www.cadanda.com
Computer-Aided Design AND APPLICATIONS

A CAD Path and Motion Planning System for Flexible Agents using a Genetic Algorithm

Miri Weiss Cohen and Sergei Kovalchuk

Braude College of Engineering, miri@braude.ac.il, sergeikov95@gmail.com

Corresponding author: Miri Weiss Cohen, miri@braude.ac.il

Abstract. This paper proposes an approach to path and motion planning for agents in 3D virtual environments. This type of planning must consider terrain settings, cost and agent features. The problem is to find the least expensive path while adapting the path to the environmental constraints. The proposed system is based on a parallel approach that simultaneously considers all three parameters in the search procedure. It uses a genetic algorithm with flexible parameters and variables adapted to different environments and agents while conforming to time and cost constraints. After defining the approach, the research also analyzes the GA model, its parameters and the defined optimization criteria and variables for solution efficiency. The proposed system attained good results and provides adequate solutions for various cases.

Keywords: Path and Motion planning, Virtual worlds, Genetic Algorithm, Flexible Agents
DOI: https://doi.org/10.14733/cadaps.2020.1143-1156

1 INTRODUCTION

Path and motion planning for various agents in virtual environments plays an important role in a large variety of applications, ranging from special effects in creative movies, interactive games and animated simulations [15]. For efficient and effective path and motion planning, planners must consider a number of factors, among them the nature of the environment itself and the specific agent. They must also seek to minimize costs.

Virtual environments vary widely and can range from static smooth environments to dynamic discontinuous constrained terrain. The problem is to find the least expensive path from point A (Start) to point B (Goal), while adapting it to the environmental constraints. Hence, planners must have the capability to adapt flexible path and motion to various terrain settings, moreover, the nature of the agent. Agents differ in size, gait and step features, thus requiring different path and motion planning considerations. For example, an agent representing David would follow a different path and move differently than an agent representing Goliath, especially in the case of discontinuous terrain. Another aspect of motion planning is cost, where energy=cost.

Different steps, even for the same agent, have different defined costs and thus consume a different amount of energy. This motion criteria tries to realistically mimic human or any other agents. Taking into account the fact, that in reality human gait motion is performed with optimal energy consumption.

The above constraints indicate that a composite solution is required for path and motion planning. Considering only one factor without the others would yield a solution that is not adaptive. More precisely, the resulting path would not meet the motion planning and minimal cost requirements and in some cases would not be realizable for the particular agent.

In this study, we propose a new approach that incorporates the following three factors: motion planning, the specific nature of each agent and flexibility of defining a different cost to each step. This parallel approach considers all three parameters simultaneously in the search procedure. The goal is implemented by means of a genetic algorithm (GA) that incorporates all the above factors while searching for the best solution. Because path and motion planning is an integral part of the decision making systems, it must be time efficient, while at the same time searching for the best solution. The proposed approach meets both these constraints.

Figure 1 illustrates the problem. In the figure, the goal appears as an orange disk close to the viewer, and three possible paths and motions lead to the goal. Paths A and B reach the goal because the agents steps are sufficiently large to overcome the discontinuity in the terrain constraint, while Path C is unable to reach the goal. Although the agents moving along Path A and Path B both reach the goal, these paths may have different costs as the agents steps are defined differently in terms of cost. For example, in both cases, the paths may have the same length but the agents steps are combined differently, resulting in different costs. The proposed system is sufficiently flexible to handle a wide range of cases. It provides the user a large range of capabilities and parameters for defining agents and terrains.

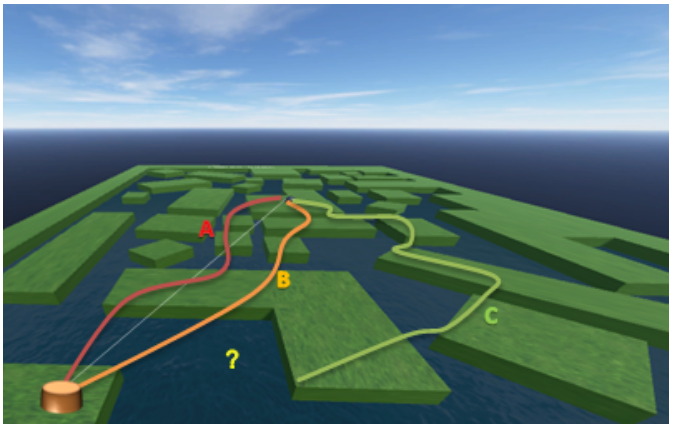

Figure 1: Illustration of the problem definition.

In this work our contributions are threefold:

- Introduces and implements a flexible system that considers all the above factors in generating realistic path and motion planning.
- Examines and researches the GA parameters influencing the solution space and results
- Explores the use of flexible parameters in portraying agents .

2 RELATED WORK

Creating realistic paths and motions for agents has applications ranging from the special effects industry to interactive games [18] and simulations. Gamers often refer to this process as "pathing" [5]. This important

problem has been the topic of extensive research. Among the early studies in this field are Bruderlin [2] and Badler et al. [1]. Bruderlin et al. proposed a new approach for path finding in video games named Direction Oriented Path finding (DOP) and compared this approach to widely used solutions such as that of Dijkstra, $A^{\star}$, visibility graph and navigation mesh. DOP is based on generating an n-sided poly-based navigation mesh, finding the shortest channel composed of a set of adjacent polygons to find the shortest path inside the channel. Kuffner et al. [12] described both path planning and manipulative planning for agents by introducing a two-stage approach that involved first finding the path in 2D using $A^{\star}$ and then using a velocity model to generate motions.

