Generating multi player maps through multi objective evolution

Tómas Guðmundsson

Kongens Lyngby 2012
IMM-MSC-2012-115

Master's thesis
Copenhagen, October 11, 2012

Supervisors : Michael Rose, Lektor, mir@imm.dtu.dk, DTU.
Julian Togelius, Associate Professor, julian@togelius.com, ITU.

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-MSC-2012-115

Abstract

In this project, the following research question is set forth: is it possible to create fair maps for a video game using multi objective evolution algorithms? A description of the video game used for this project, Civilization V, is provided as well as an overview of other map generation methods, and research being done in the field of procedural content generation. A definition for what is fair is made and expressed through functions, that evaluate maps for the video game Civilization V. These evaluation functions express five distinct perspectives on how fair maps are perceived. The fitness functions are designed to conflict as little as possible with each other. A method is defined as to how this theory is applied in practice to generate maps for Civilization V. The evaluation functions are applied on maps from the game's map generation method, and compared to maps that have been evolved with the method provided by this project.

Summary

I dette projekt er følgende problemformulering fremsat: er det muligt at skabe retfærdige baner til et videospil, ved brug af 'multi objective evolution' algoritmer? Vedlagt er en beskrivelse af videospillet anvendt i dette projekt, Civilization V, samt et overblik over andre metoder til generering af baner, ligeledes er forskning indenfor feltet 'procedural content generation' vedlagt. En definition af hvad der forstås ved retfærdige baner er fremstillet og udtrykt gennem funktioner, der evaluerer baner til videospillet Civilization V. Evalueringsfunktionerne udtrykker fem distinktive perspektiver på hvad der forstås ved retfærdige baner, samtidig med at der stræbes efter mindst mulige uoverensstemmelser. Defineret er en metode til, hvordan denne teori anvendes til at generere baner i praksis i Civilization V. Evalueringsfunktionerne er anvendt på baner fra spillets metode til banegenerering og sammenlignet med baner genereret af den førnævnte metode.

Preface

This thesis was prepared at Informatics Mathematical Modelling, the Technical University of Denmark in partial fulfilment of the requirements for acquiring the M.Sc. degree in computer science and engineering.

This project focuses primarily on evolutionary algorithms. A method is created for this purpose that uses fitness functions and a cascading elitism evolutionary strategy. The fitness functions are also used to evaluate maps that have been created with this method or from the video game Civilization V. A map generating technique is also introduced which uses a variation of brush drawing.

This project is based on research performed in the procedural content generation research field, computational intelligence field and evolutionary algorithms. This project is heavily influenced by the work done by Julian Togelius, Georgios Yannakakis et. al.

Lyngby, October 2012

Tómas Guðmundsson

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Is it possible to develop an algorithmic method for designing levels, that provides for an interesting yet balanced game-play in a video game? Video games are as varied as they are many. Thus applying broad terms like balanced and fair is in and of itself a vague description. Chess is a good example of a game that is unbiased and fair, with each player starting with the same amount of pieces, where each piece has the same ability for both players. Chess is also a good example of how, with enough time and practice, players can become more skilled than others. Games can be a form of entertainment and if they do not entertain us we are less likely to play them. Games that are unfair or biased to some players, may risk that the these players would not want to play that game. It must thus be a goal of game designers to ensure their games are fair and balanced.

What does it mean for a game to be fair and balanced? A fair and balanced game can be described as when every player has an equal chance of achieving victory. When there are no competing sides, such as in puzzles, the balancing moves from player versus player to player versus game. In genres like first person shooters, racing, sports and real-time strategy, it is possible to distinguish where fairness should be measured. A race car offered to one player that is worse than every other player receives is unfair. A gun in a first person shooter, that kills with one shot while every other gun needs ten shots to kill is unfair. These kinds of biases can contribute to a fair game, so long as there exists biases to every player that can be considered as equivalent. For example, a gun that has ten times the firepower of other guns can be met with armor that can absorb ten times the firepower of other armors. When there are unmet biases, some players will have an advantage and thus the game is considered unfair. To level the playing field, game developers try to balance their games by decreasing or increasing the strength of a game's elements or introduce a handicap on some players. The aim is to provide all players with equal chance of victory.

1.1 Problem statement

The idea behind this project is heavily influenced by the work done by Togelius et. al. [13] where the vision is to create content for video games that is unique for each player. It goes even further and attempts to create content that is based on what is believed to be fun for the player. Games that would incorporate this kind of content generation, would allow all their users to experience a game in the way they think is fun. Gone are complaints of stale material or request for creating new content after the old content becomes boring. Defining what is fun for each player is hard to do and is very qualitative. Defining what is fair is also qualitative, but can be described quantitatively.

The goal for this project is to be able to create maps for the video game Civilization V, that can be though of as fair. This includes coming up with a definition of what fair is. To create these maps an evolutionary algorithm will be used. There are two main parts to that, generation and testing. A map generation algorithm will be devised as well as fitness functions to evaluate the maps. This allows the evolutionary algorithm to find maps that are fair by definition. The definition will be determined by the fitness functions. Using these methods, maps will be created for Civilization V where players will have an equal chance of achieving victory.

1.2 Thesis structure

This thesis starts by explaining the concepts used in the research, such as evolutionary algorithms, map generation and procedurally generated content, as well as describing the application domain, which in this case the game Civilization V. Then a detailed description about the fitness functions that are used to evaluate maps generated in this project as well as maps created by the game itself. That paves the way for chapter 5 where the method created for this project is described. The results from experiments performed are then presented which is followed by a discussion and conclusions.

CHAPTER 2

Civilization V

Civilization V is a game made by Firaxis Games and published in September 2010 by 2K games[4]. The fifth in a series that dates back to 1991 with the release of Sid Meier's Civilization. Civilization V is a turn based strategy game that focuses on expansion and world domination. World dominance is the key objective which can be acquired through various means. The game was very well received critically with a Metacritic score[1] of 90, on August 24th, 2012 [15].

Like many video games today, Civilization V has had additional content created for it since its release. Various fixes have been made to the game since its conception. With downloadable content added to the game, it gets expanded in terms of resources and playable civilisations among other things. In June 2012, Firaxis released the first expansion pack to this game called Civilization V: Gods and Kings[5] where new features such as religion and espionage were added. Civilization V, without any additional content, contains 19 civilisations, each with their own specific bonuses and some with specific units. The game also includes non playable characters such as barbarians and 28 different city states.

[1]Metacritic is a website that calculates average scores from different reviewers.

2.1 Game Description

Civilization V is a tile based game, where each tile is in the shape of a hexagon. The game takes place on a map which consists of these tiles.There are 9 different types of terrain that can be placed on a tile. Each terrain has a host of possible resources and features that can be placed on those tiles. The game has 28 different resources split into three categories, Bonus, Strategic and Luxury. Not all Resources can fit on every terrain, for example you will not find bananas on an ocean tile and you will not find a whale on a mountain tile. Resources will have different effects to a player. Bonus resources mainly yield food, strategic resources only yield production and luxury resources mainly yield gold.

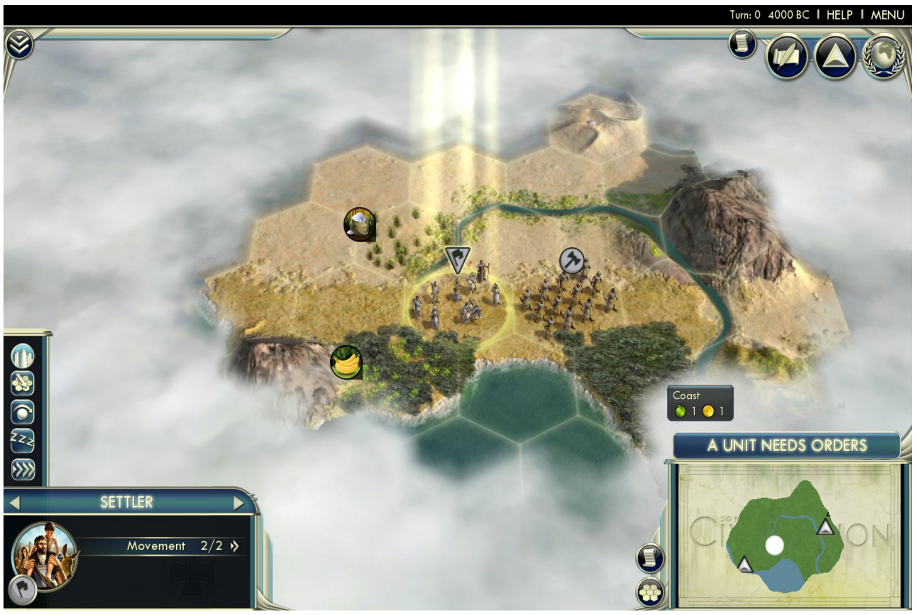

Figure 2.1: A screenshot of a typical start in Civilization V using the Earth map. A river starts from player starting position and floats to the sea.

In Civilization V, there are several meta-resources that benefit a player. These meta-resources are, food, production, gold, culture, happiness and science. With the expansion pack *Gods and Kings* religion is also introduced. Food harvested by a city increases that cities growth of population. Production is used for producing buildings and units. Culture is used to issue civilisation wide policies and increase the influence of the civilisation. Happiness affects the productivity of the cities as well as growth. Science will further your civilisation knowledge of technologies. Gold can be used in various ways, to purchase influence over land, units, buildings and to trade for peace. There are various ways of generating each of these throughout the game. A big component of meta-resource generation comes from the land a player occupies. Each unique luxury resource that gets introduced to the civilisation generates extra happiness. Grassland provides food while Plains provides food and production.

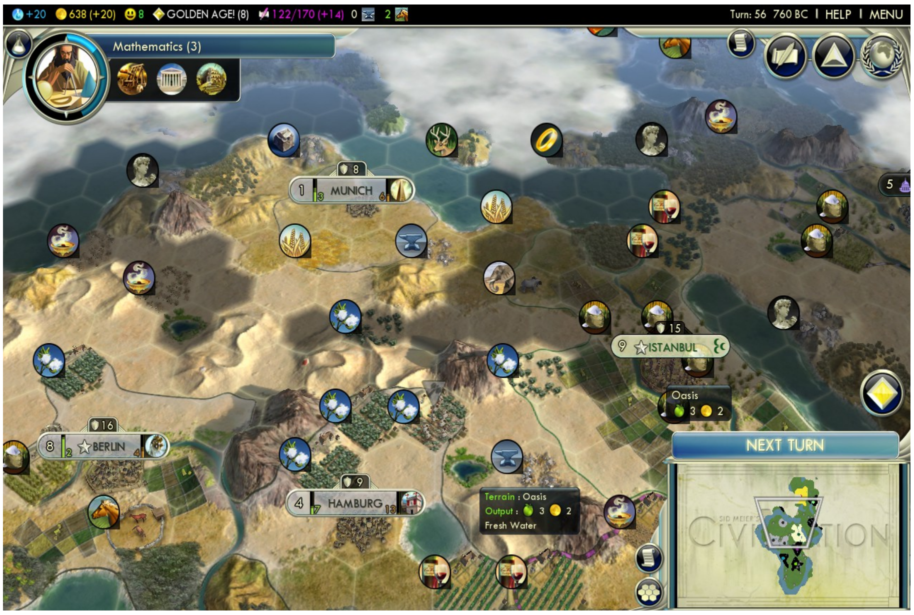

Figure 2.2: A screenshot of resource diversity and iron distribution between two capitals.

