The Application of the Genetic Algorithm based on Abstract Data Type (GAADT) Model for the Adaptation of Scenarios of MMORPGs

Leonardo F. B. S. Carvalho, Helio C. Silva Neto,
Roberta V. V. Lopes, and Fábio Paraguaçu

Federal University of Alagoas (UFAL - *Universidade Federal de Alagoas*),
Institute of Computing (IC - *Instituto de Computação*),
Campus A. C. Simões, 57072-970 Maceió, Alagoas, Brazil
`{lfilipebsc,helio.hx,fabioparagua2000}@gmail.com,`
`rv2l@hotmail.com`

Abstract. The importance of using Artificial Intelligence in video games has grow in response to a need in showing behaviors and other game elements in a closer accuracy to what is seen at the real world. In order to assist this need, this paper takes advantage of the context of MMORPGs (game worlds with rich and interactive environments where important events occur simultaneously) to demonstrate the application of the artificial intelligence technique of the Genetic Algorithm based on Abstract Data Type (GAADT) in changing the features of game maps of MMORPGs due to the passage of time, in an attempt to reproduce what is seen in the real world.

Keywords: Artificial Intelligence, genetic algorithms, GAADT, games, MMORPG.

1 Introduction

The Genetic Algorithms (GA) [6] are Artificial Intelligence (AI) algorithms that employ the evolutionary principle of the survival of the fittest to solve algorithmic problems using an heuristic search based on metaphors combining principles of search algorithms and biology.

In videogames, GAs are often used when the implementation of AI involves must deal with problems that have many possible outcomes or lots of variables that ought to be accounted to achieve proper results [4]. In this respect, this paper presents an instantiation of a GA known as GAADT (Genetic Algorithm based on Abstract Data Type) to solve a problem that fits both these characteristics, which is: "to change the features of pre-existing MMORPG (Massively Online Role-Playing Game) game maps in order to create new maps".

The problem above has a great level of abstraction. Thus, it must be stated that the aim of this research is not to create new maps from scratch. In fact, the instance of GAADT that will be presented by this paper uses existing game maps as input and manipulates them to create new maps. Game maps may house various types of objects; however, only the type capable of representing the different geographical images that make up its scenario (known as tile) will be referred here.

After a game map is given as input to GAADT, the algorithm will start to create game maps that have some geographical features that are different from the ones of the input map, meaning that they have one or more group of cells whose tiles do not match the ones that are found in the input map for those specific locations. These differences occur in respect to the beginning of a season (spring, summer, autumn or winter) and to the unique geography originally depicted by each cell. However, it is not mandatory for a game map or its regions to have four properly defined seasons. Moreover, it is also unlikely that the transitional period from one season to another will ever repeat itself, which is a response to the fact that the values defining the extent of a season, its rainfall and even the time taken for switch between seasons are randomly chosen for each execution of GAADT.

The use of GAADT to change the topology/geographical features of scenarios of game maps - which was first shown in [3] - contrasts with other algorithms employed for this purpose due the fact that GAADT is an AI algorithm, while other identified approaches have a common link to computer graphics. Some of these approaches are listed in Table 1.

Paper	**Technique**	**Limitations**
[1]	Visualizations of dynamic terrains using a multiresolution algorithm with structural changes.	The algorithm is restrict to create a single kind of alteration in the game map (craters).
[7]	Multiresolution algorithm coupled with real-time optimally adapting meshes [5].	The model seems to apply only to real-time car driving in game maps simulating an off-road type of environment.
[10]	Use of multiresolution meshes to represent geometric objects. Each set of meshes corresponds to a different representation of a same object.	Unfit to systems requiring online updates.

Table 1: Different approaches for changing the geography of game maps

Now, this paper will be divided into different sections that better clarify the use of GAADT as solution to the problem of changing the geographical features of a game map. Following this section, Sec. 2 will detail the concepts of the adopted game maps. Next, Sec. 3 will present the concepts of GAADT and the instantiation used here. Last, Sec. 4 will show the results and conclusions drawn from this research.

2 Map Concepts

The game maps of this paper were created using an open-source third-part application known as Tiled [8]. Additional modules were created for this application in order to accomplish the objectives of the research. The relevant game map concepts are depicted in Figure 1 and described next.

