Optimization in Strategy Games: Using Genetic Algorithms to Optimize City Development in FreeCiv

Optimization in Strategy Games: Using Genetic Algorithms to Optimize City Development in FreeCiv

Ian Watson, Damir Azhar, Ya Chuyang, Wei Pan & Gary Chen

Abstract

There is a growing demand for the use of AI techniques in videogame development to improve user interaction and immersion beyond that achieved by the great increase in sophistication seen in other areas of development like graphics and sound. Videogames have hence become an increasingly important testing ground for AI research, with the lucrative nature of the industry being an additional bonus. This project deals with the implementation of a genetic algorithm (GA) to optimize city development for a turned based strategy game Freeciv, with the hope of greatly reducing the need for micro – managing various facets of city development, a problem that often plagues games of this ilk. This paper is an interim report that covers issues key to the project, and provides a discussion on the work done so far.

1. Introduction

Over the last few years the videogame industry has become a huge and important entertainment industry, with sales revenues in America alone exceeding 9.9 billion dollars in 2004 [1]. Hardware advances have led to vast improvements in the areas of graphics and sound which whilst being beneficial for the player, have also made it a lot easier for them to detect nonsensical or questionable behaviour on the part of non player characters (NPCs). Therefore the use of AI techniques to improve NPC behaviour has become increasingly more important for videogame developers to make their games standout in what has become a highly competitive market [2].

This project deals with the implementation of a genetic algorithm (GA) that will be used to optimize city development for Freeciv, an open source turn based strategy game similar to popular titles like Civilizations and Age of Empires. As with other strategy games of this type, in Freeciv the player takes control of a civilization of their choice and manages its development, with the ultimate goal of leading their civilization to becoming the "Greatest Civilization" [3]. This involves a lot of micro – management, particularly with regards to dealing with the various factors involved in city development. This process can be tedious, especially with large civilizations made up of numerous cities, and new players not familiar with the various intricacies of the game often face a very steep learning curve. The aim of this project is to implement a genetic algorithm that will run in the background and adjust the city development factors so as to optimize city development, leaving the user free to deal with other facets of the game. As the game environment is dynamic, this optimization process will have to be ongoing, adapting to environmental changes as they occur at each turn.

This interim report looks at issues key to the project and details the work done so far. In the following section, a brief discussion of related work using GAs in videogames is given. This is followed by a more detailed look at Freeciv than the one given above. Section 4 discusses genetic algorithms and section 5 deals with implementation details. The report ends with summary and a description of the goals for the rest of the project. An appendix of this group's learning objectives will also be provided.

2. Previous Work

There has been some research into the application of GAs in implementing videogame artificial intelligence systems. Cole et al. [4] used GAs to fine tune Counter Strike bot behaviour. Counter Strike is a popular first person shooter, and its game AI uses a rule – based system with a set of hard – coded parameters (thresholds) to determine bot behaviour. The more extensive the set of parameters the more realistic the bot behaviour possible, but fine – tuning these parameters is a difficult process of trial and error and requires extensive knowledge of the game's strategy [4]. A GA was implemented to fine – tune parameters dealing with bot weapon selection and aggressiveness, resulting in bots that performed as well as those with manually tuned parameters [4]. Myers and Rylander [5] used GAs to plot efficient paths between two endpoints in a 3D landscape that would take into account elevation in addition to distance and obstacles. The resulting paths were more realistic then those obtained using a form of memory limited A* search commonly employed by strategy games like Age of Empires 2 [5]. Kendall and Spoerer [6] utilized GAs to evolve scripts used to solve various stage maps in a scaled down version of Lemmings, a popular 90s puzzle game. They discovered that the propagation of strategies learned from earlier runs on simpler maps (as opposed to having a random initial population), increased the speed with which a GA would evolved towards a successful strategy.

GAs have also been used as a machine learning tool for board games. Alliot and Durand [7] developed a GA to evolve the parameters of the evaluation function of an Othello program resulting in an improvement in performance. Graphics and animation is another related field where GAs have been used with successful results. In his paper *Evolving Virtual Creatures* [8], Karl Sims describes the implementation of a GA that created virtual entities by determining both their morphology and the neural circuitry involved in moment. Different fitness functions were used to direct the evolutionary process of these creatures towards different movement behaviours (e.g. walking, jumping and swimming) resulting in novel morphological configurations.

