RENDERING OF CALCITE CRYSTAL

MASTERS PROJECT

Advaita Waikar
i7916121

MSCAVE NCCA 2010-11

INDEX

LIST OF FIGURES

Chapter 1

Introduction

Crystals are visually very appealing. They are very unique in some of the properties they exhibit and also in optical effects they create. The different variations in the types of crystals make it difficult to create a specific shader to suit them all. This is especially true if the various optical effects are to be included. After careful consideration and research over the various types of crystals, I choose a calcite to work with for this project. I aim to recreate a calcite crystal, which has some unique properties such as birefringence.

Other important factors towards achieving a realistic crystal such as specularity, reflection, refraction would also be included and creating a sense of depth. It was also important to use efficient and fast, rendering methods since working with finite resources is important in the industry.

Chapter 2

Previous Work

From a visual aspect, calcite crystals have some similar physical properties as many other materials such as ice, gemstones, as they are polymorphic, translucent as well as amorphous [PR].

Weidlich, on a paper on birefringence produces successful results on combining two refractive rays where accurate physics are incorporated in the rendering process. [WE]
The figure below shows the comparison between a real calcite crystal (left) and a rendered one on the right. The standard equations of physics applied to this were optimized for faster render times.

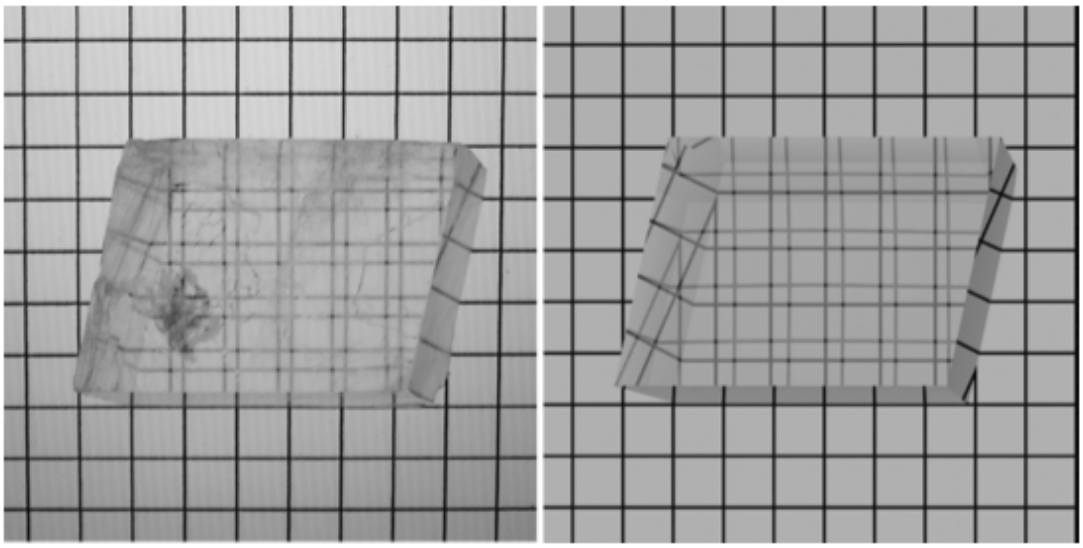


Figure 1: On the left - a real colourless calcite.
On the right - a rendered image of birefringence (double refraction). [WE]

Techniques used on other translucent materials were researched.

Pixar used diffused and scattered illumination for shading various food products in the movie Ratatouille. Using a combination of a brickmap and rgb irradiance gave very good results for softening the look of the object. [XE]

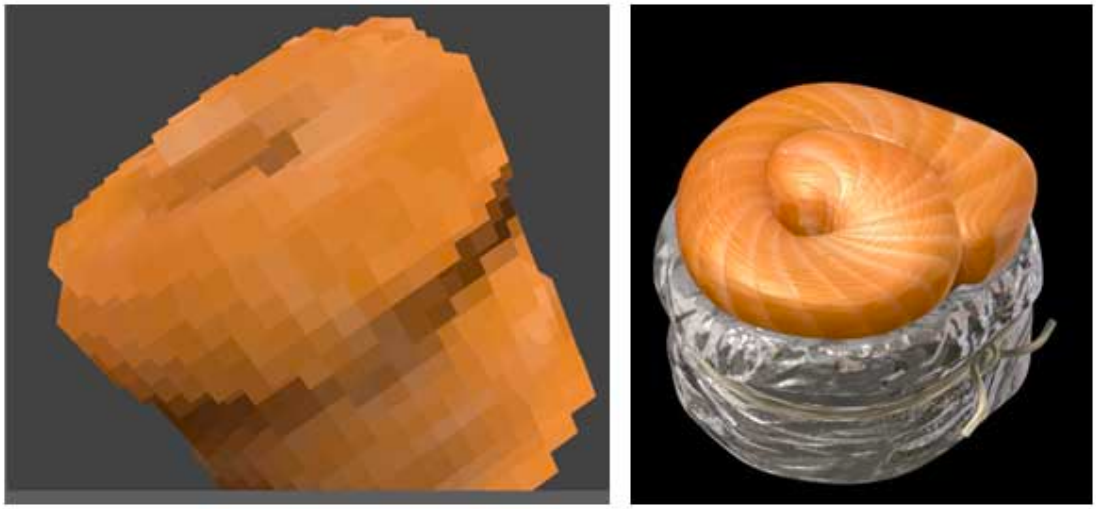


Figure 2: Brick map (on left) and rendered image (on right) used in Ratatouille. [XE]

Rushes used a point bake subsurface scattering method for producing realistic renders for organs and other organic materials for the show 'Inside the Human Body'. [RU].

WETA researched on a new quantized diffusion model that gives remarkable results for translucent materials. It is more accurate for materials, which are more absorbent. They also provide alternatives to the more traditional route of point cloud based subsurface scattering. Data is also stored in a new system that is easy for traversal. [DE11]

Figure 3: Render of cream milk using quantized diffusion model made by WETA. [DE]

Translucency effects have also at times been faked due to large the computing complexity and long render times. In Pirates of the Caribbean, many of the backgrounds containing ice and other complex materials were matte painted. [FO]

From the overall research, it was apparent that a successful and feasible method to go forward would be to use the two-pass subsurface scattering method using point clouds. It has also been used in many movie productions including 'Harry Potter and the Chamber of Secrets' and 'Pirates of the Caribbean 2'. [PR05]

Chapter 3

Technical Background

Calcite can have various appearances based on its use and chemical composition. Calcite ($CaCO_3$) is a mineral formed from carbon, which is a type of limestone. Hence it is widely used in construction as marble or limestone. It has other uses in different fields such as an acid neutralizer, sorbent, chemical fertilizer etc.
Visually, calcite has a vitreous or glassy lustre. Hence, it is analysed and implemented in a approach similar to that of rendering glass, ice, marble etc. which has similar properties of translucency, refraction etc. [GE]

3.1 : Physical Structure and Appearance

Calcite typically, has a trigonal-hexagonal structure. Its appearances vary from being chalky to glassy. Calcite is generally white or colourless but also appears in yellow, brown, and pink shades. With the exception of colourless calcite, it has a high rate of absorbing light. This gives the calcite different hues. When light passes through the crystal, some frequencies of the light wave are absorbed which result in the colour. It is also prone to exhibit phosphorescence occasionally. [GE]

Calcite is a dielectric and hence can be polarized when an electric field is applied. The images below showcase its diverse appearances.

Figure 4 : Calcite in the form of oolitic limestone. [GE]

Figure 5 : Transparent Calcite [GE]

Figure 6 : Transparent calcite showing double refraction [GE]

Figure 4 shows its chalky and more opaque rock-like appearance.
In figure 5, we can see that calcite has a yellowish shade due to absorption of light and has a marble like texture. Figure 6 has a colourless calcite showing the properties of double refraction. More on this will be discussed in the next topic.

3.2 : Optical Properties

Calcite shows uncommon and interesting properties such as birefringence and polarization effects.
The refractive index of a material is the factor by which the electromagnetic radiation speed decreases when it travels through a material or rather inside it. This loss in speed is relative to that in a vacuum. When a wave travels between two mediums with differing refractive indexes (at an angle), the phase velocity or speed of the wave changes. This causes the direction of the wave to change. This phenomenon is known as refraction. As explained before, calcite is a dielectric and hence an incident ray of light changes its direction when it travels through the material. The resultant ray is called refracted ray. A calcite crystal is also uniaxial as two of the components of its dielectric constants are equal. [PA]

Birefringence is explained with the figure below.

