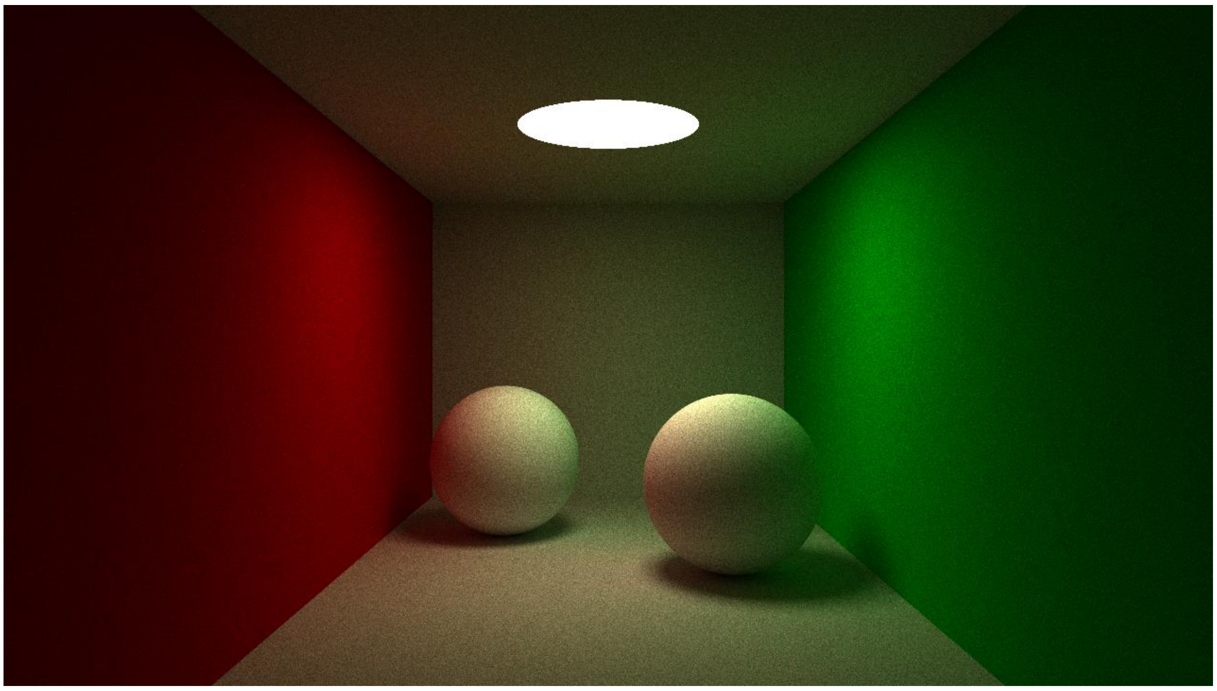
Path Tracer Practical Assignment

Advanced Graphics course 2012-2013, Utrecht University.
Marries van de Hoef



1. Overview

The goal of this assignment is to practice complex shader development, and to get a practical understanding of path tracing. Most of your time will probably be spent analyzing problems in the path tracer shader.

You will create a basic, but functional path tracer which uses the parallel processing power of the GPU. The project should be built on the simple framework you have created in the Mandelbrot assignment.

The knowledge and experience you obtain through this assignment prepares you for the future of graphics in games, which lies in massively parallel programming. Additionally, some parts have a more tangible relevance to game development.

The assignment is less detailed than the Mandelbrot assignment, so you are required to do more research yourself. The gray parts in the assignment are identical to the previous assignment, but <u>no less important</u> than the white parts. They give you a quick overview of the differences.

2. Organizational Matter

Deliverables

- The source code. It should compile directly in VS2012 (no external dependencies). Do not submit intermediate build files and other temporary files.
- The executable and any dependencies. It should run directly.
- A screenshot or link to a youtube video.
- Report.

Report

The report should contain the following elements:

- The student number.
- Additional references of any type used for the assignment.
- Instructions for the application.
- Listing and a short description of the implemented additional features.
- A short description of the code structure.
- The location in the code where the rays are generated using the view-projection matrix.
- A short explanation why your implementation is unbiased (including references to the code).
- A short theoretical analysis describing the hardware performance bottleneck(s) of the application.

