Videogame environments are constantly becoming larger and more complicated. There has been a recent trend in shooter games towards large scale battles between many players on giant maps. Other games like Rockstar Games’ GTA and Mojang’s Minecraft have giant open worlds. In order to run with decent performance, these games give the player the illusion of a large scale world, while the game only processes the details of nearby objects the player can see.

Capstone Project
Advisor: Michelle Hugue
Email: drmeesh@gmail.com
Time period: Spring Sem 2015
Hours Completed: 125 as of 4/24

Goals & Approach
The goal of this research project is to gain a better understanding of methods used to speed up games by only processing vital information. I researched this by looking at what data structures have been used in popular games, and by testing how they perform in different situations.

Quadtrees
Quadtrees are used to partition space into quadrants. Start with a square. Split it into 4 equal squares. You can split one of those squares into 4 even smaller squares, and so on.

Rendering Terrain
Terrain is commonly represented as a table where, given an x and z (longitude and latitude), will give you the y (the altitude). Renderers construct a triangle mesh from sampling this table.

Dynamic Detail
The process of rendering terrain can be made much faster by reducing the number of triangles. Detail becomes less important the farther it is from the viewer, so we can use fewer triangles on farther regions.

Use of Quadtree
We can use a quadtree to split up a landscape into square regions. The algorithm I implemented splits these squares smaller and smaller the closer they are to the viewer. Every square gets rendered with the same number of triangles. Smaller square, constant number of triangles = more detail per square unit of space.

Quadtrees" source="https://en.wikipedia.org/wiki/Quadtree"

Reflection
I have always been interested in game engines and rendering, so I had a lot of fun working on this project. At one point I actually spent 24 hours straight coding this. From this project I learned two important lessons:
1. C++ is a very complicated language and I never realized how challenging it could be. I ran into massive roadblocks due to cyclic dependencies.
2. There is a cooperative community of graphics hobbyists out there who publish research papers. I have a greater appreciation for this community now that I’ve read some of their work and done some research of my own.

Acknowledgements
Thanks to Dr. Michelle Hugue for being a helpful, supportive mentor. Thanks to Dr. Peel and CP Scholars for giving me the opportunity to work on this fun project.