Besides resources, the Civilization V map also includes features. These features are varied and serve the purpose of making the map more interesting. Some features can only appear once in each map such as Mt. Fuji or the Rock of Gibraltar. Other features are for example, Marshes, Forests and Oasis. Features also add various yields to tiles. For example, a forest grants one extra food on a tile, but decreases productivity in a city. See Appendix A, B and C for a complete list of terrain, resources and features, respectively.

On top of features and resources, Civilization V allows players to improve a tile. In all there are 20 improvements that are related to what type of terrain or feature is located on the terrain. For example, a land tile with an Oil resource on it, can be improved with the Oil Well. A sea tile with an Oil resource on it, can be improved with the Offshore Platform. A Citadel can be constructed on all land terrain which increases defence for army units while a Camp can only be constructed where Elephants (Ivory), Furs or Deer can be found. This aspect of the game is excluded in the research done by this project.

The player starts the game in the Ancient era. By technological advancement and expansion of the player's civilisation he progresses through different eras to the so called Future era. It is completely up to the player what era starts when and players can research the technology of Railroads in the 1600's or in the 2000's. With this in mind, a game of Civilization V can be quite complex and vast. For this project only a partial subset of the whole game is considered, which is described in section 2.3, taking everything into account is beyond the scope of this research project.

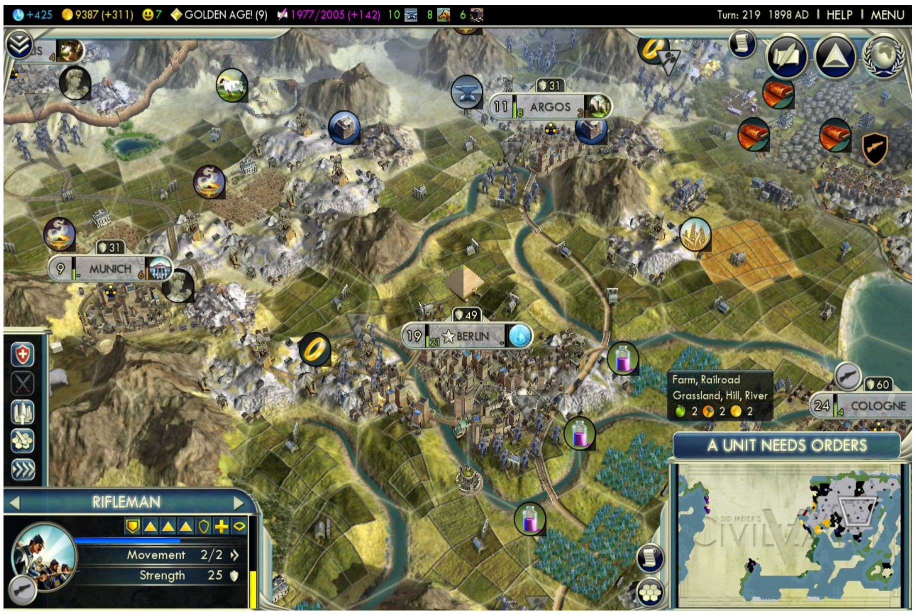

Figure 2.3: A screenshot of a game that is in turn 219, demonstrating resource distribution between cities and improvements.

2.2 Civilization V Map Generating Methods

Civilization V uses several methods to generate maps. A full list of the methods can be found in Appendix D The implementation of these methods in the scripting language can be found in the software development kit that comes with the game when purchased through Steam[2]. The methods provided with the game are tailored to an all round experience with the game, single and multi-player, duels and free for alls as well as team based games. The game also provides some realism when offering players to play a map of the earth, which is pre-determined and does not go through the script generation of the random maps.

One method is the Fractal generation. Designated as "Civ5's default map script"[6] this method serves as a base method from which others expand upon. After performing the fractal generation of the map, creating continents of land on a canvas of Ocean terrain. Then terrain is distributed over the land, based on the latitude of a tile to imitate the climate of the earth. This way, snow is only present on tiles that are in the top and bottom 25% of the map. Next rivers are added and lakes. This method then goes on to divide the map into regions for civilisations and selecting starting positions for players. It continues by placing resources, first it adds luxury resources, then strategic and then bonus resources. This method does quite well in creating maps that generally look like a reasonable substitute for real landmass.

Skirmish is another map generation method which is described as "Optimized for 1v1 or two-teams multi-player"[7]. Unlike the fractal method it allows users to define some input parameters. Users can define what will be the dominant terrain for the map. Be it Grassland or Desert this allows users to aid the map in selecting what will be the most dominant terrain on the map. This effectively makes the surrounding environment of every player starting position have the same terrain. In itself, that will effectively make the map more fair since it excludes possibilities of a players starting in a Desert while other players start in fertile environment. Users can also declare how much water will be added to the map, if it should be completely dry or contain rivers and seas. As mentioned in the description of this method, it is focused on games featuring multiple players competing against each other. To combat any shortcomings of resource distribution the game adds what is called *Strategic Balance*, which essentially places 6 iron resources and 6 horses close to every players starting position. This method gives players different maps every time that are very similar and predictable. After a player has configured this method it is quite clear what kind of map he is about to play. Whether or not there will be lakes or seafaring or if there are any mountainous or snowy regions, are assumptions the player can ignore while playing a map using this method. Due to its characteristics, this method is suitable for competitive Civilization V play.

[2]http://store.steampowered.com/app/8930/ - accessed 15th of September 2012.

2.3 Partial representation Civilization V

In this project a subset of Civilization V will be considered. As the game is very complex and vast it is necessary to limit the scope of this project to what is more manageable. Any consideration to later stages of a map in Civilization V were not considered, that is terrain improvements. Rivers and lakes were excluded from this project. Civilisation specific advantages were not included as well as, technologies, culture and happiness. This subset can be viewed as a snapshot of the map at the start of the game. By excluding these factors it is possible to compare results achieved in this project with this subset. For example, including civilisation specific advantages would require the map generation, in addition to player starting position, include what civilisation the player is. This would prove to be too much work for this project. Therefore, all players are assumed to be playing the same civilisation. A snapshot of the beginning of the game is only considered to make comparison easier.

Chapter 3

Techniques

Constructing maps for video games can be done in various ways. There are different algorithms that will achieve different goals. One approach can be described as top to bottom, where a map's most visible features, for example rivers and roads are created first and finally creates the terrain. Another approach can be described as bottom to top, where the terrain is first created and the approach ends by applying rivers, flora and other features.

Various algorithms and techniques exist today. Some are easily available while others are locked inside a company's intellectual property. The video game industry has been one of the leading pushers for advancement of algorithms. The difference in quality of graphics from video games in 2005 is vast compared to games made in 1995. A consequence of that improvement is better graphical quality in other industries such as movies and TV. Researching new algorithms and methods is important to keep the advancement going. This project aims to find a technique that can help the video game industry.

3.1 Map generation techniques

The map generation in the Civilization V game does fractal map generation. A fractal map generation is a fast algorithm that creates ridge lines that form land. One way is the mid point displacement technique. This is useful for 2 dimensional maps creating the outline of the map. A characteristic of this technique is that it is self-similar. Self similarity is when an object is similar to the subset of itself, magnified. With the mid point displacement technique being useful for 2 dimensional maps, an approach to three dimensional maps is an algorithm called diamond-square[17] algorithm. It takes an extra step to alter the height values as well. These techniques create realistic looking maps and terrain but it can be hard to control these algorithms since they are non deterministic. In Civilization V, once a map has been outlined then different techniques are used to fill it with terrain, resources and features.

Another method used in map generation is called Polygonal Map Generation. From random point it creates Voronoi polygons, smooths the polygons out and treats these polygons as a parcel that can be a part of a land or sea. After choosing what polygons are land, which are sea and which are lakes, it looks at elevation. Elevation is defined by distance from the coastline and then extra features are added to the map. A detailed description of this algorithm can be found in [1].

Video game developers do not always choose map generation methods. The fact is that when using map generation methods, they are supplied with either a random seed or a set of parameters. Even configuring generation methods with parameters will never amount to the same granularity of constructing a map by hand. Designing maps for games can be less time consuming than creating map generation algorithms. Video game developers benefit from the safety in manual map design. With manual map design it is easier to ensure all maps are playable and do not break the game. Today the quality of hand made maps is greater than many map generation methods. That is why some video game developers simply create all maps by hand. Maps that are procedurally generated can replace the hand made maps once these map generation methods get improved.

3.2 Procedural content generation techniques

A paper about search based procedural content generation has been written by Togelius, Yannakakis et. al [11] that "contains a survey of all publish papers known to the authors" on the subject. It classifies procedural content generation (PCG) in 5 ways.

- *Online* versus *Offline*
- *Necessary content* versus *Optional content*
- *Random seeds* versus *Parameter vectors*
- *Stochastic generation* versus *Deterministic generation*
- *Constructive* versus *Generate-and-test*

Online PCG algorithms are usually run right before or during game play thus having the requirement of generating content quickly. That is to say, that online PCG can not take more time generating content than reasonable expectations of users waiting. Offline PCG usually runs separately and can take longer time in generating content. The content generated is then available to be used in the game after the algorithm has finished. Necessary content in a video game is that if it is removed from the game, then it becomes unplayable, examples are protagonists, game rules, levels. Optional game content can be additional weapons, levels, flora and music. Seeds used in random number generators will generate random numbers to affect the content while parameter vectors allow user configuration. Choosing between those two depends on how much control over the content is wanted. Deterministic generation provides the same identical content based on input parameters where as stochastic can provide different content each run with the same parameters. Constructive algorithms spend some time in assessing the content it has created and makes sure it is "*good enough*" after it has created it once. Generate-and-test algorithms create the content and then evaluate it. After one pass it then regenerates content based on the evaluation results and continues until the content is "*good enough*".

Procedural content generation is not that widespread today but some of the research that has been done is exciting. In their paper, **Controlled Procedural Terrain Generation Using Software Agents**[10], J. Doran and I. Parberry create land forms aimed for computer games and flight simulators. With intelligent software agents they generate terrain elevation height maps according to some constraints. In their proof of concept they created five different agents that serve different purposes, for example creating coastlines, creating beaches along the coastline and creating mountains. The outcome is realistic landmass that is unpredictable but controllable.

