EVALUATING FITNESS OF PATCH-BASED TERRAINS IN THE USE OF VIDEO GAMES

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Lohit Bhurale

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

March 2017

Fargo, North Dakota

North Dakota State University
Graduate School

Title

EVALUATING FITNESS OF PATCH-BASED TERRAINS IN THE USE OF VIDEO GAMES

By

Lohit Bhurale

The Supervisory Committee certifies that this **disquisition** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig
Chair

Dr. Saeed Salem

Dr. Svetlana Kilina

Approved:

March 9, 2017
Date

Dr. Brian Slator
Department Chair

ABSTRACT

This paper deals with current an existing evolutionary algorithm used in procedural terrain generation content for video games which is gaining interest to lessen the development costs and meet desired user requirements in creating levels, maps, and 3D terrain. Our approach demonstrates the advantages of a two-level interactive parent selection mechanism with seamless patching done during the generation process, using genetic algorithm and evolutionary strategies. Genetic algorithm is applied with crossover and mutation to evolve the layout of the patches. On the other hand, evolutionary strategy is also evaluated. We have conducted a series of runs, resulting in visually comparing the terrains evolved through genetic and evolutionary strategy to find the best fitness for the generated terrains. Since the evolution is done through a two-level interactive process, the final selected terrains from the genetic and the evolutionary approach are evaluated by comparing the features that meet the user's expectations.

ACKNOWLEDGEMENTS

I would like to thank Dr. Simone Ludwig for her continuous support throughout my research work. Without her guidance it would not have been possible for me to come this far. I would also like to thank Dr. Saeed Salem and Dr. Svetlana Kilina for agreeing to serve on my graduate supervisory committee. Special thanks to Dr. Kenneth Magel and the Department of Computer Science for giving me the opportunity to pursue my higher studies at North Dakota State University.

I would also like to thanks my dad, family, and friends who has supported me throughout my graduate studies. Special thanks to Mary Bush Pfeifer for all her support thorough out my graduate studies.

DEDICATION

To my Family!!!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

1. INTRODUCTION

One of the most important aspects of graphical applications is the use of artificial terrain generation techniques (ATGT) representing a real or imaginary world. A large number of applications use these techniques, which are based on natural selection. Terrain generation techniques (TGT) have evolved as a broad-based field devoted to design, development, and analysis of finding solutions to complex search problems. They have been successfully applied in areas such as robotics [1], music generation [2], scheduling [3], aircraft design and simulation techniques [4], to name a few. These techniques are also being gradually adapted in fields such as landscape visualization, computer animation, and virtual video games. However, the artificial terrain generation technology is more pronounced in the field of video games since it finds a lot of relevance to create maps or levels for nature established virtual terrains**.**

The virtual world of games and movies requires graphic models of great detail. Natural and architectural objects are very complex and usually contain many motifs and elements, which are organized into one complete complex model. This whole process of designing the high quality graphic models simulating the real environments is a cumbersome task since it requires skilled manpower, time and multiple iterations to achieve the task. A number of techniques such as Computer Aided Design (CAD) systems were used to create three dimensional models through the use of algorithms and specially written program codes, [5] [23]. However, this process still requires human intervention since the software has to be operated manually and parameters defined. Individual units in the complex system can be combined to give the final product, but this is still deemed to be a difficult iterative process. Partial or semi- automated design processes such as generative or procedural designs have been developed to write simple pieces of codes, which help visualize the objects on the screen without the creation of underlying

shapes. Therefore, the manual effort is decreased. Procedural design techniques are discussed in detail in the future sections.

Most of the ongoing research in the virtual terrain field is focused on visualization of large terrains to achieve frame rates interactively. This process is composed of terrains evaluated by human evaluators in achieving the best desired level of terrain. A large number of detailed algorithms (LOD) have been developed. [5][6]. As a result, the rendering process speed is increased by using simple versions of geometry for objects that have lesser visualization importance. The advent of graphic acceleration hardware has further catapulted the virtual terrain field to an altogether different league by interactive presentations and detailed terrain models representing the outdoor phenomena. Games such as Halo2 [7] as shown in Figure 1, are replete with visually appealing outdoor terrains, which is proof that current technology can be put to full use and “real time believable visualization” is possible.

Figure 1: An Outdoor Scene from Halo 2. [7]

1.1. Requirements of Terrains in Video Games

The choice of terrain features depends on the genre of the game since it is the deciding factor for the kind of virtual terrain to be used. Application of procedural terrain generation techniques in the game genre of real time strategy games requires some fundamental aspects, which were elucidated by Olsen in one of his works [8]:

- For the characters in the game to traverse as much as possible the terrain must be flat enough.
- Visually appealing maps and strategic game play also requires terrain features such as hills, valleys, hillocks, etc.

Most of the game genres require these fundamental criteria. Nevertheless, it also depends on the specific genre since they might have their own scalability and features. The following are some of the example genres and their requirements [9]:

a. Role-playing games: Large environments and vast townships, which can be explored through length and breadth; e.g. Skyrim.
b. Flight simulators: Requires realistic terrain since it is viewed from an aerial perspective.
c. First person shooter games: Requires virtual objects and terrain features with a high relative density on small scale maps.

1.2. Artificial Terrain Generation Techniques (ATGT)

Evolutionary computation as the name suggests finds its basis in Darwin's theory of evolution and Mendel's concept of genetics. According to Darwinism, the chances of survival and reproduction of species is facilitated by the process of natural selection. This paves way for

small amount of variability in individuals that is then inherited in the subsequent generations. Modern evolutionary synthesis, which encompasses Darwinism and Mendelism is the backbone of many different types of evolutionary computations.

Procedural terrain generation (PTG) of late has been using evolutionary computation techniques through different approaches. Terrainosaurus, a terrain generation program was the brainchild of Ong et al. and was later implemented by Saunders. This algorithm is mainly used for generating a family of similar terrains [10]. Ashlock et al. applied evolutionary algorithms to generate fractal styled terrains [9]. Walsh and Gade used evolutionary algorithms on terrain generators to automatically adjust parameter values. Though this approach does not generate an altogether new terrain, it works to adjust the existing parameters [9]. Multi-objective Evolutionary algorithm were developed by Togelius et al. which was used in the creation of maps in real time strategy games. The well known game StarCraft successfully employed this approach [11]. In contrast to the above methods, Raffe et al. utilized patches of smaller terrain maps, which are essentially decomposed smaller parts of sample terrains to generate a bigger terrain. This is known as patch based mechanism, where "rendering" is done to smoothen the joined patches.

The present paper attempts to use the patch-based terrain model developed by Raffe et al. [12] and extend it with Evolutionary strategies. While the genetic algorithm uses the processes of selection, crossover, and mutation resulting in the layout of patches, evolutionary strategy uses selection and mutation without the use of crossover in generating the layout of patches.

2. RELATED WORK

Several researchers have addressed ATGT for a long time, as a result of which many techniques and algorithms have been developed. To set up a comparison with real time algorithms, Saunders [10] proposed the following list of features that terrain generation algorithms should possess:

a) Low requirement of human interaction.

b) High degree of human control.

c) Complete intuitiveness to control.

d) Ability to evolve terrains at entirely arbitrary level of details.

e) Swiftness for real-time and dynamic applications.

f) Complete ability to generate wide variety of terrain features.

g) Exhibit support towards extension of new types of terrain.

Another important aspect in terrain generation is the way a terrain is represented based on the data gathered.

Height map is one of the most widely used methods of terrain rendering. It is usually represented by a three-dimensional array with coordinates of vertices. Of the three coordinates, x and y coordinates are evenly spaced while the third coordinate z represents the height of each vertex, as shown in Figure 2. The neighboring vertices are joined to form triangles from which the solid surfaces are rendered. PTG involves techniques, which rely on the generation of certain programs and algorithms [13].

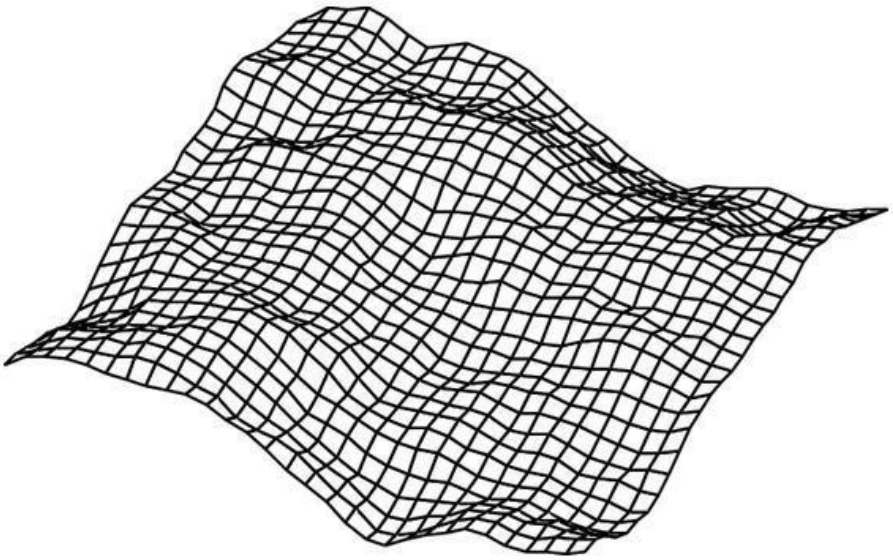


Figure 2: A Discrete height map example [13]

2.1. Artificial Intelligence used in Terrain Generation

Terrain plays an essential role in the user's experience of the game, setting the scene and mood in story-based gameplay and providing strategic and competitive gameplay in multiplayer games.