- **Map**: a game world scenario consisting of a set of rectangular overlapping grids (known as layers) that have identical width (w_{map}) and height (h_{map}) values and are so that $\{w_{map}, h_{map} \in \mathbb{N} | (w_{map} > 0) \wedge (h_{map} > 0)\}$.
- **Layer**: a map grid that houses elements for building the map. A layer is located at a specific depth that is inferior to the l_{map} amount of layers of the map, hence $\{l_{map} \in \mathbb{N} | l_{map} > 0\}$. The width ($w_{map}$) and height ($h_{map}$) values of a layer indicates the number of cells it has in that direction. Each cell has the same width (w_{cel}) and height (h_{cel}) that are given as $\{w_{cel}, h_{cel} \in \mathbb{N} | (0 < w_{cel} \leq w_{map}) \wedge (0 < h_{cel} \leq h_{map})\}$. Thus, the game map is a cube whose elements are placed regarding their distance (x and y axis) and depth (z axis) from the map origin at $[0, 0, 0]$.
- **Tile**: a map object that occupies a single layer cell and has an image assigned to it that represents a geographical element. A tile has w_{cel} width and a h_{tl} height so that $\{h_{tl} \in \mathbb{N} | (h_{tl} = j \times h_{cel}) \wedge (1 \leq j < h_{map})\}$.

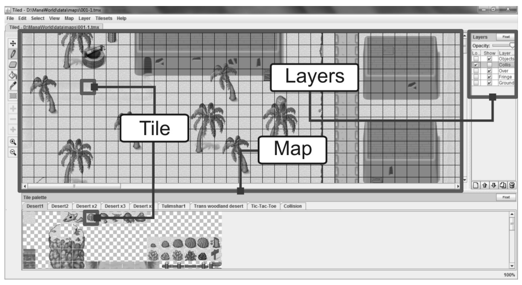


Fig. 1: Visual illustration of the game map concepts

3 GAADT's Instantiation to the Problem

As a detailed presentation of GAADT would be too long for the scope of this paper, this section will instead present the concepts of GAADT while simultaneously demonstrating its instantiation to the intended problem of the paper. Additional details of GAADT are available in [9].

3.1 Basic Types

Def. 1. Basis: the elementary genetic unit of chromosomes. Here, a basis is a $b = (x, y, z, tl, tp, in, fl)$ element that corresponds to a map cell at the x, y, z location that has the tl tile assigned to it, which is additionally identified by the tp description of its type of geography, by the unique index in of this description and by the fl flag that indicates if the tile of this cell requires a collision.

The bases of GAADT are part of a B set that is defined here according to a $DC = \{(x, y, z, tl, tp, in, fl) | (0 \leq x < w_{map}) \wedge (0 \leq y < h_{map}) \wedge (0 \leq z < \tilde{l}_{map}) \wedge (tl \in Tile) \wedge (in \in \mathbb{N}) \wedge (fl \in Boolean)\}$ set that has the $\tilde{l}_{map}$ value of the domain of the z coordinate as the amount of map layers intended exclusively for the allocation of tiles. The $Tile$ set of DC defines the domain of the tl tile objects and is so that $Tile = \{(w_{cel}, h_{tl}, gId, Img) | (Img \in Image) \wedge ((w_{cel}, h_{tl}, gId) \in \mathbb{N})\}$. The gId value is the unique index that identifies a tile.

Hence, the set of bases for this instance of GAADT is $B = DC \cup B_\lambda$. The B_λ set accords to GAADT's assumption that exists a b_λ innocuous basis whose data does not influence the identity of the gene that contains it. In turn, as this paper assumes that bases at different x, y, z coordinates are in fact different bases, the algorithm requires a $B_\lambda = \{b = (x, y, z, tl, tp, in, fl) | (tp = tp_\lambda)\}$ set of innocuous bases, whose elements have the same tp value that marks them as cell with no information.

The bases are arranged in groups in order to create genes. Each gene is a particular characteristic and thus, requires that its bases are connected in some way. For this reason, the creation of genes accords to a set of "formation rules" called Axioms for Formation of Genes (AFG).

