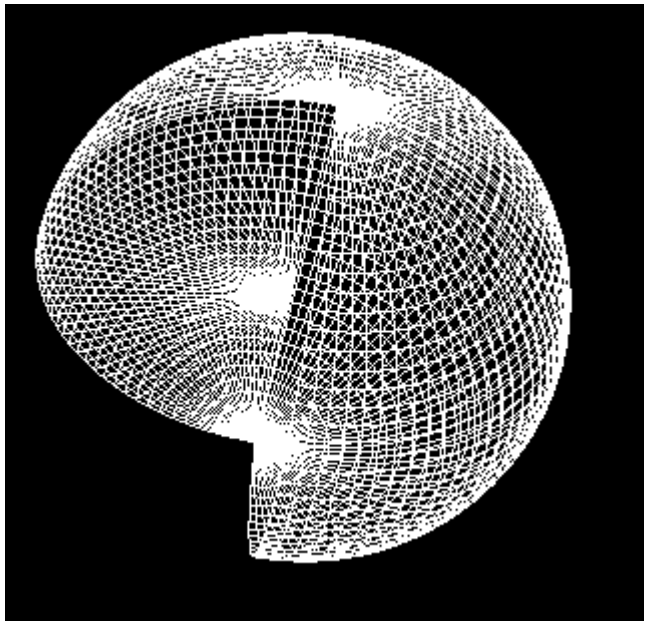
Threading: Procedural Generation of a sphere

By Nick Taylor

The Application

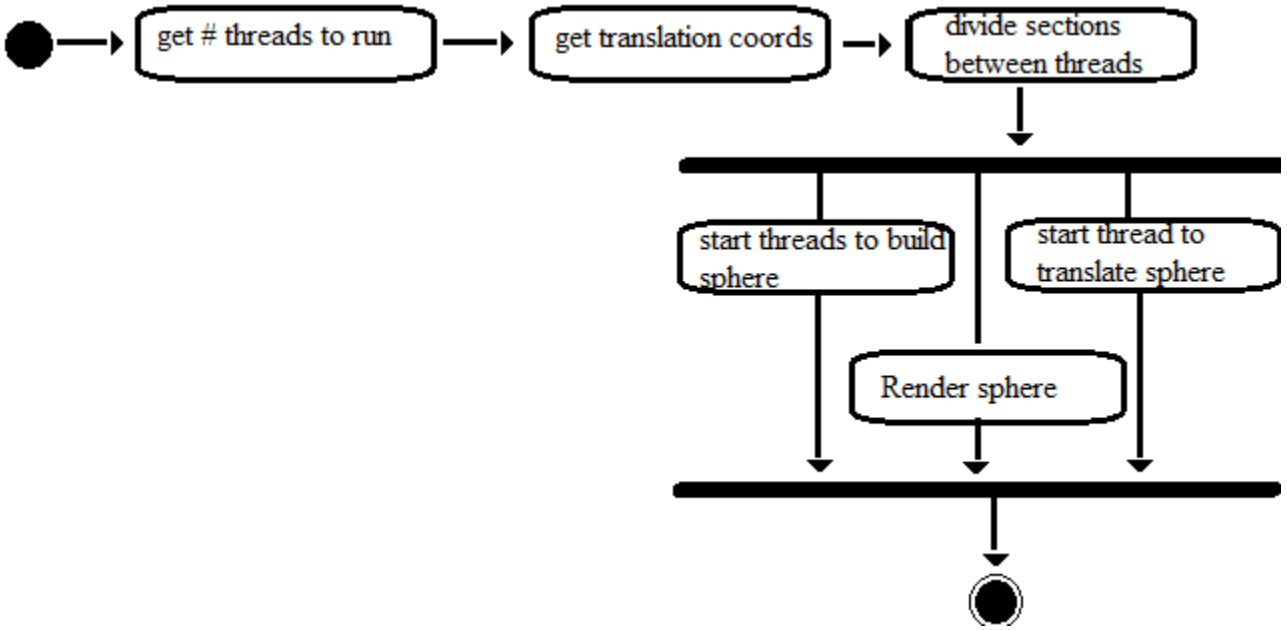
- Procedurally generates a sphere
- Fills vertex array with x, y & z coordinates.
- Then moves the sphere a requested distance.

- Sphere is built longitudinally section by section.
- If a section has been built it can then be translated.



The Code

- The Code Structure is fairly simple. Other classes are just for interface purposes.
- Key Functions: task_division, build_sphere, translate_sphere & draw_sphere



Threading

- The Application has 2 main thread functions, building the sphere and translating the sphere.
- Making the sphere is done by a group of threads of variable number. Each thread is given a number of longitude sections to build.
- Translating the sphere is always done by a single thread, though could potentially be multi-threaded as well.
- Once the sphere has finished being moved the locations of each vertex is output and the program closes.

Threading Interaction

- The Build threads and translate threads are started at roughly the same time. But the translate thread cannot do any work until there are sections to move.
- Once a section has been built in by a build thread it updates and array.
- The translate thread is constantly checking for built sections. If the mutex for a section isn't locked (aka that section is currently being built) the thread will check if the section has been built and if it hasn't been already moved. If these conditions are met the section is moved.
- Once all section are moved the thread will end. Once it does the main function will see that and fire off a thread to output the entire vertex array. Once that thread is joined the application is exited.

Reflection

- The task was pretty fun, threading a program can be a challenge.
- My initial idea was scrapped. Difficult to measure performance, difficult to thread.
- Threading seems to benefit large tasks with clear separate sections.
- Using Boost Libraries was a great improvement.