Kingdon Barrett and Andrew Bair

Genetic Algorithms

Professor Joe Geigel

Project CP2: Problem Definition

Intelligent Cameras with Genetic Algorithms

Overview

For our project, we will attempt to produce using evolutionary programming an algorithm to solve the problem of camera control in a third-person video game. We will make some simplifications to the problem for the sake of ensuring a viable project, but I hope to gain some general insight into the problem at large, and perhaps some code that can be reused in the Tuesday Studios flagship product, Attention Lens.

Instance

An instance of the problem in the general case would include as parameters a particular motion of some objects in a 3D world, over a period of time. To simplify the problem, we can assume that we know the future; for the purpose of getting this project off the ground, we will simplify the instance even further to a single moment in time; there is no motion in this world.

Further, the world will be represented as a simple 2D plane. [Fixed] points of interest are represented as X-Y coordinate pairs. The problem instance's camera is preset with a particular fixed field-of-view, that is, the polar distance across the screen in degrees.

Solution

A given solution will embody the location and orientation of a particular

camera. This is simply an X-Y coordinate pair, and a degree measure indicating the direction to the center of the field-of-view. In a temporally-aware algorithm, perhaps all of these values would be vectors indexed by time.

Output

The fitness of a particular solution will be determined by the total percentage of all points of interest that are captured within the field-of-view of the camera; ideally our camera can see every point of interest in the world, all at once. Also accounted for in the fitness algorithm will be the average distance between the camera and visible points of interest; if we can zoom in on the action without sacrificing information, we should certainly do that.

Implementation Plans

First order of business: we are planning to complete CP1 ASAP. I would have come to you sooner to ask for an extension, except that all of my good excuses happened well after the due date. (Sorry!) I hope you will still be willing to accept CP1 for some credit as soon as Wednesday 1/10.

Before 1/18 (CP3), we plan to have formally defined our genotype and phenotype in Python, using the pygene library. Within another week (CP4), we will have defined and implemented our fitness algorithm, and hopefully a visual representation to go with.

By 2/1 (CP5), we will add capabilities for crossover and mutation. At this point, it will be possible to seed our algorithm with a random sample population, and hopefully to evolve these genes into fit cameras! Before the CP6 deadline (2/8), we will have produced some sample problems which our algorithm can solve, and a report including example output. The remaining time (until 2/23) is

reserved for “making it cool.”

Possibilities for extensions to the project spec are many. A graphical front-end would add to the perceived value of the project in a possible demonstration setting. Temporal awareness built into the algorithm would bring this closer to a realistic solution for the very real problem of camera control in games. Stylistic control and a plugin architecture are planned for future iterations of Attention Lens, but are unlikely to surface this quarter. Integration with CubicEngine, the event-driven framework for Tuesday Studios development, would make for the most challenging and technically impressive extension.

(FYI: CubicEngine is a work-in-progress written in Ruby. Sort of. It's written on top of SWIG, which from my understanding does the translation from Ruby to C. That's because all of the really useful graphical libraries are written in C or C++. Sounds like a mess to me, but it's portable across architectures! But throw also Python into the mix? All I can say is eew...)