Balder and Liu [14] extended their work by using hardware acceleration that involves combining spatial search with inverse kinematics and collision detection. Lui and Kuffner [13] introduce a Finite-state machine (FSM) to define the movement capabilities of a particular agent was presented in while incorporating $A^{\star}$ searching. Given the motion trajectories of all the moving objects, the search algorithm used two interrelated data structures: First, a tree with nodes that record explored states in the FSM, which continually expanded during the search; Second, a priority queue of FSM states ordered according to cost. These states represent potential nodes to be expanded during the next search iteration. Gleicher et al. [7] describe a method which entails building a data structure of motion clips and then replaying these clips to generate motion. The motion graph serves as a policy to specify a motion, represented by a sequence of transformations, for the CG model skeleton in accordance with its state in space. Dynamic Animation of Human Walking is described in [5]. The authors propose the Key frame-Less animation of walking (KLAW) system which represents a hybrid approach to the animation of human movements that combines goal-directed and dynamic motion control. The approach operates at three levels: conceptual abstraction (high level), gait refinements (middle level) and physical abstraction (low level). Jaklin et al. [9] describe a system for differential virtual environment preference path finding known as Modified Indicative Routes and Navigation (MIRAN).

A different approach was presented by Kenwright [11], where animated human movements were represented in the form of a low-dimensional rigid body comprising joint torques that produce worm like movements. While this method facilitates creative movements, it lacks characteristic steps and is not flexible in representing constraints. A more recent approach by Zhang et al. [24] also proposed combining $A^{\star}$ search strategy and poster adaptation in 3D, converting the path result into motion clips and testing for the lowest value of connection. The work presented by Juarez et al. [10] outlined three cases of clips: frontal walking, arm constraint frontal walking and lateral walking. The authors defined a set of behaviors and adapted these to the correct movement using density-driven triangulation of the environment. Superheros animation for unrealistic human agent abilities were introduced by [3]. Recent surveys on robot path planning are detailed in [16], [23], [21], the authors provide a wide range of methodologies for solving path planning trajectories with or without constraints. the surveys provide a detailed comparison on complexity and adaptation to specific environment. The path planning methodologies do not incorporate adaptive motion possibilities and path planning optimizations in the same time.

3 PROPOSED APPROACH

The motivation for using Genetic Algorithm (GA) is its advantage as a stochastic search algorithm and a problem-solving methodology [17, 8]. It has advantages such as flexibility, adaptation, global search capability, and its suitability for parallel computation.

GAs have been used to solve difficult problems with objective functions that are multi-model and multi-constrained [19]. A wide range of genetic operators can be used to perform the crossover, mutations, and reinsertion of potential solutions, among others, to generate the offspring population [22]. Some encoding may cause the algorithm to create unfeasible solutions, or may change and enlarge the search space, making it difficult to converge[6].

The following algorithm represents the basic stages in finding the best path and motions. The algorithm is self-explanatory and provides a good overview of the major stages.

Algorithm 1 - GA Path and Motion Planner

Input: initial position S_{pos}, goal position G_{pos} , list of possible movements $M_{1\cdots n}$.
Output: path Solution between S_{pos} and G_{pos} that consists of a sequence of movements from $M_{1\cdots n}$,and angles.

```
1: procedure GAPATH(S_pos, G_pos, M_1...n)
2:     Population ← GenRandPop(M_1...n)
3:     while pathFound == false do                ▷ Run generation until pathfound RunXGens
4:         for all Path p in Population do                    ▷ Execute sequence angles of path
5:             RunPath(p, S_pos)                                        ▷ E_pos path coordinates
6:             p.RemainigDistance = Vector3.Distance(G_pos)
7:             for i = 1:pathLength do
8:                 p.PathEnergy += p.step[i].Energy
9:             end for                ▷ CollisionPenatly calculated RunPath if collision in path
10:            p.PathEnergy += p.CollisionPenalty
11:            if p.remainingDistance < 1 & p.PathEnergy < Threshold &
    p.fitness < bestFitness then
12:                pathFound = true
13:                Solution = p
14:            end if
15:            Population = GenNewPop(Population,S_pos, G_pos)
16:        end for
17:        newSolution = RunGenerations(Population)
18:    end while
19: end procedure
```

3.1 Solution encoding

The first stage entailed encoding the solution. The goal in this encoding is to calculate the type of steps (allowing repetition) and angles necessary for the animated character to reach its destination. In order to solve this non-trivial NP-hard problem, the solutions were encoded into the chromosomes of the genetic algorithm using the value encoding method, where the values for the chromosome strings take the form of pairs (#Step, angle).

(#Step,Angle)	(#1, 27)	(#0,0)	$\cdots$	(#6,10)

Since such a problem can be very complex and its solution can be quite lengthy, a long chromosome with NULL movements (Do nothing in step 0, with zero angle represented by (#0, 0)), is defined. A path is made up of steps, where each Step S_i has its own energy cost Si_{Energy}, calculated according the data values assigned to the motion of the step. The steps are flexible and the user can define them easily using the systems' GUI. In the following example, the step options incorporate five different steps (to be chosen by the algorithm).

The Si_{Energy} costs are modeled in the following manner:

#Step i	Si_{length}	Si_{Energy}
1	0.25m	X
2	0.50m	2x-ϵ
3	0.75m	3x-ϵ
4	1.00m	4x-ϵ
5	1.50m	6x-ϵ

3.2 Evaluation function

The second stage was to determine the fittnes function. This requires calculating the general cost of the path (PathEnergy) by summing all the steps in the path as indicated by the following:

$$PathEnergy = \sum_{s_i \in Path-Steps} Si_{Energy} \tag{1}$$

Moreover, the way in which the parameter Si_{Energy} is calculated must discourage multiple step repetitions for the solution. This is based on the logic that if the solution steps are repeated more often, the amount of energy consumed will be higher since ϵ will be subtracted fewer times.

$$Si_{Energy} = \frac{Si_{Length}}{Si_{Length}} \cdot Sl_{Energy} - \epsilon \tag{2}$$

A path of length *2m* can be solved either as *2xS4* or as *8xS1*. Based on the way in which the variable Si_{Energy} is defined, an energy path of 2x$S4_{Energy}$ yields better results.

For every path, the variable `PathRemainingDistance` is saved. This variable indicates the distance from the goal after the selected path has been followed.

