Major Project Report

Tom Minor - Level H
Major Project
Bournemouth University - NCCA

May 26, 2016

Project Overview (͡° ͜ʖ ͡°)

Main Aspects

1. Fractal Renderer
2. FX Work
3. Pipeline

Initial Research

1.1 Trying out existing stuff

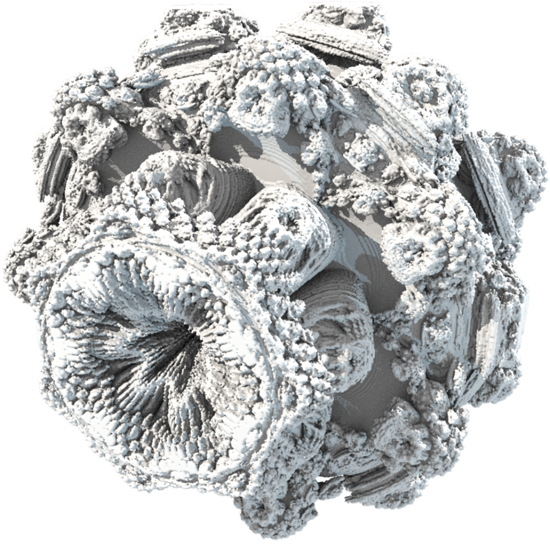
1.1.1 Fractals in FX Software

Looked at developing fractals in Houdini, may not need to develop a custom tool to visualise

Volumetric mandelbulb

Voxel Based

Slow to compute



Disney used houdini on BH6, but it's much too slow for our needs.

Need a custom tool

Houdini is still good for creature effects though, example noise shader goes here

1.2 Reading the documentation and tutorials

Pipeline

- Dropbox isn't enough
- Need to version
- More effort at beginning of the project, having backups of every asset could save our asses in the end
- use perforce
- Other scripts can be added to the main pipeline suite
- Referencing pipeline required custom hooks that modify all referenced paths to be relative to $CONTACTROOT, automated, idiot proof
- Referencing in 10 files and manually clicking student popup is tedious, on save and submit remove student/education flags
- Forces the team to think in a collaborative way, they literally cannot work on the same asset at the same time
- Took some getting used to, but now the artists are used to version control and it's benefits even if it's tedious at times
- Main development time spent on PySide Qt GUI stuff, after the initial time spent learning the P4 API and commands
- Technically cross application, the P4 and GUI side of things will work whereever pyside is available. Needs a few app specific tweaks such as file saving commands etc to work properly but wouldn't take long to port
- Various wizards to automate asset/shot/lookdev file structure generation because kyran made the layout super complex
-

Renderer

- How best to make this artist friendly (Nodegraph)
- What technique (ray marching)
- Use Optix because it's faster than what i can do
- Nodegraph needs runtime compilation, at least for geo
- Does require a little hacking to get runtime code generation
- Use NVRTC to compile code at runtime and plug into preexisting functions in the optix code
- Use hacky system commands to call nvcc directly if using CUDA 6.5 or less, aka, the uni systems
- Initial dev time for node graph, runtime compilation etc is long, but in theory will allow for rapid iteration once it works
- Everything is based on demo scene stuff, can use shadertoy as reference for loads of effects

3.1 Scene Management - Code Reflection

Need to change scene based on node graph

3.1.1 PTX Patching

- Initial attempt, required research into the .ptx format
- Use NVCC to compile CUDA program into ptx code and patch into the Optix ptx code, then load into Optix
- Works on my machine and university workstations, but potentially undefined behaviour and not officially supported
- Relies on my own string handling functions

3.1.2 Optix Callable Programs

- Discovered this later in the project after reading the Optix documentation fully, initially didn't notice what it was because it's not used very often compared to the other aspects of optix and isn't really clear unless you know what it does.
- Use NVCC to compile CUDA program into ptx code and then tell Optix to use this as a 'callable program', basically replacing the ptx patching process with a well defined, built in functionality.

3.2 Rendering Strategy

- Render tiles
  - For larger frame sizes this will give an opportunity for the calling program to return quickly and handle events
  - Will use less GPU RAM, which is vital for HD frames on GPUs without lots of memory
  - Has the potential to simplify implementation of other rendering algorithms in the future, for example Bidirectional Path Tracing stores light paths in an array that can become very large for huge images but is manageable for small tiles
- Monitor the Optix rendering in a separate thread and copy over to host memory every half a second or so, this keeps the GUI responsive without over saturating the PCI-E bus (GPU ¿¿ CPU memory transfer) with constant memory copies.

Conclusion

Bibliography

[1] http://dctsystems.co.uk/renderman/angel.html. http://dctsystems.co.uk/RenderMan/angel.html. Accessed: 25th April 2015.