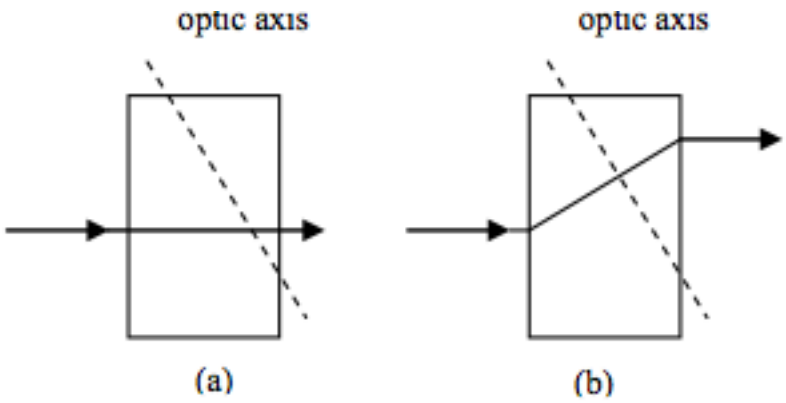


Figure 7: Ordinary and extraordinary ray [PA]
(a) shows an ordinary ray
(b) shows an extraordinary ray

In (a), the incident ray is linearly polarized perpendicular to plane formed with the optic axis and the direction of incidence. The ray simply passes the material with the direction unchanged and is known as the ordinary ray (o-ray) with the refractive index (No). It also complies with Snell's Law.

In (b), the incident ray is linearly polarized parallel to the plane formed with the optic axis and direction of incidence. The direction of the wave is deflected with respect to the incident ray. The resultant ray, which has an independent refractive index (Ne), is known as an extraordinary ray (e-ray).

Thus, each type of ray has a different refractive index and it results in a doubling effect when an object is viewed through the crystal (as seen in figure 6).

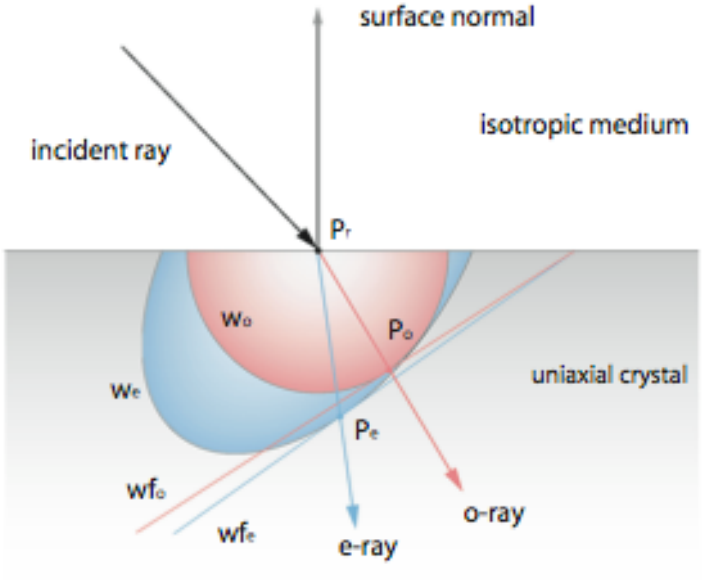


Figure 8 : Refraction in calcite [WE]

As illustrated in the figure above, a new direction cosine has to be obtained for the extraordinary ray. The two critical angles are calculated from the incident ray, RI of o-ray (No) and RI of e-ray (Ne).

$$\theta_{e1} = \frac{\left(n_e^2 - n_o^2\right)\alpha\left[n_o\beta\sin\theta_i + \gamma \cdot G\right]}{\sqrt{n_e^2\left[n_e^2\left(n_e^2 n_o\gamma^2 - n_o^3(\gamma^2-1)\right)^2 + (n_e^2 - n_o^2)\cdot G\right]^2}}$$

and

$$\theta_{e2} = \frac{F \cdot G}{\sqrt{n_e^2[n_e^2\left(n_e^2 n_o\gamma^2 - n_o^3(\gamma^2-1)\right)^2 + \left(n_e^2 - n_o^2\right)\cdot G\right]^2}}$$

Also,

$$F = n_e^2\gamma^2 - n_o^2(\gamma^2 - 1)$$
$$G = \sqrt{F \cdot \left(n_e^2 - 1 + \cos^2\theta_i + \left(n_o^2 - n_e^2\right)\cdot\beta\sin\theta_i\right.}$$

The equations above [WE], make computation easier. It is assumed that the incident ray passes through air and then the material. Its refractive index 1 is included in the formulas.

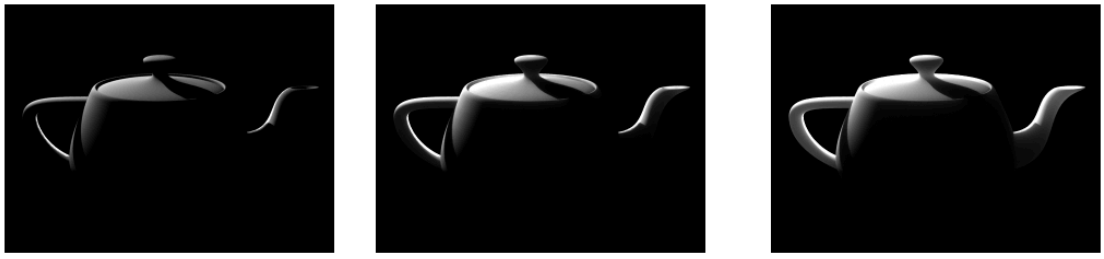
3.3 : Renderman

Renderman Shading Language was chosen as the preferred language for this project as it is an industry standard and can work standalone as well as with Maya, XSI and Houdini. 3Delight was also used at times, which is a Renderman compliant compiler.

As renderman is a scanline renderer, it works faster than any ray-tracing ones such as Mental Ray. It is also more efficient to produce subsurface scattering effects using point cloud and occlusion. This shall be discussed further on.

3.4 : BRDF and BSSRDF

Bidirectional Reflectance Distribution Function (BRDF) model calculates the light reflected from the surface. It assumes that light, which is incident on a surface is reflected back from the same position on the surface.
A material is considered to be translucent because light enters the material and either gets absorbed or leaves the material from a different position on the surface. Hence, the use of a BSSRDF model is more prudent. In a Bidirectional Subsurface Scattering Distribution Function, light incident on a material is scattered and reflected from a different point. This is the basic function of subsurface scattering that gives it the appearance of translucency and colour bleeding.



(a) (b) (c)

Figure 9: Comparison between BRDF and BSSRDF. [PR05]

(a) Shows a BRDF model where areas not hit by light are dark.
(b) Is with subsurface scattering with a short mean path. The scattering of light on the surface is apparent.
(c) Is also subsurface scattering but with a long mean path. The scattering is more distinguished here compared to (a).

The importance of using subsurface scattering for translucent materials is essential and hence, using a BSSRDF model was the appropriate choice.

3.5 : Environment Map

Image based lighting enhances realism of an image. In reality, light is diffused depending on the colours of the surrounding as well as the object itself. The figure below shows the difference that an environment map makes on the image.

Figure 10 : SSS with and without environment map [JE]

CG scenes are illuminated from the image supplied as an environment map. The images are typically HDR (cube or radiance map) and produce localized reflections. This can be seen in the image below.

Figure 11: Reflections from image based lighting [BJ]

Image based lighting is essential for translucent materials as much of the realism to be portrayed is due to the colour bleeding, reflectance, and, other such factors.

3.6 : Diffuse Approximation vs. Photon Mapping

Photon mapping one of the processes used for global illumination, which is essential to produce photorealistic renders. Lighting data is stored in a special data structure, also known as a photon map.
However, it is a very expensive process with a greater time complexity.
Diffuse Approximation is a process where multiple scattering is used for the diffusion process. It is done so in the following steps -

1. Baking the diffusion surface transmission of direct illumination into a point cloud file
2. The ptfiler program is used for diffusion simulation
3. A brickfile is created from the point cloud file
4. Using the brickfile, the image is rendered with subsurface scattering

The image below shows the comparisons in results between photon mapping, BRDF and BSSRDF.

Figure 12: Comparison between photon mapping and diffuse approximation models. [JE]

Due to ease in computation and accuracy in results, BSSRDF has been used.