2.1.1. Terrains and Landscapes

Terrain is defined as "The physical shape, configuration or general unevenness of a surface, considered with reference to variation of height and slope or to irregularities of the land surface" [28]. Two-dimensional images and three-dimensional models are used to depict the terrain properties. Landscape, on the other hand is the composition of land along with other living and natural entities. It is not defined by scale but defined by an interacting mosaic of patches relevant to the phenomenon under consideration (at any scale). Landscapes serve as the main building blocks and also adds realism. The digital data is obtained by scanning, modeling or generating the object. This is followed by the rendering process of the terrain [14]. Figure 3, displays an example of terrain and landscape

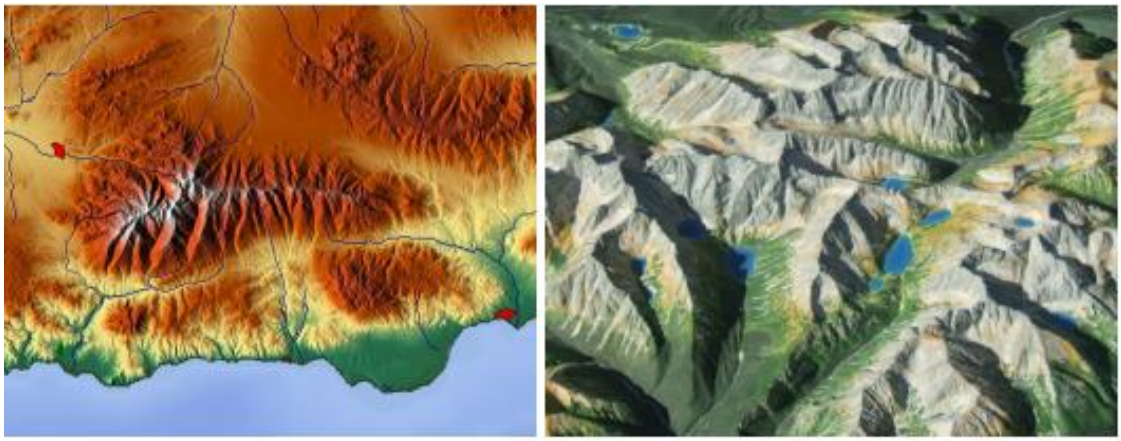


Figure 3: 2D relief image of Sierra Nevada (left) and 3D relief model of High Tatras (right) [14]

2.2. Traditional Terrain Generation Techniques

Traditional techniques in terrain generation can be categorized into two main groups [13]:

I. Measuring

II. Modelling

III. Procedural techniques

2.2.1. Measuring

These techniques utilize real-world measurements from elevation data and produce digital elevation models. Satellite images and land surveys are the common remote sensing techniques used. These techniques require minimal human intervention, producing highly realistic terrains. Designer control is one important aspect, which cannot be ignored since the designer might have to spend considerable amount of time searching for real-time data if the terrains design and features are complex and specific.

2.2.2. Modelling

3D modeling programs such as Maya [15], 3D studio [16], Blender [17], or specialized terrain editor programs such as SimCity4 or SimEarth [18] are used to manually create the

terrain morphology. This technique is deemed to be one of the most flexible techniques. Skilled professionals/designers are required and is usually a time taking process. The creative skill of the designer is an important aspect, since it reflects on the final terrain generated and its closeness to reality.

2.2.3. Procedural Techniques

The procedural techniques can be further divided into Physical based, spectral and fractal techniques. Figure 4 gives an overview how traditional terrain generation technique is applied to create terrains, which will be dealt in detail in Section 2.3 Procedural terrain generation (PTG).

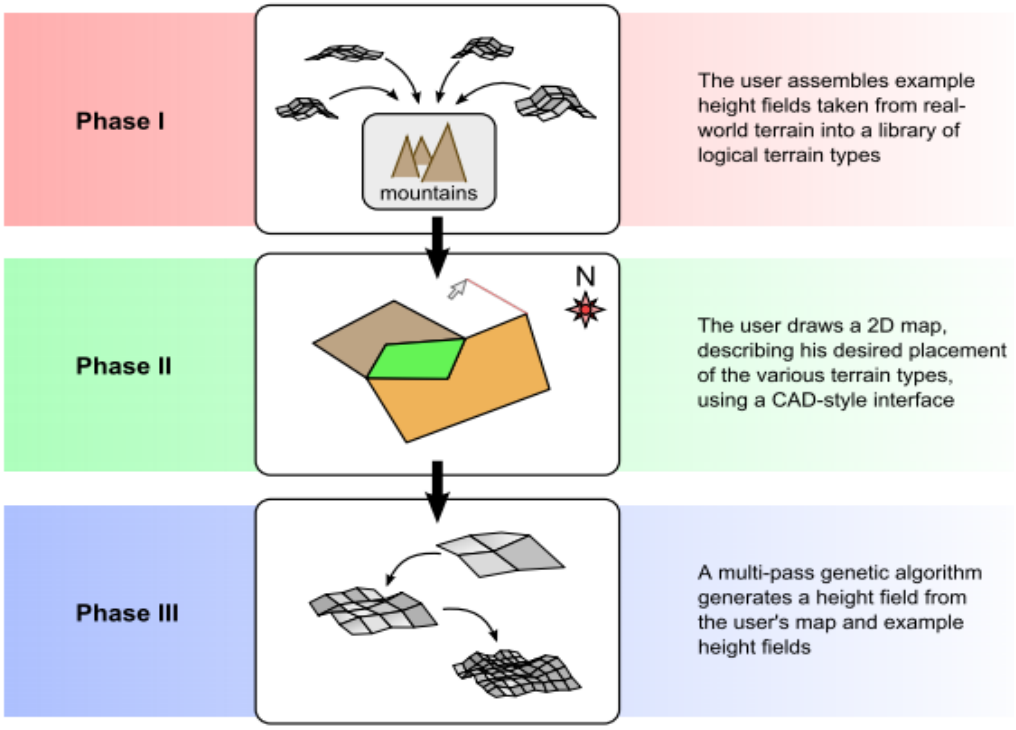


Figure 4: Three phases of traditional terrain generation [10]

2.3. Procedural Terrain Generation (PTG)

Procedural design or techniques rely on the generation of programs or algorithms in generating terrains. The procedural techniques can be further divided into the following classes.

2.3.1. Physical Based Techniques

These techniques put physical laws into best possible use, since physical processes such as wind and soil erosion, thermal and plate techniques form the basis for simulating the various natural phenomena of terrain evolution [7].

2.3.2. Spectral Syntheses

This technique converts the frequency components into altitudes but this technique suffers from certain fallacies such as isotropism and statistical homogeneity, which are the least desirable characteristic of a real-world terrain.

2.3.3. Fractal Techniques

Terragen and Gensurf [19-20] are some of the popular tools based on fractal algorithms. Fractal techniques are characterized by the concept of self-similarity, as shown in Figure 5. The object subset exhibit similarity to themselves and the object as a whole. This property allows for scalability since the terrain looks like a terrain upon scaling up. The speed and ease of implementation makes it one of the most desirable algorithms in the gaming world. However, the characteristic of self-similarity makes it easy to recognize the terrains.

Among the different procedural techniques discussed here, erosion simulation is expensive, though it adds reality to the terrain. Erosion stimulation and fractal techniques also suffer from certain fallacies such as highly stochastic nature and limited control over the terrain type generated. Patch based terrain generation is an alternate method of terrain generation. The algorithm developed by Zhou et al. [21] is based on a basic sketch of a layout, and a terrain is generated matching it.

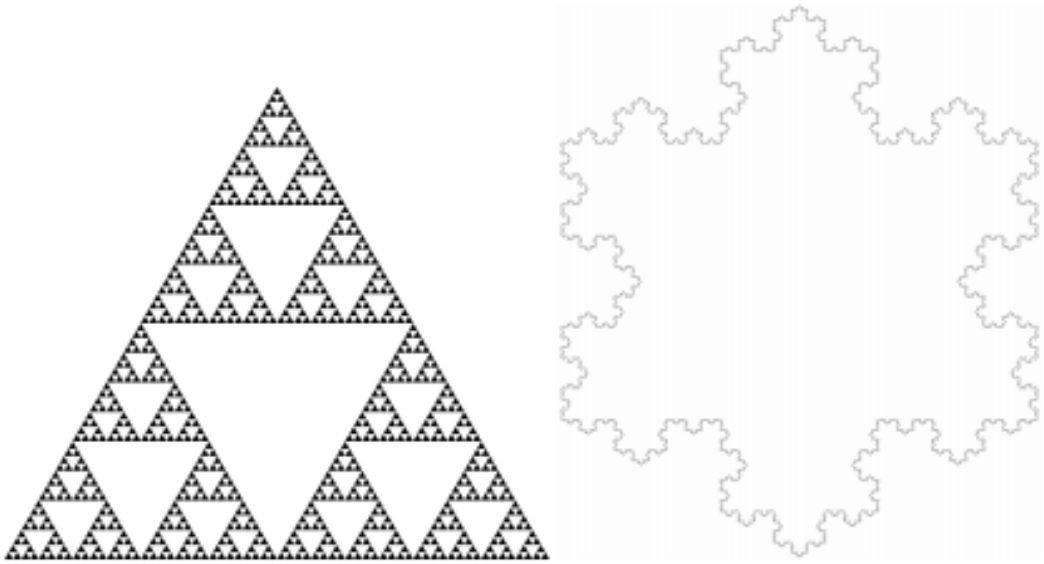


Figure 5: Self-similar planar shapes [14]

2.3.4. Evolutionary Terrain Generation Techniques

Evolutionary algorithms (EAs) are bioinspired algorithms. There are four main classes of EAs:

- Genetic Programming
- Genetic Algorithms
- Evolutionary Strategies
- Evolutionary Programming

Among these, Genetic Algorithms and Evolutionary Strategies are good candidates. EAs have been adopted by several researchers in this field to add more control over terrains generation process and to produce terrains, which meets user preferences such as position and determination of mountains and valleys.

Table 1 provides descriptions of the different evolutionary approaches in PTG. To the best of our knowledge, these five algorithms are the only publications where evolutionary

algorithms are applied in PTG techniques. Each approach is detailed with author name, the technique used, and their advantages and disadvantages [9].

Table 1: Evolutionary approach in generation of terrain [9]

Approach	**Technique**	**Advantages**	**Disadvantages**
Ong et al.	Generate 2D outline of terrain with sketch of terrain boundaries. Height map data provided by the user.	Useful in generating family of similar terrains. Possible usage in games such as flight simulators that need large, natural terrain.	Making slight variations however require appropriate samples and it is unlikely that a suitable terrain is generated.
Ashlock et al.	EA with L – learning system, generate fractal style division.	Effective in generating fractal style division	Algorithm cannot produce terrains that differ from fractal terrains.
Walsh and Gade.	Uses EA to automatically adjust parameters values of a terrain generator.	Introduction of right parameters can increase the applicability of EA.	Has limited applicability to games. Feature scale parameters require a predesigned map.
Frade et al.	Used Genetic Programing (*GenTP*) to create height function.	Better exploration of solution space due to the use of mutations and preventing the height function from becoming too long.	Not suitable for flatter surface terrains.
Togelius et al.	Multi-Objective Evolutionary algorithm(MOEA) Height-maps which creates mountains.	Initial results helped in generation of complete maps.	In comparison to other algorithms, this one contains only basic detail, due to genotype representation.

2.4. Patch Based Terrain Generation

The use of a two-level interaction selection scheme involving parent selection and gene selection is a unique method targeting control over patch-based terrain generation. This ensures that every patch of each parent is marked separately for mutation and crossover. Later, this is used in the two-level interaction evolution process for evolving and stitching smaller patches into larger terrains. In this method of terrain generation, several smaller height maps of equal sizes are stitched together to form a larger size map. The following are the advantages of this method of terrain generation:

- Avoids repetitive height calculation of vertex for every round of new terrain generation.
- Retaining good local features and swapping out undesirable features.