3. Freeciv

As mentioned previously Freeciv is a turn based strategy game, which is essentially a free open source version of Sid Meier's Civilizations. The player takes control of a civilization of their choice, and then manages the expansion of this civilization by building cities, and directing economic, scientific and military development. The ultimate

aim of the game is for the player to lead their civilization into becoming the greatest civilization. There are three possible ways of achieving this [3]:

1) Eradicating all other civilizations on the map.
2) Reaching the space age and being the first civilization to have a spacecraft reach Alpha Centauri.
3) Having the highest score at a preset deadline.

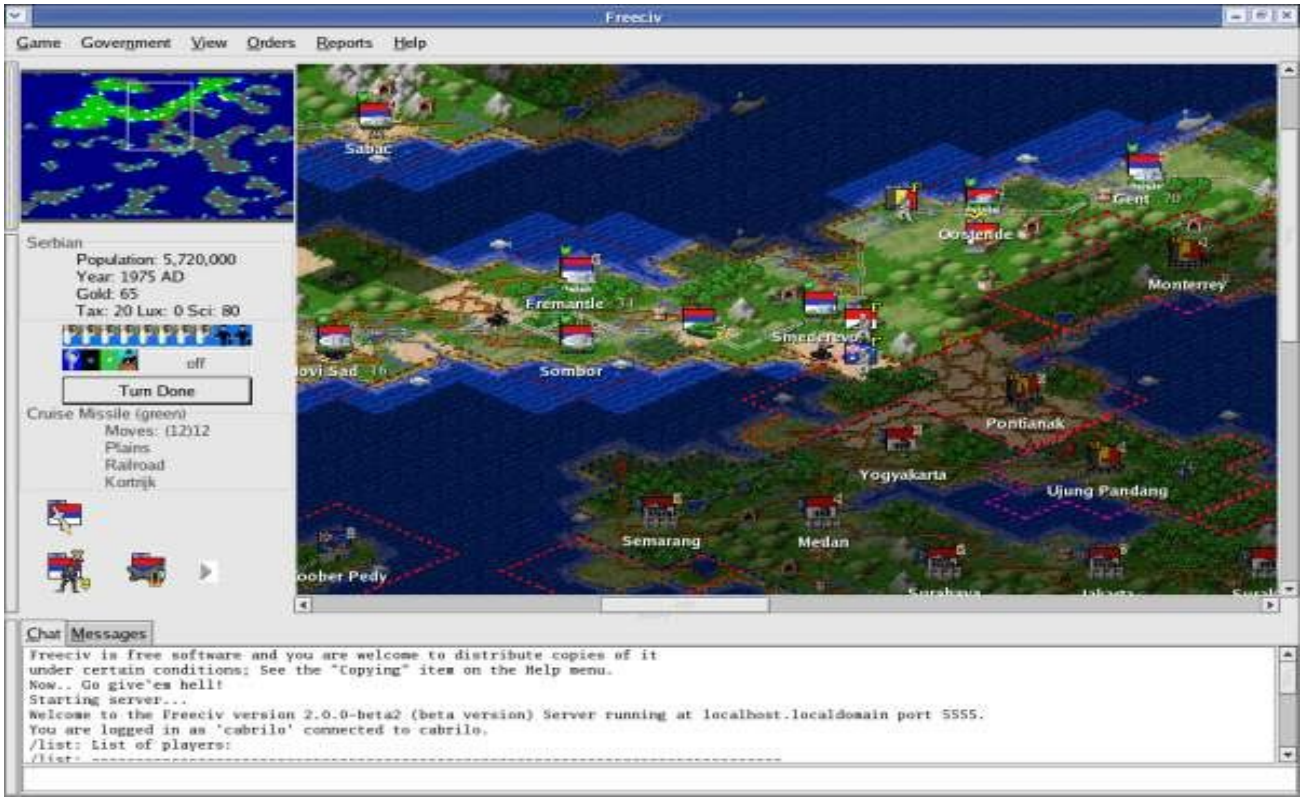

Figure 1: Screenshot of Freeciv user interface [9].

