's	#Indirect SM's	FPS
1280x720	5	5	6485	20	55
1280x720	10	5	1625	20	65
1280x720	30	5	185	20	84
1280x720	45	5	85	20	95
1280x720	90	5	25	20	100

Table 5.2: Varying the Number of VPL's Per Ray (Impact on FPS)

Resolution	Angle	VPL's Per Ray	Total #VPL's	#Indirect SM's	FPS
1280x720	5	5	6485	20	55
1280x720	5	4	5188	20	56
1280x720	5	3	3891	20	61
1280x720	5	2	2594	20	62
1280x720	5	1	1297	20	78
1280x720	5	6	7782	20	52

Results – Accurate Shadows

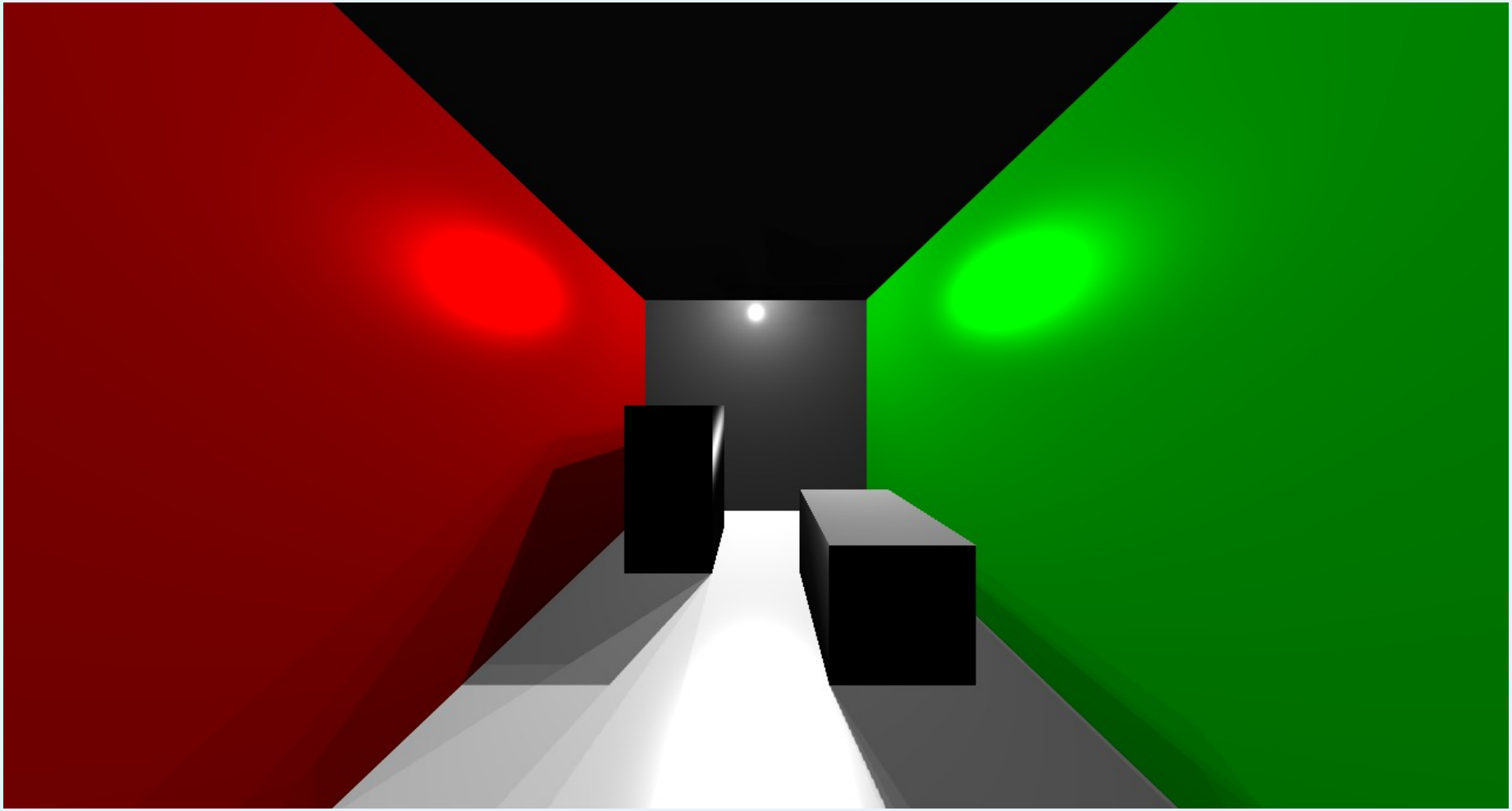
Table 5.3: Varying the Resolution Size. (Impact on FPS)