Zhou et al. in 2007 introduced an algorithm known as "Poison seam removal algorithm" a variant of poison image editing technique [21]. This algorithm emphasizes a patching system for generating a 3D terrain structure. Later, the "Roof tiling overlapping algorithm" was introduced by Raffe et al. in 2011, as shown in Figure 6 [22] The roof tiling algorithm forms the basis of our present paper. Table 2 shows the differences between the two algorithms [12].

Table 2: Difference between poison seam removal and roof-tiling overlapping algorithm

Poison Seam Removal Algorithm	**Roof Tiling overlapping algorithm**
Requires a feature detection algorithm to extract patches	Utilizes a sample height map to extract patches in a nested looping manner
Involves searching for a best fit patch and manipulation to match the users sketch map.	Patches are placed in a grid like fashion similar to texture synthesis algorithms.
Uses feature sketch in controlling patch layout	Uses evolutionary algorithms allowing interactive adjustments in the terrain.
Area around the seams need to have similar height values for height maps.	Stiches patches of varying heights.

2.4.1. Roof Tiling

The method of stitching of patches is a very important component of the patching technique. Using inappropriate methods will render the borders between the patches to become clearly evident. Mutation operations might result in patches of varying heights such as a mountain cliff and a valley lying adjacent to each other. Patches of varying heights are stitched together utilizing minimal seams and is akin to the laying of tiles on the roof top. All the patches overlap their neighboring patches. The quality of the terrain depends on the overlapping technique used and the size of the overlap region. The transition in the overlap region is smoothened by using a cubic spline interpolation function.

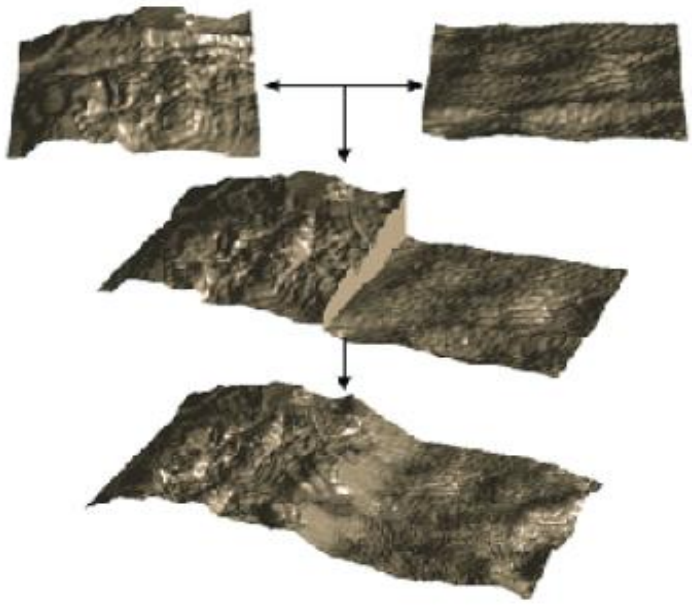


Figure 6: Two patches first joined together with no overlapping [22]

2.4.2. Basic Concepts Related to Biological Evolution

Since the roof-tiling algorithm is based on parent and gene selection required for a two-level interactive method for evolution of patches throughout the generation process, we need to understand the core concepts of biological evolution.

DNA: DNA stands for Deoxyribonucleic acid**.** It is a double helical structure and contains the genetic information for faithfully replication to the offspring.

Chromosomes: Compact strings of DNA packaged into a nucleus of the cell.

Genotype: Genotype is the genetic information encoded in the organism.

Phenotype: Phenotype refers to the physical observable characteristics of the organism, which is the result of the DNA.

Reproduction: Reproduction is the process of creating offspring involving parents, by inheriting genetic information from them.

Crossover: Crossover is the process of exchange of certain portions of gene segments during the synthesis of offspring DNA.

Mutation: Mutation is a small heritable change in the DNA of the offspring resulting in minor variations to a non-mutated crossover. The probability is however very low.

Survival of the Fittest: Over the period of several reproduction cycles only the robust properties of DNA are retained whereas the weaker properties of DNA are lost resulting in lower chances of survival. This is also referred to as evolution [23].

3. APPROACH

3.1. Genetic Algorithm

As discussed earlier, the Genetic Algorithm is a flexible and adaptive algorithm and is credited to find good optimized solutions to most problems. The Genetic algorithm usually follows the following pattern, as shown in Figure 7.

A group of individuals is used to create a population or is sometimes chosen randomly. Later this undergoes evaluation until best results are achieved. In general, this algorithm uses three steps: selection, crossover, and mutation. The process of selection involves the evaluation of each individual in the given population on the basis of fitness; the greater the fitness, the greater are the chances of being selected. This process ensures that the parent features are transmitted throughout the generation. One or more offspring is created with crossover, which requires a pair of individuals. Mutation is then applied to swap parts of the new individuals to be produced. Therefore, at the end of the process of generation, an optimal process is found.

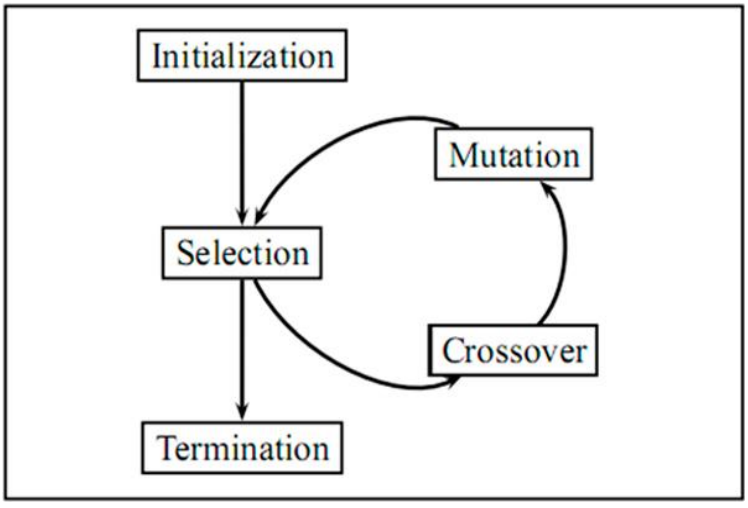


Figure 7: The basics steps of GA: Selection, crossover, and mutation [26]

3.2. Initial Population

Certain conditions need to be taken into consideration in selecting the initial population. Each patch is considered as a gene, and the chromosome is structured as a fixed grid, i.e., the evolutionary operator's crossover and mutation have to deal with substituting the patches and the number of patches cannot be increased or decreased. Initially, the population can either be offspring fetched by the user providing sample terrains, from which patches have been extracted or it can also be randomly generated. When compared to previous contributions in this field, this algorithm does not deal with altering the height values of individual patches rather it changes the placement of patches provided by the user in the parameters [24].

The user specified samples or the default terrains are required for population to the Patch Database. After initiating the run, patches are extracted from these sample terrains and put into a database with each ID, so that they can be used throughout the execution to generate terrains.

3.3. Fitness

Terrain generation being a two-level interaction process uses interactive fitness. To select the parents for the next generation, the user selects between 0 and number of parents to be base parents for next generation. For now, the number of parents per each generation is a maximum of two parents selected. The following scenarios are possible:

- If no parents are selected in each generation the next generation is randomly generated, no features from the current generation would be able to passed to the next generation.
- If one parent is selected in the current generation then the next generation will consist terrains, which are mutated version of that one parent only.

- If two parents are selected in the current generation then the population in next generation will be result of crossover between those two parents, resulting in continuous exploration of terrains where one crossover is done mutation will be applied on each offspring.

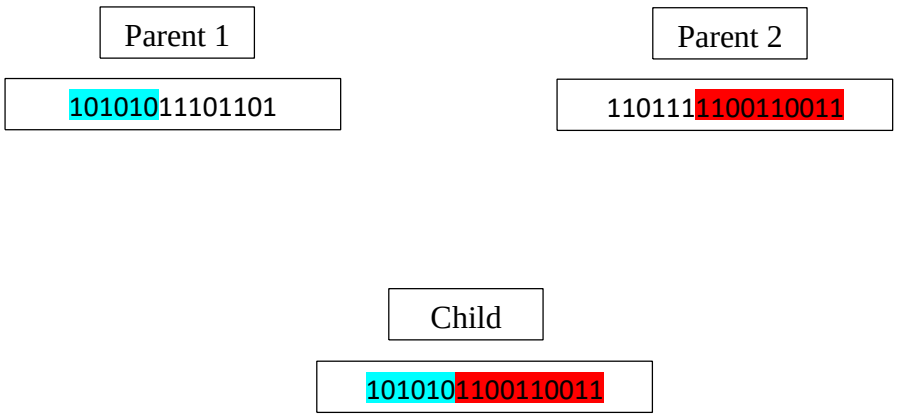
3.4. Crossover

Crossover is a genetic operator that combines two parent chromosomes to reproduce new set of chromosomal offspring. A crossover operator is necessary since new chromosomes may contain some better features than the base parents selected at the time of crossover.

Crossover can be performed in multiple ways, which are discussed below in detail.

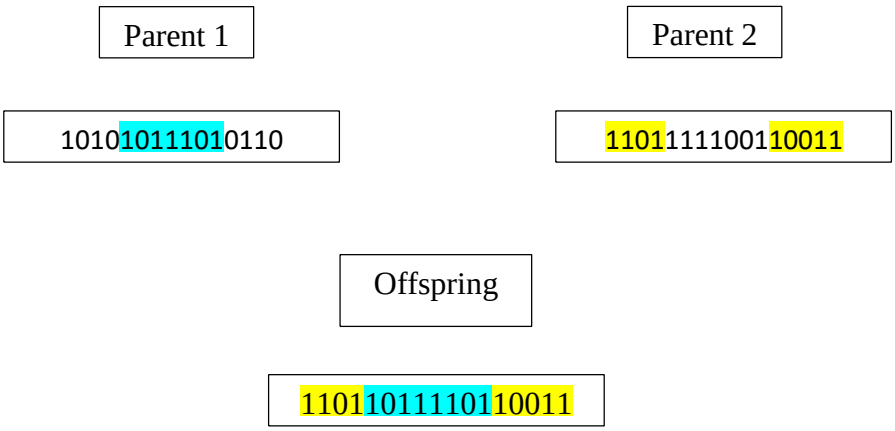
3.4.1. One-Point Crossover

In this, the crossover operator selects a crossover point with the chromosome and mix matches the parents selected, resulting to produce set of new offspring. The following example shows 2 parents selected for one-point crossover:



3.4.2. Two-Point Crossover

In this process, the crossover operator randomly selects two crossover points and mix match the two parents in between two points and reproduce two new offspring. The following example shows 2 parents selected for two-point crossover:



In our problem representation, we use two-level interactive process where the user can select what patches to select and apply crossover on those patches accordingly. The number of parents is selected and crossover is applied on the selected parents to produce the offspring in next generation. In this process the parents are mix matched and crossover is performed on a patch by patch basis, i.e., to ensure each patch undergoes the crossover process. The following represents how each offspring is evolved when the parents are selected.

