NPRG045 Project Proposal

Student: Vojtěch Pröschl
Supervisor: Mgr. Vojtěch Tázlar

September 11, 2022

1 Abstract

The project aims to create a path tracer, which will be distributed as a plugin for the software SideFX Houdini. The plugin will be represented as a node in the Houdini's rendering context. The user will be able to control the renderer's output using that node, i.e., selecting geometry target, camera and adding lights to the render, and changing render properties (e.g., sample count). The renderer will also support some of the Houdini's internal nodes as its input, so the user will have minimal work in order to render the scene using this render.

2 Features

2.1 Input

Geometry Users will be able to use custom geometry in the form of Houdini's SOP nodes containing polygonal data.
It is possible to assign specific attributes to geometric elements in Houdini, which can modify the scene's appearance. The render engine will be able to read and utilize a few of them

Color geometry attribute representing point color "Cd" will be supported by the rendering engine.

Normal geometry attribute representing vertex normal "N" will also be supported - even more, it will be required to render the geometry correctly.

Shader geometry attribute representing shader "S" assigned to the primitive will be supported..

Lights User will be able to add lights to the scene, which will modify the appearance of the scene. Lights will be represented via Houdini's nodes linked to the renderer node.

Camera User will be able to change camera properties, such as its position, rotation, and its field of view. The camera will be also represented with Houdini's node that will have to be linked to the renderer node.

2.2 Output

Render frame The renderer will be able to render a single frame - its number will be specified in the rendering node. It will then be rendered via the render frame functionality.

Render animation Rendering animations will also be possible - the user will specify the frame range which should be rendered in the rendering node. It will then be rendered via render frame range functionality.

Export Images will be rendered into the MPlay window, which has the functionality to export images (or their sequences). Therefore this functionality will be satisfied by rendering the images directly into Houdini's rendering window called MPlay.

2.3 Path Tracer

Basic functionality Path tracer will utilize Monte Carlo methods to compute scene illumination.

Global illumination The path-tracer will be able to compute global illumination.

Reflections The path-tracer will be able to compute light reflections, which will behave according to a certain BRDF. The shader will drive that.

Refractions The path tracer will be able to compute light refractions. It will be driven by the index of refraction and transparency coefficients provided by the shader.

Light The path tracer will support area lights of rectangular shapes.

Camera The path tracer will support a camera that has an adjustable focal length, position, and rotation.

Sample count Users will be able to modify the number of samples per pixel, which will alter the quality of the output picture.

2.4 Shading

Custom C++ Shaders API will allow the programmer to implant his own shader into the scene. The shader will be added to the renderer's internal shader hash table. The user will be able to add "S" primitive attribute to the geometry, which can link the shader in the hash table to the geometry.

List of shader attribtues

- Color
- Normal
- Roughness
- Transparency
- Index of refraction

Basic BRDF Part of the project will be an implementation of basic BRDF, which will be able to source rays according to a particular probability distribution. Also, it will be possible to evaluate the BRDF.

2.5 Interface

Utilising existing nodes Users will be able to use nodes in the Houdini. This includes the Camera, Light, and Geometry node. Those nodes will be supported only partially by the renderer.

Custom render node To use the render, the user will have to create a custom node representing the render. Its interface will allow the user to do various things.

Setup the scene User can add the lights, camera, and geometry to the scene using the rendering node.

Change rendering settings user will be able to change rendering settings in the node, i.e., sample count, frame, and frame range to render.

Render the scene Most importantly user will be able to render the scene from the render node. The rendered scene will be displayed in the MPlay window mentioned before.

3 Technologies

3.1 Target platform

The primary platform for which the project will be developed is macOS and Ubuntu.

3.2 Programming language

C++ will be the primary programming language used to solve the problem alongside CMake. Other programming languages such as Bash will likely be used for some utilities.

3.3 Framework

The main framework to solve the problem will be HDK which is the shorthand of the Houdini Development Kit. It is a set of libraries used directly by developers of commercial 3D software SideFX Houdini. Those libraries provide the programmer with an API to communicate with Houdini's interface.

3.4 Houdini

Houdini is a commercial software used to solve various tasks in the area of visual effects, motion design, and more. It provides a set of procedural tools which is unrivaled in the industry. The solution will be presented as an extension to Houdini's library of nodes - the user will create the node in the rendering context, and it will communicate with ot