Appeared in Proceeding of 2011 Conf. on Computer Vision, Graphics, and Image Processing, Aug. 2011.

SMART CHARACTER IN A FIRST-PERSON SHOOTING GAME

[1]Chun-Chieh Chen (陳俊傑), [1]Tsai-Yen Li (李蔡彥)

[1]Computer Science Department,
National Chengchi University, Taipei
E-mail: {s9425,li}@cs.nccu.edu.tw

ABSTRACT

Controlling a 3D character in a virtual world in real time is a challenging problem because the complexity of human motions and the limited time available for computation. Consequently, most 3D games use a simplified mechanism to place canned motions at designated locations regardless what a user inputs. In this paper, we aim to design a 3D character that is capable of generating 3D motions dynamically according to the inputs of the user as well as the constraints of the environment. The motions are synthesized at real time from a motion library organized in a motion graph. The traversal of the motion graph is represented and incrementally maintained in a data structure called *Feasible Motion Tree (FMT)*. By designing appropriate tree expansion and motion selection algorithms, a useful FMT can be maintained incrementally within a limited time budget. We use a first-person shooting game as an example to demonstrate the effectiveness of this method. We think that game designers can make good use of the enhanced interactivity with the smart character to design more enjoyable games.

Keywords *intelligent avatar, real-time 3D interaction, feasible motion tree, time-budgeted computation*

1. INTRODUCTION

Interactivity is a critical issue in designing a good game or similar virtual environments. As the computer hardware is continuously improved, more computing power can be invested in interactivity in addition to graphics rendering. In a game, the interaction design is typically part of game logics and usually is scene dependent. For games with virtual characters, this could mean that how the characters interact with a user is specified at design time. Consequently, the characters in a game usually can only display canned motions at designated locations of a scene. A new user may find the game entertaining because he/she cannot predict the actions of the characters. However, after several runs of practice, the users may get bored easily because these actions become predictable. Therefore, it is highly desirable to have a more intelligent character that can plan its motions according to the inputs from the user as well as other possible constraints from the environments.

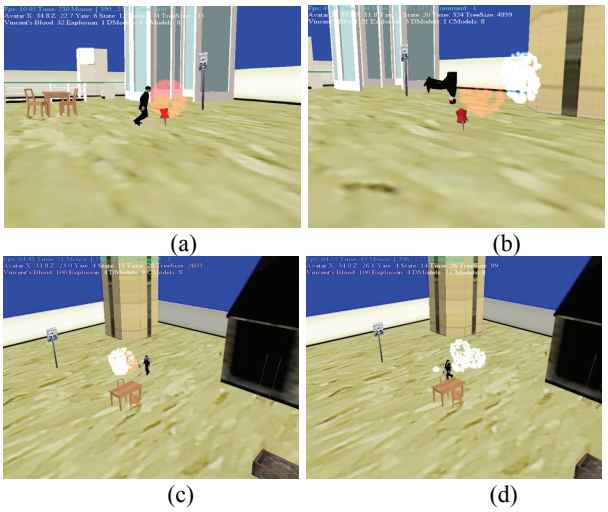


Fig. 1. (a) and (b) are snapshots of the implemented system showing the character taking actions (side move and leap over, respectively) to escape the bullets (in red and grey). (c) and (d) show the snapshots of the character in action before and after escaping from a group of bullets. The smoke cloud was caused by the bullets hitting the ground.

In this paper, we aim to design practical data structures and algorithms to realize a smart character that can choose appropriate motions from a motion library in reaction to the user inputs. The motion planning ability of the intelligent character is demonstrated through a first-person shooting game where the user can move and shoot the computer-controlled character freely. Some snapshots of the game in action are shown in Fig. 1. The character will strive to avoid being shot whenever possible by selecting appropriate captured motions organized in a motion graph. Since the available time in each frame for selecting an appropriate motion is limited by some desirable frame rate, the motion graph is

traversed and maintained incrementally in a data structure called *Feasible Motion Tree*. Strategies for maintaining this tree and selecting the promising motion will also be described in this paper. Experimental data obtained from the implemented game will be reported to illustrate the effectiveness of the mechanism.