- Firstly, one is chosen from the selected parents to be the base parent for the child, which is now the clone of that parent. This parent can be chosen randomly so that each child can have characteristics of its base parents. Based on the crossover rate, there will be uniformity in the offspring, being similar to either of the parents.

- After the base parent is selected its patch-map is copied to the child where each patch-map is given a crossover probability.
- After the probability value is given to the crossover rate parameter then the patch is swapped to the patch in the same position on the other parent patch-map. As shown in Figure 8.

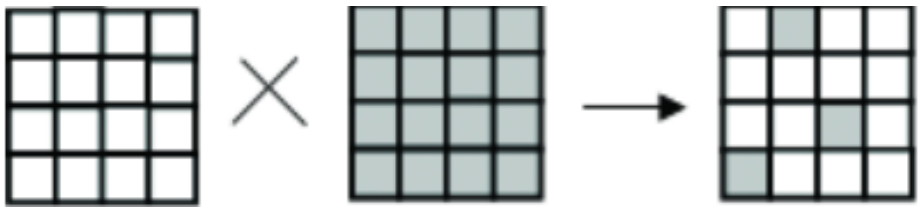


Figure 8: Crossover operation example, white parent is the base parent with crossover rate of 0.2. [22]

3.5. Mutation

An existing patch map can be manipulated to introduce randomly selected patches through the process of mutation. Either one parent can be selected after cloning or two parents are selected after the crossover process. Once the child is established, mutation probability is employed to assign single patch to the child's patch map like a patch by patch algorithm.

The following scenarios are possible:

- If the mutation probability is lesser than the mutation rate, then the patch is replaced by a random patch from the patch database.
- If the mutation rate is 0 and only one parent is used, the child is an exact clone of the parent.
- If mutation rate is 1 all the children are generated randomly and will not bear any similarity to the parents.

3.6. Evolutionary Strategy

Similar to Genetic Algorithms (GA), Evolutionary Strategies (ES) follow the same principle of natural selection evolution as a process to solve parameter optimization problems. The procedure of evolutionary strategies starts with first mutation & selection mechanism with two individuals per generation only, and the process is ended with randomly generated chromosomes, which are to be evaluated with the next generation process [25].

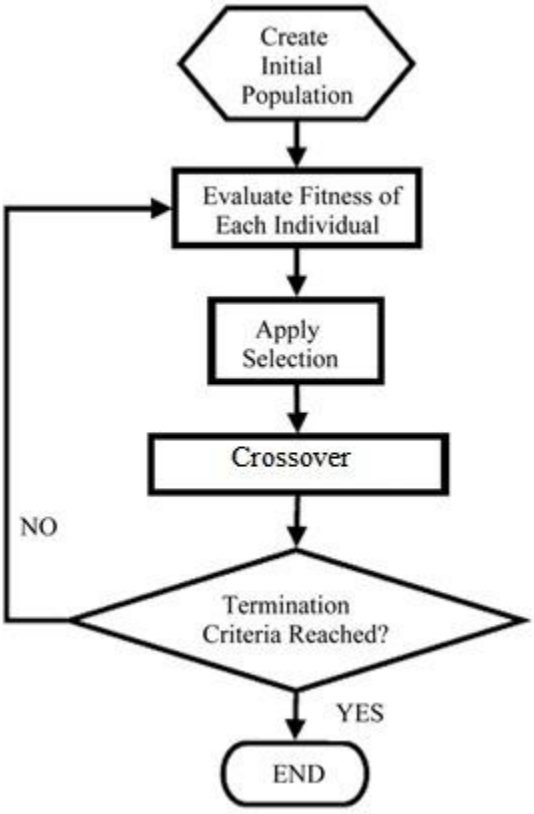


Figure 9: Strategy represents selection process modifications [27].

In this present paper, a comparison is made between two different EAs – genetic algorithm and evolutionary strategy. Each of the algorithms consists of parent selection, crossover and mutation for generating three dimensional terrains. However, the “crossover step” is “eliminated” in the evolutionary strategy. An example of an evolutionary strategy operation is

shown in Figure 9. In the further experimental section we evaluate the fitness of terrains by visualizing and comparing the results and execution time of both algorithms.

4. IMPLEMENTATION DETAILS

Raffe developed the evolutionary Terrain Tool. It is designed as a virtual terrain development tool for interactive designers and completely written in Matlab [22]. It considers sample heights terrains and analyzes them into patches with the help of the Roof-Tiling-overlapping algorithm. The roof-tiling overlapping algorithm stitches the selected patches from left to right side to get the desired terrain. In this section, we will briefly discuss the inputs required for this application to run, namely the number of iterations, patch size, population size, number of parents, overlap size, terrain resolution, crossover rate and mutation rate. Tables 3-5 represents in detail the functions and assigned values required for the application to run. The Roof-tiling overlapping algorithm is analogous to laying tiles on the roof of a house. This algorithm helps to stitch patches of various heights together with minimal seams, the number of patches, and overlap percentage of the terrain. The user can then use the two-level interactive evolution process, while varying patch sizes, mutation/crossover, and overlap sizes to form a full–fledged terrain.

4.1. Patches

Patches are small sample terrains essentially necessary for building the height-map. They are height-maps of smaller resolution divided when combined will form a full-sized terrain with their dimensions that would be equal to terrain resolution parameter.

4.2. Height-map

As discussed earlier a height-map is a two-dimensional matrix with dimensions, equal to the terrain resolution parameter for a full sized generated terrain or a full patch resolution for patches. Each of the values is specified in the form of the matrix between 0 and 1, this is the

process of rendering of the terrain. An example of height-map before and after the terrain generation is shown in Figure 10.

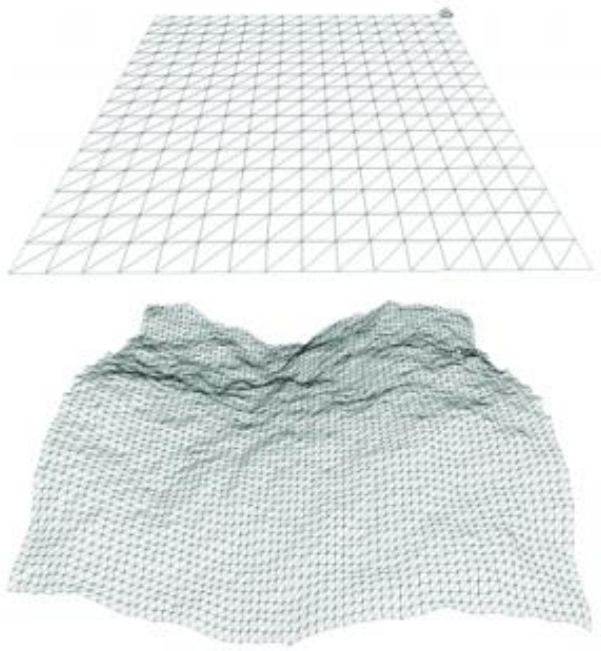


Figure 10: Flat height-map and height-map after terrain process [22]

4.3. Patch-map

A patch-map is a 2D-matrix with dimensions equal to square number of patches parameter. Each value in this matrix is a set of coordinates, which indicates it as a matrix ID that links to a unique patch within the Patch Database. When a height-map is generated, the IDs in the patch-map is used to query the patch database, extract the chosen patches to use their height-map data, sew the patches together with overlapping process and then save the final size terrain as a height-map. Figure 11 presents an overview of how sewing is done in between the patch-maps also all important terrain parameters to form a full-fetched terrain.

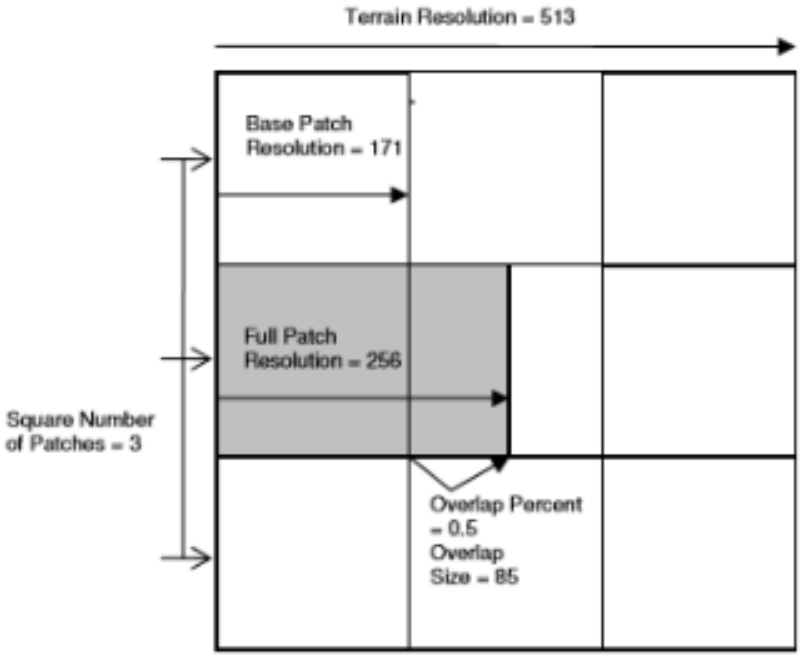


Figure 11: Overview of how patching is done with their important values

4.4. Specified Terrain Creation Parameters

This section briefly gives an overview, defines the various initial parameter. As shown in Table 3 to be considered for creating and evolution of terrains.

Table 3: User specified parameters

Parameter Name	Value	Description
Terrain Resolution	E.g. 129,257,513	Resolution of the terrain that will be generated, provided in the program.
Number of patches	2,3,4,9	Number of patches in a row/column in a generated terrain. The number of square patches are developed in Base patch resolution which later is used to overlap the patches, decide the overlap percentage according the user preferences.
Overlap Percentage	0-1	The percentage states how much of each patch is to overlap the patch around.

4.4.1. Dependent Parameters

Table 4 represents the dependent parameters and functionalities for terrain resolution, patches, and overlap size.

Table 4: Dependent parameters

Parameter Name	Calculation
Base Patch Resolution	= terrainResolution / square Numberof Patches
Overlap Size	= basepatchResolution * overlapPercentage
Full Patch Resolution	= basepatchRes + overlapSize

4.4.2. User Specified Evolution Parameters

Evolution parameters required for terrain generation are generally initiated by the user, basically depending upon a user desired goal with the requirement he has got to generate the terrain. Table 5 shows how the user specified parameter values can be varied depending on the user specified goal:

Table 5: User specified parameters

Parameter Name	Value	Description
Sample Terrain	List of height-map file names	A text file is used to represent the height-map value of each terrain, value between 0-1 which is represented in the form of matrix.
Population Size	1 - 8	To Specify how many terrains are generated and displayed per generation. The number of base parents for the generation of new offspring is set to value of 8.
Number of Parents	2	The maximum number of parents that can be selected in each generation to produce offspring, is set to 2.

