Real-time Hierarchical Movement Animation Control for Autonomous Virtual Humans

Weibin Liu, Liang Zhou, Weiwei Xing, and Baozong Yuan

Abstract— **Due to the great diversity of individuals, the variety of behaviors, and the potential complexity of environment, creating autonomous virtual human in realistic virtual environment has been a challenging area in virtual reality research. This paper focuses on autonomous movement animation control for synthesizing the life-like actions of virtual human. A hierarchical structure of autonomous movement animation control is described. Optimized A* algorithm is implemented for pathfinding in the high level; steering control is applied in the middle level for incorporating the surrounding dynamic environment, avoiding obstacles and tending toward safety; collision detection copes with the collisions in physics-level, and skinned mesh animation produces realistic action animations of virtual human. Experiments have been carried out in simulated virtual environment, which demonstrate the robustness and efficiency of the presented approach which makes the autonomous virtual human's movement in virtual environment appear intelligent, smooth and natural.**

Key words—**Autonomous behavior, obstacle avoidance, path planning, virtual human.**

I. Introduction

Combining artificial intelligence with virtual reality techniques to create autonomous virtual human in realistic virtual environment has been a heated topic in the area of virtual reality research in recent years [1-7]. There are two types of virtual humans as defined by N.I. Badler[1]: Agent, a virtual human figure representation that is created and controlled by computer programs or simulator; Avatar, a virtual human that represents and is controlled directly by a real human. Many efforts have been made in the research of virtual human, which mainly include geometry modeling, physics modeling, perception and cognition, decision-making and behavior modeling. Nowadays, there are numerous applications of computer graphics, computer animation and simulation where it is necessary to model life-like virtual humans with high level autonomous behavior control ability, such as human factors analysis, city planning, architecture visualization, military simulations, virtual reality, gaming and other interactive entertainments.

We proposed an integrated AI framework for the perception and behavior decision of autonomous virtual human in [8], as shown in Fig. 1, which mainly implements an Octree frustum culling-based visual perception system and a decision network-based behavioral decision making system. Movement control forms the lowest level of AI technique in the autonomous virtual human model, which is a core component responsible for synthesizing movement behaviors of virtual humans and must be able to get the information about the environment, steer the virtual human's movement to pursue some goal, avoid the obstacles and collisions with other characters, tend toward safety and escape from the dangers. In this paper, we focus on the hierarchical real-time autonomous movement control for synthesizing life-like movement behaviors of virtual humans in virtual environment. Optimized A* algorithm is implemented for pathfinding, steering control is applied for incorporating the surrounding dynamic environment, collision detection copes with the collisions in physics-level, and skinned mesh animation produces the realistic action animation of virtual human. Experiments have been carried out in simulated virtual environment, which demonstrate the robustness and efficiency of the presented approach that enables the autonomous virtual humans' movement animation in virtual environment appear intelligent, smooth and natural.

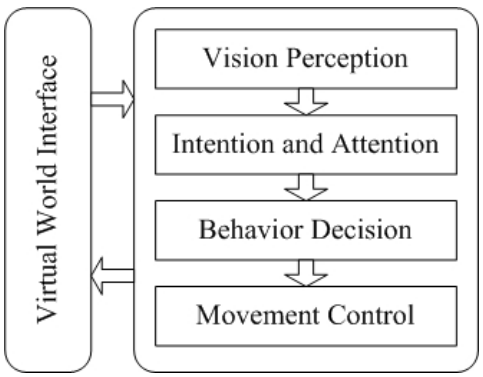

Fig. 1 Integrated AI Framework of Autonomous Virtual Human

Manuscript received January 12, 2010. This work is supported by National Natural Science Foundation of China (No.60801053), Beijing Natural Science Foundation (No.4082025), Doctoral Foundation of China (No.20070004037), BJTU Research Foundation (No.2007XM012), BJTU Scholastic Scholarship Fund, Beijing Excellent Doctoral Thesis Program (No.YB20081000401), China Basic Research Program (973) (No.2006CB303105).

Weibin Liu is with the Institute of Information Science, School of Computer and Information Technology, Beijing Jiaotong University, Beijing, 100044 China (e-mail: wbliu@bjtu.edu.cn).

Liang Zhou is with the Institute of Information Science, Beijing Jiaotong University, Beijing, 100044 China.