Each solution is evaluated by the `PathScore` of the calculated path. `PathScore` comprises the variables `PathEnergy` and `PathRemainingDistance`, each of which has a weight factor to determine how it influences fitness.

$$PathScore = \frac{1}{LenFactor \cdot (PathRemainigDistance + 1) + EnergyFactor \cdot PathEnergy} \tag{3}$$

where:
`LenFactor` - defines the importance of the `PathRemainingDistance` parameter to the fitness. As `LenFactor` becomes larger, `PathRemainingDistance` becomes more significant in calculating the fitness. These leads to the selection of solutions that arrive closer to the target at the expense of path efficiency considerations.
`EnergyFactor` - determines the significance of `PathEnergy`, that is, the importance of path optimality.

These two factors help users control their preferred method of searching for the path. By increasing `LenFactor` relative to `EnergyFactor`, they assign less importance to path optimality. The path with the largest `PathScore` is selected as the best solution for that specific generation. The evaluation variable `PathScore` takes the `PathRemainingDistance` parameter into consideration in order to compare both complete and incomplete paths, while assigning a lower (better) score to complete paths. By changing the `LenFactor` and `EnergyFactor` parameters, the user can affect the solution selection

In certain situations, such as that of a map with no direct routes to the goal, by increasing the `LenFactor` the user increases the significance of complete paths, thus assigning them a higher fitness score for selection. By means of the fitness function, the user can determine the significance of each part of the equation.

4 GENETIC ALGORITHM PARAMETER EVALUATION

In this stage of the study we examined the quality of the solution for various world maps based on the values of LenFactor and EnergyFactor. This stage was performed to evaluate relevant parameters for concluding the best values for serving best results [20, 4]. Accurate testing of the influence of these parameters involved setting a fixed value for all other parameters in the algorithm. The following were set: Length of a chromosome - PathLength = 30, Step energy control EpsilonMov = 0.1, Population size = 1000, Crossover rate = 0.9, Mutation rate = 0.03.

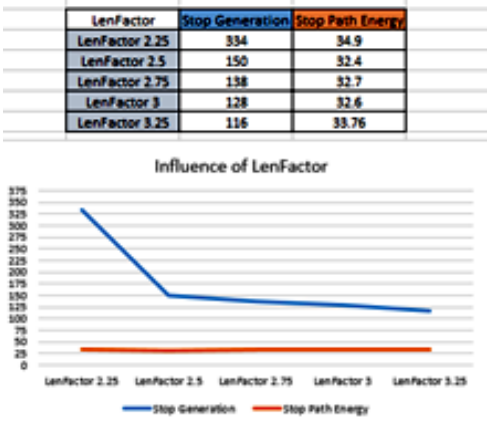

Figure 2: Influence of LenFactor on different aspects of the quality of the solution in World-1.

In the evaluation process, the genetic algorithm was generated 40 times for each value change and then averaged. In these experiments, a limit to the algorithm run time was set. A run with over 1000 generations that did not yield an adequate result was deleted.

Figure 2 depicts the Generation and Path Energy where the GA optimization process has reached the termination point. These two criteria variables, provide measurements of solution quality. The figure depicts that as LenFactor becomes larger compared to EnergyFactor, the results are achieved faster since the solutions closer to complete paths will be selected for further recombination. The effect of their contribution to the solution, will be taken into consideration in lesser importance. By the end of the run, the path will be sufficiently refined to meet the stop condition of path energy below a certain threshold (default value is defined). In the tested map World1, the solution of a somewhat straight path is quite trivial so that finding a complete path is straightforward and consumes a minimal amount of run time. Refining the path to achieve better results (fewer and longer steps) is more complex. LenFactor of 2.5 was chosen as a default for this world since on average this value yields better StopPathEnergy while not influencing run time. As StopPathEnergy improves, so does the quality of the solution.

In Figure 3 both graphs indicate that as the ratio between LenFactor/EnergyFactor gets smaller, the amount of run time needed to find a solution increases. This ratio is what determines the quality of the solution. As shown in Figure 3 (left) increasing the EnergyFactor up to 0.7 improves the StopPath Energy without harming the StopGeneration. However, when the EnergyFactor exceeds 0.7, the ratio between LenFactor/EnergyFactor is no longer beneficial, yielding unacceptable results. The time required to reach a solution increases drastically without much improvement in the StopPathEnergy parameter. These experiments clearly indicate that a value of 0.7 for the EnergyFactor parameter is better and more optima, yielding a higher quality solution together with the previously found LenFactor = 2.5.

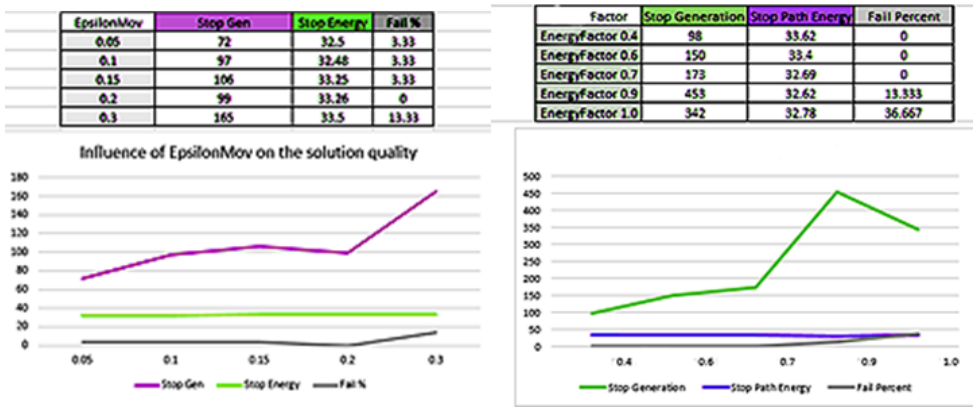

Figure 3: Influence of `EnergyFactor` on different aspects of solution quality (left) influence of `PathLength` on different aspects of the solution quality (right).

As path length increased, the algorithm was forced to find a longer solution, resulting in more generations that were not adequate and longer run time. As Figure 3 (right) depicts, a value of `PathLength` = 30 yields the best solution with respect to `StopEnergy` and does not increase `StopGeneration` by a significant amount. For this reason, the algorithm default was set at `PathLength` = 30.