4.5. Implementation Evaluation

In this section, we will discuss the interaction of the application with the user interface. It provides a friendly two-level interactive evolution with an overview on how to consider the parameters i.e., selection for square number of patches, overlap percentage and other evolution details. The application is implemented in Matlab. The GUI screenshot in Figure 12 shows how one can choose parents by selecting the check box under each terrain and then click the evolve button to generate the next level of parents. As mentioned in Table 5, one can select between 0 and 2 (number of parents) for now. In the next generation as shown in Figure 13, the first two terrains on the top left are exact clones of the parents, whereas the remaining terrains carry similar features of the selected parents.

Figure 12: GUI screenshot for two-level interactive process with selected parents

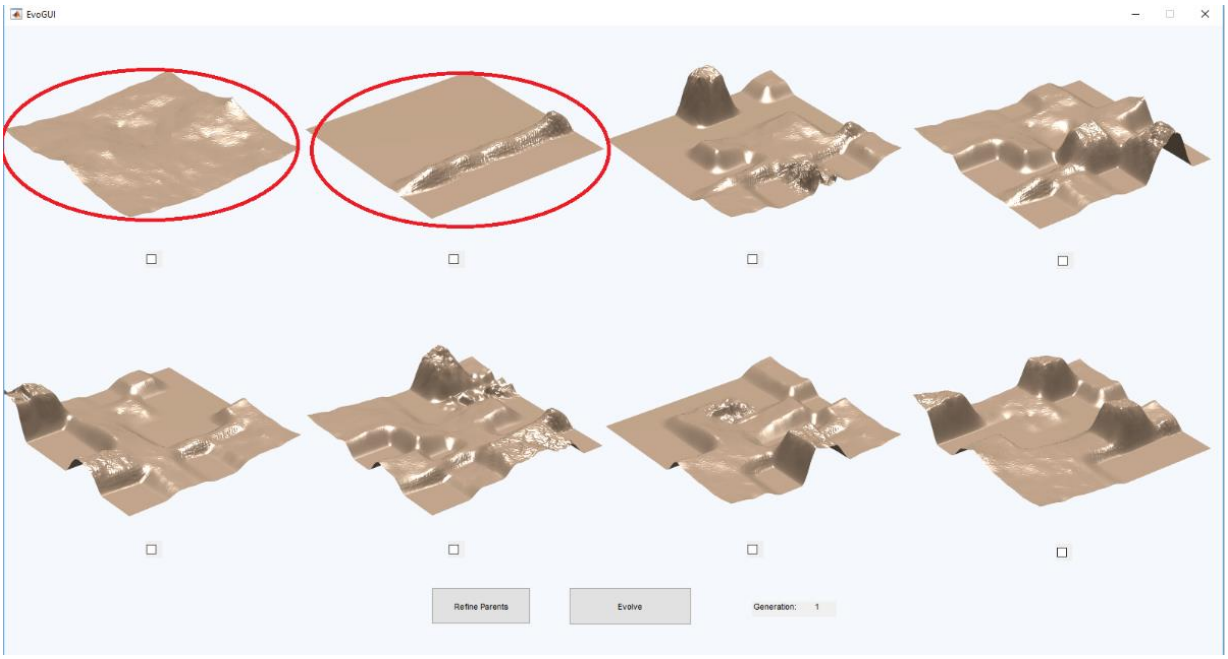


Figure 13: GUI represents the first two parents in top left selected in initial generation

4.5.1. Varying Square Number of Patches Parameter

The images below show how the square number of patches parameter is varied while all other parameters are constant. Figures 14 - 16 represents how randomly generated patch-maps are used for all the terrains generated, here the size of terrain resolution is 513 with overlap percentage of 0.5.

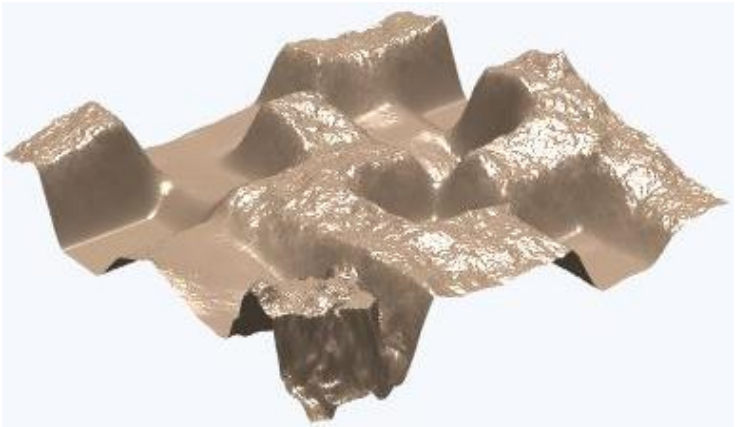


Figure 14: Square number of patches = 2

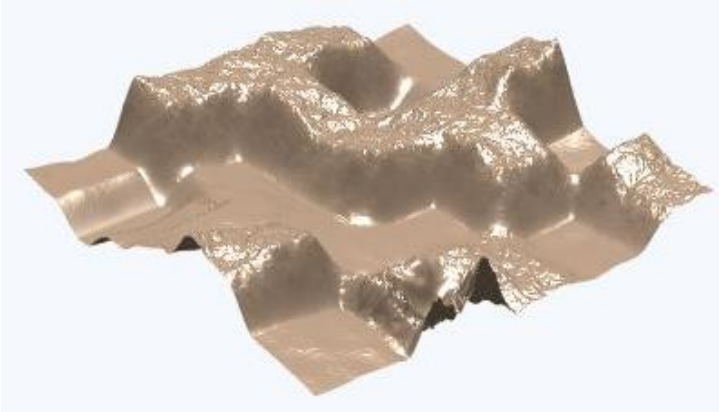


Figure 15: Square number of patches = 4

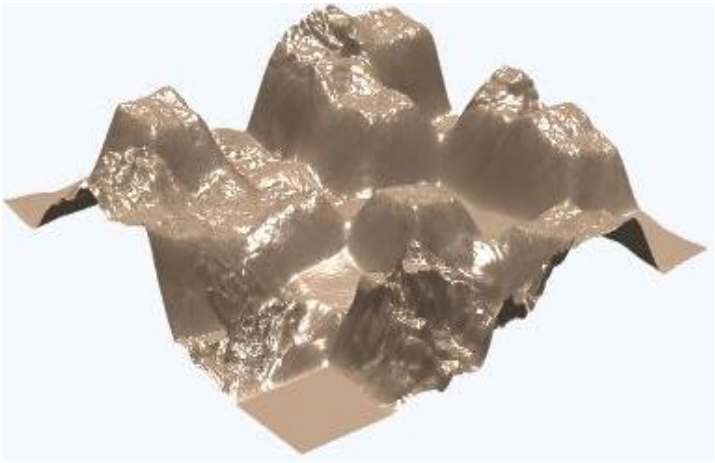


Figure 16: Square number of patches = 8

As we can see in Figures 14 – 16, an increase in the number of patches results in many more visible features between the patches. The more the number of patches to represent the terrain, the higher is the chance of features to be discontinued between the adjacent patches. This results in undesired features and is a time taking process to get rid of.

4.5.2. Varying Overlap Percentage Parameter

The following images depict the effect of varying the overlap percentage while keeping all other parameters the same. The values are kept constant i.e. Square number of patches = 5 and terrain resolution = 513.

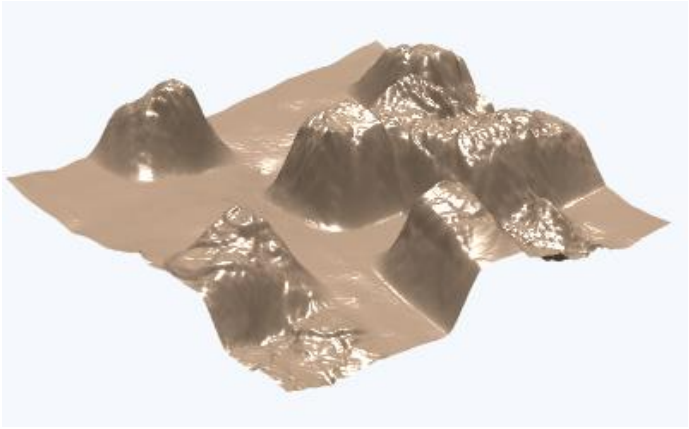


Figure 17: Overlap percentage with 0.1

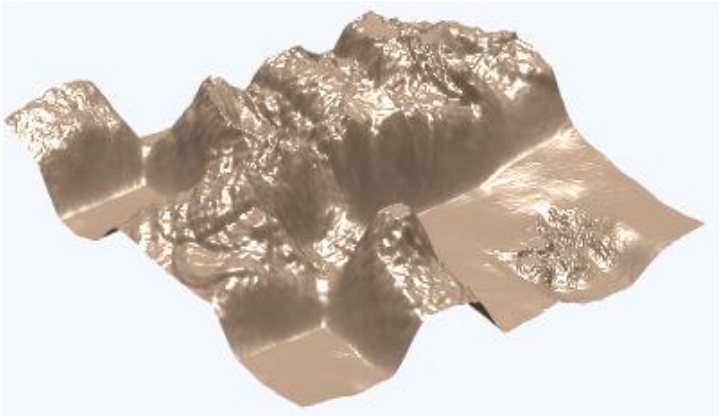


Figure 18: overlap percentage with 0.5

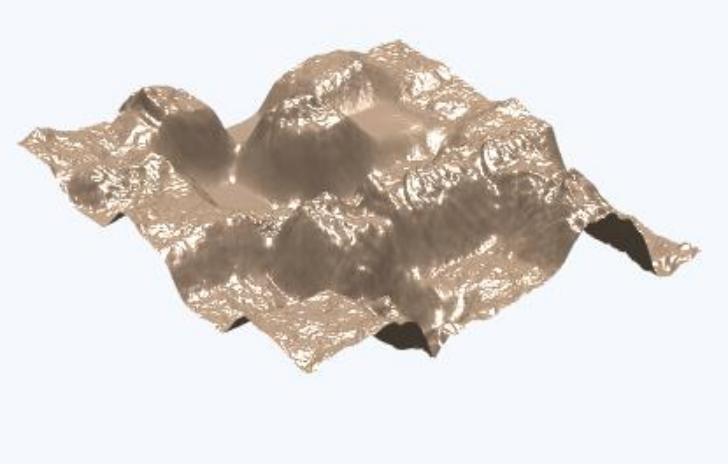


Figure 19: Overlap percentage with 0.9

As predicted and seen in Figures 17 - 19, the overlapping patches allow more seamless flow between adjacent patches. However, this does not solve all the problems. For example, as

we saw earlier, increasing the number of patches leads to discontinuous feature alignment between patches resulting in an illogical terrain. Figure 20 depicts a terrain resulting from a high overlap percentage value with a large number of patches.

