Online Procedural Content Generation of Starcraft Maps

Master's Thesis

Jakob Melnyk, jmel@itu.dk
Jacob Claudius Grooss, jcgr@itu.dk

June 1, 2015

Contents

1 Introduction

This report is the result of our Master's Thesis at the IT-University of Copenhagen, spring 2015.

The goal of the thesis was to apply both constrained novelty search and multiobjective evolutionary algorithms to do online procedural content generation (PCG) of maps for Starcraft, analyze the quality of the generated maps and compare the trade-offs required in order for each algorithms to generate maps fast enough to b e used in online PCG.

2 Map Representation

A map in Starcraft 2 is represented as a map of tiles, where a tile can have have a a height (impassable for ground troops, and 3 heights that ground troops can traverse), cliffs between heightlevels, ramps that connect heightlevels, buildings, resources and Xel'Naga towers (provides vision while a unit is nearby). The map can, of course, also contain units but they are irrelevant for our purposes. The only units a basic map contains are the initial 5 (**6?**) workers that always start next to one's starting base, so we can ignore units in our generation.

We did not have direct access to the map representation used in Starcraft, so we had to create our own map representation. As we wanted to use evolutionary algorithms (a subtype of genetic algorithms), we decided to split our representation into two parts: A genotype and a phenotype.

2.1 Genotype

The genotype contains a list of the items the map should contain and their position on the map. The position uses a a radial system, where the midpoint is the middle of the map. Each item has an angle and a distance. The angle determines the angle from the middle of the map (with 0^o being right and going counterclockwise (**Check if correct**)). The distance is represented as a percentage of the distance from the middle of the map in a straight line to where the angle will hit the edge of the map.

2.2 Phenotype

Where the genotype represents the genetics that make up an object, the phenotype represents the actual object. As mentioned about, a map is made up of heightlevels and various other things.

For our genotype, we decided to use two 2D arrays to represent the map. One array contains the heightlevels, ramps and cliffs, and the other contains "items" in the map, such as bases, resources, and Xel'Naga towers. We split it into two because that allowed us to place both a height and an item on the same tile in an easy way.

The phenotype has a few functionalities that are used for creating the map: Smoothing the heightmap that has been generated through the cellular automata and placing cliffs in the heightmap.

The smoothing works by iterating over every single tile and checking the neighbouring tiles (using the Moore/extended Moore neighbourhood). If the tile does not fit in, it is changed to something that fits better. The changes are made in a clone of the heightmap, so changes do not affect subsequent tiles until next iteration. This smoothing is repeated over a number of generations, at which point the clone is saved as the actual heightmap in the phenotype.

Cliff placement is a simple procedure. We again iterate over all tiles on the map and check its neighbouring tiles (using the Von Neumann neighbourhood). If any of the neighbouring tiles are of a lower heightlevel than the current tile, the neighbour tile is transformed into a cliff.

The phenotype is also used when we create a visual representation of the map. The visual representation is created in two steps: First we draw the heightmap as a bitmap, where each heightlevel has a different tile icon. After that, every item is drawn on top of the already-existing bitmap, in order to avoid items being overwritten by heightlevels. When the item have been drawn, the map is saved as a .png file.

3 Cellular Automata

A cellular automata is a form for model can simulate artificial life and is often used in complexity studies. Cellular automatas are often used in computational tasks (e.g. procedual contant generation), mathematics and biology.

A cellular automata consists of a grid of cells of any size and a set of rules. The grid evolves over a number of iterations(generations?) by applying the rules to tiles that fulfill the requirement for a given rule. This result in a semi-controlled evolution that is able to create very varied results.

Our cellular automata is used to generate the initial heightmap of the phenotype. This map is smoothed out, but forms the basis of the search as it ultimately is what is used to check if a map is feasible or not.

The cellular automata is initialized by randomly seeding the map with the different height levels through the use of random number generation. The default odds are the following, but can be changed by by the user:

- **Height2**: 0.0 - 0.3
- **Height1**: 0.3 - 0.6
- **Height0**: 0.6 - 1.0

The CA starts at the top of the list and works its way down until it reaches a heightlevel where the random number fits into the range. Then it moves on to the next tile and repeats the process.

When we create maps, we only generate half the map and then turn/mirror it onto the empty map in order to save time. We do something similar in the CA, where we seed half the map plus 10% of the map's height/width (whichever is relevant for the half). Each iteration works on the same area.

Had we only worked on exactly half the map, we would have had issues where the tiles in the middle row of the map would have "empty" neighbours. This would mean that the middle row of tiles would behave in unintended ways, which in turn would mean that the middle of the map would look weird.