2. RELATED WORK

In the literature of computer animation, motions of human figure can be generated with three categories of approaches: *key-framed*, *dynamics*, and *example-based*. Among these approaches, the example-based animation is the most popular approach in recent years. It usually refers to the animations acquired by the technique of motion capture. Because the captured motions are reproductions of real human motions, they are the most realistic. However, the captured motions lack flexibility in situations where the human model or the environment is different from the time when the motions were captured.

Making captured motions more reusable for different characters and environments is the focus of most recent work in the computer animation literature. For example, space-time constraints or optimal control strategies are often used to retarget the motions to a new character [3][4]. Another common objective of this approach is to synthesize a new sequence of motions from a motion library. For example, Kim et al. used signal processing techniques to segment motions into elementary clips that can be used to synthesize new motions [6]. Gleicher organizes elementary motion clips into a motion graph that connect nodes (motion clips) with similar connection frames and use motion wrapping and blending techniques to generate smooth transition between two motion clips [7]. Lee et al. [8] organized the motion clips in a two-level graph with motion groups in the higher level and individual motion clips in the lower level.

Controlling the motion of a 3D character is rather difficult given the control devices, such as keyboard and mouse, available on common desktop computers. Therefore, several intelligent user interfaces have been proposed to make motion control for 3D character easier. For example, in [5], a real-time motion planning algorithm was used to generate compliant motions for the animated character to avoid obstacles. The motions were generated at run time but limited to the upper-body motions that were generated procedurally. Lee et al. [8] used three types of interfaces (menu, sketching, and images) to help the user select appropriate motion clips from the motion library. In [1], the idea of using feasible motion tree has been proposed to generate feasible motions for a 3D character to navigate through a static environment with controls from a third-person view. In this paper, the character is controlled by the computer (with the second-person view), and the user plays the role of adversary to create dynamic obstacles for the character to escape.

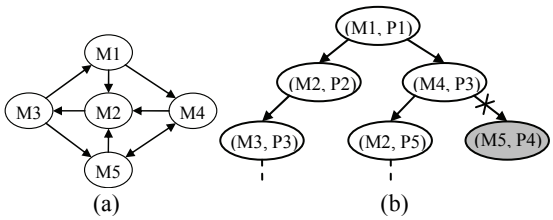


Fig. 2. Example of (a) motion graph and (b) feasible motion tree.

3. SYSTEM OVERVIEW

We first define the problem of generating the motions of a character in a shooting game considered in this paper, describe what feasible motion tree is, and then give an overview of the system by describing the flowchart of the system as well as its components.

3.1. Problem definition

In the experimental scenario of a shooting game, we assume that we are given a geometric description of the static objects in the virtual environment. The kinematics and geometric data of the virtual character are also given. A set of motion clips organized in a motion graph [7] are also available for the character to choose. Dynamic objects of known geometry such as bullets and howitzers fired from the user's first-person view at run time when the user clicks on the mouse. The dynamic objects will use a constant velocity and follow a straight-line path after they are fired. The user can move his position and orientation (x, y, z, tilt, and pan) in the 3D environment by pressing designated keys on the keyboard. The objective of the computer-controlled virtual character is to make good use of the motion clips in the motion graph to escape from the dynamic objects whenever possible while avoiding penetrating the static obstacles.

3.2. Definition of Feasible Motion Tree

The system makes use of a data structure proposed in [1] called *Feasible Motion Tree* (FMT) to store the motion clips that are viable in the near future. The feasible motion tree is the result of traversing a motion graph from a given position of the character. For example, assume that the motion graph of the character is given in Fig. 2(a). Each node in the graph represents a motion clip (denoted by *Mi*) while each directed arc represents a feasible transition from the source clip to the destination clip. A node in the FMT represents execution of a motion clip at a given position (denoted by *Pj*). For example, in Fig. 2(b), the character is currently executing the motion clip of M1 at position P1. The current node is then denoted by (M1, P1). According to the motion graph in Fig. 2(a), the feasible motion clips from M1

