Proceedings of the Seventh International Conference on Machine Learning and Cybernetics, Kunming, 12-15 July 2008

AN AI TOOL: GENERATING PATHS FOR RACING GAME

CHARLIE IRAWAN TAN, CHANG-MIN CHEN, WEN-KAI TAI, SHI-JIM YEN

Department of Computer Science and Information Engineering, National Dong Hwa University, Taiwan
E-MAIL: wktai@mail.ndhu.edu.tw, d9421003@em94.ndhu.edu.tw

Abstract:

In this paper, we propose an AI tool for generating plausible paths of racers based on the A* algorithm. User can define the race by providing a race course of 3D model and weights of the devised turn and heuristic functions in our system. The search space for path-finding is represented by a grid. Then, the proposed cost map generator automatically generates necessary information of the race course including cost value for each cell in the grid, feature cells, etc. Different from the traditional A* problem, in our research the obstacles are dynamic and there are multiple sources and destinations. Our approach generates the path on the basis of time slots and smoothes it by Gaussian filter. By the devised heuristic function we have a natural racer behavior. Also, our AI tool for path finding has been verified in a horse racing game, demonstrating realistic and exciting racing.

Keywords:

A*; Path finding; Heuristic function; AI tool; Racing game

1. Introduction

Game AI improves and enhances game experience and gameplay. To make the behavior of a character appear more intentional will also make it seem more intelligent. In order to generate a realistic racing impression of a racing game, the path on which a racer is passing is very important. To provide good interactions in the game, the developer often uses the player path as the reference for generating the AI's racing path. But sometimes, the game is lack of user interference, for example in the game demo, the auto run mode, etc. In such case, the excitement of the racing is fully controlled by the computer's AI without user's help.

In this research we propose an AI tool to generate a set of plausible paths that the racers will advance in a racing game without any guidance from the player. We utilize A* algorithm, a well-known algorithm to solve path-finding problems, and modify it to satisfy our requirements (as shown in section 3.1.). In order to control realism of the path and at the same time fulfill any constraints required by the race, we devise two functions, turn deciding and heuristic functions, which are responsible for deciding the sequence of racers who will execute the A* algorithm and how the algorithm is executed respectively. We assume that the race course is circular on the ground for simplicity and generality since most of the racing courses are circular.

In our system, a 3D model of the race course is given first to the cost map generator tool. Then, the tool automatically discretizes the model and represent it by a grid which is the search space of the path finding method. Also, the tool calculates some necessary information such as center points, feature cells, the cost value for all cells, target cells, etc. for the grid automatically yet user only needs to provide beginning and finish positions of the racer on the innermost course. Of course, depending on the race requirements, user may change the costs on the cost map, weights of the turn deciding and heuristic functions, and the random speed generating parameters in our system. Using the speed lists that are randomly generated, cost map that defines the course, and weight parameters that control the behavior of the racers, our path generator generates the path in the A* component on the basis of time slots. The path finding method in the A* component is different from the traditional path finding problem due to dynamic obstacles and multiple sources and destinations in the search space.

We briefly summarized our contributions as follow:

- We extend A* path finding algorithm to solve a path finding problem with dynamic obstacles and multiple-source multiple-destination.
- A turn deciding function and a heuristic function are devised to obtain plausible paths effectively. In addition, we formalize a mechanism to carefully weight the costs in the function so that we have a natural racer behavior.

The remainder of this paper is structured as follows: In Section 2 we will discuss related work. Section 3 explains the proposed method and tools. Section 4 shows the results of our proposed method. Finally, in Section 5 we conclude this paper and discuss ways to improve our current work in the near future.

978-1-4244-2096-4/08/$25.00 ©2008 IEEE

2. Related Works

Generating a racing path is actually a problem of motion planning and path finding which have been studied extensively in many field of study including virtual reality, AI, and robotics, etc. There are many directions in this topic such as robotic motion planning using a probabilistic approach [6][7], crowd motion simulation [8][9], and group movements [1]. Our work has similar objective as the crowd and group motion planning but with different constraints. Because each unit in the group (the racer) has not only similar target position but they should precede others when reaching the target.