Chapter 4

Implementation

4.1 Aim and solution

The ultimate result desired is a realistic render of a calcite. With the various techniques to be applied, it is essential to get the most amount of processing done with least amount of resources. While many different techniques were encountered, the most feasible solution was chosen. Many of the passes applied have large time and space complexities. Hence, it is necessary to choose options, which are the most efficient and optimal without compromising the quality of the result. It is also an aim to create a completely procedural shader with no use of textures.

Aside from choosing optimal algorithms, the shaders should be flexible and scalable. The user must have the option to enable/disable and tweak various factors so that rendering can be suited to his/her needs.

4.2 Pipeline

As explained in the previous section, Renderman has been selected as the shading language to be used.
A deformed cube has been modelled in Maya and included in the RIB scene using the RIB render feature in 3Delight.
The shaders work in two passes and the images are produced for the different passes as per the options chosen by the user.
Finally, the images are composited in a tool (eg. Nuke) to get the final render. The detailed shading process will be explained in the next section.

Maya -> RSL (Pass 1) -> RSL (Pass 2) -> TIFF images -> Nuke

4.3 Shading Process

The shading process can be roughly divided into 2 passes -

1. Pre pass
    a. Rib file "pre_pass_scene.rib" is created for pre pass.
    b. Displacement "disp_shader.sl" and pre pass shader "pree_pass_shader.sl" are applied. This shader creates a point cloud "pt1.ptc" and outputs display channels such as diffusion, occlusion etc.
    c. Using the ptfilter utility, the point cloud is used for approximated diffusion and to collect subsurface scattering values. A new point cloud "pt2.ptc" is created.
    d. The brickmake utility is used to convert the new point cloud into a brickmap "brickfile.bkm". (ptc2brick is used in 3Delight).
2. Crystal Shader
    a. New rib file "finalscene.rib" with provisions output in multiple display channels is used.
    b. The displacement shader and crystal shader "crystal_shader.sl" are compiled.
    c. Images are composited in Nuke or any other compositing software to get the final image.

The computations for the display channel passes are stored as functions in a C++ header file "passes.h".
Functions from external libraries such as Slim and Advanced Renderman [AP] have been used for noise calculations, remapping, ramps, clamping etc. These are included in the following files

1. "noises.h"
2. "crystal.h"
3. "pattern.h"
4. "filterwidth.h"

4.4 Display Channels

4.4.1 Renderman AOV

Generally, Renderman produces a beauty pass with rgb or rgba components. Arbitrary Output Variables (AOV) allows the use of display channels by storing data related to that display channel only. It's working is described in the figure below.

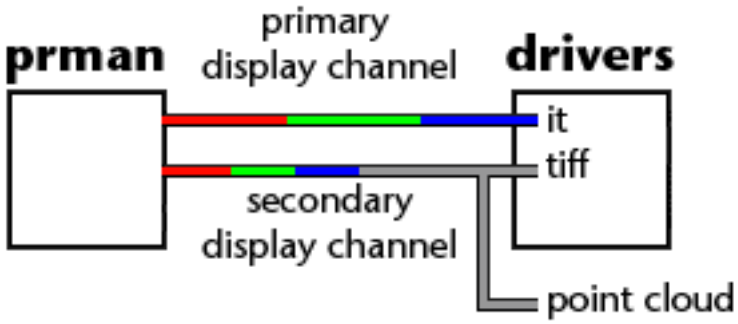


Figure 13: Working of AOV and piping. [KE]

Variables with the required colour values can be piped to separate channels. Having multiple passes gives the user more control over the final look of the image. This is implemented in both shaders.

The following passes are available in the pre pass -
Ambient, diffuse, diffuse indirect, diffuse environment, occlusion, and base noise (or albedo).

The following passes are available in the crystal shader pass –
Diffuse colour, diffuse environment, opacity, occlusion, reflection occlusion, refraction, 2nd refraction, surface noise, specular, and subsurface scattering.

4.4.2 Direct Illumination

Direct Illumination is used to calculate irradiance in generation of point cloud. The irradiance is the amount of radiant energy per unit time. It is based on ambient occlusion (_occlusion), diffuse indirect

(_diffuseIndirect) and diffuse environment (_diffuseEnv). It is calculated as

irrad += _diffuseIndirect +(Kenv * _diffuseEnv)+((1-Kenv) * _occlusion);

irrad is passed as radiance while generating the point cloud.

4.5 Displacement Shader

The displacement shader used deforms the surface randomly. Two layers of noise are applied to produce different results.

The first layer is a fractional Brownian motion function that randomly creates dents on the surface. This is used to make the surface less uniform. The frequency for this is low with a small lacunarity.

The second layer is also a fractional Brownian motion function used to create a more uniform noise pattern over the entire surface. The frequency is large and the noise is quite subtle.

A displacement bound is used to limit its size. It is important to note that it is used in both parts of the shading process.

4.6 Point Cloud

A point cloud is a data file in which each point is represented by a set of vertices in a 3D co-ordinate system. It is a convenient way for scanners like Renderman to store data. In the pre pass we store the illumination values in a point cloud. It is generated using the bake3d() function. [PR05]

We also pass the following values –

The display channels –

The display channels area, _radiance_t, _albedo and _diffusemeanfreepath are passed.

Interpolate –

This is an important parameter to set as micropolygons are converted to points and we prefer the micropolygon centers to be used, not edges. Hence there are no duplicate points at the edges. It is set to 1. [PR05]

Area –

It is the approximate area of one micropolygon. There are two strategy's used here

Dicing – it computes micro-polygons using their geometry without smooth derivatives.

Shading - It computes the area of micro-polygons using their geometry and produces smooth varying areas.

We use dicing as we get the original micropolygon area instead of a smoothed one. It is important to have this option.

Radiance -

Radiance is the amount of light that is emitted or passes through a region that comes under a specified angle and in a specified direction. It is the total amount of emission or reflection. The radiance supplied is irradiance, which is explained in section 4.4.

Albedo –

Albedo is the reflecting power of a surface. We pass the value of base noise (see section 4.8.2) as the albedo. Marble noise is used to generate that.

DMFP –

Diffuse mean free path is the average distance that a light ray travels before it reaches a surface and reflects. The longer the dmfp, the softer its appearance is.

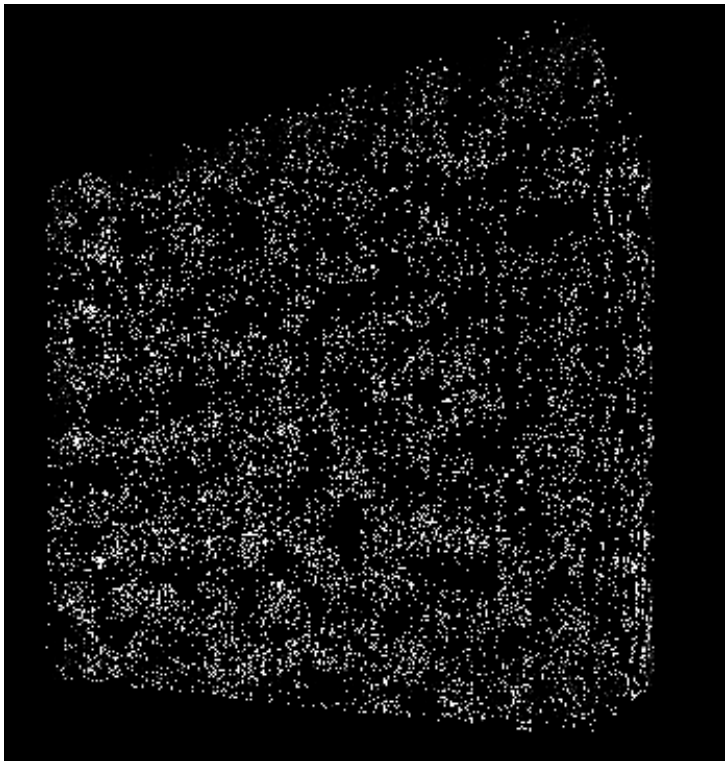


Figure 14 : Point Cloud generated from pre pass. (Own Image)

This point cloud is not of much use unless we obtain the ssdiffusion values. This saves the subsurface scattering properties.

We use the ptfilter utility to obtain the ssdiffusion values. In ptfilter, we can specify albedo, diffusemeanfreepath, and the refractive index. There are also preset materials such as marble, potato, apple etc. The absorption and scattering coefficients can also be given. Ptfilter is a very heavy process but can be optimized. In a multithreaded computer –threads n can be used to specify the number of threads. This speeds up the process remarkably. –progress can be used to keep track. The main factor to modify the quality and speed of the process is maxsolidangle. By default, the value is one. Smaller values lead to more accuracy but slower processing times. Hence, a balance must be maintained. [PR05]

It is necessary to optimize this process. With a solid max angle of 1 and 4 threads, it took 105 mins to process on a dual core machine.