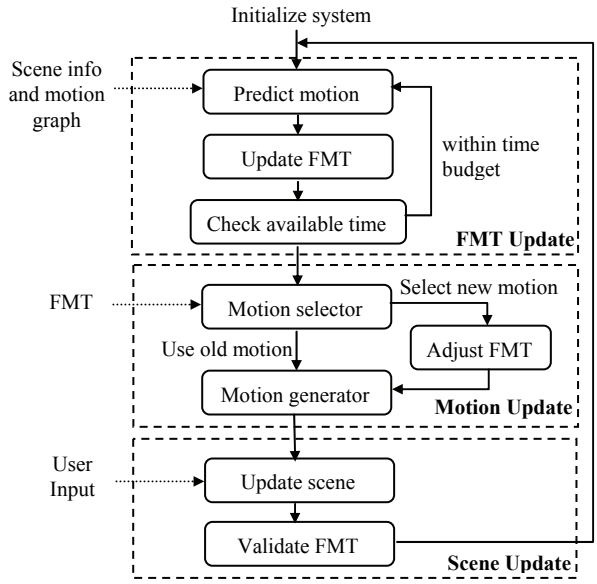


Fig. 3. System flowchart.

include M2 and M4. Therefore, the children of (M1, P1) consist of (M2, P2) and (M4, P3). However, not all motions are feasible due to the constraints of the static and dynamic objects in the environment. For example, in Fig. 2(b), the node (M5, P4) is marked forbidden after performing collision checks. Note that the feasibility of a motion clip depends on the time and the location where the motion is performed.

3.3. System architecture

For the given problem in a shooting game, we have designed a system to realize an intelligent character that can take appropriate actions in response to user inputs. As shown in Figure 3, the system consists of three modules: FMT update, motion update, and scene update that are gone through sequentially in each loop of a game. In the FMT update module, the FMT is maintained with a best effort according to the available time computed from the desirable frame rate specified by the user. Therefore, in this module, whenever the time is not used up, the system will try to grow the FMT as much as possible according to an exploration strategy described in the next section. In the module of motion update, if the current motion clip is used up and a new clip needs to be selected, the motion selection mechanism will be triggered. The motion selection strategies will be described later. If a new motion is selected, the FMT will be updated accordingly. In the module of scene update, the list of dynamic objects will be updated according to status of the existing objects and the new inputs, if any, from the user. The current FMT will then be updated according to the new status of the dynamic objects.

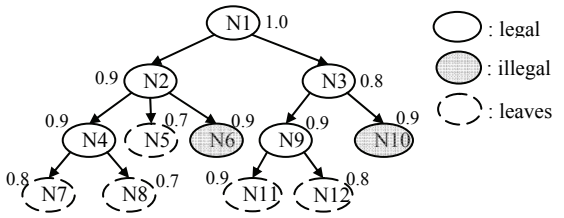


Fig. 4. Example of node priorities in the feasible motion tree. The number beside each node is the base priority.

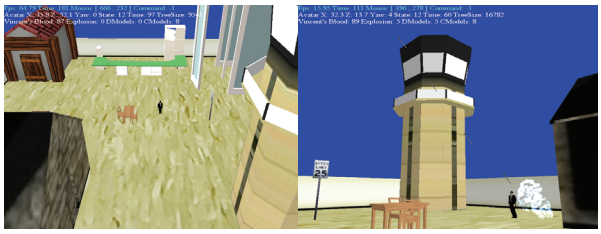
4. MAINTAINING FEASIBLE MOTION TREE

As mentioned in the previous section, the feasible motion tree is maintained in an incremental fashion with best efforts in a given time budget of each frame update. It means that the system needs to make good use of the given time budget to update the FMT. Therefore, a good exploration strategy that can grow the most promising portion of the tree would be important. Similarly, given a FMT, how to select a child motion clip that is the most promising for avoiding future collisions with the static and dynamic objects is also critical. Therefore, two key components in maintaining a good FMT are tree exploration strategy and motion selection strategy as described below.

