Asian Journal of Information Technology 15 (24): 5206-5210, 2016
ISSN: 1682-3915

A Novel Approach for Rendering Interactive Animation Using Non-Deterministic Finite State Transitions and Warshall Algorithm

D. Narashiman and T. Mala
Department of Information Science and Technology, CEG, Anna University, Chennai, India

Abstract: This approach gives a new dimension for rendering interactive animation using Non-deterministic Finite Automaton (NFA) and Warshalls algorithm to enhance reusability, smooth transition and fast rendering of the animation frames. In general an animation can be broadly classified into two type's static and dynamic animation. Dynamic animation or interactive animation scenes changes instantly depend upon the user interaction. Designing interactive animation is more challenging and time consuming. In this approach each animation action are represented as individual states in state transition diagram. Depending on the user input and present state the animation activity changes dynamically from current activity to the target activity. In all these transition the intervening states form the current activity to user initiated activity is determined by using warshall algorithm. This approach avoids the problem of specifying the intermediate states statically and also enhances the reality of the animation and it as very useful in complex application like generation of automatic animated sign gestures for the given sentence.

Key words: Face animation, interactive animation, non-deterministic finite automaton , sign gestures, warshall, state sequences

INTRODUCTION

Tomlinson (2005) categorize animation into linear or static and interactive animation and also list the difference between them. The motion pictures, cartoon movies come under the linear category and computer games are an example for interactive animation. The difference between them depends on their adoptive ness, rendering, transition, interaction and movement. The process of creating interactive animation with human model or avatar is laborious and also depends on the human behaviour. In order to render the human movement clearly the animator and programmer should avoid complex animation design, unnecessary and unrealistic movement of humanoid but consider minor details like facial expression and body language. But, developing these minor details is tough and needs a wide knowledge about the human behaviour. Minor details are designed as a separate frame and reused in the animation sequence whenever needed to reduce the designing time. Badler *et al*. (2002) developed a system EMOTE for representing and parameterize agent behaviours, however this system records the minor detail but it fails to detail personality of eye movements, head movements, gait and communal behaviours. One of the predominate domain where these minor details plays an important role is automatic generation of animated sign gestures. The sign gestures are basically depends on the hand shape, hand location, hand orientation, hand movement and facial expression, hence all the details with respect to these components should be carefully rendered. Papadogiorgaki *et al*. (2006) proposed a system VSigns that render sign gesture from sign notation but it not able animate sign such as contact gestures. FootSee an interactive animation system proposed by Kang Kang Yin and Yin and Pai for tracking full body motion based on the foot movements. But, the system works for fixed foot direction.

Wu (2012) used a graphic function to test finite state machine which connects the human action and implement interactive character. The limitation of the work is that it does not facilitate animation reusability. Liu and Popovic (2002) describe a novel method for constraint detection method that automatically determines and analysis input motion and render animation realistically. The motion signal processing is also used for animation by Bruderlin and Williams (1995). Watanabe *et al*. (2013) proposed a system for interactive animation software which run parallel threads along with the main thread in order to parallelize interactive animation, this also reduce the rendering time and render a smooth and effective interactive animation. Lau and Kuffner (2006) created a novel behaviour planning approach which develops realistic motions for animated character. However this method depended on more number of details as inputs.

Corresponding Author: D. Narashiman, Department of Information Science and Technology, CEG, Anna University, Chennai, India

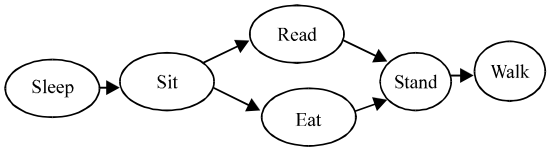

Fig. 1: Sequence of human natural action

Most of pervious research use Finite State Machine (FSM) transitions to fix a particular real life human activity for example walk. The walk cycle consists of following action moving right leg, moving left leg, moving right arm, moving left arm and moving whole body. Each action is a state in the finite state transition. When more than one human activity like walk, sleep, run are to be render jointly each activity should be blended properly and the transitions between the activity should be clear if there exits multiple paths from one activity to another activity.

Figure 1 is a sequence of human natural action. If sleep is the current state and walk is the end state, the human does not directly achieve this end state from the current state. Either the sequence sleep, sits, read, stand and walk or after sit it can be eat, stand and final state. In this type of transition FSM is not effective so NFA is used. To enhance the reusability of animation frame and to overcome the above mention problem a novel approach is proposed in this study. This approach applies the concepts of non-deterministic finite state transition along with warshall's algorithm to enhance the blending function and transit from current activity to next activity and another leverage of this approach is determining intermediate states dynamically.