Def. 2. Gene: a $g = \langle b_1, \ldots, b_n \rangle$ "micro region" of the game map that meets the requirements of the AFG set of rules.

$$\begin{aligned} AFG_1 = \{ \ & \forall g = \langle b_1, \ldots, b_n \rangle \in G \ \forall (b_\alpha = (x_\alpha, y_\alpha, z_\alpha, tl_\alpha, tp_\alpha, in_\alpha, fl_\alpha), \\ & b_\beta = (x_\beta, y_\beta, z_\beta, tl_\beta, tp_\beta, in_\beta, fl_\beta)) \in \{b_1, \ldots, b_n\} \mid z_\alpha = z_\beta\} \end{aligned} \tag{1}$$

$$\begin{aligned} AFG_2 = \{ \ & \forall g = \langle b_1, \ldots, b_n \rangle \in G \ \forall (b_\alpha = (x_\alpha, y_\alpha, z_\alpha, tl_\alpha, tp_\alpha, in_\alpha, fl_\alpha), \\ & b_\beta = (x_\beta, y_\beta, z_\beta, tl_\beta, tp_\beta, in_\beta, fl_\beta)) \in \{b_1, \ldots, b_n\} \mid \\ & (tp_\alpha = tp_\beta) \wedge (in_\alpha = in_\beta)\} \end{aligned} \tag{2}$$

$$\begin{aligned} AFG_3 = \{ \ & \forall g = \langle b_1, \ldots, b_n \rangle \in G \\ & \exists_1 (b_{x^+} = (x_{x^+}, y_{x^+}, z_{x^+}, tl_{x^+}, tp_{x^+}, in_{x^+}, fl_{x^+}), \\ & b_{x^-} = (x_{x^-}, y_{x^-}, z_{x^-}, tl_{x^-}, tp_{x^-}, in_{x^-}, fl_{x^-})) \in \{b_1, \ldots, b_n\} \\ & \forall b = (x, y, z, tl, tp, in, fl) \in \{b_1, \ldots, b_n\} \mid ((x \leq x_{x^+}) \wedge \\ & (x_{x^-} \leq x)) \wedge (0 < (x_{x^+} - x_{x^-} + 1) < (2 \times w_{mapa}/3))\} \end{aligned} \tag{3}$$

$$\begin{aligned}AFG_4 = \{\ & \forall g = \langle b_1, \ldots, b_n \rangle \in G \\ & \exists_1 \left(b_{y^+} = (x_{y^+}, y_{y^+}, z_{y^+}, tl_{y^+}, tp_{y^+}, in_{y^+}, fl_{y^+}),\right. \\ & \left. b_{y^-} = (x_{y^-}, y_{y^-}, z_{y^-}, tl_{y^-}, tp_{y^-}, in_{y^-}, fl_{y^-})\right) \in \{b_1, \ldots, b_n\} \\ & \forall b = (x, y, z, tl, tp, in, fl) \in \{b_1, \ldots, b_n\} \,|\, \left(\left(y \leq y_{y^+}\right) \wedge\right. \\ & \left.\left(y_{y^-} \leq y\right)\right) \wedge \left(0 < \left(y_{y^+} - y_{y^-} + 1\right) < (2 \times h_{mapa}/3)\right)\}\end{aligned} \tag{4}$$

The genes are arranged in groups in order to create chromosomes. Thus, each chromosome is an aggregation of characteristics and is a unique element that does not repeat itself within any present or future population of GAADT. The creation of chromosomes obeys its own set of "formation rules" known as Axioms for Formation of Chromosomes (AFC) that validates their sequences of genes. Figure 2 shows some examples of bases, genes and chromosomes that were obtained according to the defined B set and the stated AFG and AFC rules.

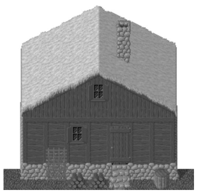
Def. 3. Chromosome: a $c = \{g_1, \ldots, g_n\}$ "macro region" of the game map that obeys the AFC set of rules.

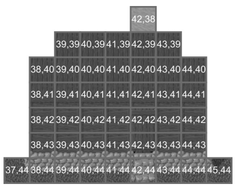
$$\begin{aligned}AFC_1 = \{\ & \forall\ c = \{g_1, g_2, \ldots, g_m\} \in C \\ & \forall \left(g_\alpha = \langle b_1^\alpha, \ldots, b_{n_1}^\alpha \rangle, g_\beta = \left\langle b_1^\beta, \ldots, b_{n_2}^\beta \right\rangle\right) \in \{g_1, g_2, \ldots, g_m\} \\ & \forall b_i = (x_i, y_i, z_i, tl_i, tp_i, in_i, fl_i) \in \left\{b_1^\alpha, \ldots, b_{n_1}^\alpha\right\} \\ & \forall b_j = (x_j, y_j, z_j, tl_j, tp_j, in_j, fl_j) \in \left\{b_1^\beta, \ldots, b_{n_2}^\beta\right\} \,| \\ & (x_i \neq x_j) \vee (y_i \neq y_j) \vee (z_i \neq z_j)\ \}\end{aligned} \tag{5}$$