4.1. Exploration strategy

Common strategies for exploring a tree include the depth-first, the breadth-first, and the best-first strategies. First, since the tree exploration process is like a search without a definite depth in our case, the depth-first search cannot be applied. Second, although the breadth-first strategy can be used, the fact of treating all motions equally may not result in the most effective exploration. According to [1], the best-first strategy with node preferences computed from statistic data or by matching user commands is more effective than the breadth-first strategy. In the application of a shooting game, since the user does not command the character directly, matching user commands is not applicable either. Instead, from the view point of the virtual character, the cost of executing a motion clip may vary greatly according to the nature of the motion. For example, taking a side walk will definitely use more energy than taking a leap-over motion.

Therefore, we have used energy consumption estimated for each motion to compute the base priority (denoted by $p_b(n)$, $0<p_b(n)<1$) of each motion clip n. The exploration priority of a node n (denoted by $p(n)$) in a FMT is computed as the product of its base priority and the exploration priority of its parent $p(parent(n))$. That is, $p(n)=p_b(n)*p(parent(n))$. Because the base priorities are equal to or less than one, the lower the node in the tree, the less exploration priority it will get. Thus, among the leaf nodes of the FMT, the node with the highest priority is always selected for further exploration when the time allows. An example of how a FMT



(a) (b)

Fig. 5. Experimental scene from (a) the first-person view and (b) the third-person view

is grown is shown in Fig. 4. The base priority of each node is indicated beside the node. The indices of the nodes in this example indicate the order of their exploration time.

4.2. Motion selection strategy

The second key component is the motion selection strategy. In the motion update module of the system described in the previous section, if the current motion clip has come to the end, a new motion clip will be selected from the FMT for execution. We think the criteria for selecting the next motion clip should be based on the exploration result of FMT. In a FMT, each legal leaf node means that there exists a path of feasible motions that can bring the character from the root to this leaf. The longer the path, the further in the future the motions are considered. Thus, we design the goodness of a path (denoted as $g(P)$) as the sum of the exploration priorities $p(n)$ for the nodes along the path P. A simple strategy for selecting the next node could be selecting one along the best path. However, this simple strategy only considers depth and disregards the breadth of the grown FMT, which could mean other alternatives of collision-free motions. Therefore, in our approach for each chide node, we take the sum of $g(P(Ni))$ of all the viable paths branching from the given child. Taking the scenario in Figure 4 as an example; each of the five legal leaf nodes (N7, N8, N5, N11, and N12) entails a path $P(Ni)$ to the root. Three of them are from N2 while two of them are from N3. In this case, the sum of g's for all the viable paths from N2 is larger than the sum from N3. Thus, N2 will be chosen as the next node for execution.

5. EXPERIMENTAL RESULTS

We have implemented the smart character in a shooting game with the language of C++ and the DirectX library on the Windows platform. The motion graph used in our experiments consists of 22 motion clips converted from the motion library provided by CMU graphics laboratory[2]. The collision detection between the character and the objects in the environment is performed on every frame of a motion clip. The collision detection algorithm was implemented based on the method of axis-aligned bounding boxes (AABB) [9].

(a) (b)

Fig. 6. Snapshots showing the traces of the motions maintained in the FMT at different situations.

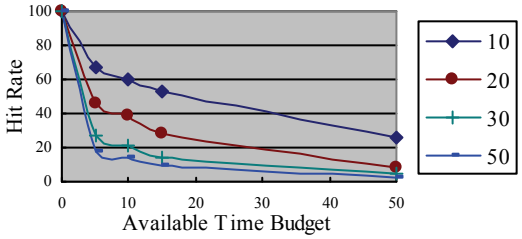


Fig. 7. Effects of shooting distance and time budget on hit rate.

The experimental scene that we have used in the game is shown in Fig. 1 and Fig. 5. The user uses mouse movement to orient the gun and mouse buttons to fire howitzers or bullets to the character in the scene. The user can also use the keyboard to change the location of the weapon in three dimensions. The computer-controlled character will strive to avoid being shot whenever possible. When the distance between the weapon and the character is close, it is unavoidable for the character to be shot. In the game, the blood of the character will be decremented according to the number of frames that a bullet takes to penetrate the body of the character. The game is over when the blood has decreased to zero. In Fig. 1(c) and 1(d), we show two snapshots taken before and after a group of dense bullets were fired. The bullets may be too small to visualize when they are far away from the camera in the snapshots. Nevertheless, we can tell their existence by the smoke clouds caused when they hit the ground. In Fig. 6, we show the traces of the motions maintained in a typical FMT in two different situations. The number of nodes in FMT may vary greatly according to the amount of given time budget in a frame. We have invited several users to test run the game, and all of them found it more than interesting than most commercial shooting game because the motions of the character is more versatile and unpredictable.