GA parameters Analysis was done by running the algorithm and analyzing its results relative to the GA parameters. The algorithm was run over 60 times for each change and the average results were used to compare the parameters. The defaults of the algorithm for World1 were based on the results of the previous experiments: `LenFactor` = 2.5 and `EnergyFactor` = 0.7.

Figure 4 (left) depicts, both low and high crossover rates. The high value represents a high rate of generations

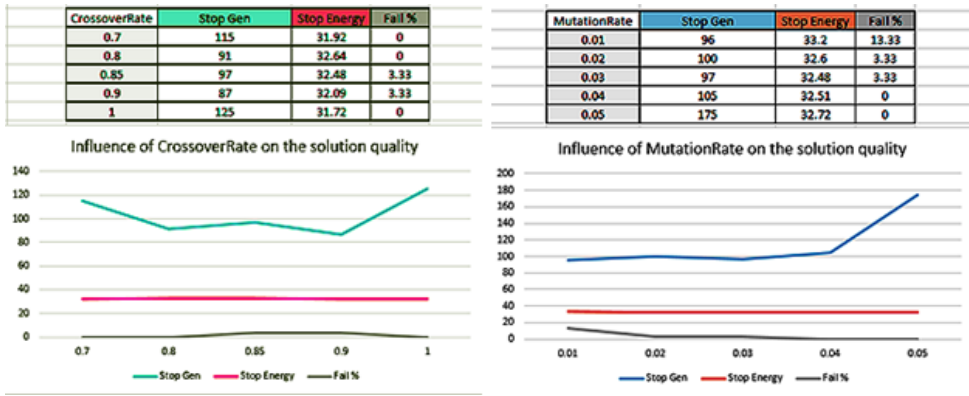

Figure 4: Influence of Crossover Rate (left) and Mutation Rate (right) on different aspects of the solution quality.

for termination, indicating that algorithm run time is slow. A high crossover rate represents a constant improvement in the offspring of each generation. Nevertheless, if the crossover rate is too high, the run time will correspondingly be too long. The default crossover rate was set at =0.85 since this value yields the best result in terms of stop energy within in an acceptable amount of time.

Figure 4 (right) indicates that as mutation rate increases, so does the time needed to find a solution. This

can be understood intuitively: If the mutation rate is higher, the chances are greater that perfectly solutions will be changed. According to the results the best mutation rate for the proposed algorithm is 0.03. This rate yields the best termination criteria (which in turn influences run time) as well as best stop energy criteria, for the solution. Most of the parameters were set to the value that yielded the best `StopEnergy`, since a lower `StopEnergy` value indicates a better solution. Nevertheless, finding a faster solution sometimes requires making compromises and choosing values yielding lower solution energy. The above experiments were used to determine the default values for the genetic algorithm that would enable it to find an optimal solution within an optimal amount of time.
The final stage was to analyze a standard single run of the GA. In this stage we show that better solutions are found as the generations progress.

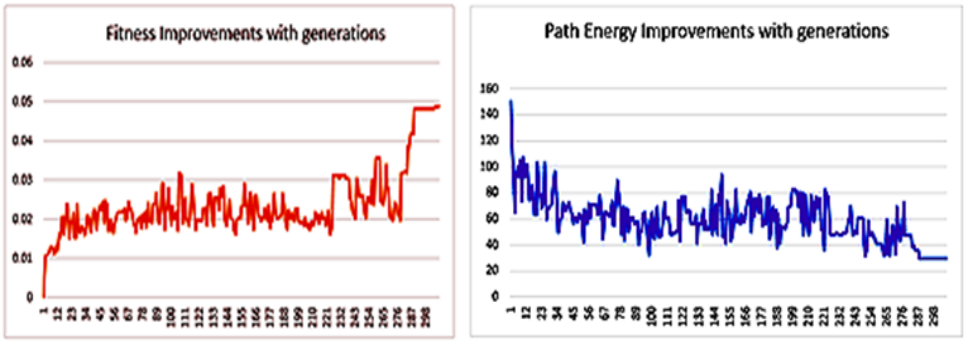

Figure 5: Graph of the improvements in Path remaining distance for the solutions in each generation.

Figure 5 depicted as expected, better solutions slowly emerge as the generations progress until the termination condition is reached. The solution selected in each generation depends both on the path remaining distance and on the path energy, as determined in the fitness score calculation. For this reason, the solution fluctuates until a good balance is reached between the two parameters to yield the best solution with the best fitness.

5 CASE STUDIES

This section describes two world environment cases. The developed system is very flexible and can receive any input, with no size or content constraints. This experimental stage comprises five defined steps in which the energy=cost values are defined by the user. The default values are in the form of uniform ascending values, as shown on the right side of the window depicted in Figure 6.

The user can easily change the values through the GUI, with no restrictions on the START or GOAL positions. The right side of the screen includes a list of parameters for the use of developers or users. These include energy factor (cost), coordinate sensitivity, speed, and GA parameters. Users can easily change these parameters in accordance with their requirements. The system enables users to choose different factors to represent different agents in order to adapt the results to each agent. Some agents are heavier and slower than others are and not all agents have the same step span. The system provides great flexibility in path and motion planning according to each particular agent. The Figure depicts a discontinuous environment in which the agents must follow a path from start to goal at minimal cost while meeting the step and system parameters.

Case Study 1: World-1:

Simulations 1-3 represent results that differ in step cost, yielding different motions for different agent. Simulation 1: The user required a path constructed from larger steps. To this end, the user imposed the

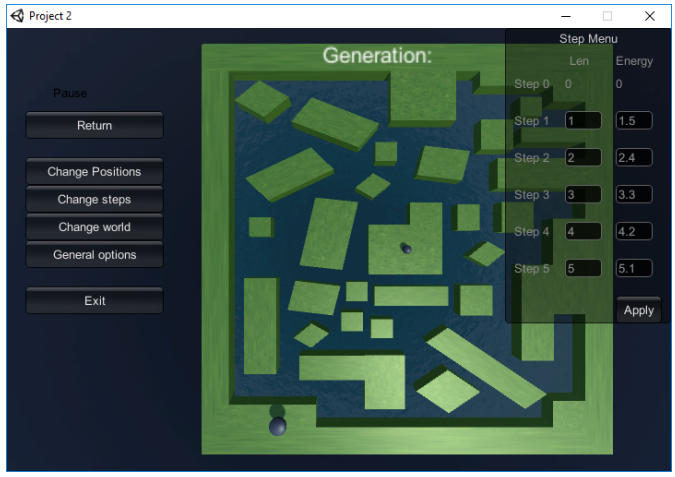

Figure 6: World-1 with defined step costs.