MATERIALS AND METHODS

System architecture: The important aspect of interactive animation is SAS (Semblance Affinity Sensation). Semblance-an out ward appearance or form, affinity-liking and similarity between the animation sequences. Sensation-impressive and conscious. The core application of this approach concentrates on the above aspects and reusability to generate dynamically realistic sequences between the current and final activity. It is difficult to predict the sequence of actions that a character may take and also a particular action to be done at that specific time. The transition also should be robust, sensible and rapid. The mentioned difficulty can be overridden and the animation sequences are generated dynamically using Warshall's transitive closure algorithm and the state transition matrix.

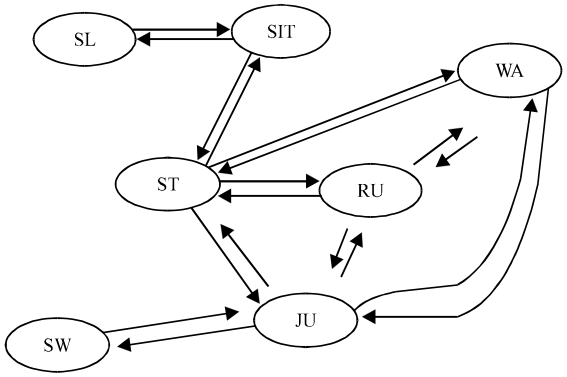

Fig. 2: A state transition diagram depicts the connection between various human activities

This architecture is two fold: first animation sequence cycle is determined and modelled. Second the state transition diagram is generated depending on the animation sequence cycle. When two are three activities for example human behaviours like sleeping, jumping is combined in a sequence, the sequence is called as animation sequence cycle. In addition to the transition form one activity to another activity, animation sequence cycle should also show various human behaviours or actions.

In order to develop a good animation sequence cycle the animator and the programmer should have a vast knowledge about human behaviours. Initially the major activities in the animation sequence cycle are decided. After that the animation sequence cycle is determined. These activities are broken into basic actions for example sleeping activity is separated into sitting, lying down action. Each action is a state in the state transition diagram. In animation these states are the role or frame of the character. The connections between the actions are determined by the edges which are also bi-directional. Some of these basic actions will be common to many major activities. Edges connections are more important in determining the transition from one activity to succeeding activity. Care should be taken in fixing the state and edges, as some of the action will be common to many activities in that case a single state is used to avoid redundancy. The transition from one state to another is determined by the user input and the present state. For a single user input there may exit transitions which branch to multiple next states. The major concern is that when transition is processed through particular edge a correct sequence should be selected dynamically from the present action to next action. The scenario is illustrated in the following Fig. 2. Here, there exits edges from stand to sit, walk, run and jump. Depending on the user input and

present state the transition take place dynamically. Where SL-Sleep SIT-Sit, WA-Walk, ST-Stand, RU-Run, SW-Swim and JU-Jump.

To adopt this situation the proposed system uses NFA and it also generalised to accept different state transition diagram but the animation sequences are pre-determined and character modelling done separately. The NFA is a 5-tuple $\{Q, \Sigma, \delta, q_0, F\}$, the parameters of concern are Q: finite nonempty set of action or state, Σ-finite nonempty set of user input, δ: a transition function mapping from $Q \times \Sigma$, q_0: belong to Q and it is the initial state, F is a subset of Q and it is the final state. The advantage of NFA over the FSM which are used in previous work is that in NFA the outcome of a state is a subset of Q whereas the outcome of FSM is element of Q.

From the animation sequence cycle the state transition diagram is drawn. Using state transition diagram the programmer develops a state transition matrix depending upon the state transition sequences. If there exit a transition between two adjacent states then 1 is marked in the corresponding ith row and jth column if there is no transition then 0 is marked in the corresponding row and column position. Using this state transition matrix the animation sequence is dynamically generated. The state transition matrix is a square and transitive closure matrix. The overall system design is shown in Fig. 3.

The menu module list various state for accepting the user selection. The user selection is processed in state

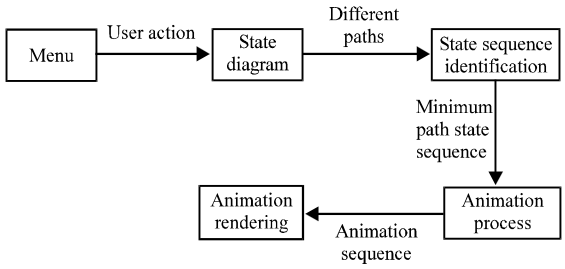

Fig. 3: System architecture

diagram module. The state diagram gives the connection between different state, when a state is said to be connected to another state iff there exists a transitive closure relation. If state A has a directed edge to state B and state B has directed edge to state C by transitive closure relation A is connected to C. There exist many transitive relations but the shortest transition among them is interesting that is many actions can be connected to many other actions but the transition should be minimum, sensible and robust in order to achieve reusability and minimum throughput. The fact is that in the state transition diagram edges are not assigned with any weightage or priority only the connection between the states is established.