In order to verify the correctness of the system, we have conducted several experiments to compare the hit rates of the fired bullets under different system parameters. In order to control the frequency and precision of the gun firing in each experiment, we have used the same set of simulated firing in different experiments. For example, in Fig. 7, we show how the distance be-

tween the character and the weapon affect the hit rates. As one may expect, the longer the distance, the more time the character has to escape the bullets; thus, the lower the hit rate. Different curves in Fig. 7 represent different time budgets. As one can expect, the longer the time budget, the lower the hit rate. Since the computation time in each frame is controlled to be within the given time budget, the performance of the system can degrade gracefully even on a slow machine. In fact, the tree exploration and motion selection strategies used to update the FMT are becoming more crucial when the computation time is limited.

6. CONCLUSIONS AND FUTURE WORK

The complexity of human motions has hindered the development of real-time control of smart characters in virtual environments such as interactive 3D games. Allowing the characters to be able to response to the inputs of the user would enhance the interactivities and fun of a game. In this paper, we have used the data structure of feasible motion tree to store the feasible motion clips of the character in the near future. The tree is maintained incrementally at run time under a given time budget in each frame. An efficient exploration strategy and a motion selection strategy have also been introduced. These techniques have been demonstrated through a first-person shooting game. The developed system has been extensively tested, and its correctness has been verified with several experiments using different system parameters.

Further experiments for verifying the effectiveness of the tree exploration and motion selection strategies are undertaken. We are also working on a mechanism to refine the motion graph at run time such that more diverse motions can be synthesized from shorter motion segments in order to avoid being shot. The game could also be made more entertaining if the character can be equipped with more motion abilities from the motion library. In addition, the character could also be designed to take advantage of the obstacles in the environment to avoid being shot. We believe that this kind of techniques will not only bring up the level of interactivity in a game but also enable new types of 3D interactive games to be developed in the future.

ACKNOWLEDGEMENT

This work is supported in part by National Science Council under contract NSC-99-2221-E-004-008.

REFERENCES

[1] C.C. Chen and T.Y. Li, "Intelligent Third-Person Control of 3D Avatar Motion," in Proc. of the 7th International Symposium on Smart Graphics, 2007.

[2] CMU Graphics Lab Motion Capture Database. Access in 2007. http://mocap.cs.cmu.edu/.

[3] M. Gleicher, "Motion Editing with Spacetime Constraints," in *Proc. of the 1997 Symposium on Interactive 3D Graphics*. 1997.

[4] M. Gleicher, "Retargeting Motion to New Characters," *Computer Graphics* (*SIGGRAPH 98 Proceedings*), pp.33-42. 1998.

[5] H.W. Hsu and T.Y. Li, "Third-Person Interactive Control of Humanoid with Real-Time Motion Planning Algorithm," in *Proc. of IEEE International Conf. on Intelligent Robots and Systems,* 2006.

[6] T.H. Kim, S. I. Park, and S. Y. Shin, "Rhythmic-Motion Synthesis Based on Motion-Beat Analysis," *ACM Transactions on Graphics*, 22(3):392-401, 2003.

[7] L. Kover, M. Gleicher, and F. Pighin, "Motion Graphs," in *Proc. of SIGGRAPH 2002*, 2002.

[8] J. Lee, J. Chai, P. Reitsma, J.K. Hodgins, and N. Pollard, "Interactive Control of Avatars Animated with Human Motion Data," in *Proc. of SIGGRAPH 2002*, 2002.

[9] G. Van Den Bergen, "Efficient Collision Detection of Complex Deformable Models Using AABB Trees," *Journal of Graphics Tools*, 2(4):1–14. 1997.