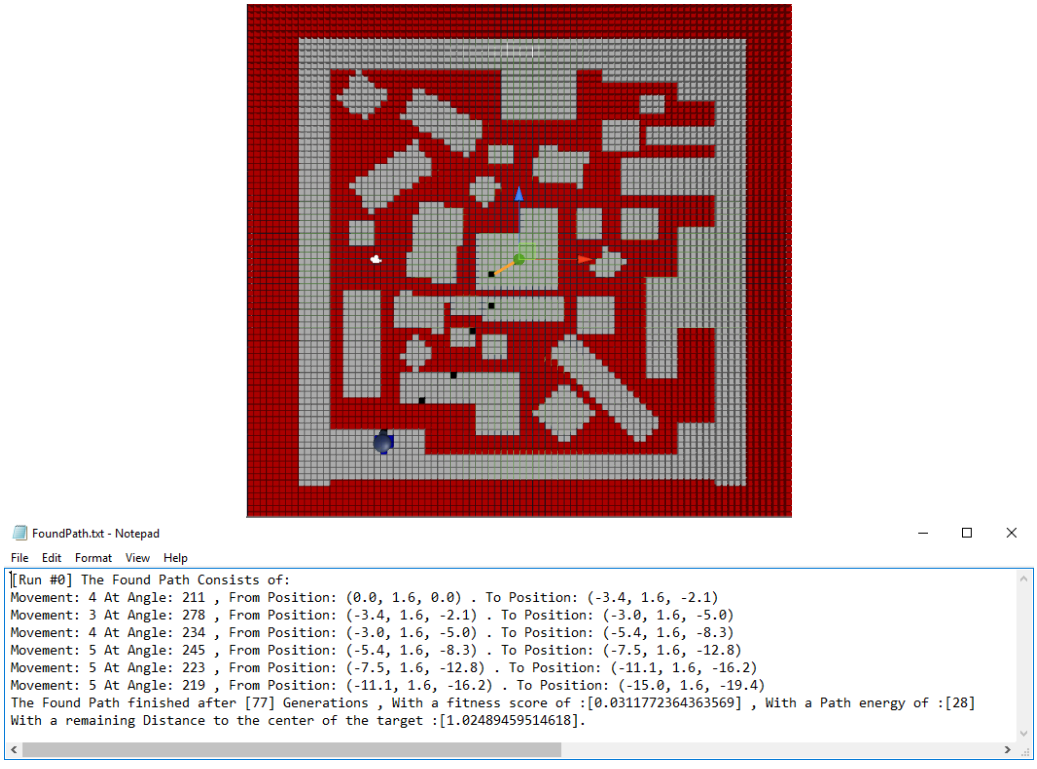

Figure 7: World1 path and motion solution,with step restrictions of 3,4 and 5.

restriction of using only steps of 3,4,5 by stipulating that the values of energy cost in steps 1 and 2 be respectively high. The actual values the user defined for the five steps were 80, 90, 4.5, 6, and 7.5, respectively.

Figure 7 depicts a 2D pixelated world that clearly shows the resulting path and motion planning. All the

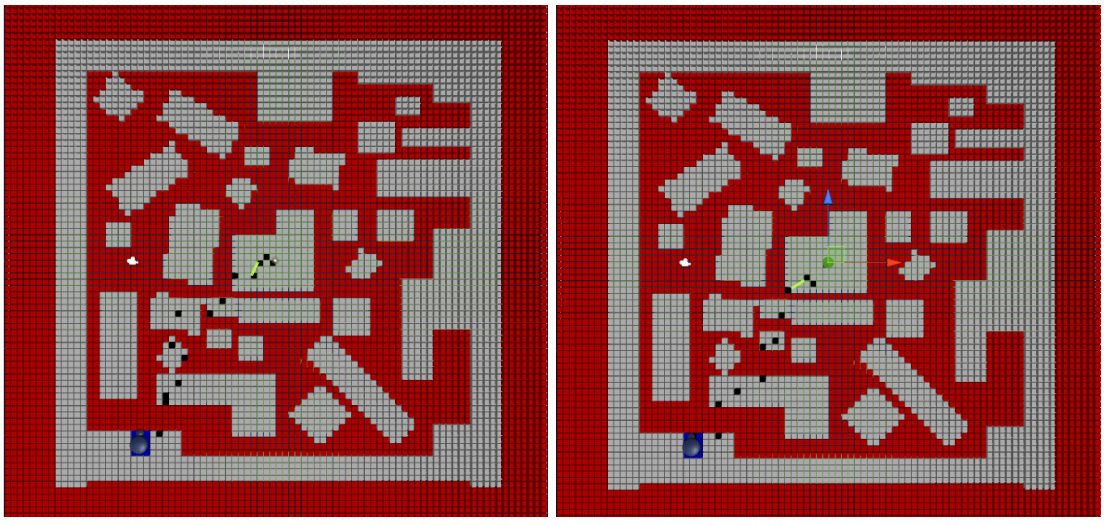

Figure 8: Simulations 2,3, where(a) and (b) the path and motion planning results.