In Freeciv, cities are the fundamental elements of a civilization. Cities can extract resources from the map square that they are located on, plus *N* nearby squares where *N* is city size (initial value one) [10]. To obtain a resource a worker has be assigned to an unused resource square. This is done automatically by the game or can be adjusted manually via the city management screen.

There are three types of resources: food, production and trade [10]. The food resource is used to grow cities. All food obtained from occupied resource squares are stored in a granary. Every turn, 2**N* food units are used to feed city inhabitants. The presence of specialized units (e.g. settlers) may increase food consumption. Once food has accumulated to a certain level within the granary, city size increases by one, making an additional resource square available.

The production resource is used to build settlers (units that are used to build new cities) and military units [10]. Settler and military unit production is determined by the player

via the city management screen. Certain units require a production resource every turn for upkeep, and building a settler reduces a city's size by one (and hence is only allowed in cities of size two or greater) [10].

The trade resource can be used for science, tax or luxury [10]. Scientific development allows for research into new types of military units. Tax acts as money that can be used to buy buildings and units, hence prematurely completing the production cycle. Luxury plays a major role in citizen happiness. Trade is adversely affected by corruption.

In addition to resources, there are other factors that affect city development. Research involves selecting a long term knowledge acquisition goal for the civilization. Achieving research goals enables new types of military units and wonders to be built and opens up access to addition governmental systems. The type of government used to run a civilization affects the amount of resources a city can extract from the map, the rate of utilization of city resources, citizen happiness and corruption levels. Wonders are special buildings that can only be built by a single player once in a single city. Each wonder has its own specific affect on a civilization, some of which directly involve city development. Lastly there are a variety of map square types (e.g. ocean, grasslands, plains) each yielding a different amount of the three resources. Certain square types provide extra options for improvement like irrigation, mining or road building that result in increased resource acquisition when those improvements are performed.

