Data driven motion transitions for interactive games

M. Mizuguchi[1], J. Buchanan[1], T. Calvert[2]

[1]Electronic Arts (Canada), Inc. Burnaby, B.C., Canada
[2]Technical University of B.C., Surrey, B.C., Canada

Abstract

In 3D video games that employ human characters, a series of animations is required to display a character's motion. The current approach is to use stored animation sequences, either motion captured or hand animated, and play them back as required. Unlike a sequence for film or video the motion needs to change according to the user's interaction with the game. There are constant unpredictable transitions from one animation into another. This paper presents the design and analysis of a framework for supporting data driven transitions that have been pre-specified by animators. This approach frees the programmers from having to determine the details of each transition and gives control to the animators. This takes advantage of the animators' skill at evaluating and tweaking the motion to produce better aesthetic results and lets the animators and programmers work in parallel.

1. Introduction

Producing convincing animation sequences for articulated figures such as human characters remains a difficult task in computer graphics. One method of simplifying the process is to create longer animations by sequencing together a series of base animations from a motion library. The order in which the animations are sequenced can be varied so that given a sufficiently large motion library a rich variety of motions can be achieved. Using motion capture technology the motion of a real actor can be recorded and an extensive library of realistic motions can be created fairly easily.

1.1. Motivation

When sequencing motions together extra care needs to be taken with the transitions between them. The pose of the figure may not be aligned in the source and target animations and the rate of change may also differ. Significant differences in the pose or speed will result in a discontinuous motion. Simple interpolation of the poses can smooth out the transition but may not produce a natural or physically realistic motion [7].

Techniques from recent motion editing research can be used to modify the base motions to produce believable transitions [3, 4, 5, 8, 9]. Although these techniques are effective, they focus on easing the task of creating animations for traditional sequential play back such as in video or film. Many of these techniques are not applicable to highly interactive environments such as games.

The difficulty in working with games is that performance is always a concern, since the characters need to react quickly. The first priority is that the characters have fast reactions and the second is that the reactions look realistic. Intensive calculations to provide smoother more natural blending are not possible if they result in slow reaction times. While the motion editing techniques referenced above are effective they are too slow to be applied in a real-time environment.

1.2. Approach

By predefining the allowable transitions and storing data about them, realistic blends between animations are achievable without great computational overhead. The simpler techniques can be employed according to the data thus avoiding poor blends. The blending parameters can be specified and tuned during development of the game.

Possibly the best benefit of this approach is that with well designed tools the animators can specify how the transitions are to occur. It is the animators who fine tune the individual motion clips but currently they have no way to specify how they are to flow together during game play. This responsibility falls on the programmers. The programmers are able to apply various techniques to blend the motions by making changes to the source code. Unfortunately, most program-

mers do not have the skill or training that animators have in evaluating and creating figure motion. Due to the lack of a proper framework, programmers are required to perform a task that is better suited for animators.

2. Transition Framework

If an animator is to take on the job of specifying the transitions, a framework needs to be developed for supporting the added communication between the animator and the programmer. The central component of the framework is the data on how to perform the transitions. Additional supporting components are required to allow the animator to create this data and the programmer to subsequently use it.

The actual data used in the game needs to be in a platform specific compact binary format so it will not be legible or flexible. Instead of reading and writing directly to this format, the editing tool could write in a flexible intermediate format. A platform specific binary file could then be compiled by the tool at a later stage. The transition data would be the same but this separation would allow data for several platforms to be generated from a single intermediate file. The intermediate file can be stored as text to allow changes to be tracked using a source code control system.

The animator needs a way of generating and viewing the transition data. The animators could use a text editor to edit the data directly. The resulting transitions could then be viewed when they are played back in the game. This provides a means with which to define transitions but it is not sufficient to meet the demands of a production environment. An easy to use tool is required so that the animators can work productively. A prototype tool was developed and is described in more detail later.

On the other end, an Application Programming Interface (API) was developed to allow the programmer to easily access the transition data and apply it accordingly. Given the current motion state all the programmer needs to specify is an ID of a transition. The API then looks up the data recorded for that ID and applies the transition according to what the animator set.

2.1. Pipeline

If the transitions are to be specified during game development then this process must fit into the production pipeline. A pipeline is the series of steps that need to be followed to get artistic content such as models and animations into a form usable by the game programmer. For a game that uses motion captured data the pipeline steps for getting animation into the game are:

1. Motion capture session.
2. Motion tracking and clean-up.
3. Tweaking by animators.
4. Export and compile into game format.