N. Sorensen and P. Pasquier wrote a paper, **Towards a Generic Framework for Automated Video Game Level Creation** [20] where they describe an approach to generate levels for video games. They use the Feasible-Infeasible Two-population genetic algorithm to advance their search of solutions. Their genetic representation is composed of "building blocks that represent units of levels". These building blocks represent a one-to-one mapping from genotype to phenotype and therefore is of variable size to represent levels of different sizes. It goes on with examples from the Super Mario Bros. [19] game having building blocks of, a block, a pipe, a hole, a staircase, a platform and an enemy. With constraints put on the generated content they are able to create levels that are playable and interesting maps for the game.

J. Togelius, R. De Nardi and S. M. Lucas, in their paper **Towards automatic personalised content creation for racing games** [13] evolve racing tracks for a car racing game. They use artificial evolution to evolve two components, drivers and tracks. Using cascading elitism as the evolution strategy they successfully evolve controllers for the race car game based on human controlled drivers. The controllers evolved were then used to create tracks that suited each player's play style. The authors of this paper deemed the progress promising but "there is much that needs to be done in order for track evolution to be incorporated in an actual game".

J. Togelius, M. Preuss et. al. wrote a paper **Controllable Procedural Map Generation via Multiobjective Evolution** [12] which focuses on creating "suitably balanced maps for real-time strategy (RTS) games". Their representation in the genotype consists of base coordinates for 3 players, coordinates for two different resource types as well as feature list. Their focus on StarCraft[2] contains similarities to this project which focuses on Civilization V. The difference in these two games, Civilization V and StarCraft are considerable although both are RTS games. For example StarCraft has a big competitive community and official competitions such as the WCG[24] where the first prize grants 20,000 USD. This puts different demands on the quality and attributes of the maps used for either game. Their objective functions focus on base placement, resource placement and paths. Paths objective functions are aimed at finding overlaps of paths as well as choke points in a map. Some of their objective functions partially conflicted with each other due to their goals with the map design. Using a Pareto front optimisation as an evolution strategy, they were not able to determine whether or not a true Pareto front could be found. A result of their paper was that on maps that were created, "appear like a reasonably fair design that is hard to understand [...] but looks interesting."[12]

3.3 Technique chosen for this project

For this project a search based evolutionary algorithm approach was chosen. Evolutionary algorithms are not always an obvious choice since they don't guarantee good results. Evolutionary algorithms are instructed by fitness functions on what content is good and what is bad. For each result that an evolutionary algorithm outputs there is no telling whether or not it is a global or a local maxima. Despite these characteristics, evolutionary algorithm method has been chosen for this project since it can find good solutions.

The idea of instructing algorithms to find content that is defined by constraints is exciting. On the horizon it looks like algorithms may be able to determine what users prefer and will be able to come up with content that is interesting and challenging to users, unique to every player. That is the motivation behind this project and the reason behind choosing evolutionary algorithms to aid in this search. The method is defined, according to the taxonomy paper mentioned above, offline generation of necessary content using random seed. All content is generated deterministically through a generate-and-test method. Cascading elitism has been chosen to be the evolution strategy.

Chapter 4

Fitness functions

In order to quantitatively evaluate a map, fitness functions have been created. These fitness functions are used to evaluate a map after it has been constructed. These fitness functions are designed in such a way that they aim to be unbiased and fair to all players in a game. The sum of the fitness functions is meant to provide a definition of a fair map.

4.1 Continental connectivity f_0

Continental connectivity simply means which players are connected through a landmass that they can reach each other. This involves searching using A* search, and blocking out terrain that is considered impassable or unembarkable. A search is made from a player to every other player. The searches that reach their end get counted up and the sum is then divided by number of players times number of players excluding that player that is being considered.

```
for (player_A in players) {
    for (player_B in players) {
        if (player_A != player_B) {
            search = AStar(player_A.start, player_B.start, map)
            search.allow_mountains(false)
            search.allow_ocean(false)
            if (search.has_reached_target())
                legs++
        }
    }
}
return legs/(players_N)*(players_N-1))
```

Figure 4.1: Algorithm for fitness function continental connectivity.

4.2 Distance between teams f_1

A fitness function is needed for determining fair starting positions. The incentive being, that with an optimal distance between players. The players will then have both the possibility of exploration and constructing a defence for their cities. If teams start too far away from each other, it will become hard to maintain an attack on the other team. If teams start too close to each other players will fight over territory with very limited resources. It would also be discouraging to inexperienced players, playing against experienced players who would have the know how to eliminate them quickly. The optimal length is considered to be diagonal of a quadrant of the map. For each player in a team, the sum of the distances to every player on any other team should be the diagonal of a quadrant of the map. For this project, a setup of two teams was taken into account. Possible extensions to this fitness functions are for example, changing a quadrant into a triangle.

```
for (player_A in team_A) {
    for (player_B in team_B) {
        search = AStar(player_A.start, player_B.start, map)
        if (search.has_reached_target())
            total_length += search.get_distance_travelled()
    }
}
fitness = (total_length/number_of_players)
fitness = fitness / sqrt((Map::DIMY/2.0^2 + Map::DIMX/2.0^2));
return fitness;
```

Figure 4.2: Algorithm for fitness function distance between teams.

4.3 Resource assignment f_a

Resource assignment is the act of assigning ownership of a resource to a player. It is a prerequisite for fitness functions 2, 3 and 4. It goes through every resource in a map and assigns it to a owner. Here A* search is used to find the closest starting position of a player. When the closest player has been found the resource in question is determined to be his property. A player will never own a resource that is closer to some other player than himself.

```
for (row in map) {
  for (column in row) {
      if (cell.has_resource()) {
          foreach (player) {
              search = AStar(cell, player.start, map)
              search.allow_embark(true)
              search.allow_mountains(false)
              if (search.has_reached_target()) {
                  if (shortest(search.get_distance_travelled))
                        closest_player = player;
                }
            }

            cell->set_owner(closest_player);
        }
    }
}
```

Figure 4.3: Algorithm for assigning resources.

4.4 Unique resource distribution f_2

In Civilization V each player receives a bonus towards his empire's happiness when it discovers a new resource. So for example, the first time the Germans find Silver, the Germans become happy. The next time the Germans find Silver, they have already been impressed by it, and its discovery will not affect the Germans' happiness. When a player starts in an area that has more varied resource distribution, it makes for a more interesting game play. This fitness function also aims at keeping the number of unique resources that each player owns equal. The fitness function keeps track of whether or not a player owns a certain resource. It then goes on and calculates the percentage of resources a player owns. The average percentage is then returned as the value for this fitness function.

```
for (row in map) {
    for (column in row) {
        if (cell.has_resource()) {
            unique_found[owner][resource] = true
        }
    }
}

foreach (player) {
    foreach (resource)
        if (unique_found[player][resource])
            cur_unique_count++
    }
    unique_per_player += cur_unique_count/max_resources
}
return unique_per_player/no_players;
```

Figure 4.4: Algorithm for counting unique resources.

4.5 Strategic resource distribution f_3

In Civilization V resources do have different usages and benefits. Strategic resources are resources that not only benefit players with the usual yield of food, production and gold, but also serve as a prerequisite to building units. This is one of the most important aspects of ensuring fairness in the game. In a world where military strength can determine a winner, not having horses or iron will surely lead to a downfall. This fitness functions counts the number of strategic resources each player has. This percentage is then divided by the optimal percentage a player should have or $\frac{1}{pl_N}$ where pl_N is the number of players. This will sum up to the average percentage of strategic resources a player owns and is then divided by the number of players. The aim is to find a map where all players have an equal share of the strategic resources on the map.

4.6 Average player throughput f_4

Each resource gives different yield of food, gold and production. This fitness function counts the actual yield of each resource per player. The relative yields for each player is compared to what other players share is. This allows the evaluation search to find maps that contain equal share of resource yields between each player. The best n tiles from the players starting position, within a circle with a radius two tiles, is also

```
for (row in map) {
    for (column in row) {
        if (cell.has_strategic_resource()) {}
            strategic_count[player]++
            total_strategics++
        }
    }
}
foreach (player)
    cur_avg = ((strategic_count[player] / total_strategics) / (1.0/
        no_of_players))
    val += cur_avg;
}
return val / no_of_players;
```

Figure 4.5: Algorithm for fitness function that counts strategic resource distribution.

included.

```
for (row in map) {
    for (column in row) {
        throughput[cell.player_owner] += cell.yield
        throughput[total] += cell.yield

foreach (player)
    foreach (neighbour in radius(2))
        neighbour_yields[player].add(neighbour.yield)
    }
    throughput[player].add(neighbour_yields[0,4])
    throughput[total].add(neighbour_yields[0,4])

foreach (player)
    cur_avg = ((throughput[player]/throughput[total]) / (1.0/
        no_of_players))
    avg_player_throughput += cur_avg;
return avg_player_throughput / get_no_of_players
```

Figure 4.6: Algorithm for fitness function averaging player throughput.

4.7 Total value of a map f_e

When a group of maps have similar values in all fitness functions this fitness function, f_e serves as an heuristic for selecting the most fertile land. This grades maps in terms of yields, that is how much gold, food and production each map has. The more there is, the more players can do. It would also be possible use this fitness function to find the map with the least resources, making it more challenging for players to play on.

```
for (row in map)
    for (column in row) {
        throughput = cell.yield

return total_throughput;
```

Figure 4.7: Algorithm for getting total throughput

Chapter 5

Representation

For this project a custom map generation has been created to generate maps. A type of brush drawing technique is used, with random resource, feature and player starting point placement. An evolutionary search is made where each map is evaluated with fitness functions and a cascading elitism is used as an evolutionary strategy.

5.1 Genotype

To effectively represent a map, a descriptive representation is stored in a genotype. A genotype needs to be able to represent a finished version of a map in smaller amount of memory. In Civilization V, maps can range from $24 \cdot 24$ or 576 tiles, up to $256 \cdot 128$ or 32,768 tiles. Each tile then contains various information vital to the game. With 28 resources and 9 different terrain, the search space for a map has grown quite considerable. In this project, the map size is usually $60 \cdot 48$, which is 2,880 tiles. This size has been chosen since in the game Civilization V, that map size has been determined as the size for 4 players.

In this project a genotype is defined of four vectors that contain the following.