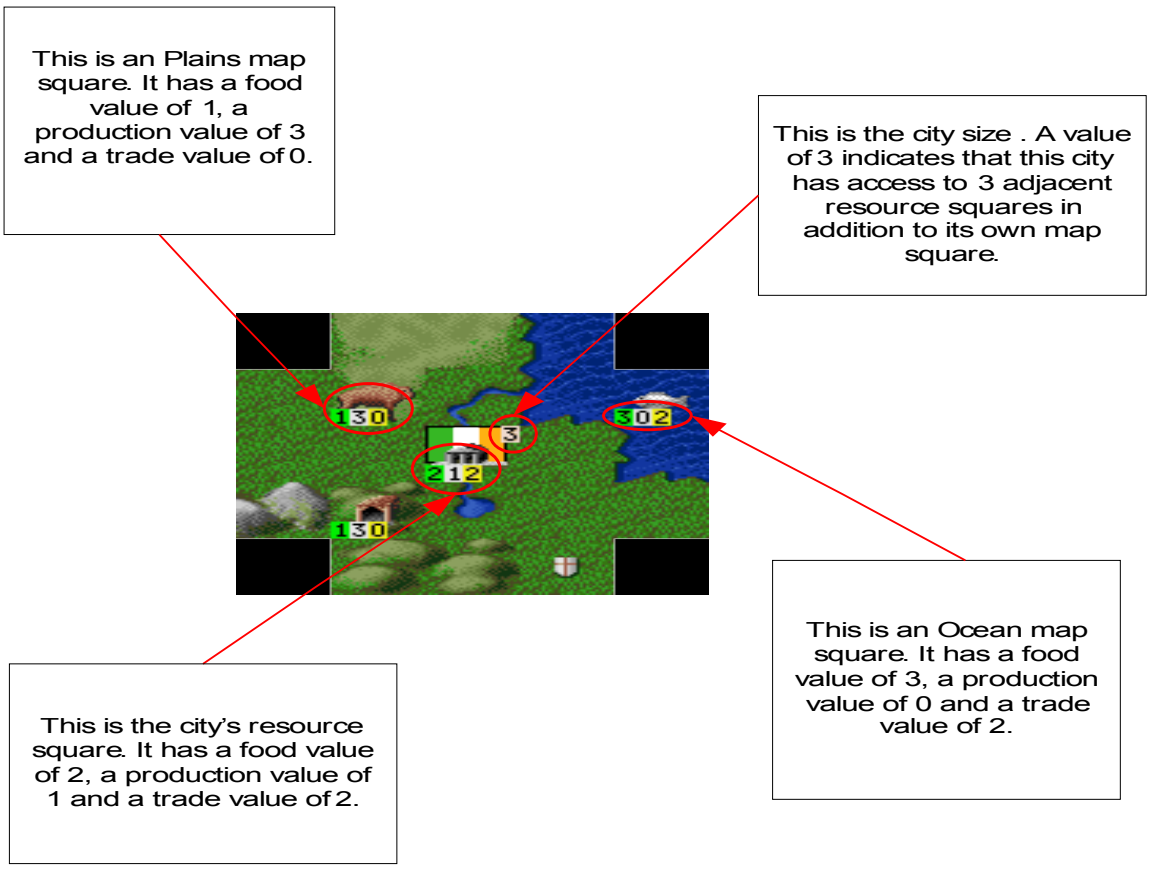

Figure 2: A single Freeciv city

4. Genetic Algorithms

Genetic algorithms are a learning methodology modeled on the process of evolution [11]. An initial population of hypotheses is used to generate successive populations of hypotheses using evolutionary procedures like crossover and mutation (a hypothesis is a possible solution to a certain problem domain). At each evolutionary step all the hypotheses within the population are evaluated to assess their quality or fitness according to a predefined function known as the fitness function [11]. Hypotheses with higher fitness values are selected with a greater probability to be members of the next generation, then those with lower fitness values. The crossover and mutation operators are used to increase diversity between members of a population. A general outline of the GA procedure is given below [11]:

- Selection: A proportion of the current generation is selected for the new generation, usually on the basis of fitness value.

- Crossover: The remainder of the new generation is made up by probabilistically selecting pairs of hypothesis and producing a new pair of hypothesis by applying the crossover operator.

- Mutation: A small percentage of the new population is then changed by applying the mutation operator.

This new generation then becomes the current generation, which will be used as the basis for the formation of a new generation at the next evolutionary step.

GA hypotheses are usually represented as bit strings which can easily be manipulated by the crossover and mutation operators. The crossover operator works on two "parent" hypothesis strings and produces two new "children" hypothesis strings by swapping selected segments between the parents [11]. A variety of crossover operators exists that differ with regards to what segments are swapped between the parents. For example with a two – point crossover, the children are generated by swapping the intermediate segments of both parents. An example is given below:

$$\begin{array}{cc} 10\underline{1100}100 & 100001100 \\ \rightarrow & \\ 11\underline{0001}110 & 111100110 \end{array}$$

The mutation operator randomly selects a single bit in the bit string and changes its value [11]. This is analogous to a biological point mutation.

$$1011\underline{0}0100 \rightarrow 101110100$$

As mentioned previously the fitness function is used to rank hypotheses within a population according to a predefined notion of quality [11]. For example if the problem

domain is a classification one, then one possible measure of quality is classification accuracy. Therefore a GA requires that the outcome of applying a hypothesis can be evaluated. This makes GAs useful in problems where the hypotheses are complex and are difficult or impossible to model. Looking at the Freeciv description in the previous section it can be seen that there are numerous interrelated factors dealing with city development, and so modeling them would be a difficult task that would require intimate knowledge of the Freeciv rule set. However the outcomes of changing these city factors are easy observable, making GAs a good choice for dealing with city development.

The fitness value of a hypothesis, as determined by a fitness function, is closely related to the probability that it will be selected for use in creating the next generation of hypotheses. With fitness proportionate selection a common selection procedure, the probability that a hypothesis is selected is directly proportional to its fitness value and inversely proportional to the fitness of other hypotheses within the population [11]. Another selection procedure, tournament selection, first selects two hypotheses at random from the current population. The more fit of these two hypotheses is selected for the next generation with a predefined probability of p, and the less fit of these two hypothesis is selected for the next generation with the probability $(1 - p)$ [11]. The ultimate aim of the selection procedure is to generate a new population of hypotheses that are more optimal on average then those of the previous generation. Thus GAs can be viewed as an optimization strategy, where the optimization is centered on the fitness function [11].