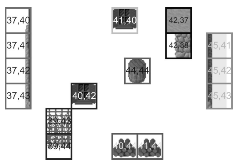
$$\begin{aligned}AFC_2 = \{\ & \forall c = \{g_1, g_2, \ldots, g_m\} \in C\ \forall (g = \langle b_1, \ldots, b_n \rangle) \in \{g_1, g_2, \ldots, \\ & g_m\}\ \exists b = (x, y, z, tl, tp, in, fl) \in \{b_1, \ldots, b_n\} \,|\, z = 0\ \}\end{aligned} \tag{6}$$

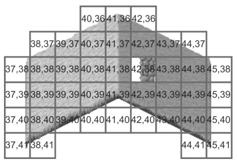
$$\begin{aligned}AFC_3 = \{\ & \forall\ c = \{g_1, g_2, \ldots, g_m\} \in C\ \forall \left(g_\alpha = \langle b_1^\alpha, \ldots, b_{n_1}^\alpha \rangle,\right. \\ & \left. g_\beta = \left\langle b_1^\beta, \ldots, b_{n_2}^\beta \right\rangle\right) \in \{g_1, g_2, \ldots, g_m\}\ \forall b_i = (x_i, y_i, z_i, tl_i, \\ & tp_i, in_i, fl_i) \in \left\{b_1^\alpha, \ldots, b_{n_1}^\alpha\right\}\ \forall b_j = (x_j, y_j, z_j, tl_j, tp_j, in_j, fl_j) \\ & \in \left\{b_1^\beta, \ldots, b_{n_2}^\beta\right\} \,|\, (z_i \geq z_j) \wedge (\forall z \in \{0, 1, \ldots, z_i\} \\ & \exists \left(g_\gamma = \left\langle b_1^\gamma, \ldots, b_{n_\gamma}^\gamma \right\rangle\right) \in (\{g_1, g_2, \ldots, g_m\} - \{g_\beta\})\right) \wedge \\ & (\forall b = (x_b, y_b, z_b, tl_b, tp_b, in_b, fl_b) \in \left\{b_1^\gamma, \ldots, b_{n_\gamma}^\gamma\right\} (z_b = z))\ \}\end{aligned} \tag{7}$$

GAADT performs its operations according to cycle of evolutionary and stagnant periods that assumes that environmental changes mark the beginning of new evolutionary periods and arranges its chromosomes in groups known as populations. Every chromosome is a potential result for the problem that the algorithm is aiming to solve and, as a consequence, it assumes that the creation of an empty population reflects an incorrect assumption of the problem, as it would mean that the intended problem does not have a solution.


(a) Chromosome


(b) Gene at the $z = 0$ layer and its x, y locations


(c) Genes at the $z = 1$ layer and their x, y locations


(d) Gene at the $z = 2$ layer and its x, y locations

(e) Gene at the $z = 3$ layer and its x, y locations

(f) A chromosome of a single gene (at $z = 0$) and a single basis

Fig. 2: Example of chromosomes with their genes along the x, y and z axes and with the bases standing as the genes' individual cells

Each loop of the cycle is known as a generation and spawns new chromosomes to GAADT's current population at the same time that removes from it the chromosomes it considers "unfit". In time, GAADT will start to converge and create populations that are increasingly similar to the one of the previous generation, until they become indistinguishable. This indicates that GAADT has reached a stagnant period, which will come to an end whenever a new environmental change occurs, forcing the evolutionary-stagnant process to repeat itself.

3.2 Genetic Operators

The genetic operators are responsible for creating new chromosomes for GAADT's populations. There are two operators that perform this task: the reproduction operator and the mutation operator.

The reproduction operator used here is a crossover operation that combines genes of two different chromosomes (parent-chromosomes) in order to create new chromosomes (child-chromosomes), while the mutation operator receives an input chromosome and switch some of its genes by new ones, thus, creating new chromosomes (mutant-chromosomes). The value that indicates the suitability of a chromosome to its environment is known as adaptation value (or fitness), which depends of the *degree* value of its genes that indicates the suitability of a gene to its environment.

