ISSN : 0974 - 7435 Volume 10 Issue 15

2014

BioTechnology

An Indian Journal

FULL PAPER

BTAIJ, 10(15), 2014 [8620-8626]

Application of artificial intelligence technology in computer game software

Xiaobo Guo, Yan Zhai
Anyang Institute of Technology, Anyang 455000, (CHINA)

ABSTRACT

With the development of computer hardware and software system, especially the development of 3D graphics technology, which greatly enhances the image of the computer game software, artificial intelligence is relative lagging and become the bottleneck that restrict the improvement of game performance. Real-time nature of computer games asks artificial intelligence computing occupy as less system resources as possible so as to provide sufficient resources to other modules of the game.After overview of the software architecture of game engine, this paper first summarizes some artificial intelligence methods the game often use but no documented; then the expansion and optimization of the game software and discusses the method of its integration with other game software module. Motion control of the game role and path planning technology are discussed in depth. Second, the artificial intelligence technology may be used in the game may to a higher level are discussed. Later, through the analysis of specific cases, way and the possibility of the learning and adaptive artificial intelligence in the game software are discussed.

KEYWORDS

Computer game; Artificial intelligence; Game engine; Script system.

ARTIFICIAL INTELLIGENCE

AI concept description

Artificial intelligence is AI in brief. Broadly, AI is research of intelligent behavior in machines, is intelligent behavior of the object created by human, and is a comprehensive cross subject of computer science, psychology, philosophy. From the concrete practice, AI refers to use a computer to simulate, extend and expand people's intelligence, so as to realize certain "machine" thinking, and can take some reasonable activity and behavior.

Artificial intelligence is very challenging and its connotation is very wide, which is composed of different fields, such as machine learning, computer vision, etc. All in all, the goal of artificial intelligence is making computers can simulate human thinking to think.

Game artificial intelligence technology is referred to as Game AI[1]. Artificial intelligence makes the roles in the game very sharp and intelligent; the hidden events and tasks make the game of more playability.

In the 1990s, designers started AI and entertainment software research and development work.

Today, artificial intelligence is the focus of the gaming research. The focus of discussion and debate is often that whether the techniques people used in the game belongs to the AI technology or not, however, many developers now take the AI as the selling point of the game development technology, however, the technology used in the game may not sweeping and profound, and even some is already outdated.

A classic AI system can be divided into four parts:

The first part: the sensory input;

The second part: the memory storage;

The third part: reasoning;

The fourth part: output decision-making behavior.

The specific process is shown in Figure 1.

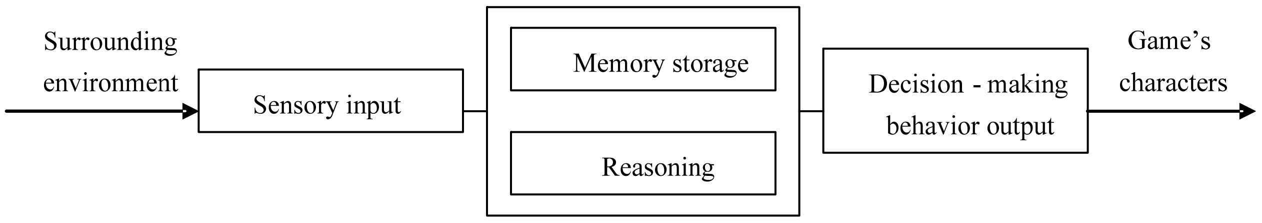

Figure 1 : Game AI system structure diagram

Academic AI and game AI

The academic study of AI emphasizes entities internal mechanisms.

Academic AI research is to clarify the internal operation mechanism in the entity so as to improve algorithm and optimization algorithm, etc when make the internal data structure increasingly reasonable.

But the application of AI in the game world pays more attention to running outside of the real game itself.

If the game players do not feel the progress compared with the old one in the progress of playing games, or even can not feel the design of programmer to intelligent environment and the role inside the game[2].Then however fashionable the designer can feel in the process of designing the game, the value and meaning of improving AI get lost.

The core principle of Game AI design is to occupy as little resources as possible, take the most simplified effective mode to copy a virtual game environment to make players fully feel the mystery of AI technology in game development and design to meet the strong pursuit of virtual world player of the game.