Using the transitive closure matrix all possible transitions from current state to user initiated state is listed. The length of each path from source to target is calculated and minimum is selected from them using warshall's algorithm. This shortest path is stored and the corresponding actions in the sequences are rendered. This animation sequence is the final output and in which intermediate states are selected dynamically. The animation of different role is rendered using Opengl. It is basically modular language with heap of graphical in-built commands. It is easy to render human role using Opengl.

RESULTS AND DISCUSSION

Performance evlautions: Five different start state to target state are selected and the transitions sequences and verified. The frame per second and CPU utilisation time are observed. The average frame per second from source state to target state is less and CPU utilisation time is more compared to tradition approach because of finding the shortest path and intermediate sequences dynamically. The different test cases are listed in Table 1 along with the CPU utilization and frame/sec. Table 2 and 3 show the FSM animation performance and NFA animation performance, respectively.

Table 1: CPU utilisation time and frame per second

Test case	Current process state	Current input state	Acceptance state	Expected sequence	Actual sequence	Average CPU utilization	Frame/sec
1	Sleep	Sit	-	Sleep-sit	Sleep-sit	13.79545	2.386364
2	Sleep	Walk	Sit, stand	Sleep-sit stand-walk	Sleep-sit stand-walk	16.82143	2.392857
3	Walk	Swim	Jump	Walk-jump-swim	Walk-jump-swim	21.66346	2.826923
4	Swim	Sleep	Jump, stand, sit	Swim-jump-stand-sit-sleep	Swim-jump stand-sit-sleep	18.79381	2.072165
5	Walk	Walk	-	Walk	Walk	19.91176	2.191176

Table 2: FSM animations performance

FSM_ANIMATION

Time (sec)	Frames/sec	OGL calls/frame	CPUs average utilization	Time (sec)	Frames/sec	OGL calls/frame	CPUs average utilization
0.0	0	0					
0.2	3	1807039	40	7.0	3	1807032	0
0.4	3	1807039	67	7.2	3	1807032	29
0.6	4	1807040	21	7.4	2	1807032	6
0.8	5	1807031	29	7.6	2	1807032	56
1.0	5	1807031	19	7.8	2	1807032	0
1.2	5	1807031	29	8.0	2	1807032	31
1.4	5	1807031	40	8.2	2	1807032	0
1.6	5	1807031	31	8.4	2	1807032	38
1.8	5	1807031	0	8.6	2	1807032	2
2.0	5	1807031	0	8.8	2	1807032	23
2.2	5	1807031	0	9.0	2	1807032	2
2.4	5	1807031	2	9.2	2	1807032	33
2.6	5	1807031	10	9.4	2	1807032	4
2.8	5	1807031	0	9.6	2	1807032	21
3.0	5	1807031	4	9.8	2	1807032	12
3.2	5	1807031	0	10.0	2	1807032	17
3.4	5	1807031	6	10.2	3	1807032	13
3.6	5	1807031	33	10.4	3	1807032	21
3.8	1	1807041	4	10.6	2	1807032	15
4.0	1	1807041	10	10.8	2	1807032	15
4.2	1	1807041	0	11.0	3	1807032	37
4.4	1	1807041	0	11.2	3	1807032	54
4.6	1	1807041	35	11.4	3	1807032	21
4.8	1	1807041	40	11.6	3	1807032	58
5.0	1	1807041	19	11.8	3	1807032	13
5.2	1	1807042	63	12.0	3	1807032	67
5.4	5	1807032	38	12.2	3	1807032	21
5.6	5	1807032	60	12.4	3	1807032	35
5.8	4	1807032	42	12.6	3	1807032	0
6.0	4	1807032	54	12.8	3	1807032	27
6.2	3	1807032	0	13.0	2	1807032	6
6.4	3	1807032	23	13.2	2	1807032	23
6.6	3	1807032	15	13.4	2	1807032	17
6.8	3	1807032	50	13.6	2	1807032	37
				Avg.	2.985294		22.61764706