A Renderman compliant renderer developed by Ian Stephenson, I initially chose to use it because it had support for geometry shaders that provided me with a simple way of creating an ice cube shape through the use of superquadrics. Unfortunately, the superquad shader did not work with shadows, the feature set is fairly dated compared to current PRMan releases and opacity support was too noisy (which presented a problem for a project that makes heavy use of translucency), forcing me to move on to using Pixar's Renderman instead.

[2] Pixar's renderman. http://renderman.pixar.com/view/renderman. Accessed: 26th April 2015.

MUST ADD ANNOTATION TO THIS

[3] Pixar's renderman documentation. https://renderman.pixar.com/view/documentation. Accessed: 26th April 2015.

MUST ADD ANNOTATION TO THIS

[4] The super egg and other super surfaces. http://www.math.harvard.edu/archive/21a_fall_09/exhibits/superegg. Accessed: 25th April 2015.

Although certain superquadrics are similiar in shape to an ice cube, notably the super egg, in the end I instead decided to write a displacement

[5] Anthony A. Apodaca and Larry Gritz. *Advanced RenderMan: Creating CGI for Motion Picture*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 1999.

Used for : Texture mapping basics, volume shader basics, brownian noise RSL function?

[6] Oliver Deusen, David S. Ebert, Ron Fedkiw, F. Kenton Musgrave, Przemyslaw Prusinkiewicz, Doug Roble, Jos Stam, and Jerry Tessendorf. The elements of nature: Interactive and realistic techniques. In *ACM SIGGRAPH 2004 Course Notes*, SIGGRAPH '04, New York, NY, USA, 2004. ACM.

MUST ADD ANNOTATION TO THIS

[7] Makoto Fujisawa and Kenjiro T. Miura. Animation of ice melting phenomenon based on thermodynamics with thermal radiation. In *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*, GRAPHITE '07, pages 249–256, New York, NY, USA, 2007. ACM.

MUST ADD ANNOTATION TO THIS

[8] Tomokazu Ishikawa, Yoshinori Dobashi, Yonghao Yue, Masanori Kakimoto, Taichi Watanabe, Kunio Kondo, Kei Iwasaki, and Tomoyuki Nishita. Visual simulation of glazed frost. In *ACM SIGGRAPH 2013 Posters*, SIGGRAPH '13, pages 14:1–14:1, New York, NY, USA, 2013. ACM.

MUST ADD ANNOTATION TO THIS

[9] Theodore Kim, David Adalsteinsson, and Ming C. Lin. Modeling ice dynamics as a thin-film stefan problem. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '06, pages 167–176, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

MUST ADD ANNOTATION TO THIS

[10] Theodore Kim, Michael Henson, and Ming C. Lin. A hybrid algorithm for modeling ice formation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, pages 305–314, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

MUST ADD ANNOTATION TO THIS

[11] Theodore Kim, Michael Henson, and Ming C. Lin. A physically based model of ice. In *ACM SIGGRAPH 2004 Sketches*, SIGGRAPH '04, pages 13–, New York, NY, USA, 2004. ACM.

MUST ADD ANNOTATION TO THIS

[12] Theodore Kim and Ming C. Lin. Visual simulation of ice crystal growth. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 86–97, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

MUST ADD ANNOTATION TO THIS

[13] Masaaki Matsumura and Reiji Tsuruno. Visual simulation of melting ice considering the natural convection. In *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

MUST ADD ANNOTATION TO THIS

[14] Takanori Nishino, Kei Iwasaki, Yoshinori Dobashi, and Tomoyuki Nishita. Visual simulation of freezing ice with air bubbles. In *SIGGRAPH Asia 2012 Technical Briefs*, SA '12, pages 1:1–1:4, New York, NY, USA, 2012. ACM.

MUST ADD ANNOTATION TO THIS

[15] Saty Raghavachary. *Rendering for beginners: image synthesis using RenderMan*. Elsevier, San Diego, CA, 2004.

MUST ADD ANNOTATION TO THIS

[16] Ian Stephenson. *Essential Renderman*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

MUST ADD ANNOTATION TO THIS

[17] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Trans. Graph.*, 32(4):102:1–102:10, July 2013.

MUST ADD ANNOTATION TO THIS

[18] Feng Xie, Mike Necci, Jon Lanz, Patrick O'Brien, Paolo de Guzman, and Eduardo Bustillo. Arctic ice: Developing the ice look for how to train your dragon 2. In *Proceedings of the Fourth Symposium on Digital Production*, DigiPro '14, pages 37–39, New York, NY, USA, 2014. ACM.

MUST ADD ANNOTATION TO THIS