From the above analysis we can see that there is a big difference between the meaning of AI in academic research and purpose in the practical application of the game world.

The technology used in game AI is simple, the AI is basically part of academic AI, namely rational behavior domain; after all game AI so far has not been able to have enough ability to simulate human thinking and behavior thoroughly.

In addition, artificial intelligence is of completely different meaning for game designers and programmers[3]. For game designers, game AI is the part of the most challenging and fuzzy part, and the highest level rules of the game. For programmers, the AI is a technical implementation of the complex rules of the game established by game designer.

LEVEL ARCHITECTURE OF THE GAME ENGINE

Three-layer software architecture of robot and game engine

Because have similar application environment, the game program can use three-layer software architecture that similar to that of a robot system structure. In robotics, the robotic software architecture must decide how to combine the model that reflects control with the model based on thinking. Reactive control is driven by sensor, suitable for real-time low level decisions, and the control thinking type can be applied to global control decisions when making decisions can not experience the whole. Hybrid architecture combines response with thinking in advance. And the most popular hybrid architecture is three-layer architecture; it consists of a reaction layer, an executive and a thinking form. Reaction layer is low level of control provided for robot, with close sensor and action cycle [4]. After executive level received instructions from the thinking level, serialized to reaction layer, playing a role of adhesive between the reaction layer and thinking layer. Thinking layer uses planning to generate global solution of the complicated problem.

Corresponding to the robot's three layer architecture, the implementation of the interface in the game engine can be divided into three layers the lower interface, middle interface and top interface. In addition to distinguish the various control and decision-making level like in the software system of the robot, engine of the three layer interface structure can bring the following benefits:

First of all, make it abstract and wrapper interface. The call interface between the upper interfaces of the lower is transparent.

Second, it promote developers clear division of labor and work together when taking different development tasks within the project teams. Implementation of image and sound, the integration of the artificial intelligence algorithms, animation data integration, the writing of the top game logic, etc., each work is clear in a unique level[5].Which often contain hundreds of large-scale projects, organization and management for the development work is very necessary, because not only designers, art workers and the programmer needs to clear division of labor, the programmer inside also need.

The game AI engine article will discuss in the behind is a module of game engine, its level is completely parallel with the level of the game engine design, and this paper is according to level of structure from low to high, so it is necessary to talk about the level of the game engine design.

Low-level interface

Low-level interface mainly provide basic 3D rendering, the physical calculation, the role of motion control and character animation system, among which the role of motion control and character animation system is closely related to the artificial intelligence.

Low-level interface motion control is the most basic operations; it only controls the role of position, direction and has nothing to do with the role of attribute. And the realization of movement associated with the role attributes, such as the realization of the velocity, acceleration, and even some high-level sports behavior is the task of the middle interface implementation. And this implementation is through the low-level interface call.

Character animation control realizes calling and mixing of animation sequence, completing real credible character animation. Likely, interface at the bottom of the animation control is for a single animation sequence control, should not contain any logical animation sequence organization, which is the task of the middle interface.

Middle-level interface

Middle interface realize the role higher level behavior of interface functions by using low-level interface provided by the movement and animation control function. For example, the user can control the character walk from one place to another place through the middle of the call interface, and character animation in the process of walk is realized by the low-level interface, and its implementation for low-level interface of users is transparent.

High-level interface

Can stipulate only high-level interface include the implementation of the artificial intelligence algorithm, namely the AI programmer can only call interface above middle level. Calling of the high-level interface, for example, can control the role on the enemy, and attack in the process of action, tactics (from a frontal attack, or from the side flank? Use a rifle, or use grenades?) Etc., it is through provides provided by specific artificial intelligence algorithm to call the middle interface functions to achieve.

High-level interface can be provided to the level designers to complete the actual contents of the game to create. These interfaces often contain fewer program logic control, can be used for non-programmers designers, this usually need to implement a simple, or scripting language close to natural language. Some game engine even provides graphical tools to facilitate the work, in this way, the personnel of programming, such as game designers, can also participate in the specific development of the game to reduce the working process.