Def. 4. Degree: the degree of a gene is a $degree : G \to \mathbb{Q}_+$ function mapping a gene to a k value known as $degree(g)$ that is so that $k \in \mathbb{Q}_+$. The $degree(g)$

function reflects a comparative stratification of the g gene suitability to its environment in respect to GAADT's intended problem.

Due the level of abstraction of the problem intended here, the *degree* function shown in (8) uses the $distri(g)$ function to exam how the micro-region of the game map corresponding to the g input gene match its surroundings. Equation (9) shows the $distri(g)$ function.

The $distri(g)$ function uses the $whiteInGene(g)$ function that combines all images within the tiles of the bases of its input gene into a single image and subject it to Canny's edge detection algorithm [2], creating a black and white image that outlines its geography (black background and white contours) and returning its amount of white pixels, which is used by (9) as an indicator of the correct placement of its geographic elements.

$$degree(g = \langle b_1, \ldots, b_n \rangle) = \begin{cases} \frac{distri(g)}{\#g} & \text{if } (pass > 1) \wedge (\exists_1 tp_{pass} \in INT\ \forall b = (x, y, z, tl, tp, in, fl) \in \{b_1, \ldots, b_n\} \mid tp = tp_{pass}) \\ 1 - \frac{distri(g)}{\#g} & \text{if } (pass = 1) \wedge (\exists_1 tp_{pass} \in INT\ \forall b = (x, y, z, tl, tp, in, fl) \in \{b_1, \ldots, b_n\} \mid tp = tp_{pass}) \\ 0 & \text{if } g \in G_\lambda \\ 1 \times 10^{-10} & \text{otherwise} \end{cases} \tag{8}$$

$$distri\,(g) = \begin{cases} 1 - \left(\frac{1}{\#g \times w_{cel} \times h_{cel}}\right) & \text{if } whiteInGene(g) = 0 \\ 1 - \left(\frac{whiteInGene(g)}{\#g \times w_{cel} \times h_{cel}}\right) & \text{otherwise} \end{cases} \tag{9}$$

In addition to $distri(g)$, (8) uses the $INT = \langle tp_1, \ldots, tp_n \rangle$ sequence whose tp types of geography are mapped accordingly to the predominant type of environment of GAADT's input map and to its randomly chosen season. The tp values of INT are the same elements presented in Def. 1 and their position within INT denotes their level of importance to the geography that GAADT expects to achieve for the game map. Therefore, tp_1 is the most important type of geography for the intended game map while tp_n is the least important one. Each call to $distri(g)$ will randomly choose a $pass$ value, $\{pass \in \mathbb{N}^* \mid pass \leq \#INT\}$, that will indicate the type of geography that will be taken into account (tp_{pass}). The G_λ set is the set of innocuous-genes, a subset of GAADT's G set of genes whose elements have no influence for the identity of a chromosome and that, for this instance of GAADT, corresponds to (10).

$$G_\lambda = \{g_\lambda = \langle b_1^{g_\lambda}, \ldots, b_n^{g_\lambda} \rangle \mid \forall\ i \in \{1, 2, \ldots, n\}\ (b_i^{g_\lambda} \in B_\lambda)\} \tag{10}$$

Def. 5. Adaptation: the adaptation of a chromosome (or fitness) is a function $adapt : C \to \mathbb{Q}_+$ that sums the degree of its genes and is defined here as $adapt(c = \{g_1, \ldots, g_n\}) = \sum_{i=1}^{n} (\Theta\,(c, g_i) \times degree(g_i))$.

The Θ function above is a weight function that maps a gene into a rational value by identifying how much of a gene's *degree* contributes to the chromosome's adaptation. Here, the weight function merely accounts if the tp value within the bases of a gene is also an element of the INT sequence, resulting in 1 if $tp \in INT$ and 0 otherwise. The *adapt* function also makes possible to calculate the average adaptation of a population of chromosomes, which is so that $adapt_m(P = \{c_1, \ldots, c_n\}) = (\sum_{i=1}^{n} adapt(c_i))/\#P$.

The value of adaptation of a chromosome makes possible to select chromosomes for specific purposes from any of GAADT's populations. One such case is the selection of chromosomes better fitted to create new chromosomes by subjection to genetic operators. The first of these operators that will be shown here is the reproduction operator by crossover of chromosomes, which heavily relays on the selection operator.

Def. 6. Selection: an operator that chooses from a population the chromosomes better satisfying an r predicate from GAADT's Rq set of requirements. Hence, $sel(P_1, r) = P_1 \cap r$.