Resolution	Angle	VPL's Per Ray	Total #VPL's	#Indirect SM's	FPS
1280x720	5	5	6485	20	55
1120x630	5	5	6485	20	82
960x540	5	5	6485	20	104
800x450	5	5	6485	20	119
640x360	5	5	6485	20	133
1440x810	5	5	6485	20	42

Table 5.4: Varying the Number of Indirect Shadow Maps (Impact on FPS)

Resolution	Angle	VPL's Per Ray	Total #VPL's	#Indirect SM's	FPS
1280x720	5	5	6485	20	55
1280x720	5	5	6485	15	87
1280x720	5	5	6485	10	106
1280x720	5	5	6485	5	128
1280x720	5	5	6485	1	157
1280x720	5	5	6485	0	161

5 Indirect Shadows Revisited



Results – Integrated Shadows

Table 5.11: 1280x720, 6485 VPL's, Resulting FPS and number of shadows from adjusting the number of steps

NumSteps	FPS	Number of Shadows
2	81	40
4	60	60
6	48	80
10	34	120

Table 5.12: 640x360, 6485 VPL's, Resulting FPS from reduced resolution and adjusting number of steps

NumSteps	FPS
2	137
6	107
10	89

Results – Integrated Shadows

Table 5.13: 1280x720, 185 VPL's, 30 degree angle, Resulting FPS from reduced VPL count and adjusting number of steps

NumSteps	FPS
2	150
6	65
10	42

- 150 FPS is the best performance that doesn't include ignoring indirect shadows altogether.

Scene Complexity Accurate Shadows

Scene	Method	FPS
6565/3249	Default Method	15
6565/3249	Indirect Lighting w/out indirect shadows	43
6565/3249	Indirect Lighting w/ 15 indirect shadows	20
6565/3249	Indirect Lighting w/ 10 indirect shadows	25
6565/3249	Direct Lighting ONLY	502
6565/3249	30 Degree Angle Between VPL Rays	70
6565/3249	Reduction down to 1 VPL Per Ray	44
6565/3249	Resolution Reduction 640x360	18
65542/32418	Default Method	1
65542/32418	Indirect Lighting w/out indirect shadows	5
65542/32418	Indirect Lighting w/ 15 indirect shadows	2
65542/32418	Indirect Lighting w/ 10 indirect shadows	3
65542/32418	Direct Lighting ONLY	173
65542/32418	30 Degree Angle Between VPL Rays	26
65542/32418	Reduction down to 1 VPL Per Ray	8
65542/32418	Resolution Reduction 640x360	1

Scene Complexity Integrated Shadows

Scene	NumSteps	Resolution	NumVPL's	FPS
6565/3249	2	1280x720	6485	28
6565/3249	6	1280x720	6485	23
6565/3249	10	1280x720	6485	20
6565/3249	2	1280x720	185	137
6565/3249	6	1280x720	185	63
6565/3249	10	1280x720	185	42
6565/3249	2	640x360	6485	31
6565/3249	6	640x360	6485	30
6565/3249	10	640x360	6485	29
65542/32418	2	1280x720	6485	3
65542/32418	6	1280x720	6485	3
65542/32418	10	1280x720	6485	3
65542/32418	2	1280x720	185	58
65542/32418	6	1280x720	185	39
65542/32418	10	1280x720	185	30
65542/32418	2	640x360	6485	3
65542/32418	6	640x360	6485	3
65542/32418	10	640x360	6485	3