5. Design Decisions

We considered two alternatives ways of utilizing a GA to optimize city development. The first alternative involves training a GA over numerous Freeciv games to generate an optimal hypothesis that could be generalized to manage city development for any Freeciv game. This "offline" option would have to be able to deal with all possible game scenarios, which considering the complicated nature of Freeciv's gameplay, would be very difficult to achieve. Another problem would be the training process, not only due to our lack of computing resources, but due to the lengthy nature of a Freeciv game (depending on the skill of the player, a single game can last for numerous hours).

The second alternative involves using an "online" GA, i.e. one that would run in the background whilst the player played the game. The GA would be used at every turn to find an optimal city development strategy for the current game state, with some form of look ahead in order to ensure that the optimization is global. An advantage of this approach is that it is more dynamic then the offline option, with the optimization process being an ongoing one. This avoids the need to be able to handle all possible scenarios, a major cause of difficulty with the offline alternative. A downside to the online option however, is that all the processing required for the GA would occur whilst a user is playing the game. As this process may be lengthy, a player may end up waiting for a significant period for the next turn. Thus when using this alternative a balance has to be achieved, between the performance of the algorithm and game playability. In the end however it was the online alternative that was selected.

6. Implementation

Work on implementation was split into two tasks each allocated to two team members. Data extraction and Freeciv GUI extension was handled by Yachu Yang and Wei Pan, and is discussed in their interim reports. This section deals with the second task which involved the implementation of a GA framework. This framework will form the basis of our GA that will be used to optimize city development in Freeciv.

The GA framework is basically a three tiered structure. At the lowest level there are ***Genes***. These are integer arrays, each representing a single city development factor. Therefore intuitively a ***City*** (the second tier) will consist of a collection of genes dealing with the various factors involved in its development. As normally a civilization consists of more than one city, the ***Genome***, the top-level structure consists of a collection of cities.

The gene structure has 3 attributes; an identifier attribute, a length attribute and a range attribute. The identifier attribute is simply a string reflecting the purpose of a particular gene (e.g. the "happiness" gene or the "food supply" gene). The length attribute stores the length of the integer array that represents the gene. For example given a city of size four which would have five possible resource sites, then one possible representation for available resources would be an integer array of length five. Each element of the array would hence represent a single resource site. The range attribute determines the set of possible values that an element of a gene's integer array can take. Binary values for example are a common choice for hypothesis representation [11]. Going back to the previous resource example a 0 could be used to indicate that a resource site is not being processed by a worker, and a 1 to indicate that it is being processed by a worker.

The city structure has a similar set of attributes. The identifier attribute will store the city name as determined by the Freeciv player. The gene array attribute is, as the name suggests, the array of genes structures that handle city development. The length attribute is simply the length of this array, i.e. the number of genes involved in city development.

The genome structure has a city array attribute, a length attribute and a genome sequence attribute. The city array attribute is an array of city structures, representing all the cities currently present within the Freeciv player's civilization, with the length attribute storing the length of this array. The genome sequence attribute is the overall array of integers representing all the genes of all the cities. In other words the genome sequence is the structure that will be used, to represent hypotheses dealing with the problem of city development for a Freeciv civilization. It is also therefore the structure used by the crossover and mutation operators.

The following is a simple example of this framework being used to represent a small civilization consisting of two cities. The development of City 1 is determined by two city

development genes of length two and three respectively. The development of City 2 is also determined by two city development genes, both of length three.