FoundPath.txt - Notepad

File Edit Format View Help

```
[Run #0] The Found Path Consists of:
Movement: 1 At Angle: 159 , From Position: (0.0, 1.6, 0.0) . To Position: (-0.9, 1.6, 0.4)
Movement: 1 At Angle: 205 , From Position: (-0.9, 1.6, 0.4) . To Position: (-1.8, 1.6, -0.1)
Movement: 2 At Angle: 244 , From Position: (-1.8, 1.6, -0.1) . To Position: (-2.7, 1.6, -1.9)
Movement: 2 At Angle: 184 , From Position: (-2.7, 1.6, -1.9) . To Position: (-4.7, 1.6, -2.0)
Movement: 3 At Angle: 243 , From Position: (-4.7, 1.6, -2.0) . To Position: (-6.1, 1.6, -4.7)
Movement: 2 At Angle: 215 , From Position: (-6.1, 1.6, -4.7) . To Position: (-7.7, 1.6, -5.8)
Movement: 3 At Angle: 190 , From Position: (-7.7, 1.6, -5.8) . To Position: (-10.7, 1.6, -6.3)
Movement: 3 At Angle: 257 , From Position: (-10.7, 1.6, -6.3) . To Position: (-11.3, 1.6, -9.3)
Movement: 2 At Angle: 304 , From Position: (-11.3, 1.6, -9.3) . To Position: (-10.2, 1.6, -10.9)
Movement: 3 At Angle: 252 , From Position: (-10.2, 1.6, -10.9) . To Position: (-11.1, 1.6, -13.8)
Movement: 2 At Angle: 225 , From Position: (-11.1, 1.6, -13.8) . To Position: (-12.6, 1.6, -15.2)
Movement: 1 At Angle: 265 , From Position: (-12.6, 1.6, -15.2) . To Position: (-12.7, 1.6, -16.2)
Movement: 3 At Angle: 263 , From Position: (-12.7, 1.6, -16.2) . To Position: (-13.0, 1.6, -19.2)
Movement: 2 At Angle: 177 , From Position: (-13.0, 1.6, -19.2) . To Position: (-15.0, 1.6, -19.1)
The Found Path finished after [200] Generations , With a fitness score of :[0.0307574899335278] , With a Path energy of :[37.400000333786]
With a remaining Distance to the center of the target :[1.28124058246613].
```

FoundPath.txt - Notepad

File Edit Format View Help

```
[Run #0] The Found Path Consists of:
Movement: 3 At Angle: 234 , From Position: (0.0, 1.6, 0.0) . To Position: (-1.8, 1.6, -2.4)
Movement: 1 At Angle: 153 , From Position: (-1.8, 1.6, -2.4) . To Position: (-2.7, 1.6, -2.0)
Movement: 2 At Angle: 212 , From Position: (-2.7, 1.6, -2.0) . To Position: (-4.4, 1.6, -3.0)
Movement: 3 At Angle: 246 , From Position: (-4.4, 1.6, -3.0) . To Position: (-5.6, 1.6, -5.8)
Movement: 3 At Angle: 268 , From Position: (-5.6, 1.6, -5.8) . To Position: (-5.7, 1.6, -8.8)
Movement: 2 At Angle: 208 , From Position: (-5.7, 1.6, -8.8) . To Position: (-7.4, 1.6, -9.7)
Movement: 3 At Angle: 273 , From Position: (-7.4, 1.6, -9.7) . To Position: (-7.3, 1.6, -12.7)
Movement: 3 At Angle: 209 , From Position: (-7.3, 1.6, -12.7) . To Position: (-9.9, 1.6, -14.2)
Movement: 3 At Angle: 222 , From Position: (-9.9, 1.6, -14.2) . To Position: (-12.1, 1.6, -16.2)
Movement: 3 At Angle: 270 , From Position: (-12.1, 1.6, -16.2) . To Position: (-12.1, 1.6, -19.2)
Movement: 3 At Angle: 182 , From Position: (-12.1, 1.6, -19.2) . To Position: (-15.1, 1.6, -19.3)
The Found Path finished after [245] Generations , With a fitness score of :[0.0340420011486435] , With a Path energy of :[34.6999998092651]
With a remaining Distance to the center of the target :[1.10254764556885].
```

Figure 9: (a) and (b) are detailed values of the steps in coordinated and angles, respectively.

steps from Start to Goal were taken from the stipulated group of steps (3,4,5) but given different costs. The lower part of the figure indicates a list of steps and their exact positions and angles. Because this world is defined as 2.5D, all heights are the same, while the user defined flexible Y coordinate as 1.6.
The lower part of the window shows the calculated parameters: fitness score and path energy (cost). The remaining distance to the center target is a factor for use by the developers and is a result of the discrete

pixelated world defined.
Based on these stipulations, the search strives to find a balance between number of generations (computing time) and energy cost of the path found. As the goal of this work is for animation purposes, computation time is crucial for achieving the smooth movements needed for realistic results. In the many of simulations run for this users requirements, termination was achieved after fewer than 80 generations, so that computation time was a non-constraining factor.
Simulations 2 and 3: Figure 8 and Figure 9 depict the results of path and motion planning for cases in which the user imposed restrictions by giving preference to steps 1, 2 and 3. The two examples also assign different energy costs for these steps, resulting in different paths and motions. Figure 9 respectively shows the detailed steps that define the motion paths as well as the resulting calculated parameters. These two examples represent for different agents with different energy costs for the steps. The last row in each figure indicates the total cost of the the path, for this assignment. Path one as indicated results cost of 37.6 units, and the second results 34.6.

Case Study 2: World-2:

Example 2 depicted in Figure 10 is a more complex world with a major obstacle, which highlights the ability of the system to perform and to provide a very good solution in various conditions, all of them defined by the user. The two simulations depicted in Figure 11 (left), differ in the costs for step sizes, adapting to the

Figure 10: World-2 with defined parameters.

simulated character in hand. Figure 11 (left) the costs for the steps are the following: 1,3 and 5 were defined as 90 units for each step, and steps 2 and 4 as a value of 4. This process, provides steps 2 and 4 a dominance of choice in the process of the GA optimization.

In Figure 11 (right) were all steps were defined in various costs but differing in less than 30 percents from each other. Figure 12 depicts the set of coordinated and angles comprising of the path and detailed the steps taken. the sum of the optimized path is indicated in the lower part of the figure, respectively.