Image Comparison
Accurate Shadows

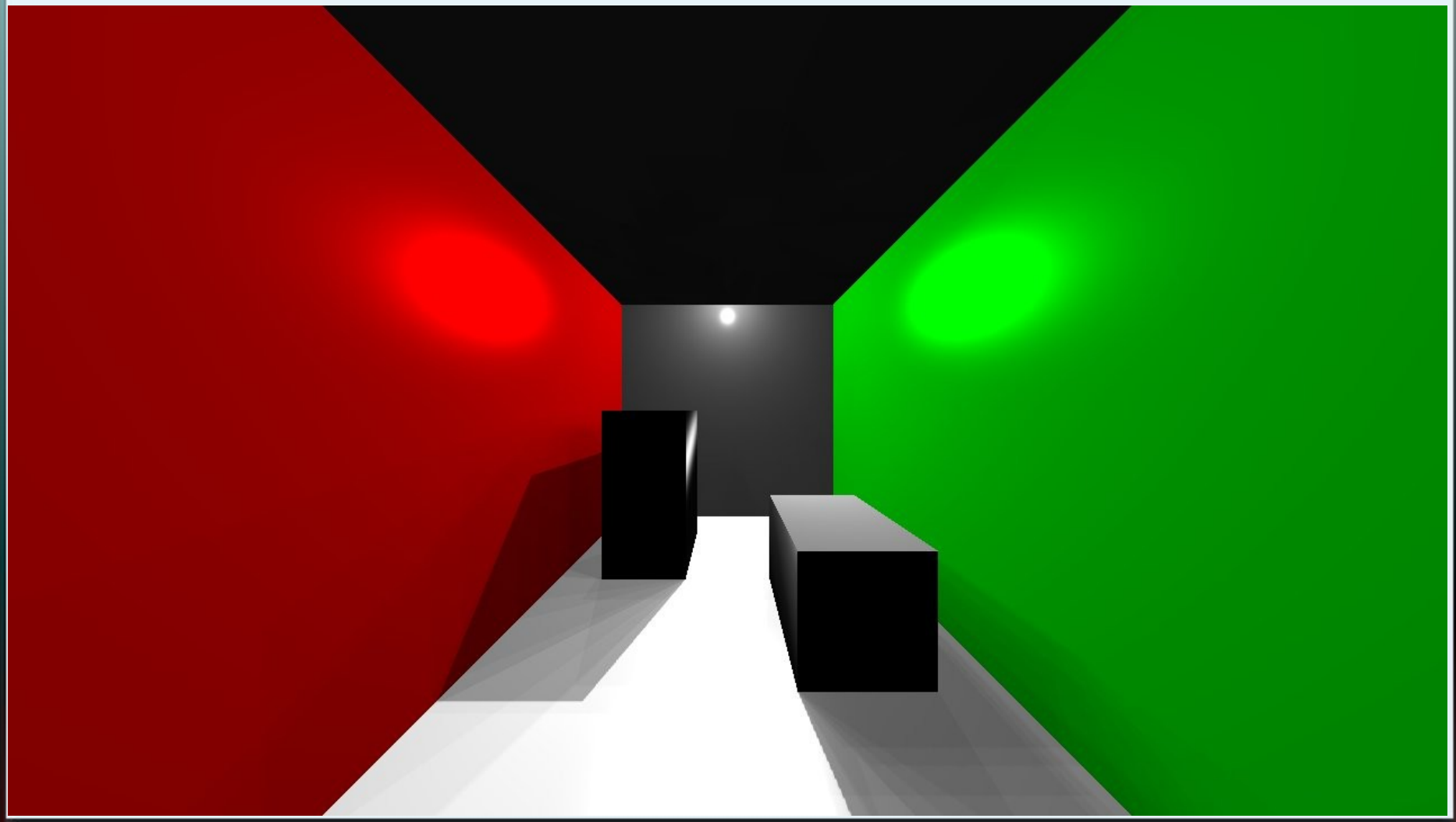
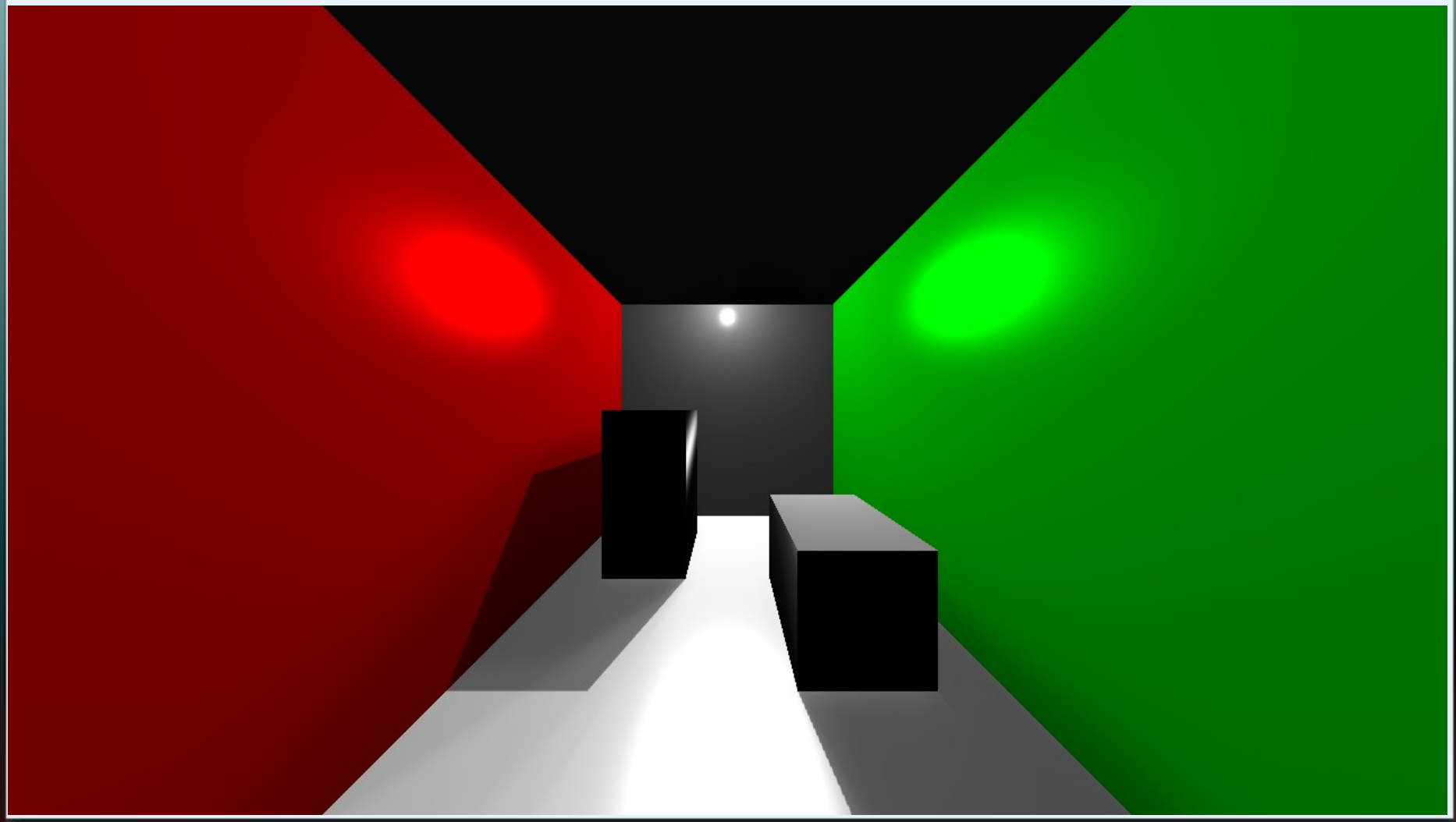