The r predicate used here is shown in (11), which has P_2 as the population of parents-chromosomes that suit r and INT_r as a subsequence of INT set at running time so that $INT_r = \{\langle tp_1^r, \ldots, tp_n^r \rangle \,|\forall i \in \{1, 2, \ldots, n\}\ \forall j \in \{1, 2, \ldots, m\}\ \exists tp_j \in INT(i \leq j) \wedge (j = i + (m - n)) \wedge (tp_i^r = tp_j)\}$.

$$\begin{aligned} r = \{P_2|\ & (P_2 \subseteq C) \wedge (c = \{g_1, \ldots, g_n\} \in P_2) \leftrightarrow (\forall b = \\ & (x, y, z, tl, tp, in, fl) \in \{g_1, \ldots, g_n\}\ \exists tp^r \in INT_r(tp = tp^r))\ \} \end{aligned} \tag{11}$$

The P_2 resulting set of chromosomes of (11) may be further split into distinct $MALE$ and $FEMALE$ by applying their equivalent predicates over P_2. Therefore, assuming the predicates $M, F \in Rq$, follows that $MALE = sel(P_2; M)$ and $FEMALE = sel(P_2; F)$. These two new sets of chromosomes are used by the fecundation operation of (12).

$$\begin{aligned} fec(c^M, c^F) = \{\{g_1, \ldots, g_m\} \,|\, & (c^M = \{g_1^M, \ldots, g_l^M\} \in MALE) \wedge \\ & (c^F = \{g_1^F, \ldots, g_n^F\} \in FEMALE) \wedge \\ & (g \subseteq \{g_1^M, \ldots, g_l^M\} \cup \{g_1^F, \ldots, g_n^F\}) \leftrightarrow \\ & (\forall g^M \in \{g_1^M, \ldots, g_l^M\}\ \forall g^F \in \{g_1^F, \ldots, g_n^F\} \\ & (domi(g^M, g^F) = g))\ \} \end{aligned} \tag{12}$$

The *domi* function of (12) receives two input genes and returns their dominant one. A g_1 gene is dominant over a g_2 gene if both express the same characteristic and $degree(g_1) \geq degree(g_2)$. Therefore, if $degree(g_2) > degree(g_1)$ the *domi* function returns g_2. Additionally, if g_1 and g_2 do not express the same characteristic *domi* returns a g_λ innocuous-gene. Here, g_1 and g_2 express the same characteristic if $\{\exists_1 tp \in INT|\ \forall g_1, g_2 \in G\ \forall i, j \in \mathbb{N}^*\ \forall b_i = (x_i, y_i, z_i, tl_i, tp_i, in_i, fl_i) \in g_1\ \forall b_j = (x_j, y_j, z_j, tl_j, tp_j, in_j, fl_j) \in g_2((i \leq \#g_1) \wedge (j \leq \#g_2) \wedge (tp_i = tp_j = tp))\}$.

Def. 7. Crossover: the crossover is a reproduction operation defined by a $cross : MALE \times FEMALE \rightarrow P$ function, which is so that $cross(c_1, c_2) = \{c = \{g_1, \ldots, g_n\} |\ c \subseteq fec(c_1, c_2)\}$.

The other genetic operator for creating new chromosomes is the mutation operator. It acts as a safeguard that prevents the removal of chromosomes having some characteristics suitable to GAADT's current environment, but that do not meet the minimal adaptation value required to be kept in the current population. Once a chromosome is mutated its adaptation value is calculated again.

Def. 8. Mutation: an operator that replaces some of the genes of its input chromosome. It is defined by a $mut \subseteq \wp(P)$ predicate corresponding to the function $mut(c) = \{c' | (\exists G_1, G_2 \subset G) \rightarrow (c' = change(c, G_1, G_2)) \wedge (c' \in cut(P))\}$.