Weiwei Xing is with the School of Software Engineering, Beijing Jiaotong University, Beijing, 100044 China.

Baozong Yuan is with the Institute of Information Science, Beijing Jiaotong University, Beijing, 100044 China.

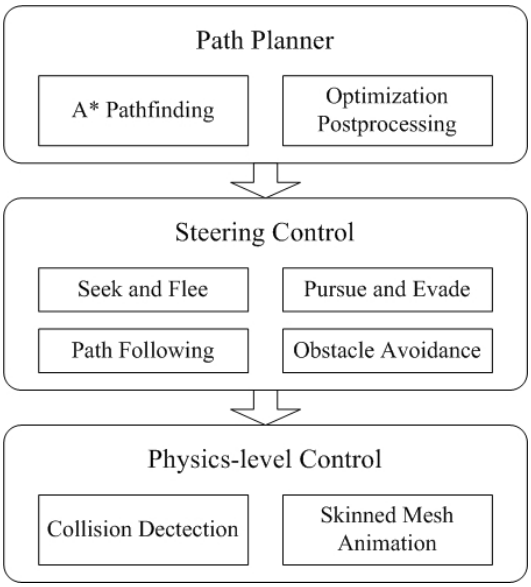

Fig. 2 Hierarchical Movement Control of Virtual Human

II. Hierarchical Structure of Autonomous Movement Animation Control

Fig. 2 illustrates the hierarchical structure of movement animation control for autonomous virtual human.

(1) In the high level, path planner is responsible for determining where the character intends to move. This is often a pathfinding algorithm, generating intermediate targets on the path to a final goal. The popular A* algorithm is chosen for pathfinding search and the optimization postprocessing is applied for smoothing the path to get rid of the zigzag effect produced by the basic A* search.

(2) In the middle level, steering control is implemented for dealing with the dynamic objects which move in the virtual environment over time and the detailed fine obstacle features on the terrain. Steering is responsible for moving the character to its goal by incorporating its surrounding dynamic environment, which avoids the static and dynamic obstacles, and tends toward safety as it moves.

(3) In the bottom physics-level, skinned mesh animation using vertex blending is implemented for driving realistic animations of virtual human and collision detection copes elegantly with the collision in physics-level to avoid the virtual human get stuck on an obstacle.

In the following sections, the details of the hierarchical movement animation control for autonomous virtual human will be expanded.

III. Path Planner

A. A* Pathfinding

To successfully find a efficient path from one location in virtual environment to another, the virtual human must be provided with some search space representation of the navigable virtual world and be able to execute search over the space. A* search algorithm [9,10] is a common solution for the search itself and has been used with great success.

A* algorithm is the most popular choice for pathfinding, which was originally applied to various mathematical problems and was adapted to pathfinding during the early years of artificial intelligence research. A* algorithm is fairly flexible and can be used in a wide range of contexts, which is guaranteed to find the best path from the origin to the destination, if one exists. A* is like other graph-searching algorithms which can potentially search a huge area of the map. It's like Dijkstra's algorithm which can be used to find a shortest path. It's like BFS in that it can use a heuristic to guide itself [9].

B. Optimization Postprocessing of A* Pathfinding

Although A* search is widely implemented in solving pathfinding problems, the basic A* has been found woefully inadequate for achieving the realistic movement required by virtual human. In Fig. 3, the left column shows two examples of the basic A* search in pathfinding, which produces an unfortunate "zigzag" effect. This effect is caused by the fact that basic A* algorithm searches the eight tiles surrounding a tile, and then proceeds to the next tile. This is fine in primitive games where units simply hop from tile to tile, but is unacceptable for the smooth movement required in real-time navigation of virtual human in virtual environment. [11] presented a postprocessing solution for smoothing the path to get rid of the zigzag effect, which takes place after basic A* algorithm has completed its path. The smoothing algorithm is similar to "line of sight" smoothing, in which all waypoints are progressively skipped until the last one that can be "seen" from the current position. The smoothing algorithm simply checks from waypoint to waypoint along the path, trying to eliminate intermediate waypoints when possible. In Fig. 3, the right column shows the optimization postprocessing for smoothing the paths achieved by the basic A* search which are shown correspondingly in the left column.