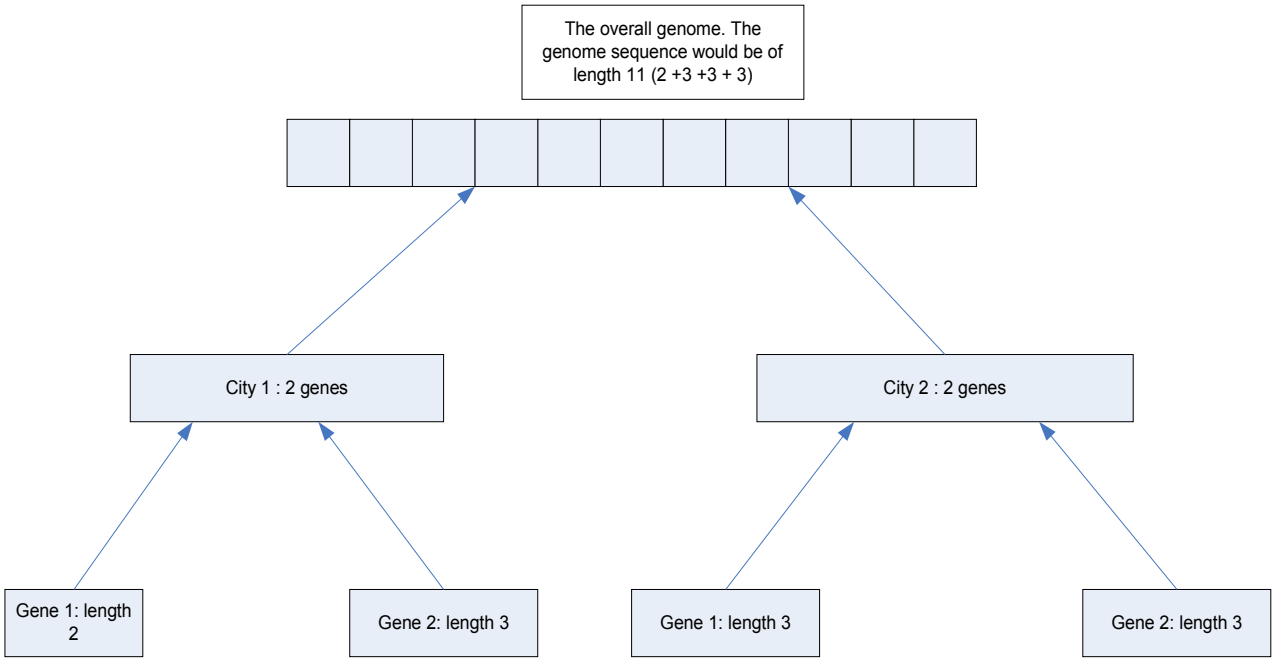

Figure 3: A graphical representation of a genome for a small civilization of two cities each with two city development genes.

The above diagram shows the genome structure that would be common to all genomes within a population if a GA was performed. Individual genomes would differ only in terms of the values stored within the genome sequence array. If only binary values are allowed, then 10110001001 and 01111101011 are valid genomes, each representing a different hypothesis.

The GA framework was made to be flexible not only because representations for the various city development factors have not been finalized yet, but to also deal with the fact that the game is a dynamic process, with changing states that might necessitate changes at the city or even the gene level.

For the selection process and GA operators, tournament selection, two point crossover and point mutations (all discussed in section 4) were implemented. The crossover operation is taken to be symmetric in order to preserve the genome structure and not waste time producing invalid genome configurations. Implementation of other selection processes/crossover techniques would be a simple task to be performed when and if required.

7. Summary and Future Work

The aim of this project is to implement an online genetic algorithm to optimize city development for Freeciv, an open source turn base strategy game. The optimization process will be an ongoing one to reflect the dynamic nature of the game. GAs have been used as AI tools for videogames in the past, with successful applications in improving Counterstrike bot behaviour and path finding in strategy games. In terms of progress made, the following things have been achieved:

- City data extraction. This data will play a crucial role in determining genome structure.
- Extension of the Freeciv GUI to provide a player with a way of interacting with the GA.
- Implementation of a flexible GA framework that will form the basis our GA.

Now that the GA framework has been built, final representations for the various city development factors must be decided upon. Our GA package then has to be integrated into the Freeciv package. Once this has been done our GA framework will initially be used in a simple scenario, e.g. optimizing city development in terms of only a single city development factor. This can progressively be extended to eventually deal with all the factors involved with city development. Various GA factors like mutation rate, population size, number of generations, type of cross-over procedures used, type of selection procedures used, etc. will be mostly be tweaked over the coming weeks to improve optimization whilst maintaining game playability.

Reference:

1. Annual U.S. videogames sales, http://retailindustry.about.com/od/seg_toys/a/bl_npd012703.htm, NPD group, January 2005.