- Vector of tile block placements
- Vector of resource placements
- Vector of feature placements
- Vector of player placements

The tile blocks consists of 5 integers, resource placement and feature placement 3 integers each and player placement of two integers. This sums up to 13 integers or 52 bytes. The size of the genome in bytes however is $G_n \cdot 52$, so for $G_n = 40$ the size will be approximately 2 kilobytes. This fits quite well to the Civilization V game, but missing still are river placement, barbarians, player's civilisation and starting era to mention a few. The genotype used in this project also includes a size variable (G_n), determining how many entries of each the genotype should have. The smaller the size the less descriptive a genotype will become. The larger the genotype, more values need to be changed in order for the search to progress. An ideal ratio between the size of the map and the size of the genotype is hard to determine. The smaller the size of the genotype the smaller the search space will be. This increases the likelihood of finding the best solution but also limits the expressibility of the method. The larger the size of the genotype the larger the search space will be. This will make it harder to find a good solution and requires more resources to do so. However in this project a larger genotype will be able to draw more on each map since more tile blocks will be used.

5.1.1 Tile blocks

A tile block is represented by the following quintuple.

$$TB = (p, x, y, w, h)$$

Where p is the pattern, x and y the two dimensional coordinate. The w and h represent the dimensions of the pattern. At initiation of the genotype, all values are selected at random from a uniform distribution. The size of the tile block vector is equal to the size of the genotype (G_n). This extremely flexible set up of patterns, also known as brushes or stamps, can be randomly generated or predefined patterns. The patterns used in this project are demonstrated in figures 5.1 through 5.10.

It is important to note that these can be of any width and height. So essentially, a map with nothing but mountains can be expressed with a genotype of size one, with the only tile block consist of the mountain terrain, and its height and width is equal to the height and width of the map in question. An unplayable map to be sure, but possible to create.

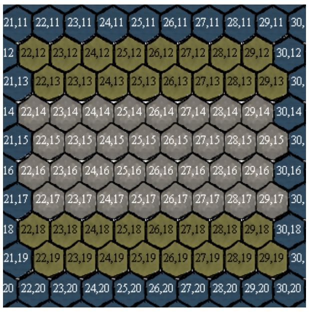

Figure 5.1: Pattern one. A top and bottom row of that span quarter each of the height of the Plains terrain. The middle half is Tundra terrain.

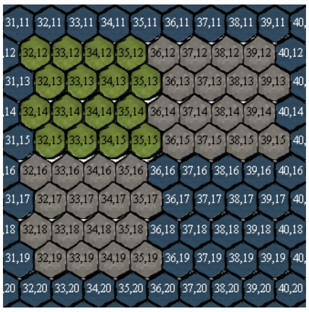

Figure 5.2: Pattern two. Top left quadrant consists of Grassland terrain while top right and bottom loft consist of Hills. The bottom right is transparent and does not modify the terrain that was there before, here it was and will stay as Ocean.

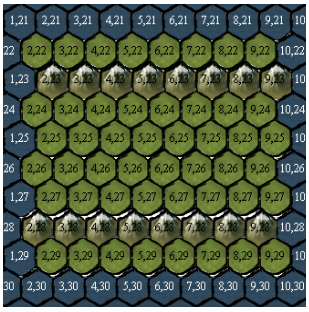

Figure 5.3: Pattern three. A Grassland pattern with two lines of Mountain that are one tile away from the bottom and top height.

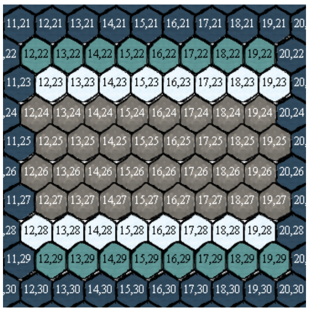

Figure 5.4: Pattern four. A pattern filled with Tundra terrain that gets padded with Snow and then Coastal lines on both the top and bottom parts.

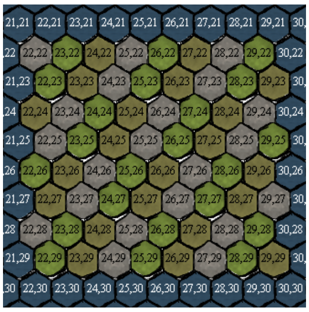

Figure 5.5: Pattern five. A mixture of Grassland terrain (green), Plains terrain (brown) and Tundra terrain (grey).

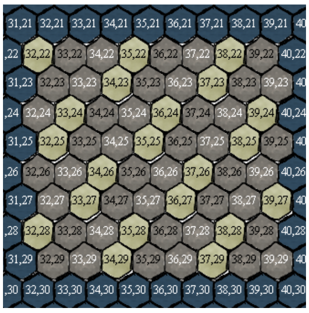

Figure 5.6: Pattern six. A mixture of Desert terrain (yellow), Hill terrain (grey, black font) and Tundra terrain (grey, white font).

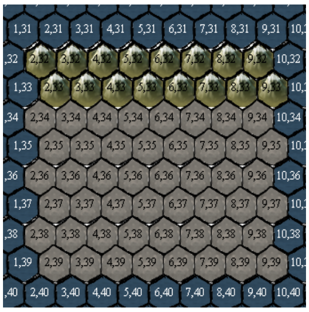

Figure 5.7: Pattern seven. A Mountain ridge on top followed by Hills.

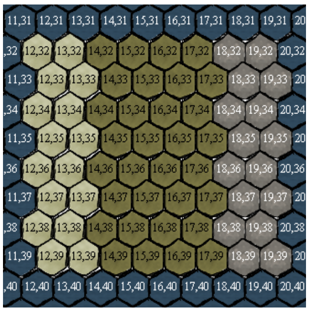

Figure 5.8: Pattern eight. Desert strip on the left and Tundra strip on the right filled up in the middle by Plains terrain.

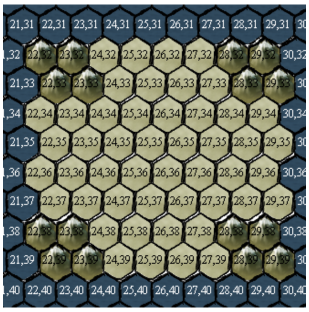

Figure 5.9: Pattern nine. Each corner filled up with Mountain terrain while the rest is filled up by Desert terrain.

5.1.2 Resources

A resource is represented by the following triple.

$$R = (r, x, y)$$

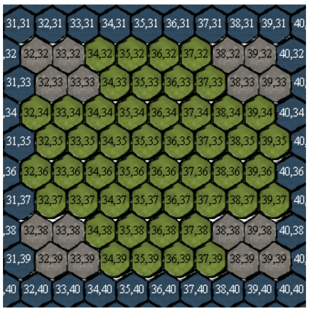

Figure 5.10: Pattern ten. Each corner filled up with Hills terrain while the rest is filled up by Grassland terrain.

Where r is the resource, x and y the two dimensional coordinate. At initiation each resource is chosen at random from a uniform distribution, the same as with the x and y coordinates. No safety checks are performed to ensure that the genotype is not broken. That is whether or not the resource is located on a terrain that is allowed by the game's rules. Of course x and y coordinates are randomly chosen within the confines of the map's dimensions. The resource vector is 8 times as large as the size of the genotype $(8 \cdot G_n)$.

5.1.3 Features

A feature is represented by the following triple.

$$F = (f, x, y,)$$

Where f is the feature, x and y the two dimensional coordinate. At initiation all values are randomly chosen. A constriction is enforced here. Some features are declared singletons, i.e. only one can exist in each instance of a map. If a feature is selected by random, that is a singleton, it is stored for later reference so that feature is not selected again. The size of this vector is equal to 4 times the size of the genotype $(4 \cdot G_n)$.

5.1.4 Player starting coordinates

A player starting position is represented by the following tuple.

$$P = (x, y)$$

Where x and y the two dimensional coordinate. The vector including player starting positions contains a two values, the x and y coordinates. During initiation these values get selected at random from a uniform distribution and enforced to be within the confines of the map's dimensions. The size of this vector is equal to the number of players have been chosen for this map.

5.2 Genotype-to-Phenotype phase

An important factor in the evolution is the genotype-to-phenotype phase. Where a fully represented entity of what the genotype is trying to capture, is created. In this project, that is a Civilization V map. This map however is a partial representation of the map in Civilization V as described in section 2.3. There was not enough time to represent a complete version of the Civilization V map in this project. Just the fact that a full length game can take up to 250 turns, and at each turn the effects of terrain and resources can be modified, is evidence that it would be out of scope for this project.

5.2.1 Pattern drawing

The pattern drawing technique is the major part how the map will end up looking. While generating the map, every tile block is looked at. On a blank canvas consisting only of ocean terrain, the tile blocks are drawn. When a tile block is drawn it will overwrite all terrain that was there before. If the current tile block reaches over tile blocks that have previously been drawn, they current tile block takes precedence. This allows for a wide array of different possibilities and expression. A pattern that is a square of grassland, might receive another square of Ocean right inside it, creating an O shaped island. Another square of coasts might then completely overwrite that island or amend it to create a C shaped island. This is quite open and various forms can be made. However if the last tile block in the tile block vector is a mountain, ocean or a coast pattern, with starting coordinates (0,0) and width and height equal to the maps. Then a very bad map is created, in fact, unplayable. The fact that this is a possibility is however good for expressibility.

5.2.2 Element distribution

While generating the map elements are distributed over the map according to their coordinates in the genotype. The genotype however can conflict with itself, for example in such a way where on one tile there is supposed to be an Ocean, but also some Elephants. When these conflicts are found, a breadth first search, spiralling outwards from the desired tile is conducted. Then a coordinate of the closest suitable terrain where Elephants can reside, according to the games rules, is returned. A rule is enforced on the spiralling search, stating that a tile can not have more than one neighbouring tile, that also has a resource upon it. This way, resources get spread over the map more equally. The broken genotype is then fixed by changing the coordinates in the genotype, as well as in the map, where the Elephants would reside. A similar thing is done with features, so Forests do not end up in a Desert or the Ocean. Fixing the broken genotype is done to ensure that this will not be needed to be done every single time a map is generated. When placing resources, it is ensured that a resource is not placed next to another resource, thus creating a cluster of resources.

5.3 Evolutionary Algorithm

This evolutionary algorithm consists of two main parts. The mutation of the genotype or population, and survival or selection. In this project a cascading elitism algorithm is used to perform the selection of the best individuals per generation. This cascading elitism is inspired by the work done by Julian et. al. [13]

5.3.1 Genotype mutation

A genotype mutation is needed to alter the phenotype. The design of the genotype ensures that small changes can result in big changes. For instance, changing a pattern of a tile block that only contains ocean terrain, to some pattern that contains three different terrains is a big change, but accomplished by only changing one value.

Every value in the genotype is modified based on two different random number generators. Both random number generators use a normal distribution to draw numbers from. The difference between them being the standard deviation. While a width and height change of 3 might be considerable, a x,y position change will not. Therefore two different standard deviations are used to address different changes in the genotype. Since all values are integers a rounding off is performed. This will result in some values will be changed considerably, while others not at all. Being a normal distribution most values will not change that much. For modifying identification numbers of tile blocks, resources and features, as well as the width and height of tile blocks, a standard deviation of 4 was selected with a mean of 0.