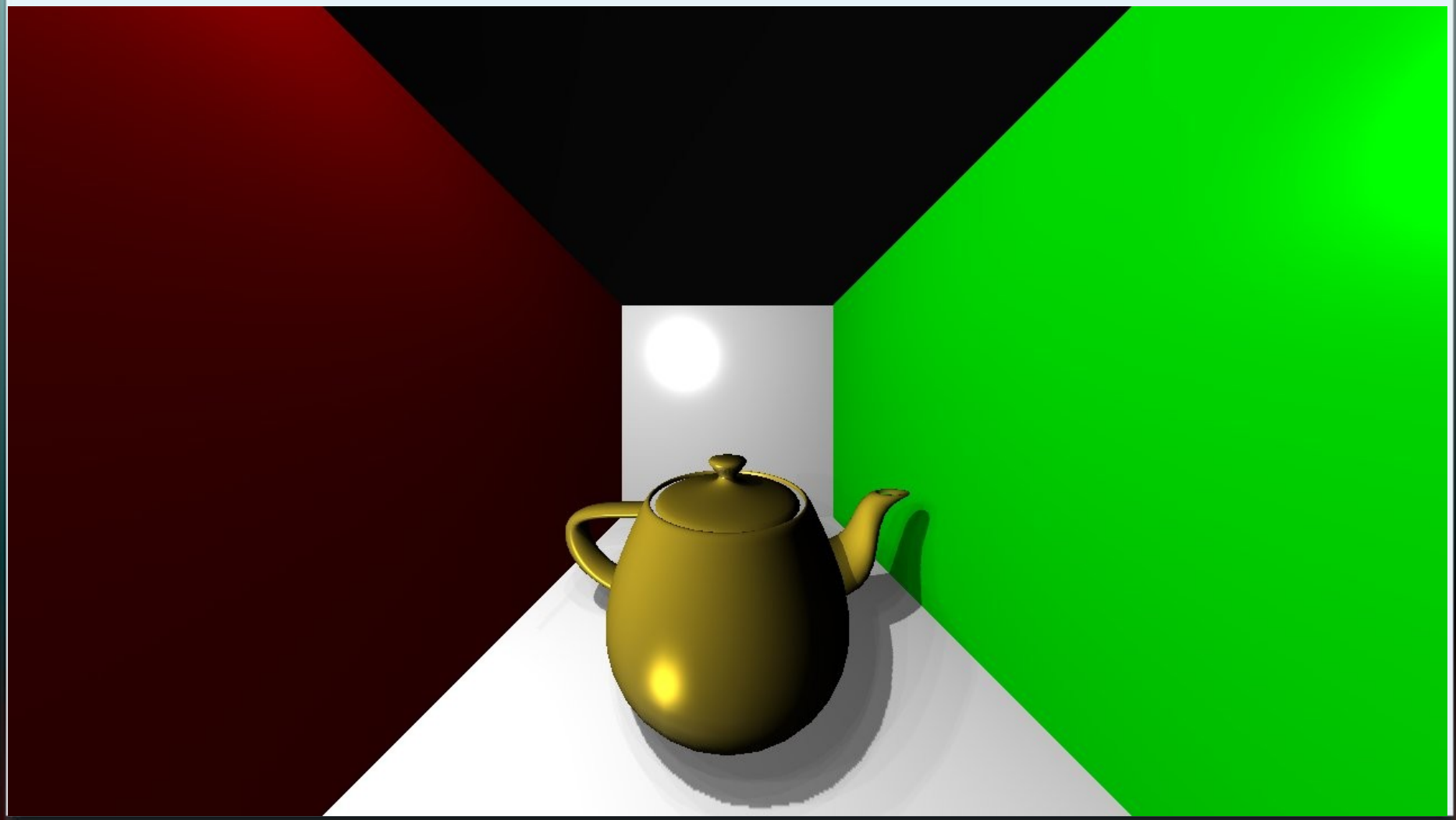