2. *Computer games with intelligence*, Johnson, D., Wiles, J., The 10th IEEE International Conference on Fuzzy Systems, Volume 3, 2-5 Dec. 2001, Page(s):1355 – 1358

3. Freeciv Game Manual, http://www.freeciv.org/index.php/Game_Manual.

4. Cole N, Louis SJ, Miles C. Using a genetic algorithm to tune first-person shooter bots. [Conference Paper] Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753). IEEE. Part Vol.1, 2004, pp.139-45 Vol.1. Piscataway, NJ, USA.

5. Myers R., Rylander B., Divide and Conquer in Genetic Algorithms: Generating Paths on Heightfields, 2002, http://upibm9.egr.up.edu/contrib/rylander/studpubs/ryan.pdf.

6. Kendall G, Spoerer K. Scripting the game of Lemmings with a genetic algorithm. [Conference Paper] Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753). IEEE. Part Vol.1, 2004, pp.117-24 Vol.1.
Piscataway, NJ, USA.

7. Alliot J.M., Durand N., A genetic algorithm to improve an Othello program, Artificial Evolution, pp. 307 – 319, 1995.

8. Sims K. Evolving virtual creatures. [Conference Paper] Computer Graphics Proceedings. Annual Conference Series 1994. SIGGRAPH 94 Conference Proceedings.
ACM. 1994, pp.15-22. New York, NY, USA.

9. http://en.wikipedia.org/wiki/Freeciv

10. An Introduction to Freeciv, http://www.freeciv.org/index.php/Introduction_to_Freeciv

11. *Machine learning*, Mitchell T. M., McGraw-Hill 1997, Page(s): 249 – 269.

Appendix: The learning objectives of this project

1 Genetic Algorithm theory

In order to apply the genetic algorithms, we need to understand the principle behind genetic algorithms. Genetic Algorithms can be viewed as a general optimization method that searches a large space of candidate solutions to seek one that performs best according to a predefined fitness function ([1]). Genetic algorithms will start with an initial population of candidate solutions, it will then evolve them by selecting the fittest individuals to the next generation population based on a predefined fitness value. Also, all members of the current population get selected to the next generation by random mutation and crossover operations. We suggest this reference [1]. It provides a good discussion on the theory behind genetic algorithms, and it is concise and easy to understand.

2 Application of Genetic Algorithm in games

Genetic algorithms have been introduced to solve different problem domains in games. In one of the recent studies [3], genetic algorithm had been applied to a first-person shooter game called Counter Strike. Genetic algorithms have been used to fine tune parameters of weapon selection and aggression setting of rule-based bots. Normally this process is manually done by developers through trial and error. In this study, they succeeded in automating the process using genetic algorithm. In one of the earlier studies, genetic algorithms can be used to procedurally generate virtual creatures and control their movement. Instead of designing the creature manually, which is time-consuming, the genetic algorithm provides a method to automatically generate a life-like creature [2].

3 Genetic Algorithm and Optimization

Optimization technique is used to find the best solution for a problem. Several optimization techniques exist, such as Hill Climbing, Simulated Annealing, Gradient Ascent/Decent and Genetic Algorithm. Genetic Algorithm are designed to efficiently search large, non-linear, poorly-understood search spaces where expert knowledge is scarce or difficult to encode and where traditional optimization techniques fail [3]. Hill Climbing, Simulated Annealing, Gradient Ascent/Decent are considered traditional optimization techniques. Suggested references: [1] [4]

Reference:

1) Mitchell T. M., Machine learning, McGraw-Hill 1997, pp.249-269.

2) Sims K. Evolving virtual creatures. [Conference Paper] Computer Graphics Proceedings. Annual Conference Series 1994. SIGGRAPH 94 Conference Proceedings. ACM. 1994, pp.15-22. New York, NY, USA

3) Cole N, Louis SJ, Miles C. Using a genetic algorithm to tune first-person shooter bots. [Conference Paper] Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753). IEEE. Part Vol.1, 2004, pp.139-45 Vol.1. Piscataway, NJ, USA.

4) Champandard A. J., AI Game Development Synthetic Creatures with Learning and Reactive Behaviours. New Riders Publishing, 2004, pp.205, pp.209-211.