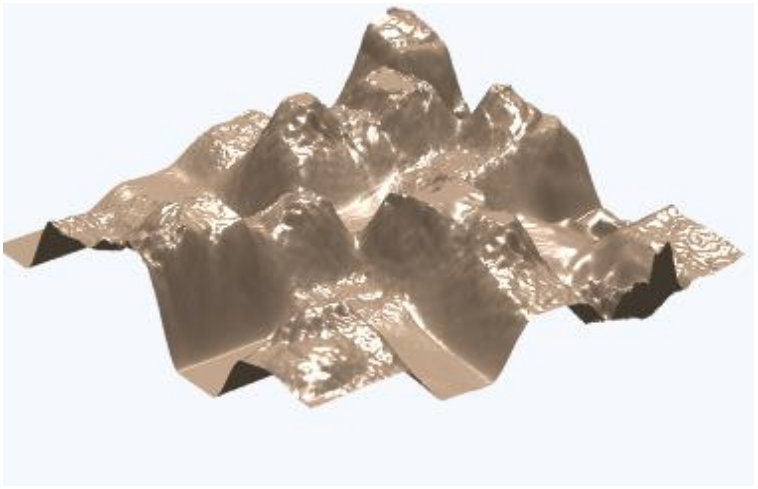


Figure 20: Overlap percentage = 0.9 & square number of patches = 8.

While the peaks of the features look far more natural with smoother transitions between patches, there is still an irregular flow and lack of coherency among terrain features.

5. EXPERIMENTS AND RESULTS

In the current set of experiments related to this paper, a regular CPU system with a 3.10 GHz, Intel i7-core processor, 16GB RAM and windows operating system was used. The template created in this experiment treads the path of interactive evolution and to evaluate the fitness of selected terrains. Choosing suitable parameter settings is one of the crucial steps in exploring the search space and therefore developing the desired terrain. In the present experimental settings, the parameters are chosen to lessen user exhaustion and thereby allow the system to generate the terrain through efficient and quick convergence.

5.1. Experimental Methods and Conditions

In this section, we use a combination of all the input parameters required to create terrains under the following conditions. A fixed map size of 512 x 512 was used, which is the number of vertices per candidate terrain. Pixels would be used instead of vertices if the height map is represented as a two-dimensional image. The number of patches constituting the terrain or the number of genes in the genetic representation determines the patch size. The optimal number of patches for the interactive evolution template is determined to be 16 patches (4x4) or 25 patches (5x5). The larger the number of patches than the optimal number leads to larger difference in average height values due to increased chances of adjacent placement of patches and therefore jagging of candidate terrains. The more the patches, the greater is the effort to control evolution leading to increased substitution of patches per offspring. A lesser number of patches leads to decreased variation in the candidates. In all of the following experiments we used a population size of 8, allowed for 2 parents of maximum per generation with combination of all other parameters. With each run in this section at the end a visual comparison of terrains is shown selected with genetic algorithm and evolutionary strategies.

5.2. Experimental Setup and Process

The following sets of experiments were conducted to achieve the desired/expected results. In this experiment, the following parameters and values as shown in Section 5.2.1 were used to achieve the result from running the genetic and the evolutionary process. Where all the parameters were static throughout run 1, in evolutionary process the crossover function was eliminated to compare the fitness of selected terrains with both genetic and evolutionary strategies.

5.3. Experiment Run 1

In this run, the population size of 8, allowing to select the user 2 parents per generation, using a crossover rate of 0.5 to ensure both selected parents were influenced to form new offspring, with a mutation rate of 0.1, with optimal number of patches for the interactive evolution template is determined to be (5x5) square number of patches 25, overlapping percentage of 0.6 and with cubical spline stitching method.

Generation1

In Figure 21, two parents were selected in the initial generation with genetic operator's crossover and mutation as we can see on the top left corner in generation 1 with the parents selected in previous generation and the remaining are offspring of the selected parents, until it reaches generation 8 as shown in figure 22.

Figure 21: Selected parents on top-left corner generated with genetic operators

Generation 8

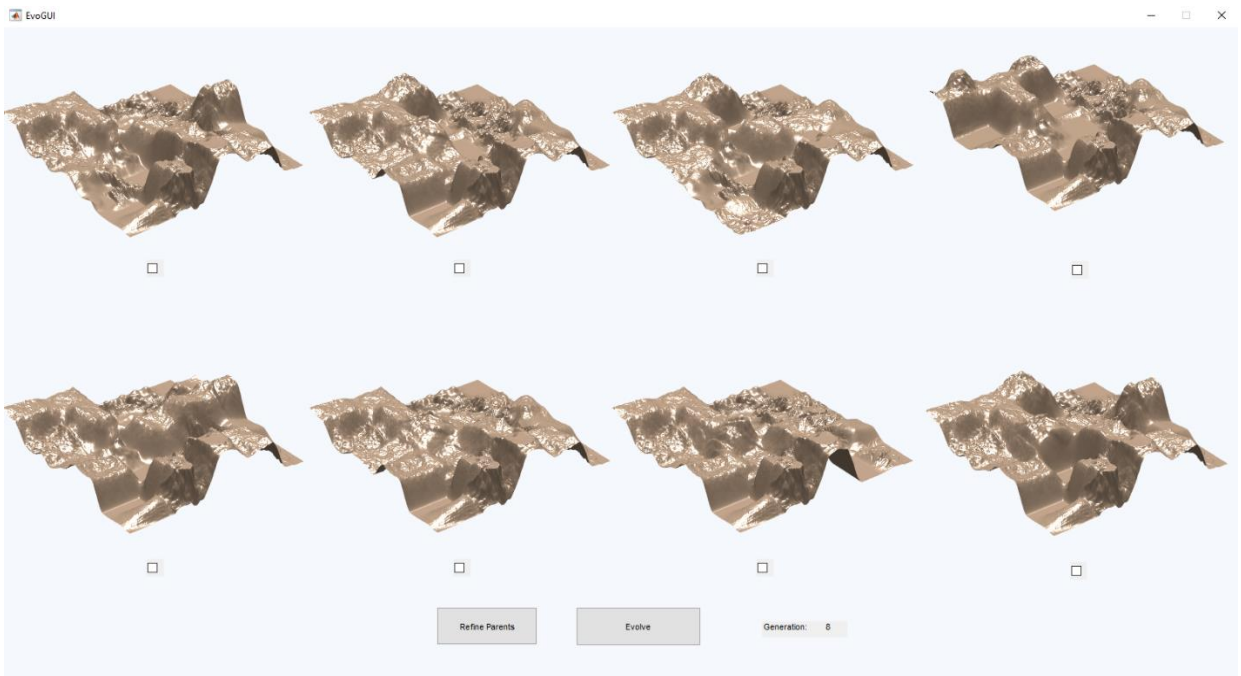


Figure 22: Evolved terrains after 8 generations

Overall, 15 generations were run in this experiment to get an overview of how the terrains are evolved from generation 1-15 and to observe if they carry the same features from the selected parents in each generation.

Figure 23 displays the difference between the final terrains selected with genetic algorithm (*on the left*) and evolutionary strategy run (*on the right*). As we can see on the left image there are few elevated peaks on the right corner of the terrain, whereas the same feature is missing in the right image. It could be possible that without the crossover operation in the evolutionary run similar features of selected parents have not been carried through the generation. If observed keenly, there are few similar features in both the images but elevated peaks on the left images makes a difference when compared to the right image, which was evolved with the evolutionary strategy process. At the end of this section, we show the elapsed time comparison for genetic algorithm and evolutionary strategy have taken throughout the generations.

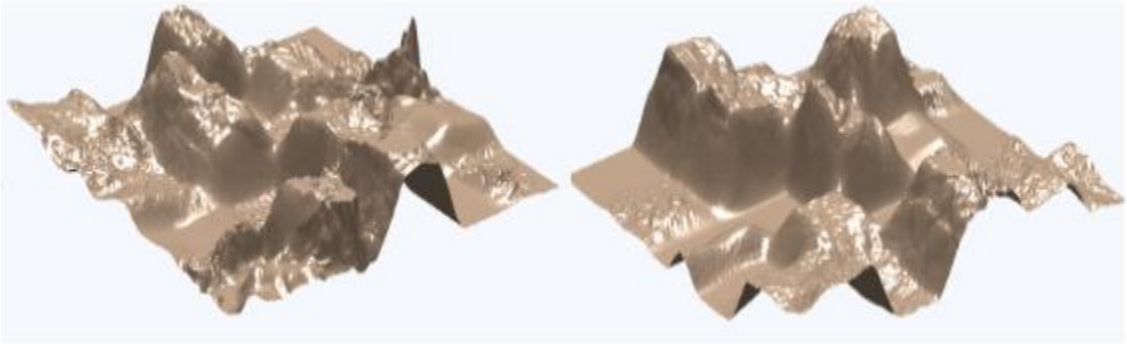


Figure 23: Final selected terrains process with genetic (left) and evolutionary (right)

5.3.1. Experiment Run 2

In this run the population size of 8, allowing to select the user 2 parents per generation, using a crossover rate of 0.7 to ensure both selected parents were influenced to form new offspring, with a mutation rate of 0.5, with optimal number of patches for the interactive evolution template is determined to be (6x6) square number of patches, overlapping percentage of 0.6 and with cubical spline stitching method. In this run an interactive gene selection process is shown in Figure 25, where one can select the features from the selected parents to keep in the next generation. This is followed if there are any undesired patches in the terrains, which the user does not wants to carry in next generation.

Figure 24 shows how after 28 generations there were some undesired patches in the terrains.