For all coordinates, standard deviation was set to a tenth of the width and height for x and y values respectively and a mean was chosen to be 0.

5.3.2 Cascading elitism

Cascading elitism algorithm is described in [13]. It suits well when optimising for several fitness functions. Fitness functions are sorted by importance, such as strategic resource distribution could be more important than unique resource distribution. In that order, maps get discarded with the worst fitness value for that fitness function. It is possible to discard different number of maps for each fitness function. For example, discarding the worst 50% of the most important fitness function, but only 20% of the second most important. This puts weights on each fitness function. It will become quite difficult to include the maps that score high in the least important fitness function, since those maps will be discarded if they have a perform badly on the more important fitness functions.
This is well suited to Civilization V and the fitness functions described earlier since they are clearly not equally important. In table 5.1 fitness functions are ordered by their importance as the experiments were performed.

	Fitness function	Comments
1	f_4	Player throughput is one of the more important ones, since the resources that belong to you and in your starting vicinity greatly affects on how quickly your empire gets built.
2	f_3	Strategic resources are quite important as has been discussed. Starting with no possibility of mounted units against players who have them by the handful is not fair.
3	f_0	Continental connectivity is important to allow players to reach each other through land instead of the need to spend resources to manufacture naval units.
4	f_1	Distance to players is somewhat important to allow players to have some time to build up there empire. It is however equally unfair to all players.
5	f_2	Unique resources give a map extra intrigue and players receive happiness bonuses for each extra unique resource, therefore not extremely important.

Table 5.1: Order of importance of fitness functions

Chapter 6

Results

In this project three main experiments were conducted to establish the validity of the work that has been done. The fitness functions are used to evaluate maps that are created by the Civilization V map generators. The effectiveness of the genome mutation is also tested and presented as well as the evolution of maps and some solution candidates are presented after evolution. All fitness evaluations rank on a scale from 0 to 1.

6.1 Civilization V maps

For evaluating the Civilization V map generation methods, 10 different maps were created each with random settings. Each map was graded 50 times, with random start locations for 4 different players. The average grade for each fitness function was then graphed on figures 6.1, 6.2, 6.3 and 6.4. The average grade for all ten maps is then presented in table 6.1. This is done to compare the averages of the different fitness functions. The average for each fitness function is calculated to counter the effects of random start locations. The average of 50 different starting locations for each map gives a general overview of how well a map is performing.

Four methods were tested in these experiments. The Fractal and Skirmish methods described in section 2.2 are used as well as the Archipelago method and the West

Method	f_0	f_1	f_2	f_3	f_4	f_e
Archipelago	0.0082	0.5601	0.5393	0.6300	0.6722	5692
Fractal	0.0498	0.5323	0.5018	0.5787	0.6269	5744
Skirmish	0.9705	0.5597	0.3918	NA	0.6970	4395
West vs East	0.8282	0.5710	0.6606	0.6896	0.6998	5125

Table 6.1: Average evaluations over 10 maps for each method.

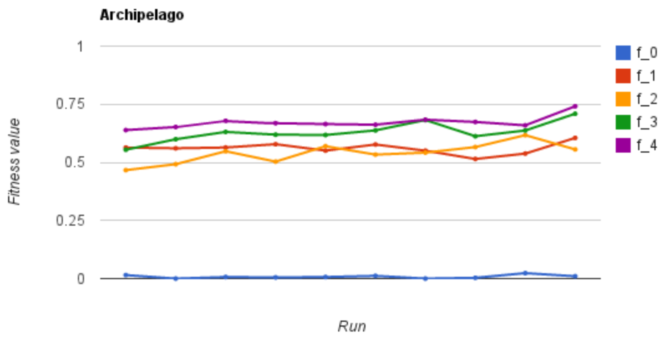

Figure 6.1: Evaluations of the Archipelago method

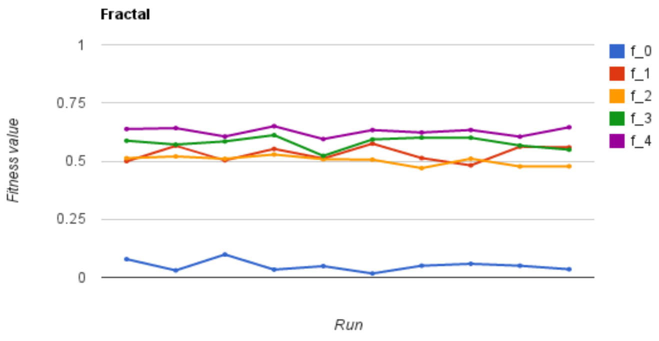

Figure 6.2: Evaluations of the Fractal method

Vs. East method. Archipelago method creates a collection of islands imitating a world for seafaring nations. The landmass is quite minimal and the amount of ocean and coasts is quite high. It is used as a candidate for a map that is not filled with land. West vs East method is a map that is split in to two sides, namely west and east. A thin line of sea separates these parts. This map method is aimed for

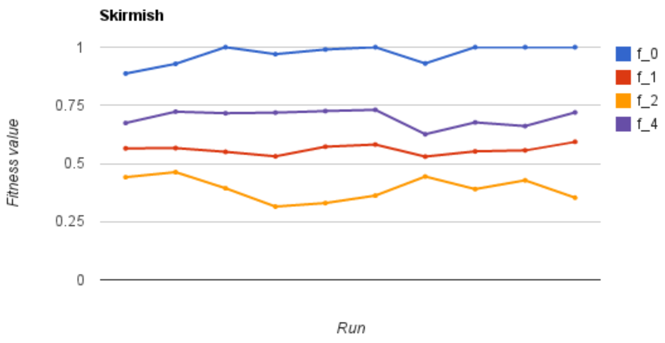

Figure 6.3: Evaluation of the Skirmish method

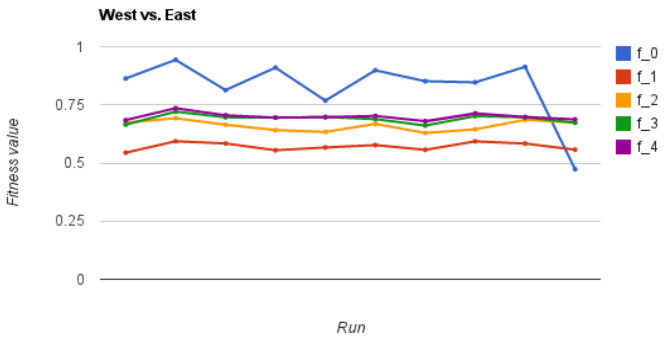

Figure 6.4: Evaluation of the West vs. East method

competitive team play where each team is located on either part. That is the reason for this method being chosen for testing.

The methods tested rank nicely. The Archipelago method does not perform well in continental connectivity as expected since it is filled with islands. On average it did not manage to get any fitness evaluations higher than 0.67. The amount of sea in this method also makes it harder to have it rank highly since resource placement happens on the small land that exists. Fractal method did not perform better and as can be seen in figure 6.2 it fluctuates somewhat between maps. The skirmish method does not seem to perform well overall and exceeds in continental connectivity and player throughput. Skirmish map does not receive a grade for fitness function f_3 since strategic resources are placed close to player starts after the game begins. West vs. East method, specifically designed for multi-player games, does considerably well in all fitness functions except distance from players to other players. The player placement is not a product of the method from the game.

6.2 Genotype mutation

To advance an individual, its genotype is mutated. This process should generate different genotypes that result in improved or deteriorated phenotypes. Since the mutation is random, not all mutations will improve nor deteriorate. Given an genotype that results in a map that gets high grades, the majority of the mutated individuals should deteriorate. Same as with bad individuals, the majority should improve. To make sure that this is the case for the mutation performed in this project, four individuals were chosen, two that performed badly and two that performed well. Each of those individuals then had 100 mutated offsprings generated and were measured against its parent. Results can be seen in table 6.2 showing how many were improved based on the parent and how many deteriorated.

Individual	f_4	f_3	f_2	f_1	f_0	**Improved**	**Deteriorated**
Good 1	0.76	0.68	0.73	0.99	0.50	26	74
Good 2	0.91	0.76	0.78	0.83	1.00	21	79
Bad 1	0.49	0.44	0.62	0.49	1.00	91	9
Bad 2	0.61	0.53	0.68	0.93	1.00	78	22

Table 6.2: Improvement and deterioration of genotypes, fitness values are from the individual used for mutation.

These results allow us to assume that with evolution that by selecting the best individuals of every generation that a local maxima of these fitness function will be reached.

6.3 Evolution

Many evolution simulations runs with different parameters were made for this project. The parameters that were tweaked were, genome size population size and number of iterations. The population size allows us to compare more individuals per each iteration. The number of iterations performed affects for how long we mutate and search for new individuals. A description on the affect of genome size is described in 5.1. The parameters that performed the best will be presented here,and can be seen in table 6.3. An average of five different runs is also presented in table 6.4

and the standard deviation can be seen in 6.5. The graph of these averages and standard deviation can be seen in 6.5 and 6.6 Screenshots of the maps made can be seen in figures 6.7, 6.8, 6.9 and 6.10.

These maps did perform well in comparison to the Civilization V maps. They scored higher using the fitness evaluations in this project. The maps generated feature no aspects that break the game and result in a non-playable map. That is to say the randomisation of the map drawing does not create unreachable areas. These maps look like they could be played in the game although, due to the nature of the map drawing technique with rectangle brushes, they look very "blocky".

G_{sz}	**Pop**	**Iter**	f_0	f_1	f_2	f_3	f_4	f_e
40	30	50	1.00	0.83	0.74	0.94	0.91	3,300
40	30	40	1.00	0.96	0.70	0.95	0.79	4,557
40	30	40	1.00	0.92	0.71	0.84	0.89	4,049
40	10	30	1.00	0.64	0.77	0.95	0.90	3,549

Table 6.3: Four results from different runs. G_{sz} stands for genome size, pop for population and iter for iterations.

f_{0avg}	f_{1avg}	f_{2avg}	f_{3avg}	f_{4avg}	f_{eavg}
0.86	0.69	0.69	0.70	0.74	4,173
0.90	0.70	0.67	0.63	0.65	4,203
0.93	0.84	0.69	0.67	0.65	4,323
0.98	0.61	0.66	0.63	0.66	3,929
0.99	0.74	0.67	0.63	0.68	4,224

Table 6.4: Averages for each fitness value from five different runs.