LOCOMOTION

Motional properties of the role

Locomotion mainly contains two attributes: Location and Rotation. To this, the role of the game can be divided into two types:

1. The role changes the direction of movement by the steering force[6]. Objects such as cars, planes, their the direction of the velocity vector must the same or the opposite compared with start orientation, the direction of the speed can only change by steering force on the head or tail. In particular, some roles ask that only when velocity is not zero, steering force can be valid. For example, for stopped the car, the steering wheel to its operation is of no avail.

2. The movement of free role, such as humans, animals, etc., the direction of the velocity vector can has nothing to do with the direction it toward.

Move randomly

Random motion can cause the feeling of wandering aimlessly. For the role of free movement, velocity vector can continuously produced randomly; role relying on the steering force to change the direction of velocity, can generate random steering force. If use completely random number to control the movement of roles, action seems unstable, the consistency of the stochastic process has the following ways:

1. Use Perlin Noise to use Noise function as a function of time to calculate force needed. But in order to get a random position while using this method, it is large amount of calculation.

2. Add a random number constantly to accumulated variables and use this variable sine as steering force. It not only ensures the chance to the left and right (by sine function) is close and certain consistency (through gradually accumulated), but also keep the randomness (through the random number).

3. Reduce the turning back[7]. An effective method is to keep the record of the location before, avoiding past them again.

4. The projected Target method. Project a target at a random direction. If can project to a place far enough, then marched toward the target. If not far enough, then select a random direction and project again until find the right target. When arrived at the target, do another project.

5. Cast a sphere right in front of the role to randomly generat target location, but the target location is limited in this sphere.

Basic tracking algorithm

1 Basic track algorithm

In the game, tracking and avoid is the most common character of behavior, it is with moving randomly, constituting the mainly used movement patterns in early game.

The most basic tracking algorithm is to constantly increase or decrease the current coordinates of the role to make it close to the target location coordinates. And avoiding algorithm is just the opposite.

This algorithm results appear relatively machinery. As shown in Figure 2, it is the trajectory of a tracker on the two-dimensional plane arriving at fixed target. According to the coordinates of target location, tracker in the beginning, x and y coordinates increase at the same time, forming a straight line trajectory with 45 degree angle coordinate system[8]. After arrived at point A, the x coordinates are the same as the goal, so only increase y, forming a straight line parallel to the y axis trajectory.

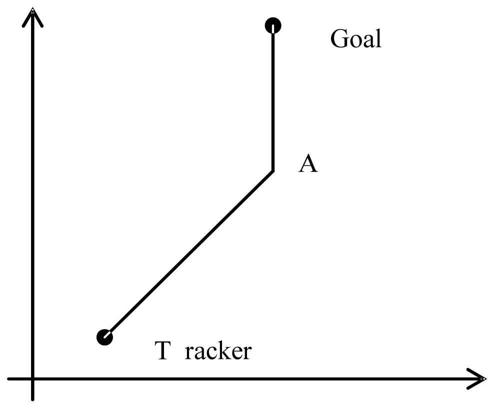

Figure 2 : The result of a basic tracking algorithm

2 Eyesight tracking algorithm

Eyesight tracking algorithm simulates the human behavior to a certain extent, appearing to be more natural.

Algorithm 1 (applied to characters with a steering force):

Set p_1 for the location of the object vector, p_2 position vector for the tracker, order

$$v_1 = p_1 - p_2 \tag{1}$$

Then transform v_1 to the tracker's local coordinate system, remembered as v_1. If coordinate y of v_2 is less than zero then apply left force of steering force, if greater than zero, applied right force of steering force. The z axis is the same way.

The corresponding avoid algorithm by force to take back.

Algorithm 2 (applied to the role of the free movement, change the tracker velocity vector directly):

Set p_1 for the location of the object vector, p_2 position vector for the tracker, order

$$v_1 = f\frac{p_1 - p_2}{|p_1 - p_2|} \tag{2}$$

Among them, the f is a scalar represent mobility, the bigger f is, the faster the steering is. In the end, join v_1 to the role of the velocity vector.

The corresponding avoiding algorithm only need to take negative value to v_1.