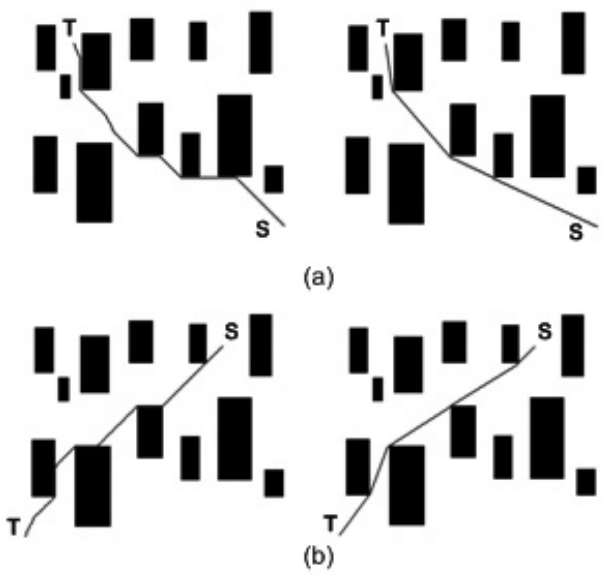

Fig. 3 Optimization Postprocessing of Basic A* Search. The paths in left column shows the results of the basic A* search, and the paths in right column show the results of the smoothing postprocessing. The black blocks represent the buildings on the terrain; "S" denotes the start and "T" denotes the termination for the pathfinding.

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

IV. Steering Control

A. Limitations of Static A* Pathfinding

A* algorithm is a static pathfinding algorithm, which can hardly fulfill the task of planning path dynamically in real time to deal with the dynamic objects in virtual environment. It uses a grid-based map representation for pathfinding, which makes it impossible to mark all the detailed fine obstacle features on the map for the A* search space, such as the trees or the small scale static obstacles in the virtual environment, due to the limited cell subdivision of the search space. Moreover, the unnaturally straight-lines fashion of A* planned path makes things even worse. So the A* path planner only generates the intermediate targets on the path to a final goal.

B. Steering Controls

Reynolds proposed a model for flocking based on local rules, as well as a system for simulating realistic autonomous virtual characters using steering forces [12]. In [12], Reynolds describes the term steering behavior, which addresses "the ability of a character to navigate around their world in a life-like and improvisational manner". Steering is responsible for moving the character to its goal by incorporating its environment. For example, a character has to avoid a collision with an obstacle during walking to his destination.

The steering behaviors are described in terms of the geometric calculation of a vector representing a desired steering force. The common steering behaviors include: seek, flee, pursuit, evasion, offset pursuit, arrival, obstacle avoidance, wander, path following, flow field following, unaligned collision avoidance, separation, cohesion, alignment, flocking and leader following. [13] proposed a family tree of the steering behaviors, which marks a steering behavior as a child of another if it can be seen as extending the behavior of its parent, as shown in Fig. 4. In our system, the following steering behaviors are mainly implemented: Seek and Flee, Pursue and Evade, Path following and Obstacle avoidance. A sidestep repulsion vector [14] is introduced in our steering computation to cope with the problem when the total steering repulsion is roughly opposite to the destination direction.

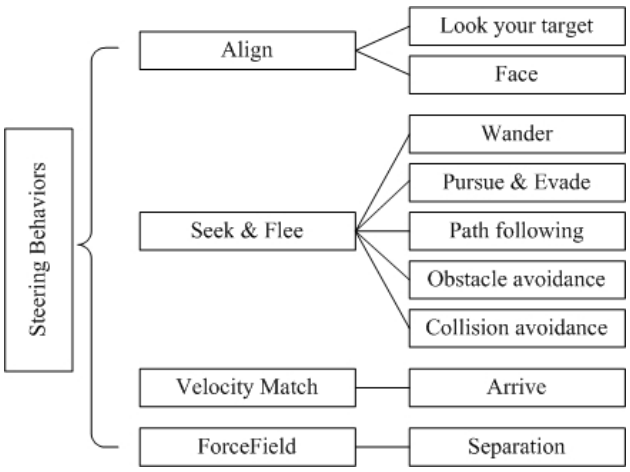

Fig. 4 Steering Family Tree [13]

C. OpenSteer Library Implementation

In our system, OpenSteer Library is implemented for computing the steering behaviors. OpenSteer [15,16] is an open-source toolkit to support construction and simulation of steering behaviors of autonomous characters in games, animation and simulations, which is the C++ implementation of Reynolds' steering model. OpenSteer uses an abstract mobile vehicle model for steering behaviors, which is defined as a point mass approximation with the following properties: mass, position, velocity, maximum steering force (acceleration), maximum speed, etc.