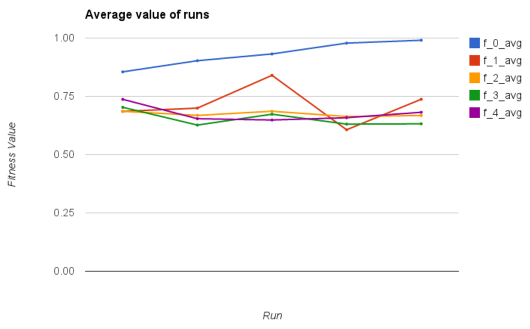

Figure 6.5: Graph of the average of the population from five evolution runs

f_{0sd}	f_{1sd}	f_{2sd}	f_{3sd}	f_{4sd}	f_{esd}
0.25	0.12	0.04	0.10	0.09	383.22
0.22	0.15	0.05	0.12	0.13	342.61
0.17	0.13	0.03	0.11	0.08	287.01
0.10	0.11	0.03	0.07	0.09	368.80
0.07	0.16	0.06	0.13	0.13	422.30

Table 6.5: Standard deviation of a population from five different runs.

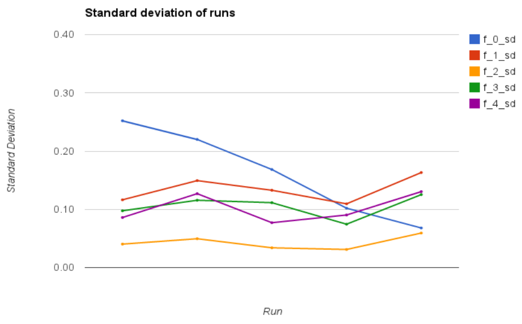

Figure 6.6: Graph of the standard deviation of the population from five evolution runs

When faced with maximising values over five fitness functions these results are encouraging. It is evident that they are not conflicting each other, but rather enforcing each other. For example, having equal amounts of strategic resources and unique luxury resources will in part, also supplement having equal amounts of throughput per player. Being designed in that way, to limit conflicts also allows for higher grades across every fitness function.

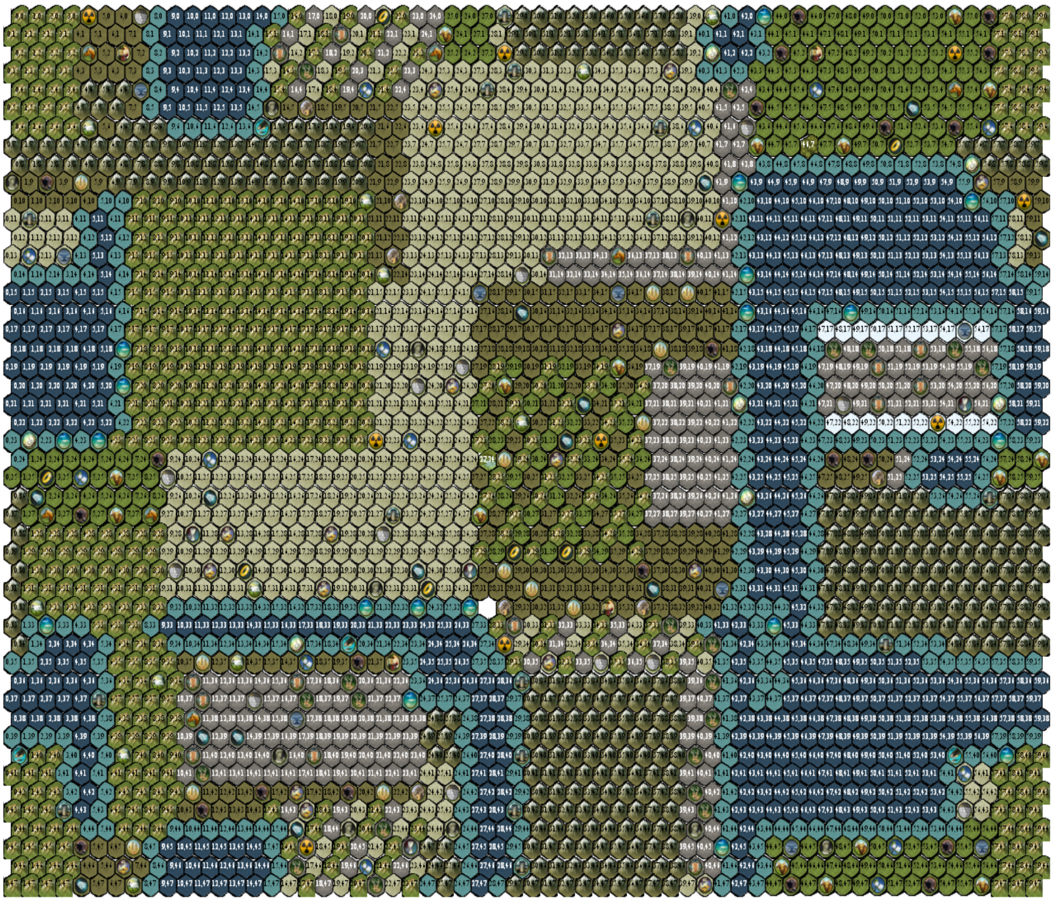

Figure 6.7: Evolution with genome size 40, out of 30 individuals after 50 iterations, with total map throughput of 3300

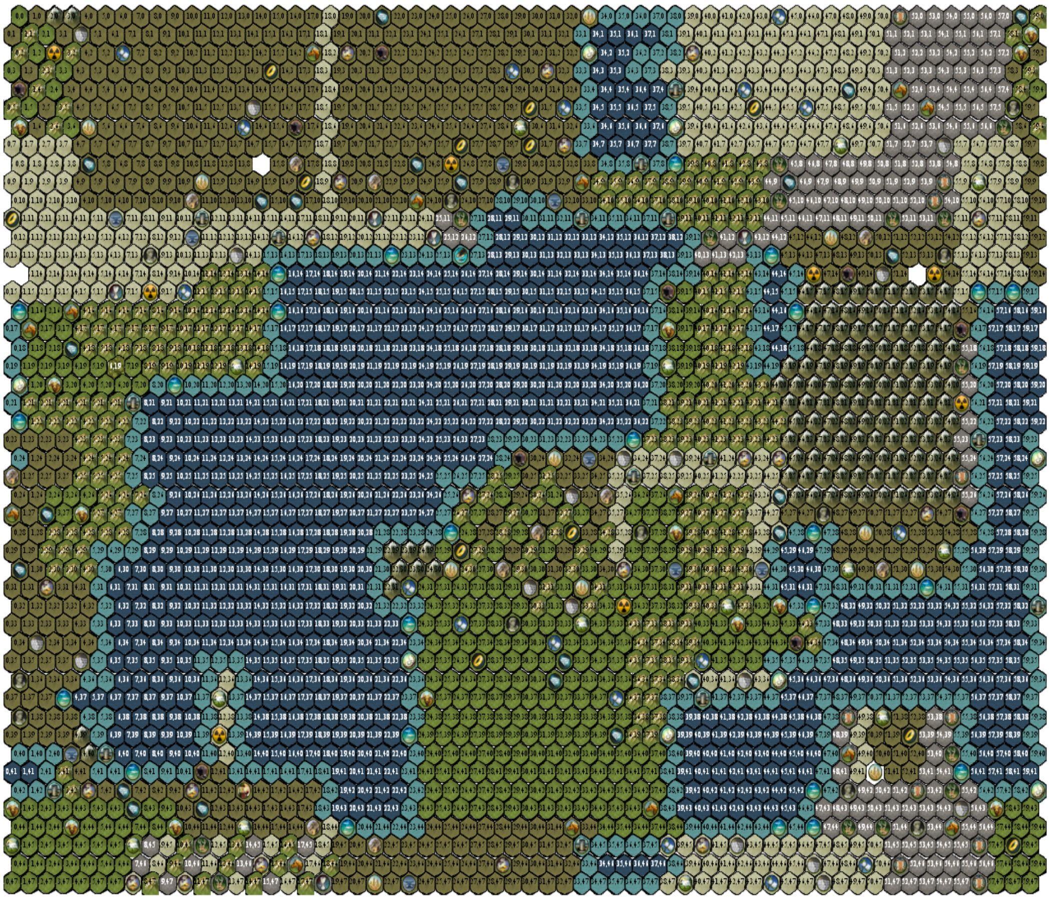

Figure 6.8: Evolution with genome size 40, out of 30 individuals after 40 iterations, with total map throughput of 4557

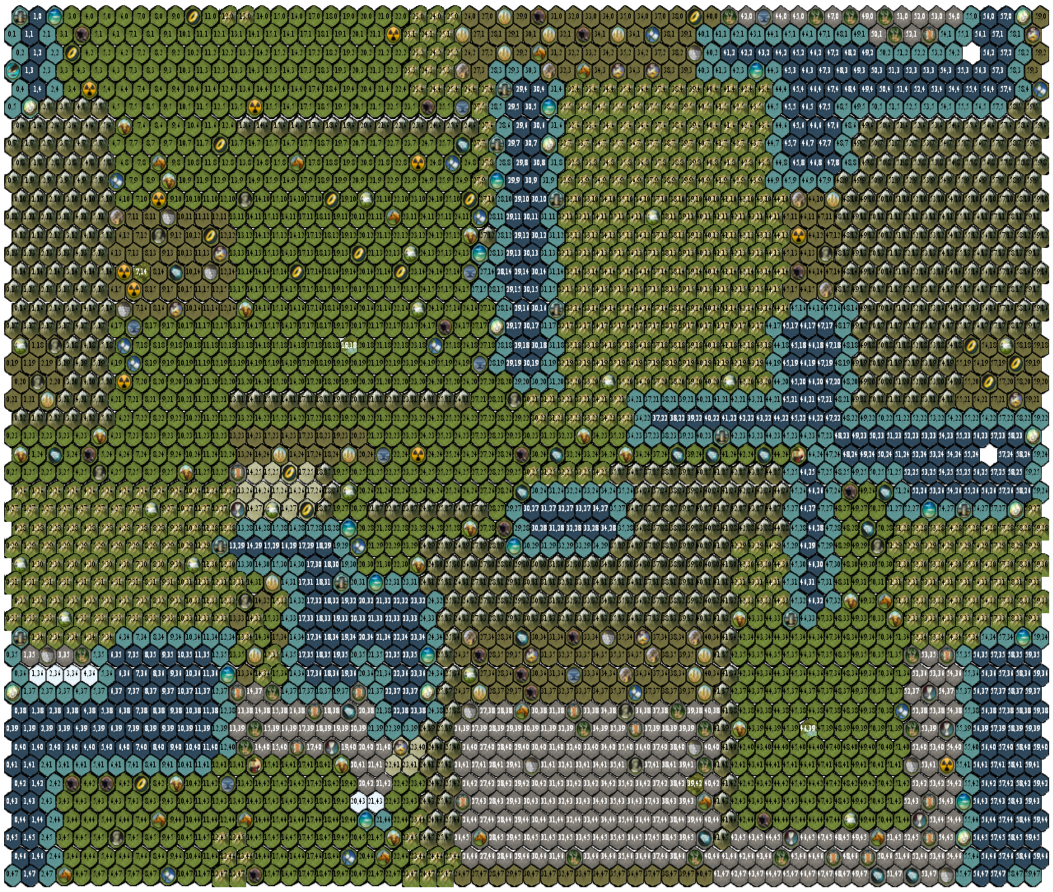

Figure 6.9: Evolution with genome size 40, out of 30 individuals after 40 iterations, with total map throughput of 4049