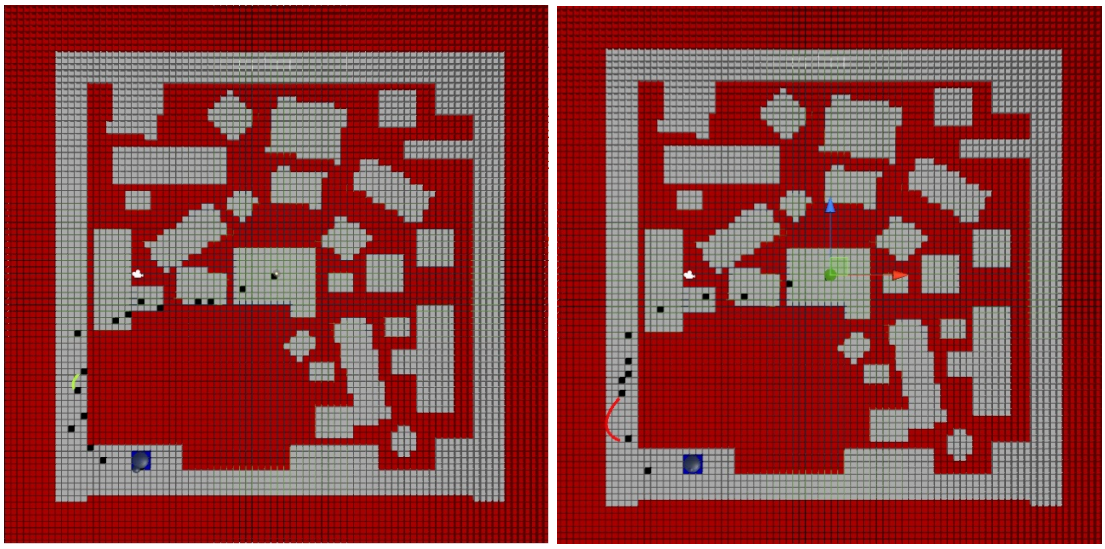

Figure 11: Simulations 2 and 3 results for path and motion planning.

ResPath.txt - Notepad

File Edit Format View Help

```
[Run #0] The Found Path Consists of:
Movement: 4 At Angle: 206 , From Position: (0.0, 1.6, 0.0) . To Position: (-3.6, 1.6, -1.8)
Movement: 4 At Angle: 205 , From Position: (-3.6, 1.6, -1.8) . To Position: (-7.2, 1.6, -3.4)
Movement: 2 At Angle: 163 , From Position: (-7.2, 1.6, -3.4) . To Position: (-9.1, 1.6, -2.9)
Movement: 4 At Angle: 197 , From Position: (-9.1, 1.6, -2.9) . To Position: (-13.0, 1.6, -4.0)
Movement: 2 At Angle: 165 , From Position: (-13.0, 1.6, -4.0) . To Position: (-14.9, 1.6, -3.5)
Movement: 2 At Angle: 219 , From Position: (-14.9, 1.6, -3.5) . To Position: (-16.4, 1.6, -4.8)
Movement: 2 At Angle: 206 , From Position: (-16.4, 1.6, -4.8) . To Position: (-18.2, 1.6, -5.6)
Movement: 4 At Angle: 200 , From Position: (-18.2, 1.6, -5.6) . To Position: (-22.0, 1.6, -7.0)
Movement: 4 At Angle: 272 , From Position: (-22.0, 1.6, -7.0) . To Position: (-21.9, 1.6, -11.0)
Movement: 2 At Angle: 260 , From Position: (-21.9, 1.6, -11.0) . To Position: (-22.2, 1.6, -13.0)
Movement: 4 At Angle: 262 , From Position: (-22.2, 1.6, -13.0) . To Position: (-22.8, 1.6, -16.9)
Movement: 2 At Angle: 40 , From Position: (-22.8, 1.6, -16.9) . To Position: (-21.2, 1.6, -15.7)
Movement: 4 At Angle: 272 , From Position: (-21.2, 1.6, -15.7) . To Position: (-21.1, 1.6, -19.7)
Movement: 2 At Angle: 329 , From Position: (-21.1, 1.6, -19.7) . To Position: (-19.4, 1.6, -20.7)
Movement: 4 At Angle: 349 , From Position: (-19.4, 1.6, -20.7) . To Position: (-15.5, 1.6, -21.4)
The Found Path finished after [2015] Generations , With a fitness score of :[0.0290232370464968] , With a Path energy of :[48]
With a remaining Distance to the center of the target :[1.05057227611542].
```

ResPath.txt - Notepad

File Edit Format View Help

```
[Run #0] The Found Path Consists of:
Movement: 5 At Angle: 193 , From Position: (0.0, 1.6, 0.0) . To Position: (-4.9, 1.6, -1.1)
Movement: 5 At Angle: 192 , From Position: (-4.9, 1.6, -1.1) . To Position: (-9.8, 1.6, -2.2)
Movement: 4 At Angle: 185 , From Position: (-9.8, 1.6, -2.2) . To Position: (-13.7, 1.6, -2.5)
Movement: 5 At Angle: 192 , From Position: (-13.7, 1.6, -2.5) . To Position: (-18.6, 1.6, -3.6)
Movement: 5 At Angle: 220 , From Position: (-18.6, 1.6, -3.6) . To Position: (-22.5, 1.6, -6.8)
Movement: 3 At Angle: 274 , From Position: (-22.5, 1.6, -6.8) . To Position: (-22.3, 1.6, -9.8)
Movement: 1 At Angle: 291 , From Position: (-22.3, 1.6, -9.8) . To Position: (-21.9, 1.6, -10.7)
Movement: 1 At Angle: 220 , From Position: (-21.9, 1.6, -10.7) . To Position: (-22.7, 1.6, -11.3)
Movement: 2 At Angle: 263 , From Position: (-22.7, 1.6, -11.3) . To Position: (-22.9, 1.6, -13.3)
Movement: 5 At Angle: 280 , From Position: (-22.9, 1.6, -13.3) . To Position: (-22.0, 1.6, -18.2)
Movement: 4 At Angle: 300 , From Position: (-22.0, 1.6, -18.2) . To Position: (-20.0, 1.6, -21.7)
Movement: 5 At Angle: 9 , From Position: (-20.0, 1.6, -21.7) . To Position: (-15.1, 1.6, -20.9)
The Found Path finished after [297] Generations , With a fitness score of :[0.0347056215678756] , With a Path energy of :[49.7]
With a remaining Distance to the center of the target :[0.438999891281128].
```

Figure 12: Detailed values of steps' coordinated and angles.