The physics of the abstract vehicle model is based on forward Euler integration. At each simulation step, the currently selected steering behaviors return the desired steering forces limited by maximum steering force and the steering forces are applied to the vehicle's point mass. This produces acceleration equal to the steering force divided by the vehicle's mass. That acceleration is added to the old velocity to produce a new velocity, which is then truncated by maximum speed. Finally, the velocity is added to the old position. In brief, the computation is as following,

steering force = truncate (steering forces, max force)
acceleration = steering force / mass
velocity = truncate (velocity + acceleration, max speed)
position = position + velocity

V. Physics-Level Control

A. Collision Detection

Steering is a practical, fast solution to obstacle and hazard avoidance, but it isn't intended to be entirely fool proof. Occasionally, collisions will occur between entities; for example, if the magnitude of a repulsion vector of steering isn't high enough, or if the specific steering vector is ignored due to the combination strategy of different steering behaviors. A virtual character can also collide with the static obstacles while avoiding dynamic hazards. So the collisions should be coped with elegantly in the physics-level of movement control system by collision detection process in order to avoid the virtual human get stuck on an obstacle.

Collision detection, also known as interference detection or contact determination, is the process of detecting pairs of geometric objects that are intersecting or are within a given proximity of each other [17]. SOLID Collision Detection Library [18] is implemented in our system. SOLID is a software library containing functions for performing intersection tests and proximity queries that are useful for interference detection of multiple three-dimensional objects undergoing rigid motion in the context of collision detection. In SOLID, the motions of objects are controlled by the client application, and the collision response is handled by callback functions. The types of response and the callback functions that need to be executed for each pair of intersecting objects are stored in a response table. Given a scene and a response table, a collision test computes the required response data for all pairs of colliding objects on which a response is defined and passes these data together with the colliding pair to the

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

callback. The response actions are defined by the client application in the callback.

B. Skinned Mesh Animation

For driving the 3D virtual human models to produce the realistic smooth action animation, the skinned mesh animation technique [19-22] is implemented.

Skinned mesh animation is a kind of skeletal animation technique, which provides a natural underlying structure for driving an articulated character such as virtual human. Skinned mesh doesn't suffer the problem of rigid body animation [20]. In skinned mesh animation, portions of the character's skin can normally be associated with multiple bones, each one having a scaling factor called vertex weights, or blend weights. To calculate the final position of the vertex, each bone transformation is applied to the vertex position, scaled by its corresponding weight. This algorithm is called matrix palette skinning [19,20]. Skinned mesh animation combines both advantages from articulated body animation and single mesh blending animation, namely flexibility and smoothness.

In our system, we animate a skinned mesh of virtual human with the "SkinnedMesh" class in the DirectX SDK Library. A virtual human can have multiple animation sequences for his different behaviors. For example, a character may have a walking sequence, a running sequence, an idling sequence, a fighting sequence, and so on. The multiple animation sequence data is represented by animation set data which has been pre-edited and embedded into .X model file of the virtual human. Once the .X model files have been loaded by DirectX API, the animation set is loaded into the animation controller. According to the output name of the behavior control module, the controller switches between the multiple animation sequences, picks up the corresponding animation from the animation set, and plays it. Fig. 5 shows some examples of skinned mesh animations of virtual character.

VI. Experiments and Conclusions

Experiments have been carried out in the simulated virtual environment. The virtual environment comprises terrain and stationary objects. Fig. 6 shows some scenes produced in our experiments. Experimental results prove the sufficiently robustness and flexibility of the proposed hierarchical control approach for producing life-like movement animation of autonomous virtual humans in the real-time interactive virtual environment.