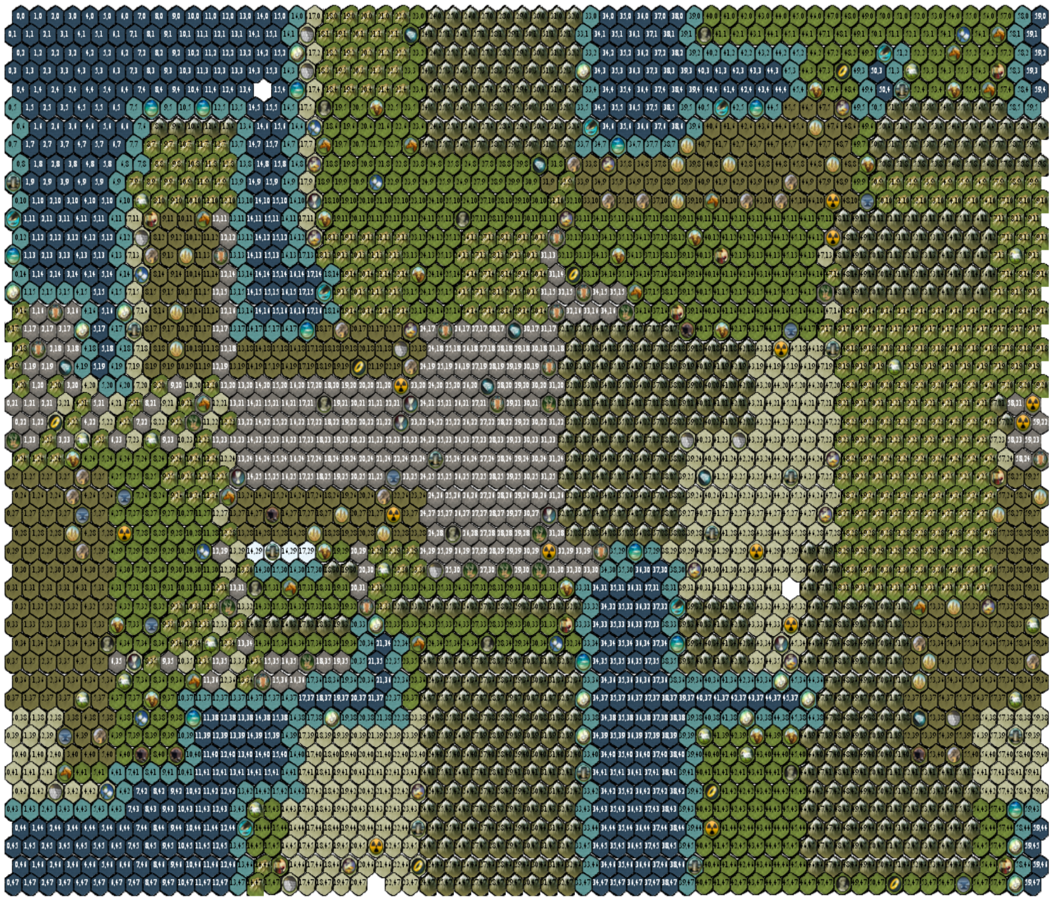

Figure 6.10: Evolution with genome size 40, out of 10 individuals after 30 iterations, with total map throughput of 3549

Chapter 7

Discussion

This project has established a solid foundation with further potential for map generation. There are shortcomings in the application but there is also great value. The results that have been presented receive a high value from the fitness functions. This tells us that the maps are fair, but only according to the definition made by the fitness functions. Assessing whether or not those maps are truly fair, is subjective to definitions of fair. The definition of what is fair, can vary tremendously. That makes it hard to find a consensus on what is fair. Based on informal discussions through online forums [1] [2] with players of the game, there is no consensus on what people think is fair or what people want in their maps. Some players voice their complaints about how resource distribution is game changing, other players think that it is a vital part of the Civilization V game and is a fun challenge rather than an upsetting imbalance. Finding a balance on what players think is fair, can be hard. The fitness functions described in this project are complete in defining what is fair. It proved hard to define them, and even harder to include more. Maximising five objectives is a challenge, and including more would pose an even greater challenge.

The shortcomings of this project are the map drawing technique and verification technique. The map drawing, while easily capturing the genotype, outputs maps that are not aesthetically pleasing. The majority of the generated maps are very

[1] `http://www.civplayers.com/index.php?section=smf&topic=10097.0`
[2] `http://forums.civfanatics.com/showthread.php?t=473236`

square, where the only detectable patterns are rectangular in shape. While the hope was that with enough overlapping rectangles, interesting patterns would emerge but this was not the case. This did not however affect the results from performing badly with the fitness functions created. This would suggest that independent of map drawing techniques, it would be possible to achieve good results from the fitness functions. Creating maps that are generally aesthetically pleasing and fair would be a possibility. Confirmation of the fairness of the results in this project is lacking. Upon close inspection one can verify whether or not the maps are fair. Subjective as that is, there is no opinion on how far resources and how much output there is shared between players. Creating a survey, asking players to rate the fairness of maps would be interesting, but not a confirmation on the issue. Opinions vary greatly among the few people that have enough knowledge about Civilization V. The greatest possibility of verification of fairness for this project is a play test, simulated or user based. The feedback from those tests would be able to verify whether or not fairness has been achieved. It would give weight to the definition of fair composed by the fitness functions described in this project. Unfortunately, this was outside the scope of this project.

This project can be further improved and used as a tool for developers as well as modders[3] aiding with map creation. A scenario would be where a user would choose fitness functions and prioritise them according to his preferences. Set the number of maps generated and iterations to be performed. With the resulting maps he would choose the one he likes the most and further improve on that map using map modification tools. This would provide the user with a map that is based on a fair map and would be improved upon by the user, aesthetically or in any other way. Another scenario is implementing this into a map modification tool. This has been done before [9] and is also a promising research field. The modification tool comes up with a map which the user modifies. After the user's modification a limited evolution around the modification is performed and suggested to the user. Reinforced evolution like this has the user dictating in what direction the map should go with suggestions and evaluations from the algorithm.

A way of evaluating the map in another way would be by using **simulation-based**[11] evaluation functions. For the Civilization V game this would include using the artificial intelligence of the non-playable characters (NPC) to play several games. For this project, it was simply out of scope but it would be possible to do. Some NPCs can play the game in different ways. Configuring a play test using NPCs, that are configured to behave the same way and have the same advantages. For Civilization V, this means that they use the same mentality configuration as well as play as the same civilisation. A satisfying result would be that every NPC has an equal share of wins over every other NPC. This method would take considerable time if graphic rendition of the game being played can not be skipped. Equal shares

[3]Modders is a term over people who modify a game

of wins for every NPC would mean that the map could be considered to be fair.

Chapter 8

Conclusions

In this project the purpose was to create maps for the video game Civilization V that were fair and balanced. Creating a definition of what a fair map is through fitness functions. Representing the map with genotypes, and phenotypes for evaluation. Creating an evolutionary algorithm that searched for candidates that were optimised over multiple objectives. All this has been done and results have been presented. There is not conclusive evidence to support the claim that the maps produced are fair but rather a demonstration of maps that fulfil the definition of fair that the fitness functions provide. For everyone that agrees with these definition these maps are fair. This work has also done the groundwork necessary to further this research on fair map generation, especially for the game series Civilization V as well as others.

8.1 Future work

There are several aspects that would improve this project and could further research in this field.

- Improving representation of the Civilization V game
- Adding output of generated maps to civilisation map files

- Adding simulation based evaluation functions
- Adding other map drawing methods

A first step to improve the accuracy of this project is to represent the Civilization V game better. That is to say to include things that have been excluded in this project such as rivers. This step would allow for a map to be generated as an equivalent of a Civilization V map. Generated maps can improve further on their compliance with Civilization V's map features, such as supporting starting era and technologies.

Creating a map file that is readable by the game is trivial and a requisite for any evaluation functions that are based on the game being played. This would open up the possibilities to get user feedback as users could play generated maps from this project. It would also greatly validate or invalidate the claims made in this project.

As mentioned above, **simulation-based** evaluation functions could greatly reinforce the claim of fair maps. This could also reinforce or discredit the fitness functions. Automated playthroughs, using non-playable characters, would output a vast amount of data that could improve this project. Time to reach a technology, an era as well as time to first civilisation extermination, could be measure the quality of a generated map.

As described in chapter 3, there are several different map drawing techniques available. Having a map drawn that is not as "blocky" would result in maps that are more realistic. Maps that contain blocks of features, such as those generated in this project, are predictable in a way and thus limiting intrigue. More patterns, turning rectangles into hexagonal and circle shapes, would also generate maps that look better, but would not guarantee more fair maps.

It is the author's true belief that this project has laid solid groundwork for future improvements that could be beneficial to many parties in the game research field, game developers, and others alike.

Appendix A

Terrain

Name	**Yield**			**Resources found on**
	Food	Gold	Production	
Coast	1	1	0	Oil, Fish, Whales, Pearls
Desert	0	0	0	Iron, Oil, Aluminium, Uranium Sheep, Stone, Gold, Silver, Gems Marble, Cotton, Incense
Grassland	2	0	0	Iron, Horses, Coal, Uranium, Cattle, Sheep, Stone, Gold, Gems Marble, Cotton, Wine
Hills	0	0	2	Sheep, Iron, Coal, Aluminium, Uranium, Sheep, Deer, Stone, Gold, Silver, Gems, Marble
Mountain	0	0	0	-
Ocean	1	1	0	-
Plains	1	0	1	Iron, Horses, Coal, Aluminium, Uranium, Wheat, Sheep, Stone, Gold, Gems, Marble, Ivory, Cotton, Wine, Incense
Snow	0	0	0	Iron, Oil, Uranium, Stone
Tundra	1	0	0	Iron, Horses, Oil, Aluminium Uranium, Deer, Stone, Silver, Gems, Marble, Furs

Table A.1: All terrains in Civilization V

Appendix B

Resources

Bonus resources	**Food**	**Gold**	**Production**
Bananas	1	0	0
Cattle	1	0	0
Deer	1	0	0
Fish	1	0	0
Sheep	1	0	0
Stone	0	0	1
Wheat	1	0	0

Table B.1: All bonus resources in Civilization V

Strategic resources	**Food**	**Gold**	**Production**	**Revealed by technology**
Aluminium	0	0	1	Electricity
Coal	0	0	1	Scientific Theory
Horses	0	0	1	Animal Husbandry
Iron	0	0	1	Iron Working
Oil	0	0	1	Biology
Uranium	0	0	1	Atomic Theory

Table B.2: All strategic resources in Civilization V

Luxury resources	Food	Gold	Production
Cotton	0	2	0
Dyes	0	2	0
Furs	0	2	0
Gems	0	3	0
Gold	0	2	0
Incense	0	2	0
Ivory	0	2	0
Marble	0	2	0
Pearls	0	2	0
Silk	0	2	0
Silver	0	2	0
Spices	0	2	0
Sugar	0	2	0
Whales	1	1	0
Wine	0	2	0

Table B.3: All luxury resources in Civilization V

Appendix C

Features

Extra information.
El Dorado provides 500 gold one time and 5 culture.
Fountain of Youth provides 10 happiness.
Krakatoa gives 5 science.
Mt. Fuji provides 5 culture.
Old Faithful provides 2 science and 3 happiness.
The Barringer Crater provides 3 science.
The Great Barrier reef provides 2 science.