The result of the new point cloud file created with ptfilter can be seen below.

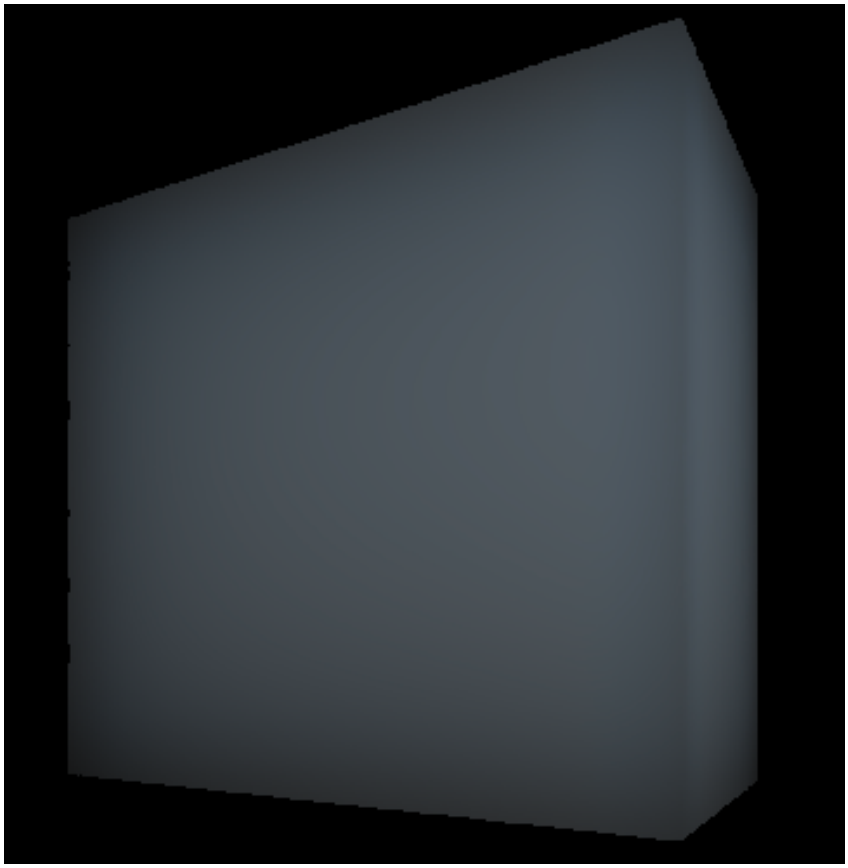


Figure 15: Point cloud with ssdiffusion values. (Own Image)

It should be ensured that "cull" "hidden" and "cull" "backfacing" should be set to 0 so that light bouncing of the hidden surfaces is calculated and all shading points are shaded. The "dice" "rasterorient" attribute is also set to 0 for an even distribution of shading points. [PR05]
It should be noted that, point cloud files can be viewed using ptviewer, which is provided with Renderman.

4.7 Brickmap

A brickmap is also a data file where point information can be stored. It is different from a point cloud as it also stores geometry information. A point cloud stores point information like colour, normal etc. for every micropolygon vertex. The data is not structured and hence has to be converted into a brickmap or kdtree. [RE]

Brickmaps are organized in a octree structure so that geometry can be viewed at different level of detail. At each level the number of points present increases.

The new point cloud file created can be converted to a brickmake file by using the brickmake command. To speed up the process,
-maxerror should be set to 0.002. [PR05]

The brickmap can be used for calculating values for the subsurface scattering pass. Brickmap files can be used with the brickviewer utility.

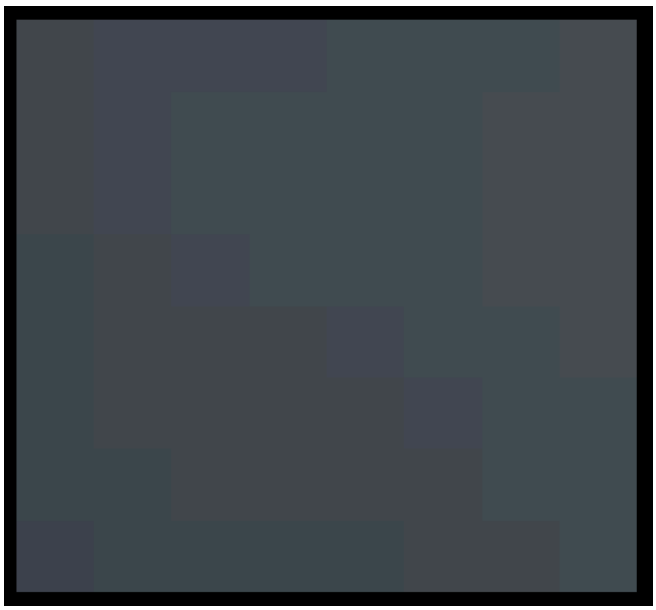


Figure 16 - Brickmap generated. (Own Image)

4.8 Shader Details

The different techniques used in the shaders to produce different image passes are discussed in detail in this section along with sample images.

Each different pass has a flag to enable or disable it. There are also attributes associated with it made available for modification. These are listed in Appendix A.

4.8.1 Surface Noise Diffuse Colour

A turbulence and fbm function is used to generate noise. The diffuse colour value is stored.

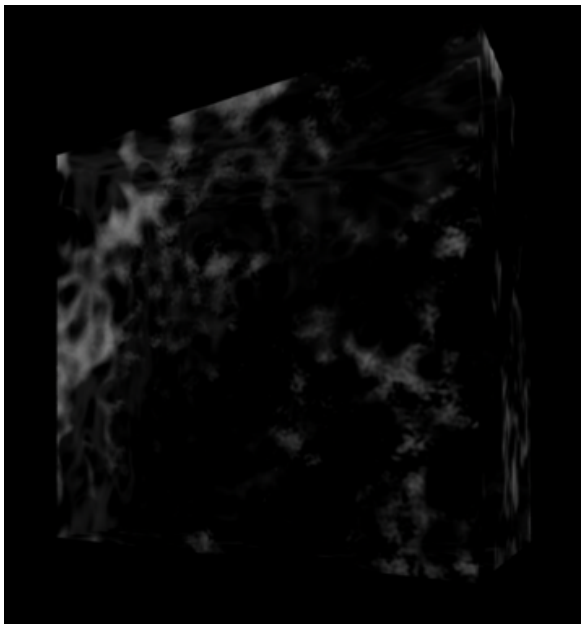


Figure 17 : Surface Noise Diffuse Colour (Own Image)

4.8.2 Base Noise

Marble noise was used to give a varying colour effect. Unless the calcite is colourless, pockets of varying colour are noticeable and add to the visual appeal of the image. Functions from the Slim library were used to achieve this. The base colour applied is in ranges of grayscale values as it is the most common colour of calcite found.

The base and crack colours (varying albedo) can be modified along with a variation factor to randomize it. The base noise is only rendered in the pre pass and is optional. However, it is quite useful to include this while compositing to add depth.

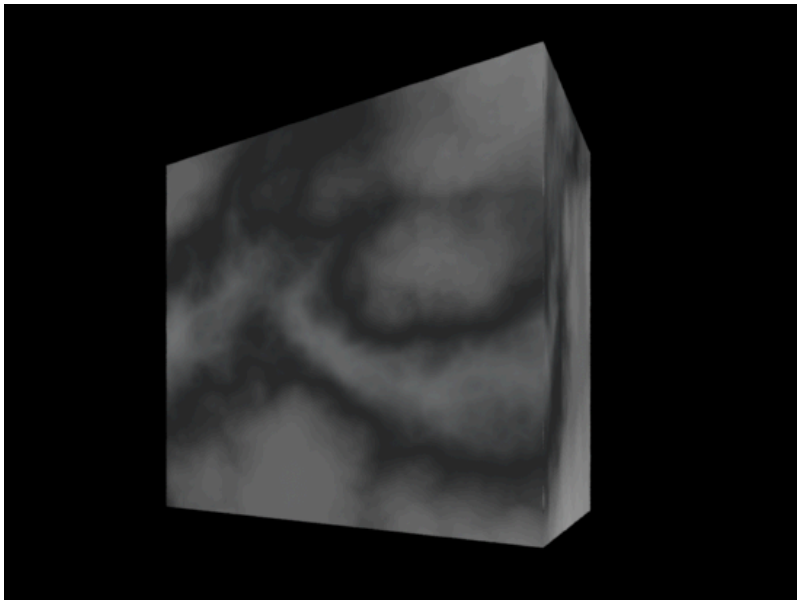


Figure 18 : Base Noise (Own Image)

4.8.3 Diffuse Indirect

The diffuse indirect pass is used to show colour bleeding, which is cause by colour transfer between surrounding objects. Colour bleeding is essential so that the object does not appear out of place. An environment map is supplied to generate the data.