Grading criteria

The grading criteria are the following, roughly in order of importance:

- Compliance with the deliverable requirements and technical requirements.
- Presence of the required features.
- Quality of the report.
- Absence of bugs and glitches.
- Code structure (object orientedness, etc.).
- Code quality (e.g. the layout clarity, variable naming clarity, commenting).
- Implementation quality (polish).

Grading and additional features

The grade depends on the grading criteria mentioned above and the additional features you implemented. The additional features can be selected from the list in this document. You are also free to think of features yourself, but make sure they are relevant to the goal and focus of the assignment. Note that the difficulty of additional features is more important than the quantity.

A research attitude towards optimizing your path tracer is highly encouraged.

Submission

The final submission deadline is on Sunday April 21st at 23:55.

Submission is done through the Submit system: http://www.cs.uu.nl/docs/submit/

3. General Remarks

Internet sources

You will probably need additional resources from the internet. Be sure not to copy any code, or code structure. Instead use the internet to learn HOW something works, and implement it yourself. There is only one exception and that is the random number generator, but don't forget to reference your source.

The source code of several path tracers can be found on the internet. Do not consult these, unless you want to resolve a specific issue with your path tracer. When looking at other source code while constructing your path tracer, you risk imitation and therefore (subconsciously) plagiarizing the other implementation.

In general, don't forget to reference your additional sources in the report (when you use them for more than answering one small question).

By following the above approach, you will ensure that you have a good understanding of the choices you made in the code. If there are doubts that you may have plagiarized, you will be requested to explain the choices made in your implementation in an interview.

Academic dishonesty

Fraud or plagiarism has to be reported to the Board of Examiners, as required by the Education and Examination Regulations. This includes for example copying code from the internet or other students, letting other students create (parts of) your assignment, letting other students copy code from your assignment, and reusing previously written code (without the teacher's permission).

Code quality

The grading criteria include the code structure and code quality. The main reason is because it shows that you understand what you are doing and that you care enough to express it.

Quality in code is not expressed by using difficult or obscure syntax constructions to show your skill, but by creating code that is easy to understand and maintain while still performing (near) optimally. You can use any coding *style*, as long as it does not conflict with the previous properties.

Support

You are encouraged to post questions on the forum when (for example) you are stuck, the assignment is unclear, or for advice on how to solve a specific problem. You are also encouraged to answer the questions of other students.

4. Technical Requirements

You have to use C++ and the Direct3D 11 API. The Direct3D 11 API can also be used on older hardware. For the assignment, you will have to use Shader Model 4.0 and therefore you need at least DirectX 10 compatible hardware.

Note that Direct3D 11.1 features may not be used (Direct3D 11.0 is fine), and neither can double precision be used. (My GPU does not support it, so your submission can't be graded then.) Additionally, you should not use the WinRT API for Windows 8 “metro” style applications.

Visual Studio 2012

You have to use Visual Studio 2012 as programming environment. There have been significant changes in this last iteration, which are favorable for this assignment. The DirectX SDK has been integrated in the Windows 8 SDK, which is part of Visual Studio 2012. There have been some significant modernizations in the integrated DirectX SDK. Additionally, VS2012 has an integrated Direct3D debugger which can be helpful. The Windows 8 SDK and VS2012 run fine on Windows 7.

Visual Studio 2012 can be obtained through DreamSpark: https://ict.science.uu.nl/index.php/MSDN

Libraries

The only additional library you need is a math library. Consult me first if you want to use other libraries (for example for additional features). Make sure that the library dependencies are stored locally in the submission, so the project can be compiled directly.

Submission

Before submitting your code, delete any intermediate build files and other temporary files. If your code is larger than 1MB, you probably forgot to delete something.

Assignment Description

You will create a simple unbiased path tracer which is entirely executed on the GPU. The user has to be able to move interactively through the environment. The path tracer is reset when the camera moves. The rays have to be generated using the view projection matrix.

The path tracer must be able to display the provided scene for comparison purposes.