The *change* function of the definition above is of type $change : C \times \wp(G) \times \wp(G) \rightarrow C$ and is defined here as (13). Additionally, the *cut* function of *mut* is an acceptance criterion of the *Rq* set of requirements, $cut \in Rq$, that is imposed to every chromosome subject to mutation and that is defined here as $cut(P) = \{C_\Delta |\ \forall c \in C((c \in C_\Delta) \leftrightarrow (adapt(c) \geq adapt_m(P)))\}$.

$$change(c, G_1, G_2) = \begin{cases} (c \cup G_1) - G_2 & \text{if } (c \cup G_1 \in AFC) \wedge (c - G_2 \in AFC) \\ c \cup G_1 & \text{if } (c \cup G_1 \in AFC) \wedge (c - G_2 \notin AFC) \\ c - G_2 & \text{if } (c \cup G_1 \notin AFC) \wedge (c - G_2 \in AFC) \\ c & \text{if } (c \cup G_1 \notin AFC) \wedge (c - G_2 \notin AFC) \end{cases} \tag{13}$$

3.3 The Algorithm

The populations of GAADT are part of an environment defined by the 8-tuple $E = \langle P, \wp(P), Rq, AFG, AFC, Tx, \Sigma, P_0 \rangle$. The P value is the current population of GAADT and $\#P \in \mathbb{N}^*$. Simultaneously, $\wp(P)$ is the power set of P; Rq is the set of environmental requirements guiding the evolution of chromosomes; AFG and AFC are, respectively, the sets of Axioms for Formation of Genes and of Chromosomes; Tx is the taxonomic classification of the chromosomes of P, which prevents them from occurring multiple times within a population and of resurging on any future population; Σ is the set of genetic operator; and P_0 is the initial population.

GAADT itself is established as the function $GAADT : E \rightarrow E$ shown at (14) and that assumes $P_t \subseteq C$ as its input population and $P_{t+1} \subseteq C$ as the next population of the algorithm, which contains the chromosomes that were produced from P_t as result of the environmental requirements imposed by the Rq set and the genetic operators of Σ. Consequently, $P_{t+1} = cross(a, b) \cup mut(c) \cup p_{cut}(P_t)$ and $a, b, c \in P_t$.

$$GAADT(P_t) = \begin{cases} P_{optm} & \text{if } P_{optm} = \{c |\ \forall c \in P_t(adapt(c) \geq k)\} \neq \varnothing; \\ P_{t+1} & \text{if } t + 2 = T; \\ GAADT(P_{t+1}) & \text{otherwise} \end{cases} \tag{14}$$

The Rq set of requirements of GAADT also contains two stopping criteria to halt its execution. The first of these criteria is defined in respect to what the instantiation of GAADT is considering as an optimal solution to its intended problem and is presented in (14) as definition of the minimal adaptation value k that must be met by the chromosomes of the P_{optm} optimal population. The last criterion establishes a maximum number of interactions (generations) that GAADT can perform before returning a result. In (14) this corresponds to the defined T value that is so that $T \in \mathbb{N}$. This last criterion ensures that GAADT's evolutionary process will eventually stop and therefore, is computable.

4 Results and Conclusions

Figure 3 shows a map provided as input to GAADT (Fig. 3a) and the map that results of its evolutionary process (Fig. 3b). The map area of Fig. 3b that diverges from Fig. 3a corresponds to the chromosome of greatest adaptation value that exists within the last population created by GAADT before the halt of its process. In this particular example, a forest input map (Fig. 3a) was subject to a period of drought that was caused by an extend summer and began to acquire geographic features commonly seen in deserts (Fig. 3b).

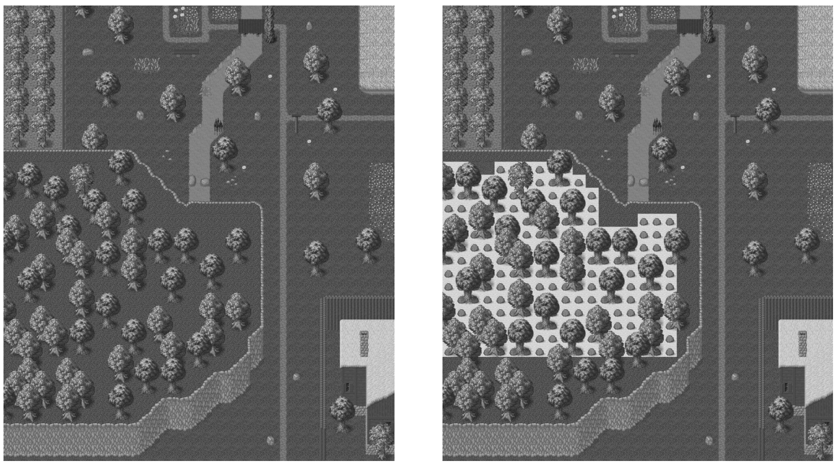

(a) Input map (forest) (b) Output map (desert)

Fig. 3: An input map and the output map created from it by GAADT

The number of interactions performed by GAADT to create the map of Fig. 3b is shown in Table 2, along with the chromosomes of greater and lower adaptation value and the average adaptation of the population of each generation. Table 2 also makes possible to identify stagnant periods of GAADT's evolutionary process, which correspond to the generations presenting identical values for the greatest, lowest and average adaptation of its chromosomes (generations 3 to 5 and 6 to 7). After seven generations with no changes in any of these three values, the evolutionary process of this instance of GAADT is halted.