6 CONCLUSIONS

This study describes a novel path and motion planning system for realization with animated characters in simulations and games. The approach uses the genetic algorithm method together with flexible parameters and variables adapted to different environments and characters. Motions are constrained by two parameters: 1) terrain, as tested on discontinuous cases, and 2) different kinds of steps, depicted by length, energy and cost. The user can adjust the value of each of these parameter values as required. The research also analyzes the GA model, its parameters and the defined optimization criteria and variables for solution efficiency. The proposed system attained very good results and provided adequate solutions in real time for various cases. In the future, the model and the software can be expanded to environments that are more complex than 3D, thus requiring different environment chromosome representations. Another interesting topic would be to examine dynamic environmental changes.

ORCID

Miri Weiss Cohen, http://orcid.org/0000-0001-5250-1016

REFERENCES

[1] Badler, N.I.; Phillips, C.B.; Webber, B.L.: Simulating humans: computer graphics animation and control. Oxford University Press, 1993.

[2] Bruderlin, A.; Calvert, T.W.: Goal-directed, dynamic animation of human walking. In Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '89, 233–242. ACM, New York, NY, USA, 1989. http://doi.org/10.1145/74333.74357.

[3] Campana, M.; Fernbach, P.; Tonneau, S.; Taïx, M.; Laumond, J.P.: Ballistic motion planning for jumping superheroes. In Proceedings of the 9th International Conference on Motion in Games, MIG '16, 133–138. ACM, New York, NY, USA, 2016. http://doi.org/10.1145/2994258.2994279.

[4] Chuang, Y.C.; Chen, C.T.; Hwang, C.: A real-coded genetic algorithm with a direction-based crossover operator. Information Sciences, 305, 320–348, 2015.

[5] Cui, X.; Shi, H.: Direction oriented pathfinding in video games. International Journal of Artificial Intelligence & Applications, 2(4), 1, 2011.

[6] Eiben, A.; Smith, J.: Evolutionary computing: The origins. In Introduction to Evolutionary Computing, 13–24. Springer, 2015.

[7] Gleicher, M.; Shin, H.J.; Kovar, L.; Jepsen, A.: Snap-together motion: Assembling run-time animations. In Proceedings of the 2003 Symposium on Interactive 3D Graphics, I3D '03, 181–188. ACM, New York, NY, USA, 2003. http://doi.org/10.1145/641480.641515.

[8] Haupt, R.L.; Haupt, S.E.: Practical genetic algorithms. John Wiley & Sons, 2004.

[9] Jaklin, N.; Cook IV, A.; Geraerts, R.: Real-time path planning in heterogeneous environments. Computer Animation and Virtual Worlds, 24(3-4), 285–295, 2013. http://doi.org/10.1002/cav.1511.

[10] Juarez-Perez, A.; Kallmann, M.: Full-body behavioral path planning in cluttered environments. In Proceedings of the 9th International Conference on Motion in Games, MIG '16, 107–112. ACM, New York, NY, USA, 2016. http://doi.org/10.1145/2994258.2994281.

[11] Kenwright, B.: Planar character animation using genetic algorithms and gpu parallel computing. Entertainment Computing, 5(4), 285–294, 2014. http://doi.org/https://doi.org/10.1016/j.entcom.2014.09.003.

[12] Kuffner, J.; Latombe, J..: Interactive manipulation planning for animated characters. In Proceedings the Eighth Pacific Conference on Computer Graphics and Applications, 417–418, 2000. http://doi.org/10.1109/PCCGA.2000.883973.

[13] Lau, M.; Kuffner, J.J.: Behavior planning for character animation. In Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '05, 271–280. ACM, New York, NY, USA, 2005. http://doi.org/10.1145/1073368.1073408.

[14] Liu, Y.; Badler, N.I.: Real-time reach planning for animated characters using hardware acceleration. In Proceedings 11th IEEE International Workshop on Program Comprehension, 86–93, 2003. http://doi.org/10.1109/CASA.2003.1199308.

[15] Lu, R.; Zhang, S.: Automatic generation of computeranimation: using AI for movie animation. Springer-Verlag, 2002.

[16] Mac, T.T.; Copot, C.; Tran, D.T.; Keyser, R.D.: Heuristic approaches in robot path planning: A survey. Robotics and Autonomous Systems, 86, 13 – 28, 2016. http://doi.org/https://doi.org/10.1016/j.robot.2016.08.001.

[17] Malhotra, R.; Singh, N.; Singh, Y.: Genetic algorithms: Concepts, design for optimization of process controllers. Computer and Information Science, 4(2), 39, 2011.

[18] Millington, I.; Funge, J.: Artificial intelligence for games. CRC Press, 2016.

[19] Simon, D.: Evolutionary optimization algorithms. John Wiley & Sons, 2013.

[20] Sivaraj, R.; Ravichandran, T.: A review of selection methods in genetic algorithm. International journal of engineering science and technology, 3(5), 2011.

[21] Um, S.; Kim, T.; Choi, J.: Dynamic difficulty controlling game system. IEEE Transactions on Consumer Electronics, 53(2), 812–818, 2007. http://doi.org/10.1109/TCE.2007.381764.

[22] Whitley, D.: An executable model of a simple genetic algorithm. Foundations of Genetic Algorithms, 2, 45–62, 2014.

[23] Yang, L.; Qi, J.; Song, D.; Xiao, J.; Han, J.; Xia, Y.: Survey of robot 3d path planning algorithms. Journal of Control Science and Engineering, 2016, 5–22, 2016. http://doi.org/10.1155/2016/7426913.

[24] Zhang, G.; Liu, H.; Huang, Y.; Zhang, S.; Lu, R.: Combining path and posture planning in 3d environment. In International Conference of Electrical, Automation, and Mechanical Engineering (EAME 2015), 425–429, 2015.