The assignment is broken down into several parts which are discussed in the next sections:

- Generate rays
- Ray tracer
- Interaction
- Incremental refinement
- Path tracer
- Additional features

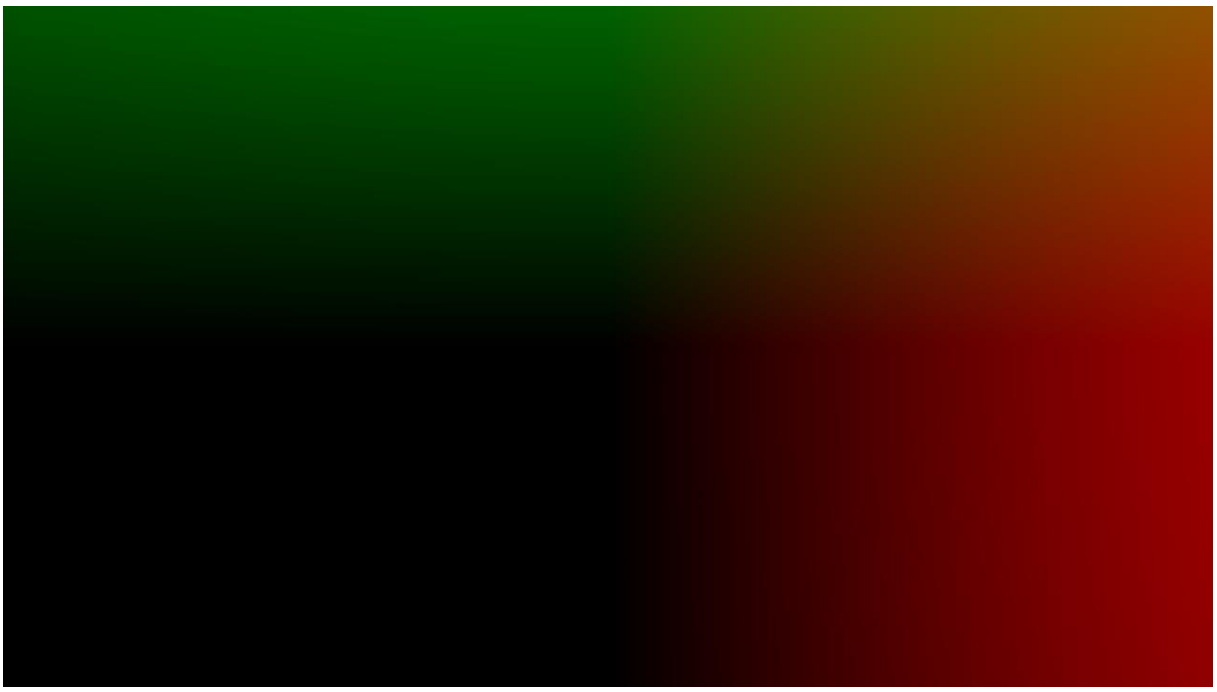
5. Generate Rays

Use the framework you created in the previous assignment to create the basis. You need to be able to run a shader for each pixel inside the window.

Generate rays from the camera position according to the view projection matrix. Recall the transformations normally executed in the vertex shader to project the 3D world onto the screen. To generate a ray from the camera through a specific pixel, we will use the *inverse* of these transformations. This allows us to obtain a point *A* in 3D space which would be projected to that specific pixel. By subtracting the camera position from *A*, we get the direction through this pixel.

To generate and invert the view projection matrix, you are advised to use a math library. For details on the DirectXMath library, refer to the previous assignment.

When displaying the direction as a color (for a right handed coordinate system using a default orientation), the resulting image should be similar to this:



6. Ray Tracer

You only need to implement ray-sphere intersection. To refresh your memory, consult the Ray Tracing 1 lecture from the Graphics course:
http://www.cs.uu.nl/docs/vakken/gr/2011/gr_lectures.html

You should use this scene description in your shader:

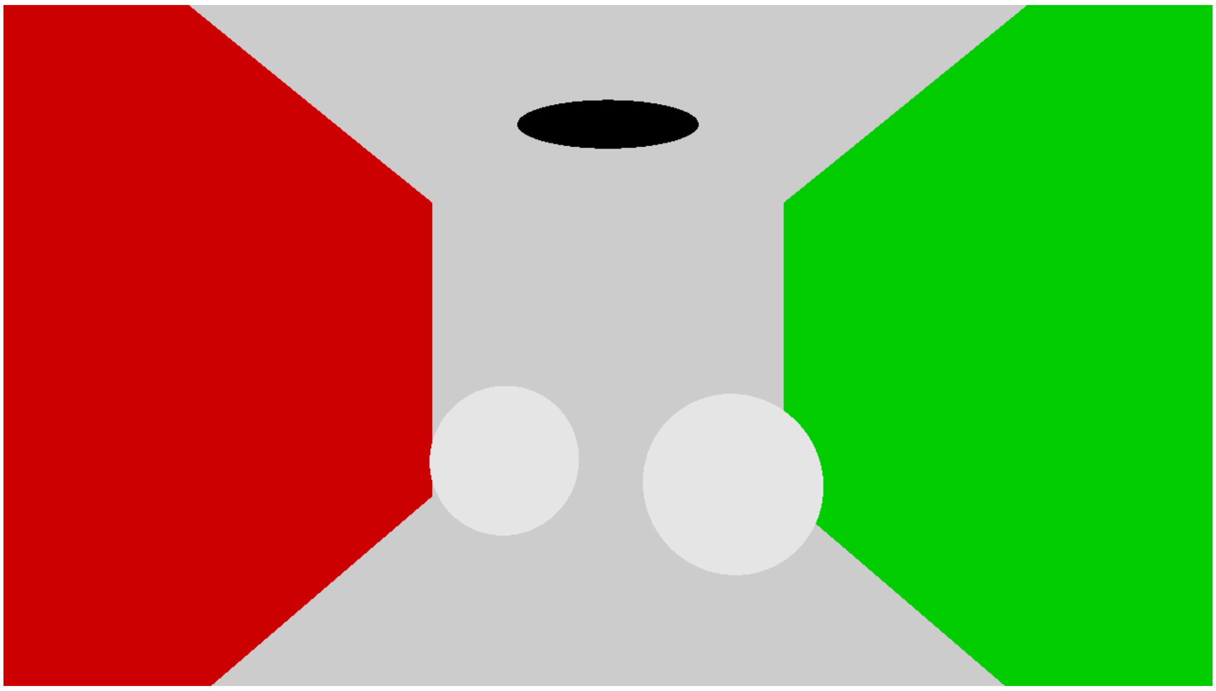
```
struct Sphere
{
	float3 Center;
	float Radius;
	float3 Albedo;
	float3 Emission;
};

static const int NumSpheres = 9;
static const float Wall = 1e4;

static const Sphere Cornell[NumSpheres] =
{
 {float3(Wall+49.0f, 21.2f,-149.0f),Wall,float3(0.0f,0.8f,0.0f),float3(0,0,0)},//Right
 {float3(-Wall-49.0f,21.2f,-149.0f),Wall,float3(0.8f,0.0f,0.0f),float3(0,0,0)},//Left
 {float3(0,21.2f,Wall+10.6f),Wall,float3(0.8f,0.8f,0.8f),float3(0,0,0)},//Front
 {float3(0,21.2f,-Wall-230.6f),Wall,float3(0.8f,0.8f,0.8f),float3(0,0,0)},//Back
 {float3(0,Wall+39.6,-149.0f),Wall,float3(0.8f,0.8f,0.8f),float3(0,0,0)},//Top
 {float3(0,-Wall-42,-149.0f),Wall,float3(0.8f,0.8f,0.8f),float3(0,0,0)},//Bottom
 {float3(-23,-25.5f,-183.6f),16.5f,float3(0.9f,0.9f,0.9f),float3(0,0,0)},//Left Sphere
 {float3(23,-25.5f,-152.6f),16.5f,float3(0.9f,0.9f,0.9f),float3(0,0,0)},//Right Sphere
 {float3(0,639.6-.21,-149.0f),600.0f,float3(0,0,0),float3(12,12,9)},//Light Source
};
```

The scene is based on work by Kevin Beason.