Results			
generation	greatest adaptation	lowest adaptation	average adaptation
1	0,924541016	0,00302477	0,572786602
2	0,946289063	0,00302477	0,611779318
3	0,9921875	0,00302477	0,63175816
4	0,9921875	0,00302477	0,63175816
5	0,9921875	0,00302477	0,63175816
6	0,993339978	0,00302477	0,672413871
7	0,993339978	0,00302477	0,672413871
8	0,996235983	0,01467041	0,713797671
9	0,996235983	0,01467041	0,713797671
10	0,996235983	0,01467041	0,713797671
11	0,996235983	0,01467041	0,713797671
12	0,996235983	0,01467041	0,713797671
13	0,996235983	0,01467041	0,713797671
14	0,996235983	0,01467041	0,713797671

Table 2: Adaptation values for the chromosomes of GAADT in creating the map of Fig. 3b

The result seen in Fig.3b and achieved by the process detailed in Table 2 shows that even thought the use of GAADT contrasts with the traditional use of computer graphics to modify the geography of game maps, it is a valid approach that provides resulting maps suitable to MMORPGs and that add an extra element of dynamic that players can explore.

However, it is the authors' believe that such results can be further improved by refining the parameters of the instance of the algorithm in order to achieve an even better distribution of the geographic elements of the resulting maps.

An identified drawback is the time required by the algorithm to obtain a resulting map. In this respect, the map shown in Fig. 3b took roughly 10 hours to be created from the map of Fig. 3a. This is a consequence of GAADT using its Σ set of genetic operators to create all possible variations of the input map that obey the restrictions imposed by the Rq set of requirements.

To solve this drawback, future approaches to this problem will focus on either implementing the GAADT instance seen here in hardware in order to reduce the processing time by taking advantage of its higher processing speeds; or, refactoring the instance of GAADT as to improve its execution in addition to generalize it so it become applicable for game maps whose structure is different from that of the game maps used here.

References

1. Cai, X., Li, F., Sun, H., Zhan, S.: Research of Dynamic Terrain in Complex Battlefield Environments. In: Technologies for E-Learning and Digital Entertainment, Lecture Notes in Computer Science, vol. 3942. Springer Berlin / Heidelberg (2006)
2. Canny, J.: A Computational Approach to Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8(6) (1986)
3. Carvalho, L.F.B.S., Neto, H.C.S., Lopes, R.V.V., Paraguau, F.: Application of a genetic algorithm based on abstract data type in electronic games. In: Proceedings of the 9th Mexican International Conference on Artificial Intelligence - Special Session - (MICAI 2010). pp. 28–33. IEEE Computer Society Press (2010)
4. Carvalho, L.F.B.S.: An Evolutive game of Checkers. End of course work, Institute of Computation, Federal University of Alagoas, Brazil (2008)
5. Duchaineau, M., Wolinsky, M., Sigeti, D., Miller, M., Aldrich, C., Mineev-Weinstein, M.: Roaming terrain: Real-time optimally adapting meshes. In: Proceedings of the conference on Visualization '97 (1997)
6. GOLDBERG, D.E., HOLLAND, J.H.: Genetic algorithms and machine learning. Machine Learning 3(2-3), 95–99 (1988)
7. He, Y., Cremer, J., Papelis, Y.: Real-time extendible-resolution display of on-line dynamic terrain. In: Proceedings of The 2002 Conference on Graphics Interface (2002)
8. Lindeijer, T., Turk, A.: Tiled Map Editor. Available at: `http://www.mapeditor.org/` (2006-2011), last accessed on December 19 2010
9. Lopes, R.V.V.: A Genetic Algorithm Based on Abstract Data Type and its Specification in Z. Ph.D. thesis, Computer Center, Federal University of Pernambuco (2003)
10. Shamir, A., Pascucci, V., Bajaj, C.L.: Multiresolution dynamic meshes with arbitrary deformations. In: Proceedings of IEEE Visualization 2000 (2000)