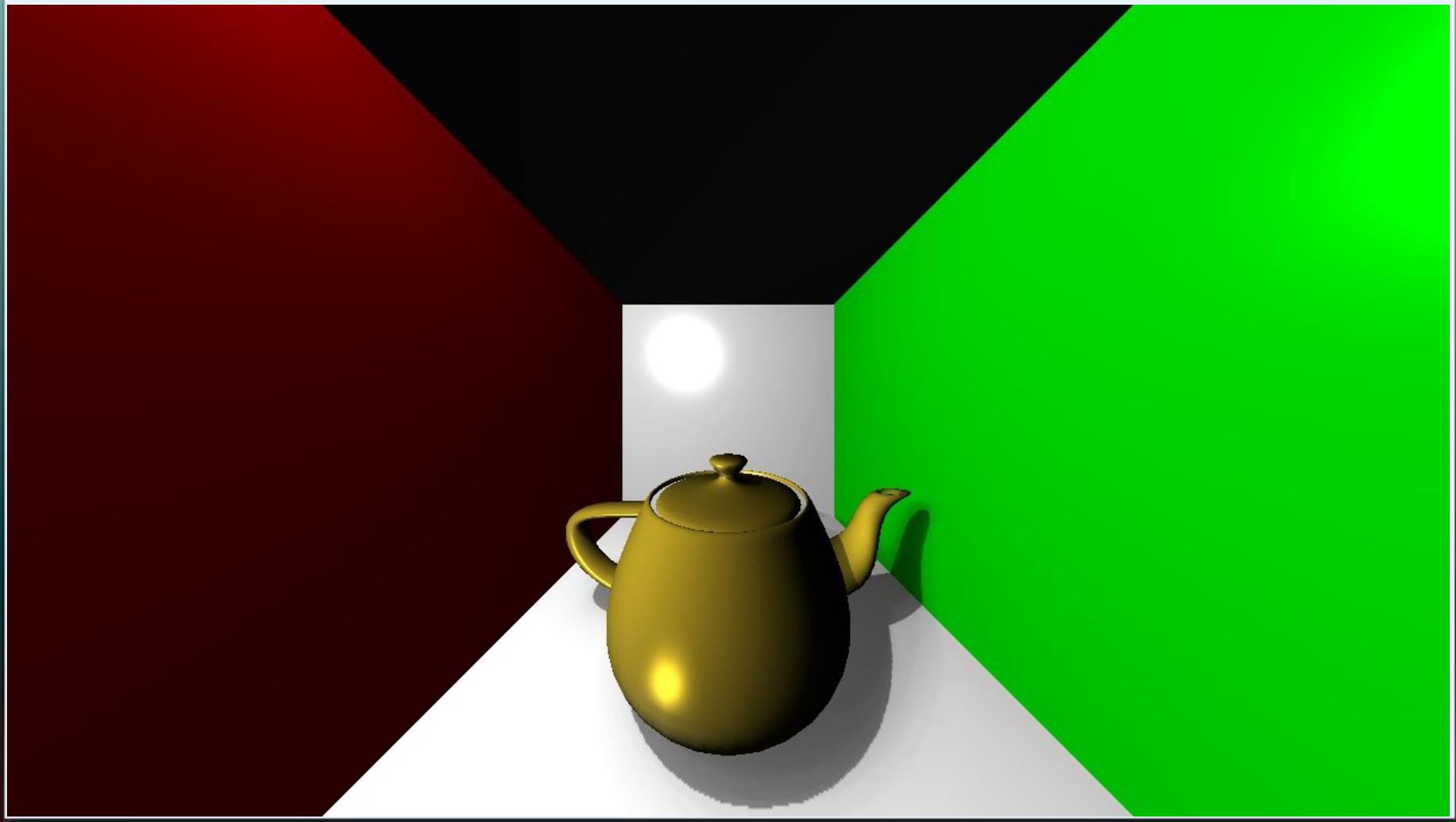
Image Comparison
Integrated Shadows



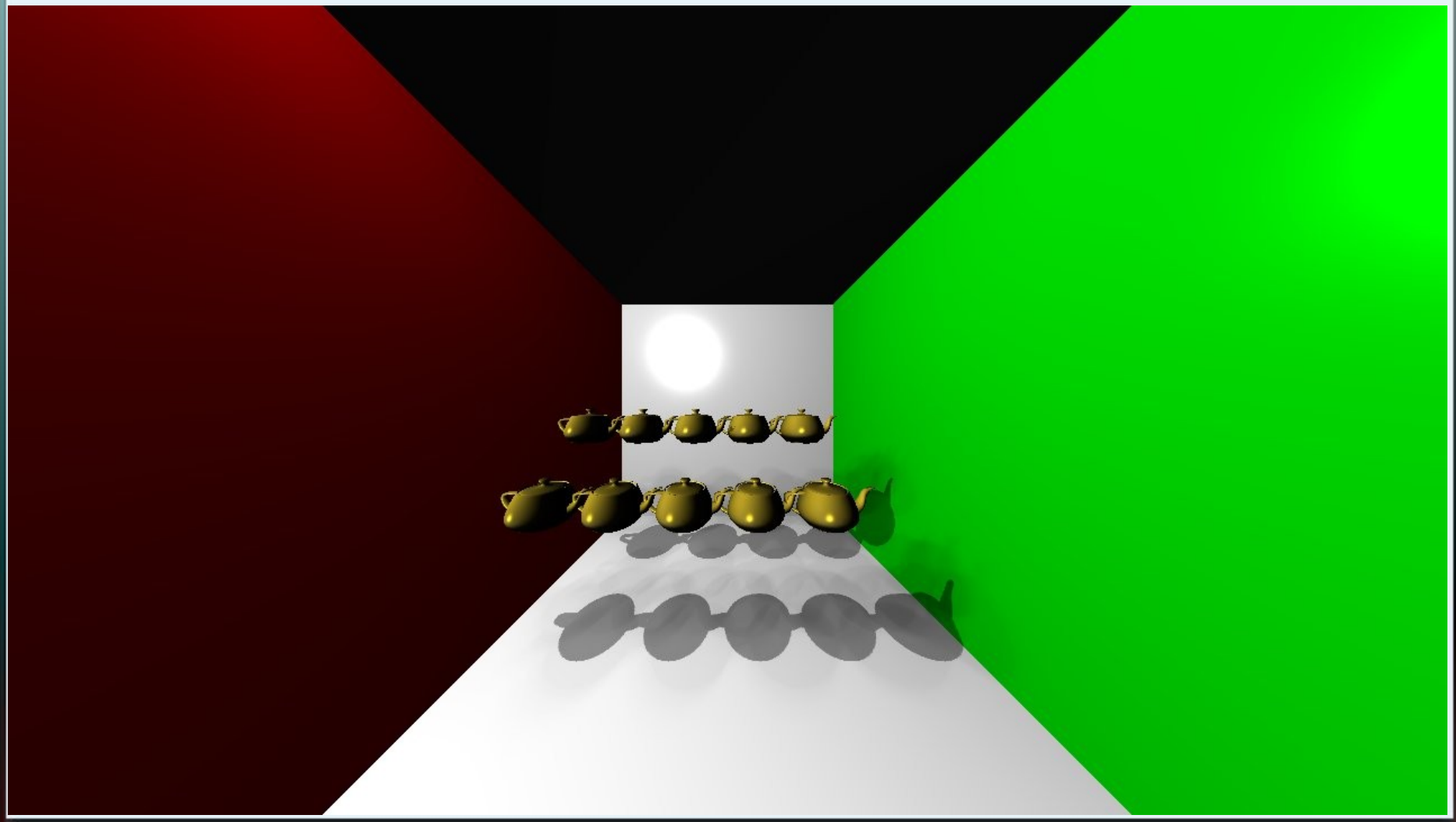
Scene Complexity
Accurate Shadows



Scene Complexity
Integrated Shadows



Scene Complexity
Integrated Shadows



Goals Met?

- Indirect illumination with indirect shadows (accurate shadows and integrated shadows)
- Fully dynamic light and objects (demo)
- Scalable (adjustable parameters to increase performance)
  - Scene Complexity(26-58FPS,65k+triangles)
  - Support variety of hardware capabilities (reduced memory version, demo on laptop)
- Implement using the GPU through the use of OpenGL and GLSL (done)

Conclusions

- Indirect shadows are expensive to compute.
  - Many shadows must be present in order to make them look smooth.
  - The number of accurate shadows we can support using this size of shadow maps is too limited and 1 shadow per shadow map is inefficient.
  - Best to approximate indirect shadows using some type of interpolation technique.
  - With further advancements in GPU memory, accurate shadows will get better.

Conclusions

- The integrated shadows gave better results than the accurate shadows in terms of FPS, memory use, and number of shadows and therefore appear more realistic since smoother shadows are more visually pleasing.

- With better GPU's, we can increase the number of shadow maps and shadows as well as increase the shadow map size and the number of interpolation steps.

Limitations

- Accurate shadow numbers are limited by the capability of the GPU.
- For fully dynamic scenes with high scene complexity, VPL count has to be lowered to limit the number of vertex calculations.
- Common VPL limitations: singularities exist at boundary edges such as where 2 walls meet resulting in darkening due to VPL's being at discrete locations.

Possible Future Improvements

- Both shadowing techniques will improve with GPU advancement.
- Use the idea of Virtual Ray Lights (Novak 2012). VRL's are different from VPL's in that they are line segments that are integrated across to remove singularities. Instead, I could use VPL's that are semicircles (wave-like) and integrate across them.
- Incorporate other related works not listed here such as matrix sampling or neighborhood optimizations.

Demo

- Desktop Hardware:
  - CPU: AMD Athlon 64 X2 Dual Core 5200+ 2.61GHz
  - GPU: Nvidia GeForce GTX 465 **(1GB mem)**
- Laptop Hardware:
  - CPU: Intel Core 2 Duo P8700 2.53GHz
  - GPU: ATI Mobility Radeon HD 3650 **(512 MB mem)**