A* path finding algorithm [5] is a very popular solution for path finding problems in games. A lot of games use A* even the commercial games. Higgins discussed on how to write a generic A* implementation that re-useable [2] and effective as possible [3]. In our method, we modify the A* algorithm so that it can be applied to racing path generation.

Biasillo [4] proposed a representation of race track that divides the track into sectors and include the track information i.e. racing path on the boundary of the sector. This method reduces the amount of data that needs to be stored, but as consequences it reduces the accuracy since many parts of the path are generated through interpolation. Furthermore such structure is inappropriate for A* implementation.

3. Method

We describe our method by presenting three important aspects of a racing. Path finding which is the most important aspect assures that the racers are on the path and do not collide with each other. Racer behavior comes next. It controls the racer behavior on how he seizes the first position and the excitement of the race itself. Finally, the race course representation which is the foundation of the simulation process defines how the calculation is actually done. We also present a refinement tool which helps fulfill the requirement of its user. With the refinement process, the quality of the race is even more guaranteed due to the customization.

3.1. Path Finding

A* is a well known path finding algorithm that is widely used in many games. It provides a good balance between speed and accuracy. Based on the branch and bound algorithm, A* maintains a priority list of possible paths which grows towards the target position (open list) and a list of even more likely paths (close list). When the target position is reached i.e. it is in the close list, we can backtrack and record the path towards the start position. In order to determine the priority of a step in a path, A* requires a heuristic function. The heuristic function will affect the path produced by A*.

Although A* algorithm is an excellent path finding algorithm, directly applying to a racing game is impracticable. Some problems of the traditional A* algorithm when applied to the racing games are:

- Time independent. When finding the path, A* doesn't take the time into consideration, i.e. it doesn't know when a position is reached.
- Single-source single-destination. In racing game we have to solve multiple-source multiple-destination problem.
- Static obstacles. Obstacles are static in A* methods while in the racing game they are dynamic mostly.

In order to solve the time independency of A* algorithm and to extend A* to multiple–source multiple-destination domain, we apply A* algorithm on each racer based on turn. We first slice the entire racing time into smaller time periods and simulate the racing separately. For each simulation, we maintain a sequence of racers each executes the A* algorithm in turn. We use the behavior of the horses as the deciding factor of this sequence. Also, we weight the behavior in the turn deciding function. The turn deciding function is defined as

$$F_{turn} = {w_s}/{l_i} + w_e D_{end} + w_c D_{center} \qquad (1)$$

where $\boldsymbol{l_i}$ is the racer's speed, $\boldsymbol{D_{end}}$ is the current distance to the end of the race and $\boldsymbol{D_{center}}$ is the current distance to the nearest center of the race course. A racer with the smallest value of the turn deciding function precedes others in turn.

In this research, the heuristic function is considered a function of time-space cost, rotational cost, and the spatial cost. The heuristic function is defined as

$$F_{heuristic} = w_t C_t + w_r C_r + w_m C_m + w_e D_{end} \qquad (2)$$

where $\boldsymbol{C_t}$ is the time-space cost, $\boldsymbol{C_r}$ is the rotational cost and $\boldsymbol{C_m}$ is the spatial cost. The time-space cost prevents racers from collision and then solves the dynamic obstacles problem. It has the minimal value 0 if there are no other racers at next selected position at a given time. Otherwise, we give a high cost to it, say 100 for example. The rotational cost $\boldsymbol{C_r}$ is the orientation difference from the previous direction. The spatial cost $\boldsymbol{C_m}$ is defined by the race course map data (please refer to sub-section 3.3. for more details). The rotational cost determines likeliness for a

racer to turn, and hence it prevents them from constantly changing direction during racing on the straight path. In the curved path, however, the spatial cost becomes the factor that makes the racer to turn by increasing the cost of outer race track.

3.2. Racer Behavior

The racer behavior is defined as a random speed list and a number of weights which are used in the turn deciding and heuristic functions. To randomize the speed list, we define an average speed, a speed variance which defines a range of speed, and an acceleration value which limits the speed changes. We limit the speed change in order to prevent instant speed change that looks unnatural.