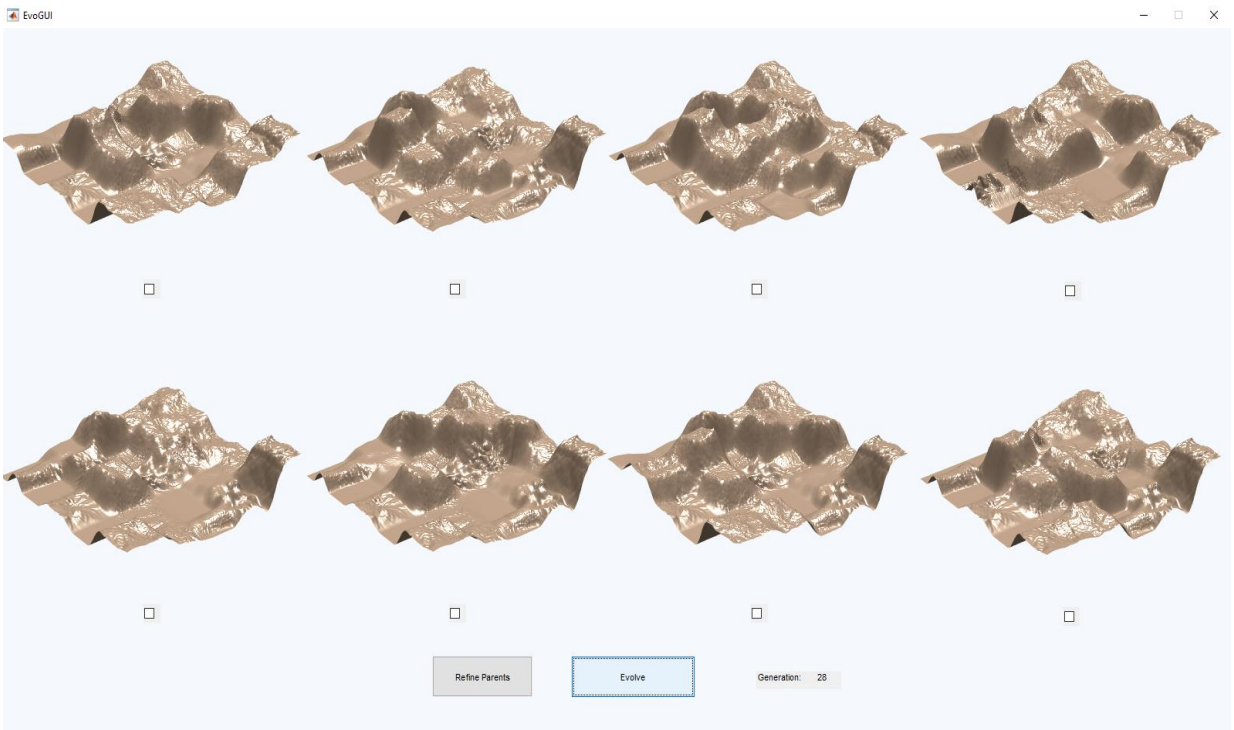


Figure 24: Undesired patches after 28 generations

However, to get rid of undesired patches the gene selection method was used as shown in Figure 25.

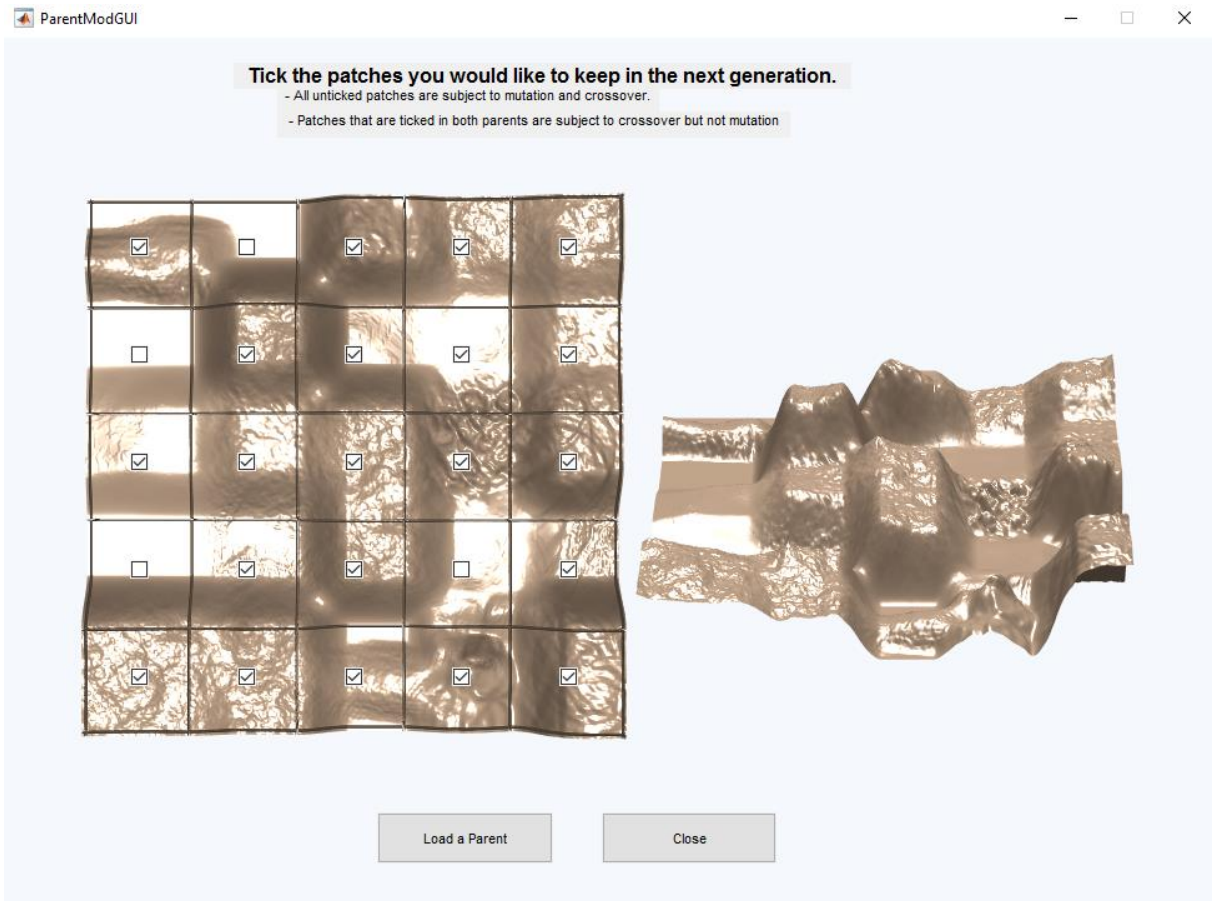


Figure 25: Gene section through parents

In the gene selection process, we have selected patches which are only required in next generation, through which crossover and mutation takes places within those selected patches only. Therefore, eliminating the undesired patches in Figure 26, the final generation of this run we can see how all the terrains are having similar features after gene selection through which the final genetic evolved terrain was selected as shown in Figure 27 (*left*).

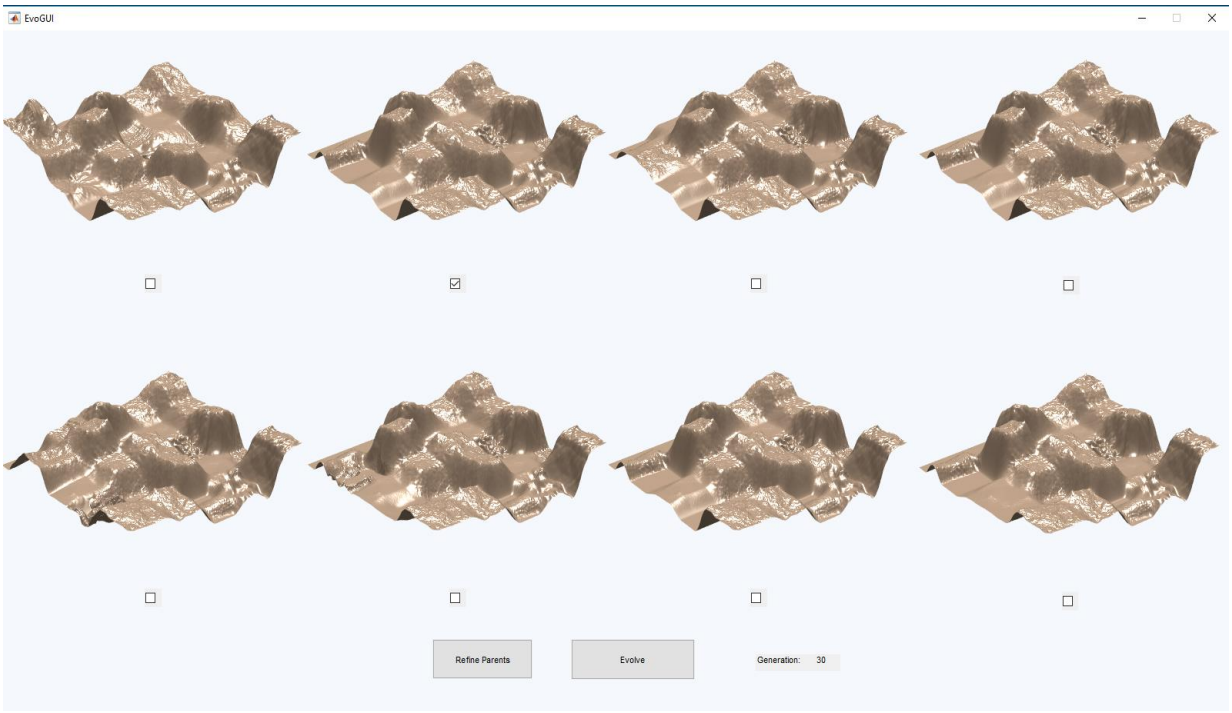


Figure 26: Terrains generated after gene selection

Figure 26 represents how the terrains are generated with similar features after gene selection. After 30 generations in this run, unwanted patches are removed with the interactive gene selection, which is one advantage of genetic operators. Figure 27 represents the difference between gene selection terrains as we can see the final selected evolutionary strategy terrain (right) has not fully evolved, whereas gene selection process was used to evolve the image on left.

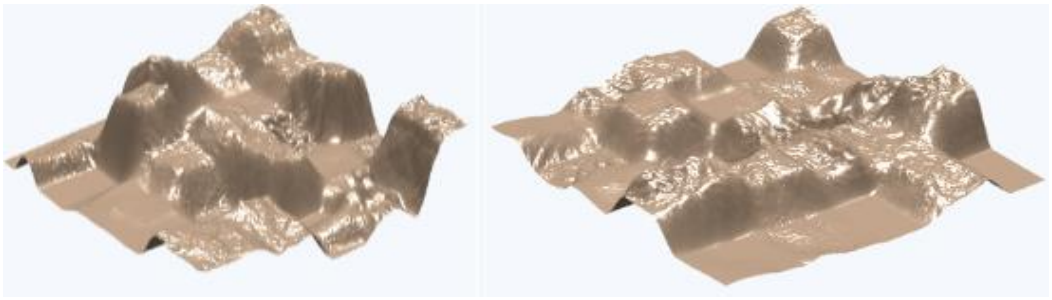


Figure 27: Final selected terrains process with genetic (left) and evolutionary (right)

5.3.2. Experiment Run 3

In this run the population size of 8, allowing to select the user 2 parents per generation, using a crossover rate of 0.4 to ensure both selected parents were influenced to form new offspring, with a mutation rate of 0.2, with optimal number of patches for the interactive evolution template is determined to be (8x8) square number of patches, overlapping percentage of 0.3 and with cubical spline stitching method. As we can see the overlap percentage is reduced to 0.3 when compared to previous runs to observe how the patches are evolving along the generations. Figure 28 represents the final selected terrains after 50 generations of genetic algorithm and evolutionary strategy operations. As it can be clearly seen with overlap percentage of 0.3 and all other parameters static, the genetic algorithm and evolutionary strategy generated terrains are not fully evolved resulting most of the patches to be flat surfaced one after the other. This indicates not only mutation and crossover operator plays a vital role in the generation of terrain but other parameters too.

