Marble Rendering

Zahra Ghavasieh

Introduction

Marble is a beautiful stone with a unique texture. This project attempts to capture this texture and its characteristics realistically and apply it in a real-time setting.

The simplest solution to this problem is to use texture mapping to map an image onto the model. However, this technique does not apply to every situation and lacks customization capability. This project would instead use procedurally generated textures as well as deferred rendering to apply approximations to subsurface scattering and reflections. A combination of these three techniques is believed to create the best visualization of the Marble stone.

Motivation

In real-time rendering, textures are most commonly implemented using texture mapping. Although this is usually enough for a simple preview, it is not very realistic. Not only is the texture image itself difficult to procure, but it may also not wrap around the object well. Marble objects are cut from the stone directly, so the veins must wrap around all the edges. Furthermore, the user does not have as much freedom when customizing a texture image.

This problem can be solved using procedural texturing. Many modelling softwares such as Blender already implement functions that can be used to create a procedurally generated marble texture. However, in these cases, procedural textures are more generalized so one must produce the desired texture manually through trial and error. There is no standard function to simply apply a marble texture. Furthermore, the end results may not appear as realistic as intended.

A marble texture must be easy to apply to a model while also be customizable without losing its realistic appearance.

Project Summary

This project aims to create a customizable marble texture in a real-time setting. It will make use of Perlin Noise to create a procedurally generated texture, the Schlick Fresnel Approximation to add reflections, and a subsurface scattering approximation technique to add more realism to this material.

PROJECT DETAILS

I. Architecture and Environment

This project will be using the GPU to produce real-time rendering. It will be implemented in C++ using the OpenGL graphics library as this is a familiar environment. Therefore, efforts will be focused more on the marble rendering portion of the project rather than the basic graphics pipeline.

II. Implementation Issues and Challenges

Creating a realistic rendition of the marble texture using procedural generation will involve much trial and error. Various approaches will need to be implemented and compared to produce the best results. Furthermore, to create enough customizability for the user while also providing a strong basis for marble textures will be a challenge.

Another issue may be the combination of the reflection and subsurface scattering approximation techniques. Deferred rendering may be a good solution for this problem, but it will take further tuning to find a perfect balance between these techniques.

III. Deliverables

This project will produce a simple program where the user can import a model and apply the marble texture in real-time. They will also have a degree of freedom concerning the appearance of the marble. This may range from the colours to the shape of the veins or even its level of transparency as seen in certain marbles.

A basic program will render a singular marble texture using three techniques in combination. These techniques include procedural texture generation using a noise function such as Perlin Noise, a subsurface scattering approximation, and a Fresnel reflection approximation such as the Schlick Approximation. If time permits, multiple examples of a marble texture will be made available, and the user will have more freedom in terms of its customization.

As the implementation will take multiple trials, each of the applied rendering technique will be thoroughly explained in a final report document. This project aims to simply produce a single or a range of realistically rendered marble textures and may be used in other projects to enhance a certain scene or further built upon for research purposes.

Finally, the progress of this project will be documented online on the following website: https://judgyknowitall.github.io/marble-renderable/

IV. Timeline

The deadline for the project demo is April 11th while the report is due a couple of weeks after. To meet these deadlines, this project will be implemented during a three-milestone schedule. Each milestone will be clearly tagged and documented on GitHub pages.

1. The first milestone will be due on March 20th. At this point, the project structure must be set up and the program must be able to load in 3-dimensional models. The user must also be able to view their model from multiple angles using keyboard and/or mouse inputs. A basic procedurally generated texture may be implemented but not yet finetuned.
2. The second milestone will be due on April 3rd. One or more textures must be implemented and can be applied to any model. A basic implementation of reflections and subsurface scattering may also be applied.
3. The third and final milestone will be due on April 10th. As this is the last milestone, the project must have fully implemented all three rendering techniques and provided some degree of user-controlled customization.

CONCLUSION

To summarize, this project aims to produce realistically rendered marble textures in real-time by applying three rendering methods. These methods include procedural texture generation, a Fresnel reflection approximation, and a subsurface scattering approximation technique. The program will be implemented in three milestones spread over a five-week period. Each milestone will be documented on the GitHub Page and the results will be made available on the final report.

References

- GitHub Pages Website: https://judgyknowitall.github.io/marble-renderable/

- Understanding Marble textures:
  - https://youtu.be/MZLbbmdCAQE
  - https://www.csie.ntu.edu.tw/~cyy/courses/rendering/pbrt-1.04/doxygen/html/marble_8cpp-source.html

- Perlin Noise / Procedural Texturing:
  - https://observablehq.com/@toja/procedural-texturing-w-perlin-noise
  - https://lodev.org/cgtutor/randomnoise.html
  - http://physbam.stanford.edu/cs448x/old/Procedural_Noise(2f)Perlin_Noise.html

- Reflections:
  - https://belcour.github.io/blog/slides/2020-brdf-fresnel-decompo/index.html#/5/0/0

- Subsurface Scattering:
  - (PDF) Image-Space Subsurface Scattering for Interactive Rendering of Deformable Translucent Objects (researchgate.net)
  - https://developer.nvidia.com/gpugems/gpugems/part-iii-materials/chapter-16-real-time-approximations-subsurface-scattering