The racer behavior is defined as a random speed list and a number of weights which are used in the turn deciding and heuristic functions. To randomize the speed list, we define an average speed, a speed variance which defines a range of speed, and an acceleration value which limits the speed changes. We limit the speed change in order to prevent instant speed change that looks unnatural.Determining a sequence of racers for A* execution is vital in our method. That means that determining the weights used in the turn deciding function is important. A bad sequence likely pushes a racer into a dead end, namely it cannot find a feasible path since possible paths are already selected by others. This is especially important on the curved course since inner course is favorable to any racer and all the racers might go to this course.

In the turn deciding function we define three weights. Higher speed weight (w_s) allows the quickest racer at the time to precede others; higher distance to end weight (w_e) allows the leading racer to precede others; higher distance to center weight (w_c) allows the racer in the inner track to precede others. In ideal case, it is reasonable that the leading racer who also has the highest speed and yet on the inner most course to precede others, but that case is not likely to happen. When we have to choose, the racer in inner track should execute first, since it is most likely to stay in its track. If the distance to center is similar, then we select the foremost racer to executer first. The reason is that if their tracks are the same, then their ranks are absolutely different. Selecting the foremost racer makes them capable to seize the inner track.

Another behavior that we define is the rotational tendency of the racer. The racer must maintain its direction on the straight course and rotate on the curved course. We control this behavior through weights w_r and w_m in the heuristic function. The rotational weight w_r reduces the rotational tendency and makes the racer maintain its direction. But when it is necessary to turn, w_m and C_m will increase the cost to force the racer to turn. The values of rotational and spatial cost must be balanced enough in order to generate a realistic racing path. If the spatial cost is too small, relative to the rotational cost, the racer will maintain its position until it reach the outer course on the curved course before it turns. If the rotational cost is too small, then the racer might not maintain its direction on the straight course.

3.3. Race Course Representation

When dealing with the path finding algorithm, the search space representation is critical. The representation of the underlying search space will have a dramatic impact on the performance and memory requirements of the path finding method and the quality of the paths the racer follows. Many approaches have been proposed to represent a search space [6]. We need a path finding representation that fits into a reasonable amount of memory and allows the fastest possible search. We discretize a race course into a grid. The cell width of the grid is set to be the longest axis of the racer so we can prevent each other racer from collision approximately. Like other discrete space, discrete representation of the race course is subject to aliasing problem, and increasing the resolution reduces this problem with performance penalty. However, grids support random-access lookup. Thus, we need to maintain a good balance between computation cost and accuracy.

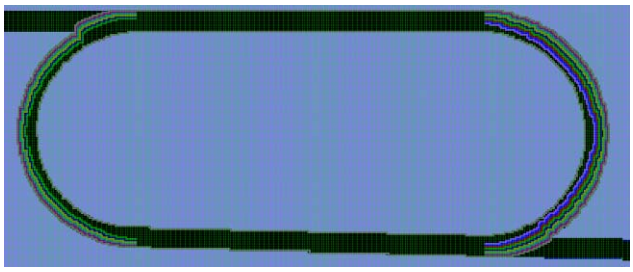

Figure 1. An example of the cost map of a race course.

Each cell is marked with the spatial cost (C_m) of the position, forming a cost map. The value of this cost is set to fulfill the requirement as mentioned in sub-section 3.2. An example of cost setting is shown in Figure 1. The black area represents the course where cost is set to minimum; the blue area represents the unwalkable area while the other color represents other cost setting. Note that we only set some specific cost on the circular track.

In order to annotate the race direction and to keep

tracks of current race length that a racer has been advanced, we number a set of feature cells sequentially and assign the number to cells which have the same race length with the feature cell. The innermost track is selected as the initial feature cells, and we mark them from the start cell sequentially. To track the race length that a racer been advanced, the cells on a line from a feature cell (the racer position) to its corresponding center of the cost map are set to the number of the feature cell. As shown in Figure 2, this sample race course has two center points (red dots), c_i and c_j. The cells (blue) at innermost course are numbered sequentially from the start cell (yellow). The cells (magenta) on the line from the racer (green) to center c_i have the number same with feature cell c_f.

