Capstone Proposal (R.3) Jonathan Pearl

Raypoint

**Project Description**

My project will allow users to express and visualize both simple and complex mathematical structures in 3D space. They will be given the capability to script in order to define the shape they wish to see. This can be accomplished in several ways. Subset check will march along each ray cast at a small defined interval from the projection and check to see if a point satisfies a condition. If it does it fills in that pixel appropriately. This method is the simplest way to define most structures but is also inefficient. Distance-estimated (DE) ray marching, does the same but will be modeled so that at any point the closest distance to any point on the structure can be calculated. Using this shortest distance it would be able to march along the ray by that amount safely. This would allow steps on the ray to be much larger in contrast to the previous method. The final method is ray tracing where the distance to the structure is known absolutely from any ray and will be used to instantly snap to the structure. These methods work on varying levels of detail depending on how much is known of the distance to the structure. Then the user will be able to apply transformations like scaling, rotation, translation, and repetition fields. They may also combine them with other structures using unions, intersections, and exclusions depending on their method of definition.

**Target Users**

This project will target those who are fascinated by mathematical constructs like fractals. It will allow them to both produce and explore these fractals as well. Those who are not technically apt will be able to quickly find constructs that are already prefabricated and explore them, while those who are able to can take hold of the scripting language at hand and formulate their own structures entirely and share their findings with others.

**Features List**

**Planned Features**

* The project will render
  + Ray marching with subset check and DE
  + Ray tracing
* The project will take advantage of the graphics card in order to perform parallel calculations
* The project will allow users to script in their own constructs to display opting for one of the three methods of defining the structures
* The project will compile down the script to be used in a non-interpretive fashion
* The project will offer an interface for viewing prefabricated constructs as well as those that the user has imported/saved.
  + From this interface, the user can pull constructs into the scene and perform applicable transformations on them and with other structures in the scene
    - Local Transformations ( scale, rotate, translate )
    - Joint Transformations between similar defined ray march constructs ( union, intersection, exclusion )
* Material based manipulations
  + Refraction, reflection, lighting
* Preview window that shows the structure currently being observed and the projection can be manipulated to be moved and rotated
  + Subdivide the screen into large quadrants and calculate for those pixels and continue subdividing breadth-first in order to supply low-resolution renderings first and then refine over time to allow for more responsive
* Record high quality stills into images with subsampling

**Stretch Goals**

* Record traversals through the structures and save them into videos.
  + Set up key framed intervals and interpolate between them to create the frames for the video.
  + Import audio to be played with the video
* Create a ray tracing prefabricated structure that allows the importing of models
  + Programmatically smooth the mesh as well as including the allowance and use of texture data
* Multi-pass and post-processing effects: Shadowing, depth of field
* Add Occulus Rift, Anaglyphic display, or 3D monitor support
* Social media integration

**Technical Specifications**

The main language of this project will be C++ with Microsoft Visual Studio as my IDE. I will also use OpenCL in order to perform hardware accelerated tasks on the GPUS of the machines running the application as well as the OpenCL platform. This will allow me to deploy OpenCL programs from my C++ application. I will use Notepad++ to create and edit external OpenCL programs. I will take advantage of Nvidia’s Nsight Visual Studio Edition in order to assist in debugging and tracing the programs I send to the GPU in the development phase. I will use hardware-neutral OpenCL as to retain compatibility with NVidia and Intel chips.

In order to create an interface for my users I will use Qt in C++ as a windowing system and content manager. I will also use Qt in combination with OpenGL in order to quickly and efficiently display a preview of the scene. If I incorporate the ability to record flights with video output I will use OpenCV in order to synthesize those videos.

**Phase 1 (2-Week Plan)**

**Week One**

* Layout UI to be populated
* Create the scripting interface
* Parse scripts and export code to be used in OpenCL programs
  + Handle type assignment to variables upon variable set and use ( integral vs floating point )
  + Allow for “outlet” variables that can be adjusted in the scene
* Allow the saving of scripts and placing into scene

**Week Two**

* Handle multiple structures in the scene
* Allow input for transformations and adjust accordingly
* Join two applicable structures to have a boolean operation set on them and create the procedures for union, intersection, and exclusion
* Add the ability to refract through surfaces that have different indices of refraction

**Justification**

I believe my project is of Capstone quality due to its technical depth. Not only does it incorporate ray marching, a subject that does not see much coverage, but I will also be learning to use new technologies like OpenCL and potentially OpenCV. An application like this does not exist namely because of the customizability my application will offer with the ability to formulate custom structures and apply transformations on them. Even before coming to Neumont I was fascinated with fractals and wanted to create an application to view the beautiful mathematical landscapes found in photos in Google Image Search and videos on Youtube.