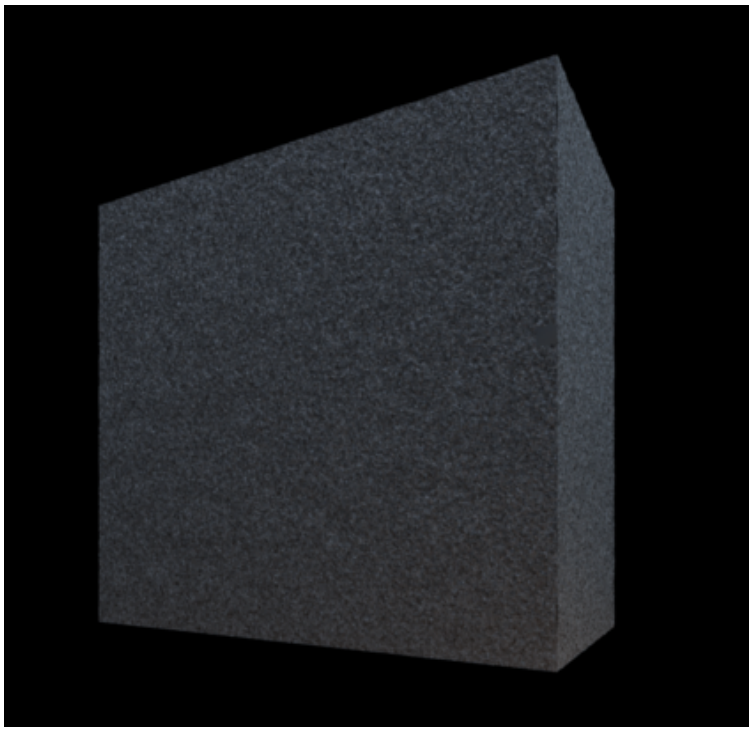


Figure 19: Diffuse Indirect pass. (Own Image)

The effect of the environment map is apparent, as the colour is reflected on the object.
This pass is quite heavy to compute. The settings for the PRman function indirectdiffuse() have been kept such that render times are minimal. Keeping a high 'maxsolidangle' ensures this. The quality of the image could be compromised if it is set too high.

4.8.4 Diffuse Environment

This pass computes values for image based lighting. Firstly, occlusion using the built in PRman function occlusion() is calculated. It is then multiplied with the environment colour value supplied by the environment map. The environment colour value is looked up in the occlusion function.

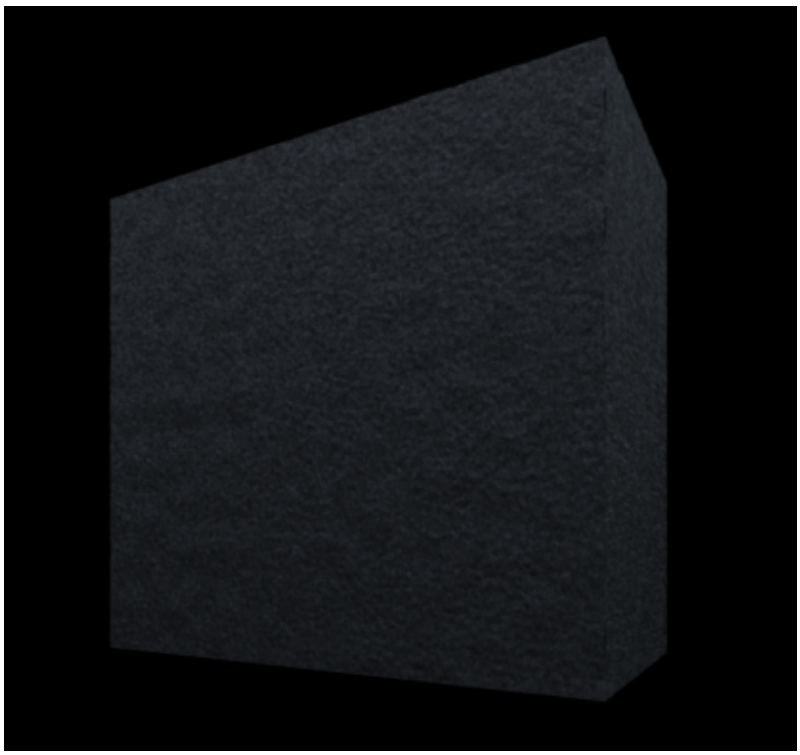


Figure 20 : Diffuse Environment Pass (Own Image)

4.8.5 Ambient Occlusion

In this pass, radiation of light from non-reflective surfaces is approximated. Spread and attenuation of light due to occlusion is calculated. As mentioned above, the PRman occlusion function is used to get ambient occlusion values. [KE]

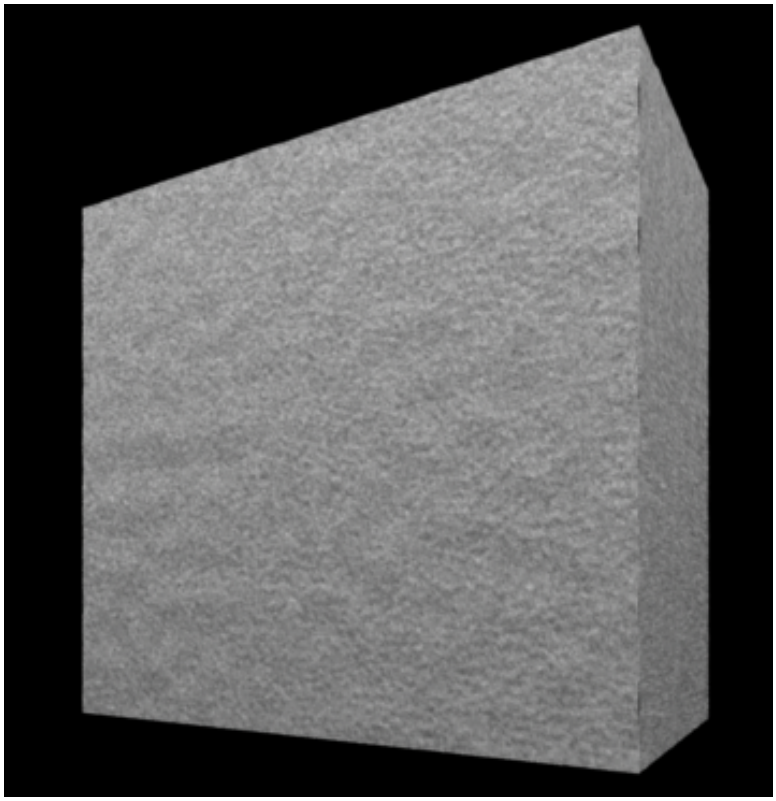


Figure 21 : Ambient Occlusion Pass (Own Image)

4.8.6 Reflection Occlusion

Reflection occlusion is trickier to operate. Essentially it is similar to ambient occlusion where instead of gathering light values, the reflection values are collected. To get this, samples are directed in the direction of reflection rather than the surface normal. [KE]

However, in renderman, the gather() is used to throw rays at the object. When the ray intersects an object, the hit colour is stored. refOcclusionSamples stores the number of samples, which is the number of light rays to be sent. refOcclusionAngle is the angle (in radians) through which the samples are sent. A small angle with a

relatively large sample rate would produce a sharper reflection. When this is reversed, the reflection is comparatively blurred.

Finally, if the ray does not hit any surface then the ray is sent in the direction of the environment to get the colour value. An environment map is supplied for this.