Fig. 5 Examples of Skinned Mesh Animation

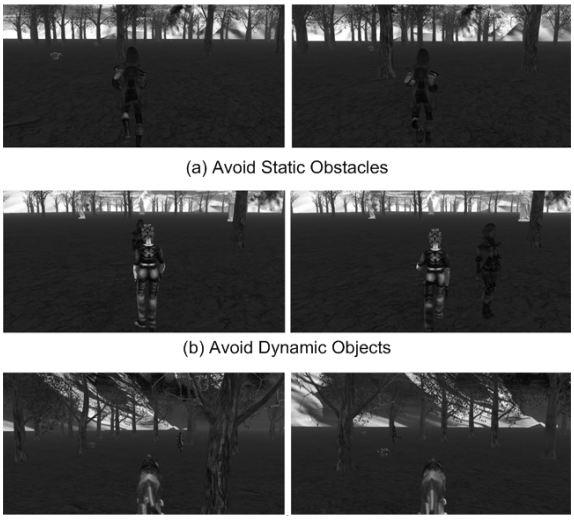

Fig. 6 Scenes of Experimental Results

References

[1] N. I. Badler, "Real-time virtual human," *The 5th Pacific Conference on Computer Graphics and Applications*, Seoul, Korea, Oct. 1997, pp.4-13.

[2] N. Pelechano, J. Allbeck, and N. Badler, "Virtual crowds, methods, simulation and control," *Synthesis Lectures on Computer Graphics and Animation*. Morgan & Claypool, 2008.

[3] SESSION: SIGGRAPH Core: Motion Planning and Autonomy for Virtual Humans, ACM SIGGRAPH 2008 classes, Los Angeles, California, USA, Aug. 2008.

[4] Q. Yu, D. Terzopoulos, "A decision network framework for the behavioral animation of virtual humans," *ACM SIGGRAPH / Eurographics SCA 2007*, California, USA, 2007, pp. 119-128.

[5] W. Shao, D. Terzopoulos, "Autonomous pedestrians," *ACM SIGGRAPH / Eurographics SCA 2005,* California, 2005, pp.19-28.

[6] S. Kshirsagar, N.M. Thalmann, "A multilayer personality model," in *Proceedings of 2nd International Symposium on Smart Graphics*, Hawthorne, New York, USA, June 2002, pp.107-115.

[7] S.J. Rymill, N.A. Dodgson, "Psychologically-based vision and attention for the simulation of human behaviour," *the 3rd Int. Conf. on Computer graphics and interactive techniques in Australasia and South East Asia*, Dunedin, New Zealand, 2005, pp.229-236.

[8] L. Zhou, W.B. Liu, W.W. Xing, and B.Z. Yuan, "Autonomous perception and behavior control for virtual human," *The 9th Int. Conf. on Signal Processing*, Beijing, China, Oct. 2008, pp.1396-1399.

[9] Amit's A* Pages. Available: http://theory.stanford.edu/~amitp/GameProgramming/

[10] P. Lester, "A* pathfinding for beginners," Updated July 2005. Available: http://www.policyalmanac.org/games/aStarTutorial.htm

[11] M. Pinter, "Toward more realistic pathfinding", 2001. Available: http://www.gamasutra.com/

[12] C. W. Reynolds, "Steering behaviors for autonomous characters," *in Proc. GDC 1999*, San Jose, CA, USA, 1999, pp.763-782.

[13] I. Millington, J. Funge, *Artificial Intelligence for Games. 2nd ed.* Morgan Kauffman Series in Interactive Technology, 2009.

[14] G. Johnson, "Avoiding dynamic obstacles and hazards," *AI Game Programming Wisdom 2*. Charles River Media Inc. 2004, pp. 161-170.

[15] The OpenSteer Community. http://opensteer.sourceforge.net

[16] S. Stiefel, "Parallelizing the OpenSteer toolkit for simulation of steering behaviors of autonomous characters using GPU," Available: http://www.stiefels.net/wp-content/

[17] M. Lin, S. Gottschalk, "Collision detection between geometric models: a survey," in *Proc. of IMA Conf. on Math. of Surfaces*, 1998.

[18] *User's Guide to the SOLID Collision Detection Library for version 3.5*. DTECTA. Available: http://www.dtecta.com/

[19] T. Akenine-Möller, E. Haines, *Real-Time Rendering: 2nd ed.* A.K. Peters, Ltd. Natick, MA, USA. July 2002.

[20] F. Luna, "Skinned mesh character animation with Direct3D 9.0c, " Retrieved December 15, 2004, Available: http://www.moon-labs.com/resources/d3dx_skinnedmesh.pdf

[21] A. Thorn, *DirectX 9 Graphics: the Definitive Guide to Direct3D*. Jones & Bartlett Publishers, April 2005.

[22] *DirectX Software Development*. Microsoft cooperation

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)