Checkpoint Report

Stephan Boettcher

November 6, 2014

Introduction

The Parallel Pathfinding project has been an enlightening one. I have dusted off many of my OOP skills and put them to great use. Thus far, I have researched many different parallel pathfinding algorithms, looked the the source code of a few different implementations of the A* algorithm, and have utilized this knowledge to generate a functioning serial implementation of the code. I have chosen to forego Bitbucket for Google Code solely because I prefer SVN to GIT for small projects due to the abbreviated syntax necessary to sync and upload code. Below is my motivation for the code design thus far, and my plans for the remainder of the project.

Motivation for methodology

The goal of this project is to develop a parallel pathfinding algorithm to be used for small UAVs or drones. This type of application of algorithms is becoming increasingly relevant in today's world. With the advent of Amazon Prime Air[1], a large number of routes will need to be calculated to optimize the delivery of packages. In an urban setting near one of Amazon's main distribution hubs (e.g. Los Angeles), the package flow to local customers can be quite large. While strictly optimal paths are not necessary for this application, minimizing travel distances must be as optimal as possible while still allowing the servers at Amazon to calculate paths as quickly as possible.

Many of the maps generated for this project have the letters AOR in their titles, which stands for Area of Regard. These maps, and subsequent maps to be generated, will mimic a map with various No Fly Zones. Larger circles will denote things like Airports, sports stadiums or restricted airspace. Smaller circles will be used to generate No Fly Zones over things like police stations, schools, places of worship, or other smaller buildings that don't want drones flying overhead. The resulting swiss-cheese style map can present an number of routing challenges to the computer.

Plans forward

Currently, a serial implementation of the exists. The code is able to pull in maps and output a bitmap image of the map and path. The program also creates or appends to a results file that catalogues the code performance. The next step is to implement a batch-processing parallel implementation of the code. Given the project is focused on generating a number of different paths for multiple drones, a batch-processing method seems to be the best first step. Next, I will implement a less-than-strictly-optimal parallel implementation of a pathfinding algorithm, such as a batch parallel distributed fridge search. I will also attempt to implement a batch GPU version of

[1] Amazon Prime Air: http://www.amazon.com/b?node=8037720011

the code, if time permits. I imagine that running this code on multiple GPUs would result in the largest speedup.

Running the code

This code requires a C11++ compiler to work. To compile the code, please go to the Code/Serial folder and type: "make". The code should then compile and generate a binary file named "Serial". Serial requires 3 inputs:

1. The total number of iterations to run
2. The either the path to the maps folder, or the path to a 24-bit color bitmap
3. a 1 or 0. If 1, only provide the path to the maps folder in the step above. If 0, give the path to the bitmap map (e.g. maps/cross50.bmp)

The program will run until the total number of iterations passed is complete. If the user passes a 1, the program will cycle through all of the maps in the folder given. If a 0 is passed, all iterations will be run on one map.