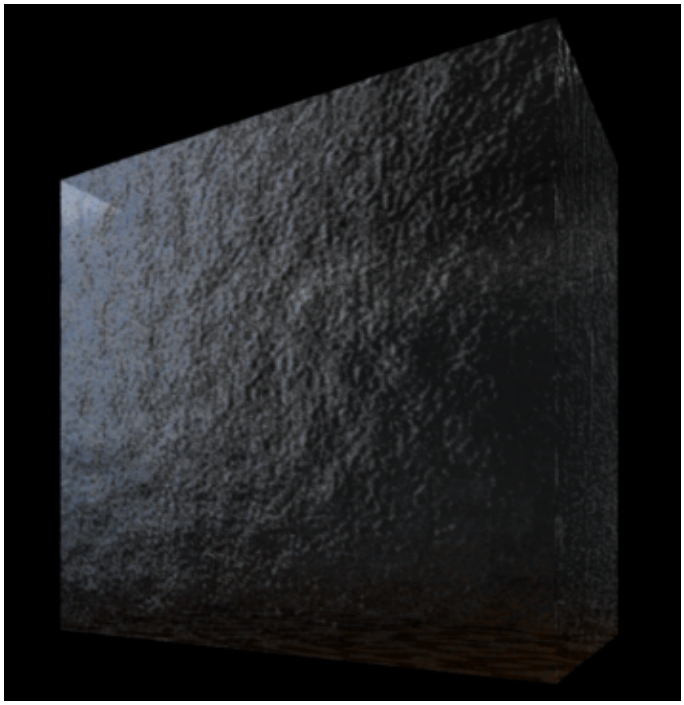


Figure 22 : Reflection Occlusion Pass (Own Image)

4.8.7 Refraction and Fresnel

Refractions in renderman are calculated using refract() and trace(). The incident ray normal (I), refractive index and face forward normal in direction of incident ray is passed to refract(). This results in the refract ray. Using the trace() the colour value in the direction of the refract ray is stored.

In the attempt to calculate birefringence, the desired result using the formulas mentioned in section 3.2 could not be achieved. It was decided to use two refraction passes with different refractive indices. The known RI values of calcite were used. The RI for ordinary ray (No) used was 1.658 and for the extraordinary ray (Ne), 1. 486 were used. [PA] However, visually it did not look very accurate. Hence,

noise was added to the extraordinary refract ray before using trace(). This way, there are two options available for using refractions, which can be used individually or together.

The refractionFlag is for computing ordinary ray refraction and birefringenceFlag is for computing extraordinary ray refraction.

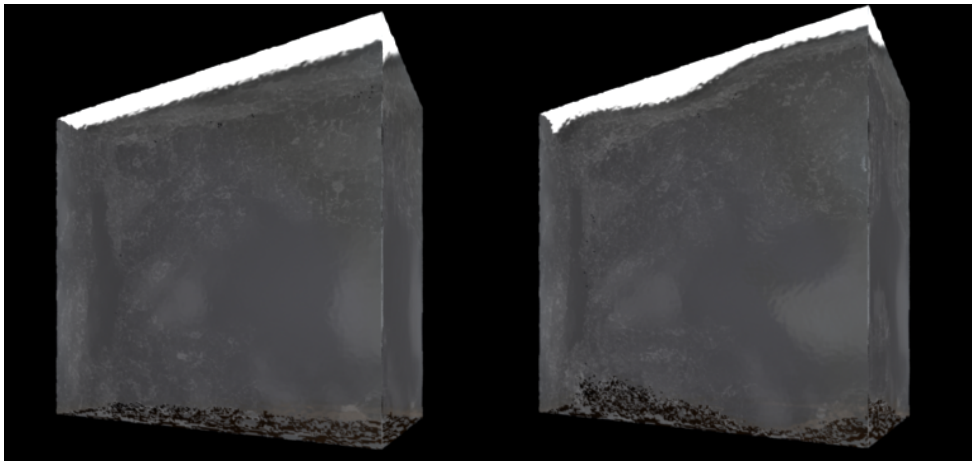


Figure 23 : Refraction Passes (Own Image)

The figure above shows *No* refraction on left and *Ne* refraction on right.

This feature is optional but the built in PRman function fresnel() can also be used to calculate the refract and reflect rays,

4.8.8 Surface Pattern (Veining)

Apart from the base noise, a calcite at times has more wear and deformations on the surface. The veining pattern was used to show more surface cracks, which would stand out. The algorithm was modified from a tile surface shader, which featured vein like cracks. Sharp turbulence and noise is used to achieve this.
A light and dark colour was mixed to get an effect similar to baseNoise but with a smoother gradient transition.

The veins added on are spread out and its spread can be controlled. A look up area is defined with a noise function and the veins are layered in the same area. This is used so that they do not look very dense. The sharpness of the veins is very high.

The effect of the veins can be seen in the figure below. You can notice the cracks throughout the surface. This can be exaggerated or diluted while compositing.

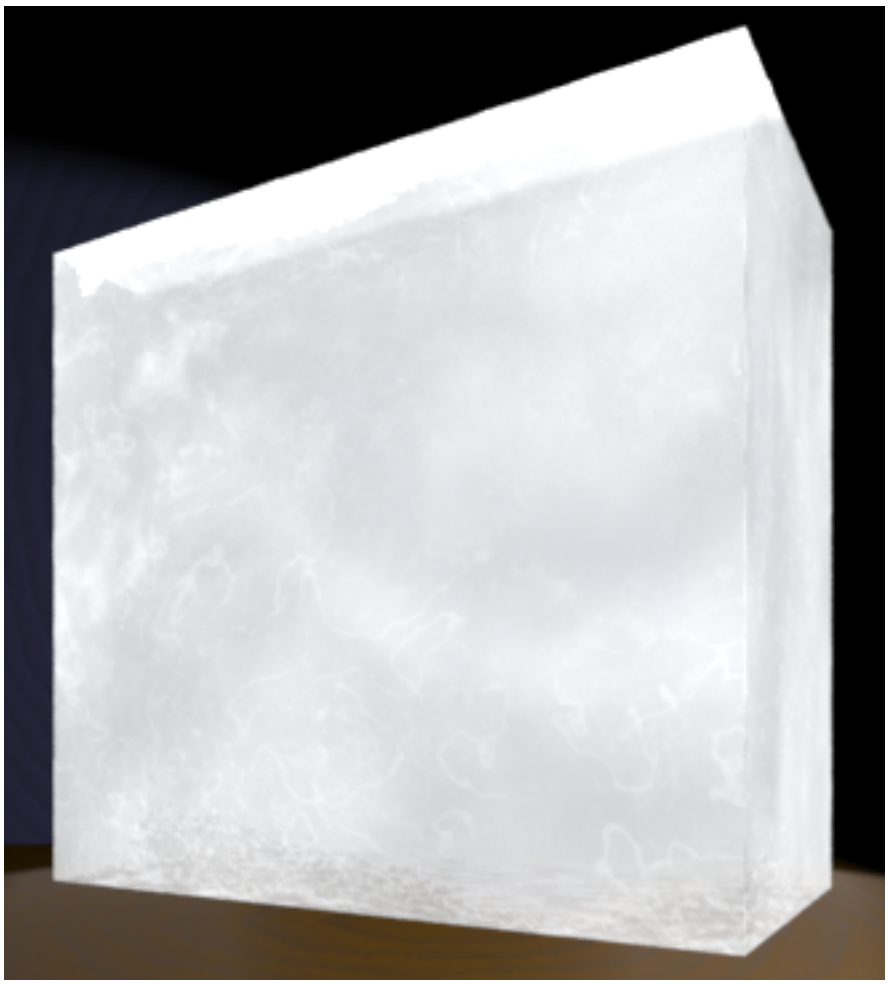


Figure 24 : Surface Veining Pattern (Own Image)

4.8.9 Surface Noise

Surface noise is an optional layer of noise that can be used as well. It adds gashes and rough patches. A colour ramp is generated and noise functions from the slim library are layered on to produce this effect. The spread of this noise over the surface can be controlled using vSpread (vertical) and uSpread (horizontal), which control the spread across u and v on the surface. The effect can be seen below.

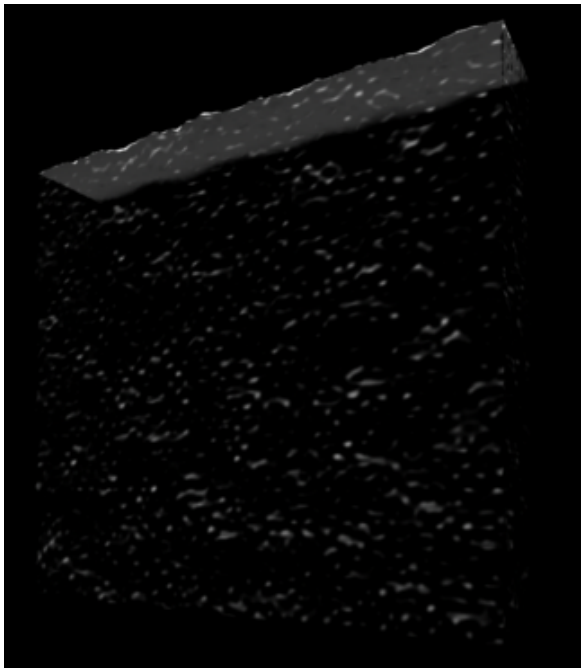


Figure 25 : Surface Noise Pass (Own Image)

4.8.10 Specular

The specular pass is generated by using the PRman specular(). It uses a vector (H), which is half way between the observer (V) and the direction of light (L).