Algorithm 2 is shown in Figure 3:

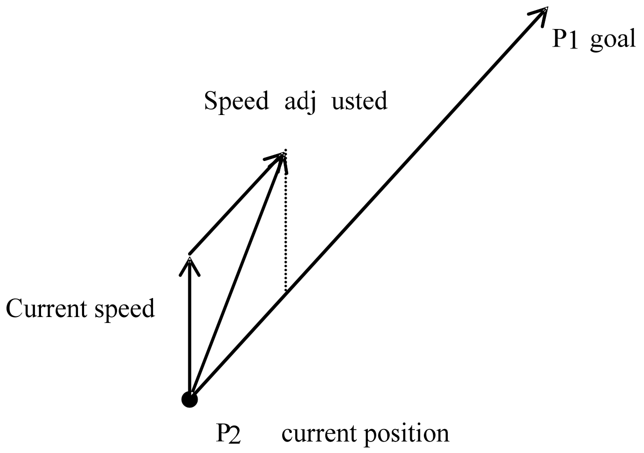

Figure 3: Eyesight tracking algorithm that change velocity vector of the tracker directly

3 Intercept algorithm

An improvement of tracking algorithm is interception algorithm: tracker predict the position of a target will arrive in (interception point), and take track targets as interception point. The simplest realization of computing the intercept point is: if the tracker in front of the goal, the calculated target on the direction of the straight line and tracker position distance nearest point as the intercept point. But the algorithm has a problem[9]. For example, one possible outcome is target tracker get to intercept point in advance. Really feasible algorithm must consider tracker and the target of the relative position and velocity. Algorithm is as follows:

Set p_1 for the location of the object vector, p_2 position vector for the tracker, v_1 as the goal of the velocity vector, v_2 as tracker velocity vector. Order:

$$t_1 = \frac{|p_1 - p_2|}{|v_1 - v_2|} \tag{3}$$

$$t_2 = \frac{1}{2}\left(1 - \frac{v_1}{|v_1|}\cdot\frac{v_2}{|v_2|}\right) \tag{4}$$

Then, position vector of the intercept point is

$$p = p_1 + (t_1 + t_2)v_2 \tag{5}$$

In the formula, t_1 reflect the distance between the tracker and the target and the influence of their respective speed on track time, while t_2 reflect the track of time because of changing direction. Target free to move can not take this factor into consideration, so the formula above can be simplified as:

$p = p_1 + t_1 v_2$ (6)

PATH PLANNING

Basic pathfinding algorithm

Pathfinding algorithm can be divided into two categories: global algorithm and local algorithm. Strategy game usually adopt global algorithm (e.g., A * algorithm), and action games usually use local algorithm. In the simplest case, pathfinding algorithm has the following:

1. When do not need to avoid the obstacles, pathfinding algorithm is track algorithm.

2. Given the obstacles to avoid, the most simple pathfinding algorithm is, project from the position of the tracker to target projection to get a straight line, if there are obstacles in the straight line, move randomly, if no obstacles in a straight line, move along the straight line toward the target. This method is feasible in the open scene, when obstacle quantity is less and the conditions are limited.

3. When there are large obstacles in the scene, you can use the following partial algorithm: from the position of the tracker to target projection to get a straight line, move along a straight line toward the target, if encounter obstacles, move along the edges of the obstacles, until the back line, continue to move along a straight line, as shown in Figure 4.

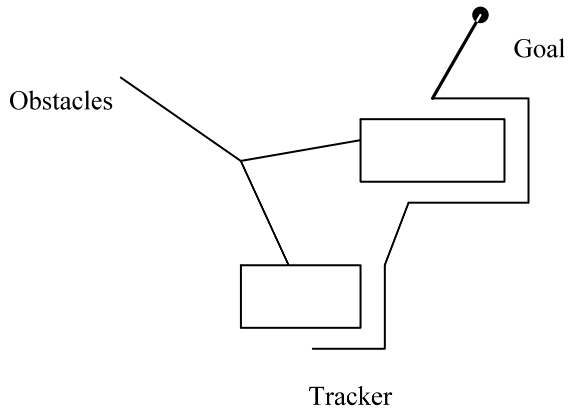

Figure 4 : Path-finding algorithm against obstacles

The improvement of this method is that, when moves along the edges of the obstacle, project to the goal continuously to get new targets projection line, if in a straight line of the next step no fellowship with the obstacles, move along the new line, as shown in Figure5.