Table 3: NFA animations performance

FSM_ANIMATION

Time (sec)	Frames/sec	OGL calls/frame	CPUs average utilization	Time (sec)	Frames/sec	OGL calls/frame	CPUs average utilization
0.0	0	0	42				
0.2	0	0	75	7.0	1	1807039	23
0.4	0	1795060	63	7.2	1	1807039	44
0.6	3	1795060	79	7.4	1	1807039	10
0.8	3	1795061	79	7.6	1	1807039	23
1.0	2	1795061	81	7.8	1	1807040	38
1.2	2	1795052	77	8.0	1	1807040	33
1.4	3	1795052	77	8.2	1	1807032	67
1.6	3	1795052	77	8.4	3	1807032	27
1.8	3	1795052	75	8.6	3	1807032	35
2.0	2	1795052	67	8.8	3	1807032	33
2.2	2	1795052	54	9.0	3	1807032	23
2.4	2	1795052	50	9.2	3	1807032	29
2.6	2	1795052	46	9.4	2	1807032	10
2.8	2	1795052	48	9.6	2	1807032	54
3.0	2	1795052	60	9.8	2	1807032	56
3.2	2	1795052	42	10.0	2	1807032	23
3.4	2	1795052	50	10.2	3	1807032	33
3.6	2	1795052	52	10.4	3	1807032	27
3.8	2	1795052	50	10.6	3	1807032	15
4.0	2	1795052	52	10.8	2	1807032	60
4.2	2	1795052	73	11.0	2	1807032	27

Table 3: Continue

FSM_ANIMATION

Time (sec)	Frames/sec	OGL calls/frame	CPUs average utilization	Time (sec)	Frames/sec	OGL calls/frame	CPUs average utilization
4.4	0	1795063	60	11.2	3	1807032	50
4.6	0	1795063	38	11.4	3	1807032	27
4.8	0	1795063	25	11.6	3	1807032	52
5.0	0	1795063	33	11.8	3	1807032	31
5.2	0	1795063	23	12.0	3	1807032	36
5.4	0	1795063	46	12.2	3	1807032	65
5.6	1	1795060	23	12.4	3	1807032	38
5.8	1	1795060	46	12.6	3	1807032	40
6.0	1	1795060	15	12.8	3	1807032	37
6.2	1	1795060	21	13.0	3	1807032	44
6.4	1	1795060	12	13.2	2	1807032	33
6.6	1	1795060	44	13.4	2	1807032	46
6.8	1	1795033	42	13.6	3	1807032	46
				Avg.	1.955882		44.58824

CONCLUSION

The proposed Interactive software animation tool is implemented by using NFA techniques. The systems not only facilitates the reuse of animation and self control of character motion, but also carry out graphics transfer states tools which can test and implement the state transition of interactive animation. It is observed that the efficiency of the proposed system is higher than the older systems. From the result it is observed that the frame per second rendering is less than the FSM based animation. By correctly giving the gestures sequences the system can dynamically generate the sign gestures.

REFERENCES

Badler, N., J. Allbeck, L. Zhao and M. Byun, 2002. Representing and parameterizing agent behaviors. Proceedings of the Conference on Computer Animation, June 21, 2002, IEEE, Philadelphia, Pennsylvania, USA., ISBN:0-7695-1594-0, pp: 133-143.

Bruderlin. A. and L. Williams, 1995. Motion signal processing. Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, August 06-11, 1995, ACM, Los Angeles, California, ISBN:0-89791-701-4, pp: 97-104.

Lau, M. and J.J. Kuffner, 2006. Precomputed search trees: Planning for interactive goal-driven animation. Proceedings of the 2006 ACM SIGGRAPH-Eurographics Symposium on Computer Animation, September 2-4, 2006, Eurographics Association, Switzerland, Europe, ISBN:3-905673-34-7, pp: 299-308.

Liu, C.K. and Z. Popovic, 2002. Synthesis of complex dynamic character motion from simple animations. ACM. Trans. Graphics, 21: 408-416.

Papadogiorgaki, M., N. Grammalidis, L. Makris and M.G. Strintzis, 2006. Gesture synthesis from sign language notation using MPEG-4 humanoid animation parameters and inverse kinematics. Proceedings of the 2nd IET International Conference on Intelligent Environments, Vol. 1, July 5-6, 2006, IET, Thessaloniki, Greece, ISBN:0-86341-663-2, pp: 151-160.

Tomlinson, B., 2005. From linear to interactive animation: How autonomous characters change the process and product of animating. Theor. Pract. Comput. Appl. Entertainment, 3: 5-5.

Watanabe, Y., S. Okamoto, M. Kohana, M. Kamada and T. Yonekura, 2013. A parallelization of interactive animation software with web workers. Proceedings of the 16th International Conference on Network-Based Information Systems (NBiS), September 4-6, 2013, IEEE, Tokyo, Japan, ISBN:978-1-4799-2510-0, pp: 448-452.

Wu, S., 2012. The implement of Animation State transitions in interactive scenes based on graphic finite state machine. Proceedings of the 4th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Vol. 2, August 26-27, 2012, IEEE, Beijing, China, ISBN:978-1-4673-1902-7, pp: 242-245.