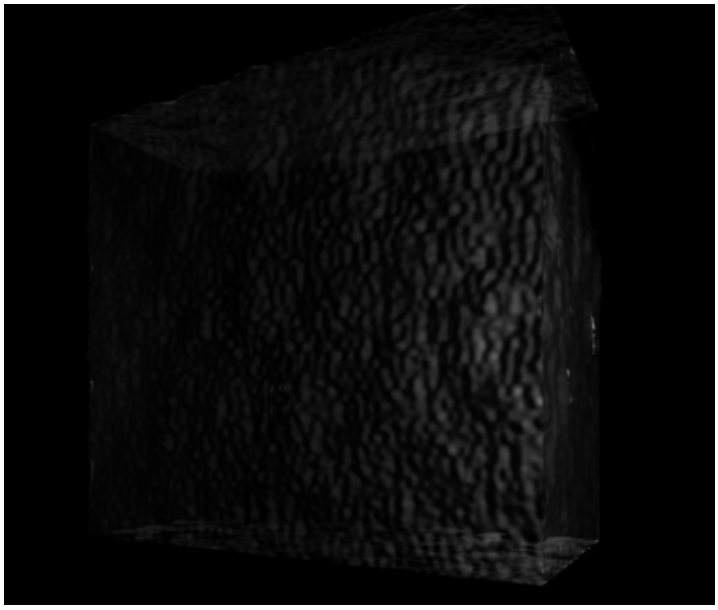


Figure 26 : Specular Pass (Own Image)

H is the orientation that the surface normal would need to be in, to get the most appropriate reflection. The distance from the surface normal can be calculated by the H.V. This distance is raised to the power of a value, which controls the highlight. [ST]
The roughness of the specular highlight can be controlled with the specRoughness variable.

4.8.11 Subsurface Scattering

As explained in section 4.7, the brickmap stores values for the ssdiffusion or subsurface scattering. These values are read from the brickmap using texture3d(). It is a function that looks up _ssdiffusion (which is specified) values from the file and stores it. It supplies the brickmap file, colour to be read and where it is to be stored. Thus, multiple scattering can be produced with this image. The resulting SSS pass is as shown below.

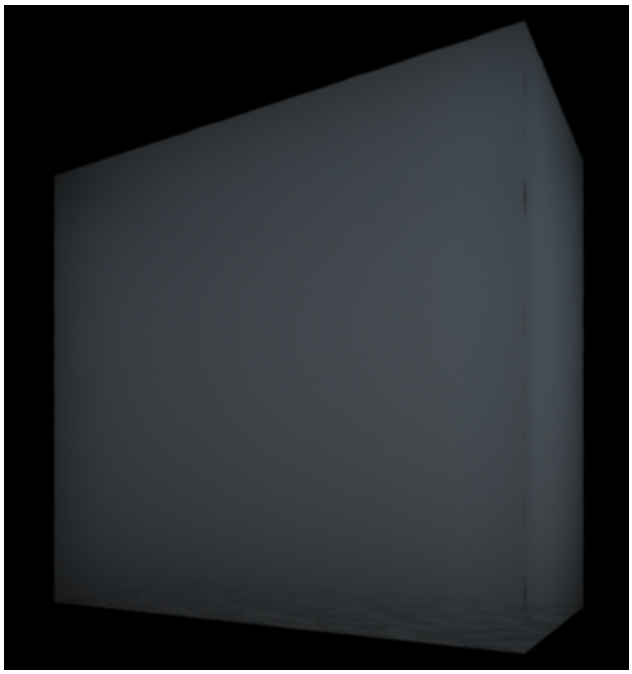


Figure 27 : Subsurface Scattering Pass (Own Image)

4.9 Scene RIB Settings

The objects, lighting conditions and other display options ate stored in a RIB file where displacement and surface shaders are applied to the object.
The camera has a 90-degree field of view.
The shading interpolation is set to 'smooth' which results in a smoothened shading of surfaces. The pixel samples are 8-bit.
The display channels are quantized to 8-bit channels as well by specifying "quantize" [0 255 0 255] "dither" [0.5] in Display.

Visibility attributes such as "int diffuse", "int specular" and "int transmission" are all set 1, so that they included while raytracing during refraction and reflection calculations.

Attributes for trace such as bias and "int displacements" are set to 0.05 and "primitive" respectively for trace().

The shadingrate is set to 1. Higher shading rates result in shorter render times as it is less detailed.

The 3-point lighting scheme has been utilized in each rib file. Spotlights have been used.

The scene objects consist of a horizontal and vertical plane and the crystal model in the center. There are two models available. One is the default geometry, a simple polygon cube. Another is a deformed cube modelled in Maya.

4.10 Final Result

The output variables for all passes are first multiplied with the opacity. The final value is arranged as shown below. Appendix A can be referred to for details about the variables.

```
Ci += _surfaceNoiseDiffuseColour * Kd * ((Kenv * _diffuseEnv)+((1.0-Kenv) * _occlusion))*_surfaceNoise ;
Ci += _reflectionOcclusion*Kr;
Ci += _sss*Ksss;
Ci += _surfaceNoise;
Ci += _surfacePattern;
Ci += _specularity*Ks * fKr ;
```

```
Ci += _refraction*Kfrac *fKt;
Ci += _refraction2*Kfrac *fKt;
```

The final composited image of the calcite crystal is as shown below.

Figure 28 : Final composited image (Own Image)

By changing displacement and other values, other variations were also obtained.

Figure 29 : Variations from same shader (own Image)

Chapter 5

Conclusion

5.1 Overview

The shaders in the end form a good and efficient tool capable of producing drastically different results depending on the user's preference. The images look realistic. Early on, it was a decided feature that the shader should be very flexible to be able to produce varying results. This has been achieved by giving the user control to almost all aspects of the shader within the bounds of it resembling a calcite crystal. The shader is also completely procedural.

5.2 Drawbacks and Improvements

The method used for shading is very efficient. However, newer techniques discovered could have been implemented.

- As mention in section 2, WETA published a paper on a quantized diffusion model, which is very efficient for translucent and absorbent materials. Implementing this would have been good learning experience. [DE11] However, it was discovered during the implementation phase of the project and hence could not be used.
- Birefringence in the shader and is not accurate to the physics of the effect. This is a factor that can certainly be improved upon. As the main aim was the visual aspect of the shader, this feature had to be given less priority.
- The two-pass process is lengthy with regards to time, as multiple files have to be generated before getting the final result. It made testing and backtracking more time consuming. Some of the passes and the ptfilter utility especially take a long time to complete.
- As an extension, the shader could have been implemented on a few more models in Maya using Slim or extended to be used in Houdini and XSI.

5.3 Conclusion

After an overview of what was accomplished and improvements to be thought about, it can be acknowledged that the project fulfilled its main goals. It produces a realistic render of calcite and is procedural. Changes can be made to the actual shader to improve some techniques used but the underlying framework is also a good base for other shaders to be based on.

References

[AP] Apodaca, A. and Gritz, L., 2000. Advanced Renderman. Morgan Kaufmann Publishers : San Francisco, USA.

[BJ] Bjorke, K., GPU gems, chapter 19. Available from: http://http.developer.nvidia.com/GPUGems/gpugems_ch19.html [Accessed August 2011].

[DE11] D'Eon, E., and Irving, G., 2011. A quantized-diffusion model for rendering translucent materials. SIGGRAPH. 30(4). Available from: http://dl.acm.org/citation.cfm?doid=1964921.1964951 [Accessed August 2011]

[DE] D'Eon, E., 2011. Render of cream. Available from: http://www.eugenedeon.com/CreamLarge.jpg [Accessed August 2011]

[FO] Fordham J. Pirates of the Caribbean 3. Cinefex. 110, 62.