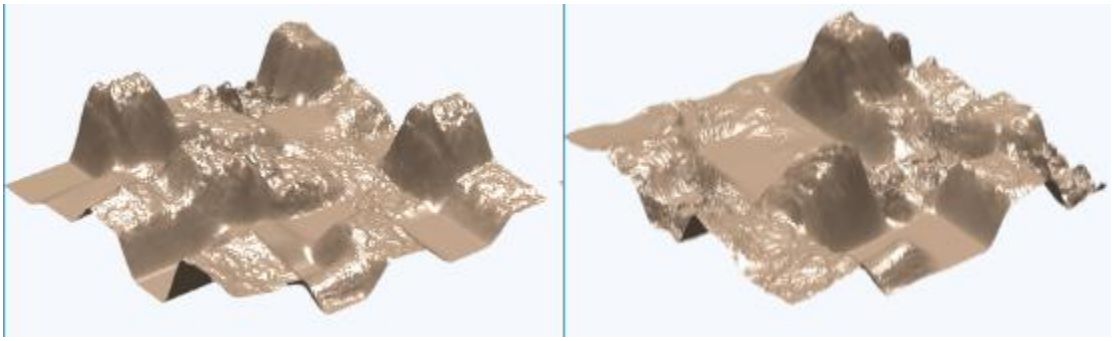


Figure 28: Final selected terrains process with genetic (left) and evolutionary (right)

Figure 28 displays the final selected terrains after 50 generations of genetic algorithm and evolutionary strategy process, still lagging in evolution of fully fetched terrains when compared to previous runs 1-2 as shown in Figure 23 and 27.

5.4. Results

As mentioned earlier in this section, we show the elapsed time difference between the genetic algorithm and the evolutionary strategy process, both approaches are efficient while running and loads quick rendered terrains after parents are selected in next generation to create offspring. Figure 29 represents a graph where both the genetic algorithm and evolutionary strategy approach are considered for time to be calculated through there generations. The graph clearly shows that the genetic algorithm approach takes more time to generate the terrains as it considers the genetic process of crossover and mutation whereas, the evolutionary strategy approach takes less time as the crossover function is eliminated.

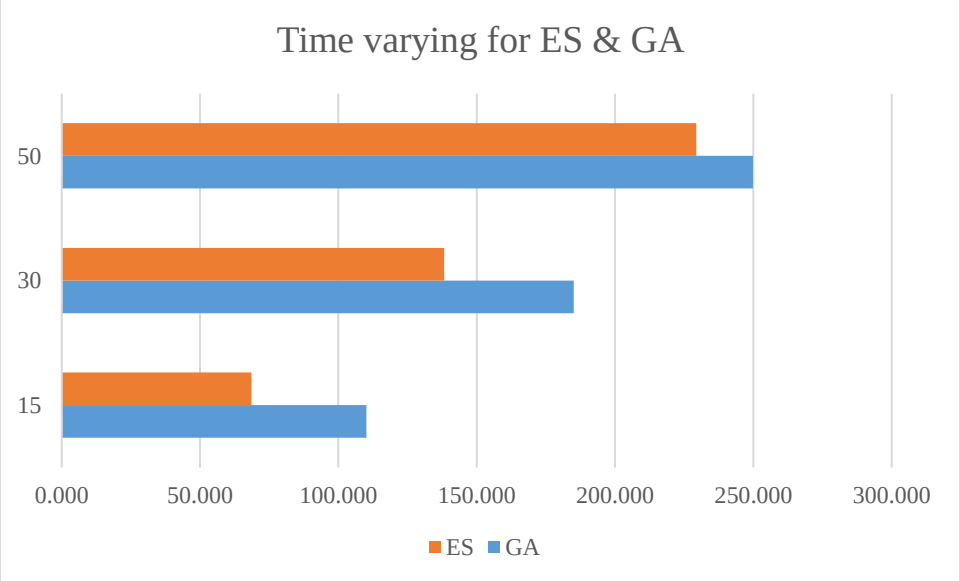


Figure 29. Graph representing elapsed time of evolutionary and genetic approach

6. CONCLUSION AND DISCUSSION

In this paper, we have used the approach of genetic algorithm and evolutionary strategy, which adds control to the process of procedural terrain generation applying it to the generation of virtual terrains. A patch-based mechanism for terrain generation was used to exercise more control over the user desired goals to generate terrains. The use of both gene selection and parent selection during interactive evolution gives the user better control over how patches can be affected by crossover and mutation. It also allows the patches to be selected in the next terrains and therefore the undesired patches can be removed in the gene selection process to obtain the desired terrains. We also provided stitching technique for height-map patches to produce seamless and featured 3D terrains. The combination of various parameters and modification of the values vary upon user desired goals. A point to be noted here is that even though both genetic algorithm and evolutionary strategy processes use the same parameters with the exception of crossover. High square number of patches leads to undesired patches and therefore leads to the evolution of unwanted features. However, with parent selection and gene selection process as shown in Figure 25, these unwanted patches can be removed. It is a time taking process since it involves the selection of gene to be required for further generations. On the other hand, the evolutionary strategy process does not have to go through gene selection process as it evolves similar or better terrains depending on the user selection. It also requires less time, which allows more generations to be run until the final terrain is selected.

The observation and analysis of terrains has exposed a few drawbacks in existing techniques, which we were able to identify. A graphical processing unit (GPU) could be more beneficial in enhancement of graphics such as display and motion control of the game. Fitness evaluation method for PTG should be placed on techniques that evaluate the map, based on user

interaction [9] [12]. Also, a metrics function can be introduced to allow for direct comparisons between terrain generation techniques. The terrains evolution process can be re-written in programming language such as C ++, by providing a plugin directly into DirectX graphics API or writing it completely in Unity.

REFERENCES

1. N. Stefano, and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology*. MIT Press Cambridge, MA, USA ©2000.
2. L. Davis. "Job Shop Scheduling with Genetic Algorithms." Paper presented at the Proceedings of the 1st International Conference on Genetic Algorithms, 1985.
3. J. H. Jensen, and P. C. Haddow. "Evolutionary Music Composition Based on Zipf's Law." Paper presented at the Proceedings of the 13th annual conference companion on Genetic and evolutionary computation, 2011.
4. J. Alonso. “Computational Methods in Aircraft Design.” (March 2002) http://adg.stanford.edu/aa241/design/compaero.html.
5. W. L. Raffe, F. Zambetta, X. Li, and K. O. Stanley. "Integrated Approach to Personalized Procedural Map Generation Using Evolutionary Algorithms." *IEEE Transactions on Computational Intelligence and AI in Games 7, no.* 2: 139-55, 2015.
6. S. Li, X. Liu, and E. Wu. "Feature-Based Visibility-Driven Clod for Terrain." Paper presented at the Proceedings of the 11th Pacific Conference on Computer Graphics and Applications, 2003.
7. Bungie Studios. 2004. *Halo 2* (software). Microsoft Corporation. Redmond, WA
8. J. Olsen. “Realtime procedural terrain generation,” Department of Mathematics and Computer Science (IMADA) University of Southern Denmark, 2004.
9. W. L. Raffe, F. Zambetta and X. Li, "A survey of procedural terrain generation techniques using evolutionary algorithms." *IEEE Congress on Evolutionary Computation, Brisbane, QLD, pp.* 1-8. doi: 10.1109, 2012.

10. R.L. Saunders, *Terrainosaurus: Realistic Terrain Synthesis Using Genetic Algorithms*. Master's thesis, Texas A&M University, 2007.

11. J. Togelius, M. Preuss, and G. N. Yannakakis. "Towards Multiobjective Procedural Map Generation." Paper presented at the Proceedings of the 2010 Workshop on Procedural Content Generation in Games, 2010.

12. W. L. Raffe, F. Zambetta and X. Li. "Evolving Patch-Based Terrains for Use in Video Games." Paper presented at the Proceedings of the 13th annual conference on Genetic and evolutionary computation, 2011.

13. M. Frade, F. Fernandez de Vega, and C. Cotta, "Breeding Terrains with Genetic Terrain Programming: The Evolution of Terrain Generators." *International Journal of Computer Games Technology*, vol. 2009, Article ID 125714, 13 pages, 2009.

14. T. Rastislav. *Generation and Visualization of Terrain in Virtual Environment*. Bachelor's thesis, Masaryk University, Faculty of Informatics, 2012.

15. Autodesk 3D design, Engineering & Entertainment Software. "Autodesk." (Feb 2014). http://www.autodesk.com/products

16. Product design collection, Included software, Trials, Autodesk. "Autodesk." (March 2008). http://www.autodesk.com/collections/product-design/included-software

17. Home of the blender project, Free and open 3d creation software. "Blender." (May 2016). https://www.blender.org/

18. The Living Planet for Amiga – Mobygames, SimEarth. "Maxis." (March 1992) http://www.mobygames.com/game/amiga/simearth-the-living-planet

19. A Home of Terragen Photorealistic 3d Environment Design and Rendering Software. "Planetside Software." (April, 2009). http://planetside.co.uk/

20. A Mapping Tool for Quake 3 Arena Video Game. “Activision.” (August, 2015). http://dmoztools.net/Games/Video_Games/Shooter/Q/Quake_Series/Quake_III_Arena/

21. H. Zhou, J. Sun, G. Turk and J. M. Rehg, "Terrain Synthesis from Digital Elevation Models," in *IEEE Transactions on Visualization and Computer Graphics, vol. 13, no. 4, pp.* 834-848, July-Aug. 2007.

22. W. L. Raffe. *Personalized Procedural Map Generation in Games Via Evolutionary Algorithms*. PhD Thesis, RMIT University, Melbourne, Australia, 2014.

23. J. Kruse, A. M. Connor. "Multi-Agent Evolutionary Systems for the Generation of Complex Virtual Worlds." *EAI Endorsed Transactions on Creative Technologies, 2015.*

24. T. Ong, R. Saunders, J. Keyser, and J. Leggett, “Terrain generation using genetic algorithms”. In Proceedings of the Genetic and Evolutionary Computation Conference, pages 1463–1470. ACM, 2005.

25. T. Bäck,F. Hoffmeister, H.P. Schwefel, “A Survey of Evolution Strategies”, “Proceedings of the Fourth International Conference on Genetic Algorithms”. 1991.

26. B. Xu, H. Lin, Z. Yang, K. B. Wagholikar and H. Liu, "Classifying protein complexes from candidate subgraphs using fuzzy machine learning model." *IEEE International Conference on Bioinformatics and Biomedicine Workshops, Philadelphia, PA, pp.* 640-647, 2012.

27. "Genetic Optimization of Artificial Neural Networks to Forecast Virioplankton Abundance from Cytometric Data." *Journal of Intelligent Learning Systems and Applications, 2013, 5, 57-66, 2013.*

28. European Environment Agency. *EIONET - GEMET Thesaurus.* March, 2012. https://www.eionet.europa.eu/gemet/