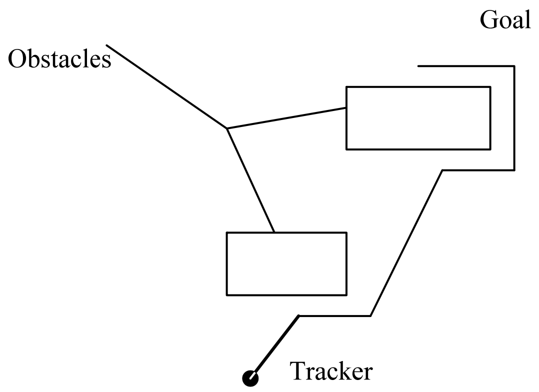

Figure 5 : Improved path-finding algorithm against obstacles

Breadcrumb pathfinding method

Idea of breadcrumbs pathfinding method is that record the path goal through, let mobile tracker get along this path. Method to record the target moving path is to leave some marker (bread) invisible to the player[10]. Tracker can move in different bread crumbs order, for example, every time a search for the nearest bread crumbs, or record left in the data structure of bread crumbs of bread crumbs, tracker order according to the time of bread crumbs to find new bread crumbs (tracker will be completely follow the path of target through. This appear unreasonable in some cases, if the target the obvious step back or circle, the tracker will completely recreate the process), or every time looking for the latest bread crumbs (this allows tracker understand latest location information of the target, in order to find the target) as soon as possible.

Can define the bread crumbs amount limit, beyond the limit is to delete the old bread crumbs, join the latest bread crumbs. Changing the ceiling can adjust the tracker's ability to track targets.

SUMMARY

Due to the development of computer hardware and software and 3D graphics technology, picture expression of computer game software is enhanced, at the same time more system resources are freed again, which makes the implementation of artificial intelligence in the game to be more complex and urgently need. The increase of game software development effort and the difficulty of data integration gave birth to the game engine; the game engine use hierarchical architecture, in order to abstract and encapsulate connecting interface. This, in turn, provides the basis of artificial intelligence and the definition of the problem. This paper first introduces the basic method. In this paper, many spaces are used to describe path planning problem, introduce the application and optimization of path planning and an effective method is proposed. For high-level logic control of game characters, this paper introduces several possible algorithms and tools to use in game, and the feasibility of its application and realization method is discussed. Finally, data driven model of artificial intelligence engine based on finite state machine and scripting system is given in detail.

ACKNOWLEDGEMENT

Henan talent training of joint fund of The National Natural Science Foundation (U1204613)

REFERENCES

[1] Marc Ponsen; Improving adaptive game Ai, With evolutionary learning [D], Netherlands, Delft University of Technology, **(2004)**.
[2] M.Corey, Miller; Evolutionary artificial neural network weight tuning to optimize decision making for an abstract game [D], USA, Air University, **(2010)**.
[3] Ira Pohl; Praetical and theoretical considerations in heuristic search algorithms [J], Problem Solving and Deduction, 55-72 **(1977)**.
[4] J.Stuart, Russell, Peter Norvig; A modern approach of artificial intelligence [M], Beijing, Tsinghua University Press, 59-136 **(2006)**.
[5] Lin Liqun; Design and application of artificial neural network theory [M], Beijing, Chemical Industry Press, 1-67 **(2007)**.
[6] Stuart Russell; Efficient memory-bounded search methods [C], In Proceedings of the 10th European Conference on Artificial intelligence, 1-5 **(1992)**.
[7] Kenneth Rosen; Discrete mathematics and its application [M], USA, McGraw-Hill, **(2002)**.
[8] Mark Weiss; Data structures & algorithm analysis in C[M], USA, Printice Hall, **(2003)**.
[9] Maltin Riedmiller, Thomas Gabel, Roland Hafner, etal; Reinforcement learning for robot soccer [J], Autonomous Robots, **27**, 55-73 **(2009)**.
[10] Jason Gregory; Game engine architecture [M], New York, A.K.Peters, **(2009)**.