[GE] Calcite. Geology. Available from: http://geology.com/minerals/calcite.shtml [Accessed August 2011]

[HE] Hery, C., 2003. *Implementing a Skin BSSRDF.* In "RenderMan, Theory and Practice", SIGGRAPH. Course Note #9, 73-88.

[JE] Jenson, H. Subsurface Scattering. Available from: http://graphics.ucsd.edu/~henrik/images/subsurf.html [Accessed August 2011]

[JE01] Jensen, H., Marschner, S., Levoy, M., and Hanrahan. P., 2001. *A Practical Model for Subsurface Light Transport.* SIGGRAPH, 511-518.

[JE02] Jenson, H., and Buhler, J., 2002. *A Rapid Hierarchical Rendering Technique for Translucent Materials.* SIGGRAPH, 576-581.

[JE03] Jensen, H., 2001. Realistic Image Synthesis Using Photon Mapping, A K Peters, Ltd., Massachusetts.

[KE] Kesson, M., 2002. CG references & tutorials. Fundza. Available from: http://fundza.com/index.html [Accessed August 2011]

[MI] Calcite. Mindat. Available from: http://www.mindat.org/min-859.html [Accessed August 2011]

[PA] Pascal, T., Usman, A. and Ododo, J., 2011. The phenomenon of nonlinear optical birefringence in uniaxial crystals. Latin-American Journal of Physics Education, 5(2), 432-437. Available from: http://www.lajpe.org/june11/19_LAJPE_522_Adam_Usman_Preprint_corr_f.pdf
[Accessed August 2011]

[PR] Primdahl, K., 2005. Fresnel reflectance, anisotropic absorption, and polarization in rendering, to show crystallographic effects. Image Synthesis Symposium, Stanford University. Available from : http://*graphics.stanford.edu/courses/cs348b-competition/.../Project_Report_C.pdf* [Accessed August 2011]

[PR05] Translucency and Subsurface Scattering. PRman Technical Notes. Available from: http://hradec.com/ebooks/CGI/RPS_13.5/prman_technical_rendering/AppNotes/subsurface.html [Accessed August 2011]

[RU] Rushes, 2011. VFX for BBC's Inside the Human Body. 3D World Magazine. Available from: http://www.3dworldmag.com/2011/05/11/rushes-creates-vfx-for-bbcs-inside-the-human-body/ [Accessed August 2011]

[RE] Render Wiki. Available from: http://joomla.renderwiki.com/ [Accessed August 2011]

[ST] Stephenson, I., 2007. Essential renderman. Second Edition. Springer : London.

[ST05] Stephenson, I., 2005. Production rendering: design and implementation. Springer : London.

[WA] Waters, Z. Photon Mapping. Available from: http://web.cs.wpi.edu/~emmanuel/courses/cs563/write_ups/zackw/photon_mapping/PhotonMapping.html [Accessed August 2011]

[WE] Weidlich, A., and Wilkie, A., 2008. Realistic rendering of birefrigency in uniaxial crystals. ACM Trans. Graph. 27(1). Available

from: http://cgg.mff.cuni.cz/~wilkie/Website/Home.html [Accessed August 2011]

[XE] Xenakis, A., and Tomson, E., 2007. Shading food: making it tasty for ratatouille. SIGGRAPH. Available from: https://renderman.pixar.com/products/whats_renderman/showcase_ratatouille.html [Accessed August 2011]

[XI] Xiao, H., Torrance, K., Sllion, F., and Greenberg, D., 1991. A comprehensive physical model for light reflection. SIGGRAPH. 25(4) 175-186.

Appendix A – Files and Variables

1. disp_shader.sl

The displacement shader used.

Data Type	Variable	Description
Float	d_freq	Frequency for dent fractal
Float	d_layers	Layers for dent fractal
Float	d_power	Power for dent fractal
float	d_fourthDim	Fourth Dimension for dent fractal
Float	d_lacunarity	Lacunarity for dent fractal
Float	d_flag	Enable/Disable flag for dent fractal
Float	r_freq	Frequency for random displacement
Float	r_layers	Layers for random displacement
Float	r_power	Power for random displacement
Float	r_fourthDim	Fourth Dimension for random displacement
Float	r_lacunarity	Lacunarity for random displacement
float	r_flag	Enable/Disable flag for random displacement

2. pre_pass_shader.sl

Surface shader for pre pass.

Data Type	Variable	Description
String	displayChannels	Channels for bake3d()
Color	albedo	Albedo for bake3d()
Color	dmfp	Diffuse Mean Free Path value for bake3d()
Color	Kd	Diffuse Colour
Float	baseNoiseFlag	Flag for base noise
Color	baseTint	Colour tint for base noise
float	crackCoeff	Factor for crack noise generated
Color	crackColour	Colour of crack in baseNoise
Float	crackVariation	Variation factor for crack
Float	occlusionFlag	Flag for ambient occlusion
Float	occlusionMaxVar	Max variation for Ambient Occlusion

Float	diffuseIndirectFlag	Flag for diffuse indirect
Float	diffuseIndirectSamples	Light ray samples for diffuse indirect
Float	diffuseIndirectMaxVar	Max variation for diffuse indirect
String	diffuseMap	Diffuse map to be used for diffuse indirect
Float	Kenv	Coefficient for diffuse environment
Float	diffuseEnvSamples	Diffuse environment samples
String	envMap	Environment Map for diffuse environment
float	diffuseEnvFlag	Flag for diffuse environment
Output varying color	_baseNoise	Output variable for base noise
Output varying color	_diffuseIndirect	Output variable for diffuse indirect
Output varying color	_diffuseEnv	Output variable for diffuse environment
Output varying color	_occlusion	Output variable for ambient occlusion

3. crystal_shader.sl

Surface shader for 2nd pass

Data Type	**Variable**	**Description**
Float	Ksss	Subsurface Scattering Coefficient
Float	Ks	Specularity coefficient
Float	Kr	Reflection coefficient
Float	Kfrac	Refraction coefficient
Float	Kenv	Environment diffuse coefficient

Float	Kd	Diffuse coefficient
Float	specFlag	Flag for specularity
Float	specRoughness	Roughness for specularity
Float	fresnelFlag	Flag for Fresnel effect
Float	bifriFlag	Flag for birefringence (2nd refraction)
Float	refractionFlag	Flag for refraction (ordinary ray)
Float	No	Refractive Index for ordinary ray
Float	Ne	Refractive Index for extraordinary ray
Float	refOcclusionFlag	Flag for reflection occlusion
Float	refOcclusionSamples	Light ray samples for reflection occlusion
Float	refOcclusionAngle	Angle for reflection occlusion in radians
Float	sssFlag	Flag for subsurface scattering
String	brickFile	Filename for brickmap
Float	surfaceNoiseFlag	Flag for surface noise
Float	uSpread	Horizontal factor for surface noise
Float	vSpread	Vertical factor for surface noise
Float	diffuseEnvFlag	Flag for diffuse environment
Float	diffuseEnvSamples	Light ray samples for diffuse environment
Float	occlusionSamples	Light ray samples for occlusion
Float	occlusionMaxVar	Max variation for occlusion
output varying color	_surfaceNoiseDiffuseColour	Output variable for surface noise diffuse colour
output varying color	_diffuseEnv	Output variable for diffuse environment

output varying color	_opacity	Output variable for opacity
output varying color	_occlusion	Output variable for occlusion
output varying color	_reflectionOcclusion	Output variable for reflection occlusion
output varying color	_refraction	Output variable for refraction of ordinary ray
output varying color	_refraction2	Output variable for refraction of extraordinary ray
output varying color	_surfaceNoise	Output variable for surface noise
output varying color	_specularity	Output variable for specularity
output varying color	_sss	Output variable for subsurface scattering
output varying color	_surfacePattern	Output variable for surface pattern

Appendix B - Manual

Instructions for using for the shaders. The following commands have been tested with Renderman.

1. Compilation of displacement shader

shader disp_shader.sl

2. Compilation of pre pass shader

shader pre_pass_shader.sl

3. Render pre pass rib file

render pre_pass_scene.rib

4. Use ptfilter to create new point cloud

ptfilter -ssdiffusion -albedo 0.830 0.791 0.753 -diffusemeanfreepath 8.51 5.57 3.95 -ior 1.6 –threads 4 –progress 2 -maxsolidangle pt1.ptc pt2.ptc

5. Convert resultant point cloud to brickmap

brickmake –maxerror 0.002 pt2.ptc brickfile.bkm

6. Compile 2nd pass crystal shader

shader crystal_shader.sl

7. Render final scene

render finalscene.rib