Name	**Yield**			**Impassable**	**Resources found on**
	Food	Gold	Production		
Atoll	1	0	1	No	
Cerro de Potosi	0	0	10	Yes	
El Dorado	0	0	0	Yes	
Fallout	-3	-3	-3	No	
Flood Plains	2	0	0	No	Wheat, Sugar
Forest	1	0	1	No	Uranium, Deer, Furs, Dyes, Silk
Fountain of Youth	0	0	0	Yes	
Ice	0	0	0	Yes	
Jungle	1	0	-1	No	Oil, Uranium, Bananas, Gems, Dyes, Spices
Krakatoa	0	0	0	Yes	
Lakes	2	1	0	Yes	
Marsh	-1	0	0	No	Oil, Uranium, Sugar
Mt. Fuji	0	1	0	Yes	
Oasis	3	1	0	No	
Old Faithful	0	0	0	Yes	
Rivers	0	1	0	Yes	
Rock of Gibraltar	2	5	0	Yes	
Barringer Crater	0	2	0	Yes	
Grand Mesa	0	3	2	Yes	
Great Barrier Reef	2	1	1	Yes	

Table C.1: All features in Civilization V

Appendix D

Map generation methods

Civ5 Map generation methods.

Name	Parameters	Description
Archipelago	World Age, Temperature, Rainfall, Sea Level, Resources	A collection of island this is built for seafaring nations. With landmass quite minimal resources are often scattered together in patches.
Continents	World Age, Temperature, Rainfall, Sea Level, Resources	A true imitation of the Earth with no similarities. Few and big continents with adequate resource distribution.
Four Corners	World Age Temperature, Rainfall, Resources, Buffer Zones, Team Setting	Four corners are a big landmass that are separated by sea and coast line that form a cross, essentially creating four corners of a map. The four corners are connected with a patch of land in the middle making it a continuous region of land.
Fractal	None	A fractal based method that results in erratic coastline and several continents.
Great Plains	None	A map designed to imitate a region of North America, Great Plains consists mostly of desert, plains and grassland with a small gulf of sea in a bottom right corner.
Highlands	Temperature, Rainfall, Resources, Mountain Pattern, Mountain Density, Bodies of Water	A mountainous map with lakes and various terrain.
Ice Age	World Age, Temperature, Rainfall, Sea Level, Resources, Landmass Types	A method that is rich with ice snow and ocean. Many continents and sparse resources.
Inland Sea	World Age, Temperature, Rainfall, Resources	Like the name suggests a large sea is centred at the middle of the map.

Table D.1: All Civilization V map methods, part one.

Name	Parameters	Description
Lakes	World Age, Temperature, Rainfall, Resources, Bodies of Water	A continuous continent with several lakes and big lakes inside.
North Vs. South	World Age, Temperature, Rainfall, Resources, Team Setting	A map that is split in two parts, namely north and south with a arid desert line in the middle to separate the map. Aimed at competitive team play.
Oval	World Age, Temperature, Rainfall, Sea Level, Resources	An oval shaped land mass with ingrown gulfs.
Pangaea	World Age, Temperature, Rainfall, Sea Level, Resources	A one continent map with small surrounding islands.
Ring	Dominant Terrain, Land Shape, Isthmus Width, Resources	A map that creates a big ring of landmass.
Skirmish	Dominant Terrain, Water Setting, Resources	A map that is aimed at all kinds of competitive player vs player games, a continuous landmass with variable water and seas.
Small Continents	World Age, Temperature, Rainfall, Sea Level, Resources	Many small continents with tight resources.
Terra	World Age, Temperature, Rainfall, Resources	Terra is a method that imitates the look of the earth without looking exactly alike. A fractal method that is instructed to create earths 7 continents.
Tiny Islands	World Age, Temperature, Rainfall, Sea Level	A island extensive map with no considerable landmass. An imitation of a Caribbean sea. Resulting in a map with sparse resources.
West Vs. East	World Age, Temperature, Rainfall, Resources, Team Setting	A map that is split in two parts, namely west and east. A thin lien of sea and coastline in the middle is used two separate the map in two parts. Aimed at competitive team play.

Table D.2: All Civilization V map methods, part two

Name	Values
Bodies of Water	Random, Small Lakes, Large Lakes, Seas
Buffer Zones	Oceans, Mountains, Random
Dominant Terrain	Grasslands, Plains, Forest, Jungle, Desert, Tundra, Hills, Global Climate, Random
Isthmus Width	2, 3, 4 Plots Wide, Random
Land shape	Natural, Pressed, Solid, Random
Landmass Types	Wide Continents, Narrow Continents, Islands, Small Islands, Random
Mountain Density	Random, Thin, Normal, Dense
Mountain Pattern	Ridge-lines, Scattered, Ranges, Clusters
Rainfall	Arid, Normal, Wet, Random
Sea Level	Low, Medium, High, Random
Resources	Sparse, Standard, Abundant, Legendary Start, Strategic Balance, Random
Team Setting	Start Together, Start Anywhere
Temperature	Cool, Temperate, Hot, Random
Water Setting	Rivers, Small Lakes, Seas, Rivers and Seas, Dry, Random
World Age	3, 4, 5 Billion Years, Random

Table D.3: Parameters for Civilization V map methods

Bibliography

[1] Amit P. amitp@cs.stanford.edu. Polygonal map generation for games. http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/, 2012. URL visited on 2012-09-10.

[2] Blizzard. Starcraft. http://en.wikipedia.org/wiki/StarCraft, 2012. URL visited on 2012-09-16.

[3] Cameron Browne and Frederic Maire. Evolutionary game design. *IEEE Transactions on Computional Intelligence and AI in Games*, 2(1):1–16.

[4] 2K Games. 2k conquers pcs with the release of sid meier's civilization v on september 21, 2010 in north america. http://www.2kgames.com/#/news/2k-games-conquers-pcs-with-the-release-of-sid-meier-s-civilization-reg-v-on- 2012. URL visited on 2012-09-02.

[5] 2K Games. 2k games - blog - civilization v expansion pack announced. http://www.2kgames.com/blog/civilization-v-expansion-pack-announced, 2012. URL visited on 2012-08-15.

[6] Firaxis Games. fractals.lua. Lua script provided with the Civilization 5 SDK of the Fractals map generation method.

[7] Firaxis Games. skirmish.lua. Lua script provided with the Civilization 5 SDK of the Skirmish map generation method.

[8] Firaxis Games. Civilopedia 5. http://civilopedia5.com, 2012. URL visited on 2012-09-16.

[9] Michael Mateas Gillian Smith, Jim Whitehead. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *IEEE Transactions on Computional Intelligence and AI in Games*, 3(3):201–215, 2011.

[10] Ian Parberry Jonathon Doran. Controlled procedural terrain generation using software agents. *IEEE Transactions on Computional Intelligence and AI in Games*, 2(2):111 – 119, 2010.

[11] Kenneth O. Stanley Cameron Browne Julian Togelius, Georgios N. Yannakakis. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computional Intelligence and AI in Games*, 3(3):172–186, 2011.

[12] Mike Preuss Nicola Beume Simon Wessing Johan Hagelback Corrado Grappiolo Julian Togelius, Georgios N. Yannakakis. Controllable procedural map generation via multiobjective evolution. 2012.

[13] Renzo De Nardi Julian Togelius and Simon M. Lucas. Towards automatic personalised content creation for racing games. In *IEEE Symposium on Computational Intelligence and Games, 2007*, pages 252 – 259, 2007.

[14] Paul Martz. Generating random fractal terrain. `http://www.gameprogrammer.com/fractal.html`, 2012. URL visited on 2012-09-07.

[15] Metacritic. Sid meier's civilization v for pc reviews. `http://www.metacritic.com/game/pc/sid-meiers-civilization-v`, 2012. URL visited on 2012-09-16.

[16] Carlos Cotta Miguel Frade, F. Fernandez de Vega. Breeding terrains with genetic terrain programming: The evolution of terrain generators. *International Journal of Computer Games Technology*, 2009(125714):13, 2009.

[17] Gavin S P Miller. The definition and rendering of terrain maps. *SIGGRAPH Comput. Graph.*, 20(4):39–48, August 1986.

[18] MouseyPounds. Civ5map file format. `http://forums.civfanatics.com/showpost.php?p=10386463&postcount=1`, 2012. URL visited on 2012-09-02.

[19] Nintendo. Super mario bros. `http://en.wikipedia.org/wiki/Super_Mario_Bros.`, 2012. URL visited on 2012-09-16.

[20] Elvis Alistar Robert Pieter van Leeuwen Phillipa Avery, Julian Togelius. Computational intelligence and tower defense games. *IEEE Congress on Evolutionary Computation*, pages 1084–1091, 2011.

[21] K.J. de Kraker R. Bidarra. R.M. Smelik, T. Tutenel. A declarative approach to procedural modeling of virtual worlds. *Computers & Graphics*, 35:352–363, 2011.

[22] Nathan Sorenson and Philippe Pasquier. Towards a generic framework for automated video game level creation. In *Applications of Evolutionary Computation*, volume 6024 of *Lecture Notes in Computer Science*, pages 131–140. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-12239-2-14.

[23] Julian Togelius, Mike Preuss, and Georgios N. Yannakakis. Towards multiobjective procedural map generation. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, PCGames '10, pages 3:1–3:8, New York, NY, USA, 2010. ACM.

[24] WCG. World cyber games awards. `http://www.wcg.com/renew/history/wcg2011/wcg2011_awards.asp`, 2012. URL visited on 2012-11-05.