You are free to modify the scene however you like, as long as there still is an option to show (and to move through) this exact scene. Note that the formatting is not important, but the numbers are. Your submission will be tested for correctness by comparing the results of this path traced scene to a reference implementation.

Remember to visit the forum and to ask questions. It will save you a lot of time (and headaches).

The ray traced image should look like this (with an aspect ratio of $\frac{16}{9}$, and a field of view of $\frac{\pi}{4}$):



7. Interaction

Allow the user to navigate freely through the environment.

Make it user-friendly. An example of a user friendly is FPS-style navigation (WASD for movement, mouse for looking around). An additional user-friendly element is capturing the mouse within the screen and releasing it when pressing escape. Note that this is just an example, you are free to use a different control scheme as long as all positions and orientations can be reached.

Make the interaction frame rate independent to guarantee the same experience on different hardware.

8. Incremental Refinement

It is not feasible to execute all path tracing iterations in a single shader pass because this requires too much time. Instead, we will spread the iterations over an indefinite number of frames.

Blend the rendered image with the previous image in such a way that the result is the average over all frames.

Try to figure out how to implement this by yourself and do some research on the internet. If you have questions or need advice, go to the forum.

9. Path Tracing

Use your own knowledge on path tracing to structure the algorithm. Check the slides to refresh your memory. Don't forget to implement the algorithm in small steps. Creating everything before you start testing will make debugging very difficult.

Your path tracer must be unbiased, and therefore must use Russian Roulette termination.

The sections below provide directions for specific aspects of the algorithm.

Random number generator

You will have to generate random numbers within your shader. You can use an implementation from the internet, but don't forget to properly reference your source.

Thorough information on GPU random number generators, including code:
http://http.developer.nvidia.com/GPUGems3/gpugems3_ch37.html

Source code for a simple random number generator can be found on Wikipedia:
http://en.wikipedia.org/wiki/Random_number_generator#Computational_methods

To convert random integer numbers to a float value in 0-1 range, you can do some bit-wise operations. Refer to the bit representation of floats: http://en.wikipedia.org/wiki/Single_precision

Random hemisphere sampling
This link provides several methods to create a random sampling on a sphere:
http://mathworld.wolfram.com/SpherePointPicking.html

Reference image
The image at the start of the assignment can be used as reference.

10. Additional Features

Implement additional features to enhance your path tracer. You are free to choose the features you want to implement from the list below. You can also think of features yourself, but make sure that they are relevant to the goal and focus of this assignment. More specifically, try to focus on improving the convergence speed and supported features of the path tracer. If you're in doubt if your feature suffices, consult me.

In general, difficulty is preferred over quantity. Original additional features are encouraged, as long as they meet the above requirements. A scientific attitude towards optimizing your path tracer (for example by doing experiments and adding them to the report) is highly encouraged.

Don't forget to mention and shortly describe your implemented additional features in the report. Add a short motivation if you implemented a feature which is not in this list. Make sure that the original scene can still be rendered for comparison purposes.

Additional information regarding path tracing can be found here:
http://www.cs.rutgers.edu/~decarlo/readings/mcrt-sg03c.pdf

Ideas
This is a list of ideas for additional features, roughly in order of increasing difficulty.

Anti-aliasing
Implement anti-aliasing by perturbing the ray direction within the pixel.

Cosine sampling
Implement importance sampling for the Lambertian reflectance function.

Mirror material
Implement perfect specular reflections.

Triangles
Add support for triangles in the scene.

Refraction
Implement refracting materials.

Depth of field
Implement depth of field by taking the camera lens into account.

Explicit light sampling
Add importance sampling which samples in the direction of light sources. This reduces variance effectively in common scenarios. Make sure that your implementation is still unbiased.

BRDF importance sampling
Add a reflectance function with specular highlights, and implement importance sampling for this reflectance function.

Geometry acceleration structure
Implement an acceleration structure for the geometry to reduce the number of intersection tests in complex scenes.

Compute Shader with path regeneration
Regenerate paths on termination to improve SIMD usage. To improve coherence, you can implement stream compaction.

Participating medium
Implement subsurface scattering for translucent materials.

Bidirectional path tracing
Implement bidirectional path tracing to improve convergence speed in difficult light conditions.