The course center is defined by two center points. If the center points reside on the same spot, then the race course is a circle. When the course is not a circle, we calculate the distance to the course center as the distance to the nearest center point if the racer is on the curved course, and the distance to the line formed by two center points if the racer is on the straight course. In Figure 2, we can see that the distance to the course center for the grid g_i (green) is $|g_i - c_i|$.

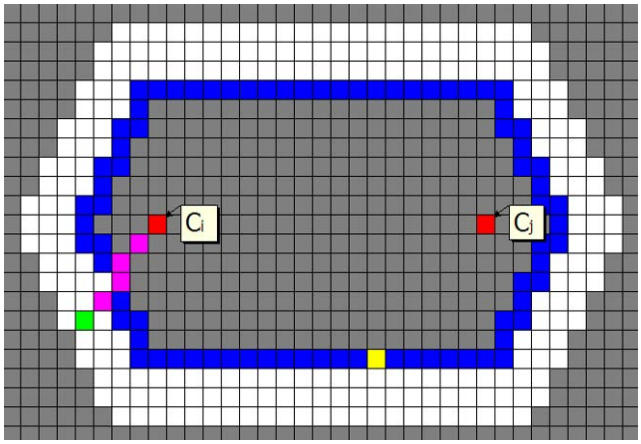

Figure 2. A sample race course illustrates two center points (red), one start cell (yellow), cells with the same number (distance to finish) (magenta and green), and innermost cells (blue) with the same cost value.

The race course is generated semi-automatically. Given a race course model, the discretized model and the center points are automatically generated. Users then need to define the start position of every racer and the finish point on the innermost course, and the system will automatically find all the feature cells and number them accordingly. Finally the cost values for all cells in the map are generated by our system automatically. If the result of the simulation is not satisfied enough, the user may modify the cost values interactively. Generally the cost of outer track is higher than the inner track.

4. Results

Figure 3 shows the user interface of the cost map generator for the sample race course mentioned in sub-section 3.3. The green mesh indicates the automatically generated grid for the race course with respect to a given three dimensional race course model. Users can interactively assign the start position and end position for each racer as shown by H1, H2, …, H6 (red) in the Figure 3.

The tool generates, divides and assigns cost to the grids automatically from a race course model. The tool also automatically calculates the centers, and with the start and finish position it generates the feature cells as shown in cells marked by a small red flag. These automation greatly improve the effectiveness of course generating process. Optionally, user can edit the data that automatically generated including costs, centers position, feature cells, cell size, course scale etc.

The default cost range is defined in the script and can be edited according to the race requirements. In our implementation, the range is from 1 to 10. The cost is distributed from the feature cell on the curved path outward. The cost increases along with the distribution, forming the lowest cost on the innermost course and the highest cost on the outermost course.

When the cost map is set and saved in a file, A* algorithm starts execution to obtain the feasible path for each racer. Due to the random access capability of grid representation, searching cells is efficient. For each cell 1 Byte is required to record a value of the cost, another 2 Bytes for a feature cell and approximately 36 bytes for other data such as course centers and start positions of each racer. Therefore, the memory space is about 107KB for the grid with resolutions 643 x 163.

A racer would likely to select a position that is available, feasible and closest to the end of race. Such position is usually available on the front in the straight course and toward the inner most course on the curved course. The leading racer should take the most advantage position provided it is capable of doing that action, namely with sufficient speed. In short, the distance, speed, and space time factors are all considered in our devised heuristic and turn deciding functions. The time-space weight of heuristic function is set to 0.25 since the existence of others in a grid is absolute restriction for the racer. The weights of rotational and spatial cost must be

balanced with the value of the cost map. With our cost map configuration, the weights of rotational and spatial cost are set to 0.4 and 0.1 respectively. This is because our cost map has relatively high range (1 to 10) and we need to add the weight of the rotational cost to make it on effect. The rest (1- w_t - w_r - w_m) is assigned to end distance weight. Each weight in both turn deciding and heuristic functions represents real factors that affect the decision making of a racer.

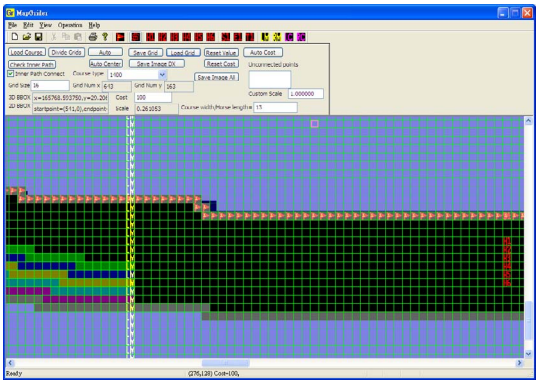

Figure 3. The user interface of the proposed cost map generator for a given race course.

Finally, we present the actual racing game screenshot in which the path the racer passing on is generated by our method in Figure 4 and Figure 5. After we simulate the race, we store the path as a sequence of position-time pair. We can use the saved path in any application by linearly interpolating positions of two consecutive pairs in the sequence and use the time information as the interpolation weighting parameter.

Figure 4. Racers on the curved track.

Using our method, the game play increases as the race become more exciting. The horses are trying to outrun the others not only by speed but also by seizing the best position on the course. Naturally they try to run on the innermost track on the curved track as shown in Figure 4, but they stop to change the track after they reach the straight track as shown in Figure 5.

Figure 5. Racers on the straight track.

5. Conclusions & Future Works

In this paper, we have presented the proposed AI tool. Conventional A* algorithm is modified in order to generate a realistic racing path for a path-finding problem of multiple-source multiple-destinations with dynamical obstacles in the racing game. The heuristic function dominates the realism of paths found. According to true racing behaviors of the racer, we have devised a turn deciding function and a heuristic function to generate plausible paths by weighting control parameters. We have carefully tested our method by applying it to a horse racing game. The result is satisfying as the game shows a realistic and exciting racing. The method is semi-automatic with minimal interaction required. A user may customize the path if necessary but mostly the automated result is sufficed.

The grid representation for the search space suffers from aliasing problem because the artifact cells make the path found not smooth. We will consider a new shape, say octagonal or other feasible shapes, for the cell instead of rectangle to reduce artifact cells. We are also interested in displacing the speed list into a function that calculates a specific speed based on the condition of the racers and environments to generate an even more realistic racing path.

References

[1] A. Kamphuis and M. H. Overmars, "Finding Paths for Coherent Groups Using Clearance", Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 19-28, 2004.

[2] Dan Higgins, "Generic A* Pathfinding", AI Game Programming Wisdom, Charles River Media, Massachusetts, 2002.
[3] Dan Higgins, "How to Achieve Lightning-Fast A*", AI Game Programming Wisdom, Charles River Media, Massachusetts, 2002.
[4] Gari Biasillo, "Representing a Racetrack for the AI", AI Game Programming Wisdom, Charles River Media, Massachusetts, 2002.
[5] James Matthews, "Basic A* Pathfinding Made Simple", AI Game Programming Wisdom, Charles River Media, Massachusetts, 2002.
[6] Kavraki L. and Latombe J.-C., "Randomized preprocessing of configuration space for fast path planning", International Conference on Robotics and Automation, pp. 2138-2139, 1994.
[7] Kavraki L., Švestka P., and Latombe J.-C., "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", IEEE Transactions on Robotics and Automation, Vol 12, pp. 556-580, Feb. 1996.
[8] Helbing D., "A mathematical model for the behavior of individuals in a social field", Journal of Mathematical Sociology, Vol 19, No. 3, pp. 189-219, Feb. 1994.
[9] Helbing D. and Molnár P., "Social force model for pedestrian dynamics", Physical Review, Vol 51, pp. 4282-4286, Feb. 1995