The first two steps are generally done once but the tweaking of the animations is an ongoing process that continues throughout development. As long as the tweaking does not drastically change the animation it can be done in parallel with animation programming tasks. Once the modified animation is re-exported it can be dropped seamlessly into the game in place of the original.

A similar pipeline is required to get transition data into a game ready state. The initial steps of the pipeline require that the programmer and animator agree on a set of transitions to be supported in the game. Once this set of transitions is created the animators can tweak the parameters to perfect the resulting motion. As with the component animations the programmers do not need to make any code changes to get the modified transitions.

3. Specifying Transitions

The framework developed supports several transition techniques and a wide variety of parameters for those techniques. This sections discusses some of these techniques.

3.1. Parameters

Simply concatenating two animations together will rarely produce a smooth transition since the poses will not be lined up. Figure 1(a) graphs a joint angle over time for two concatenated animations. There is a pronounced discontinuity at the transition time which results in a pop in the motion.

If the initial source animation has not run to completion, then a simple way to smooth the motion is to blend the remainder of the source with the target as in Perlin [7]. The end of the source animation is overlapped with the the target and the target is progressively blended into the source over the interval. The interpolation is controlled by a weight function that progresses from 1 to 0 over the blending time. Figure 1(b) shows the result of a blend with a linear weight function. The blend removes the harsh discontinuity at the transition.

Blend Length

When using a blending technique the length of the blend must be decided upon. Longer blends produce smoother transitions but they also water down the target animation. A rather exuberant kick may become a shallow one. The optimal blend length depends on how much the poses are out of alignment. If the poses and tempo are matched exactly then the blend length can be 0. If the poses are seriously out of alignment, then a longer blend time will be needed to smooth them together.

As a general guideline, 10 frames provides a good starting point for 30 frame per second animations. At 5 frames the transition becomes discontinuous for most cases and 20 frames is generally more than enough.

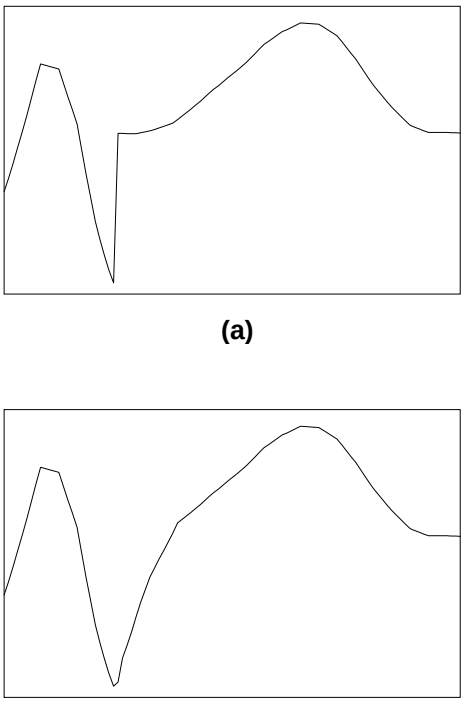

Figure 1: *Graphs of a joint angle over time: (a) two animations are concatenated resulting in a discontinuity, (b) the same animations are blended with a fall-off.*

Start Frame

When blending from one animation into another the blend does not need to start at the first frame of the target animation. An internal frame in the target motion may correspond better to the current pose in the source motion. If a target frame is selected where the character is supported by the opposite leg an undesirable attenuated step results. The correspondence of leg poses can be based on which leg is supporting the figure and the angle between the legs. This value is called the leg phase [1].

Figure 3 illustrates the results of 3 transitions from a jog to a run with only the target frame varied between them. In (a) there is a strong correspondence in the phase. The feet strike the ground properly and the legs move in a believable cycle. In (b) the start frame is set to one with a different support leg. The resulting motion contains a short hop midway through the blend. In (c) the correspondence is improved but the motion still contains an undesirable loop at one of the foot contact points.

When specifying a transition, the animator can manually select a target frame or select that the target frame be automatically selected by matching the phase angle at runtime.

Modification and Correction

The blending technique only takes into consideration the current motion state of the figure. The result of the transition does not take into account additional state data that can be supplied during game play. It may be desirable to incorporate additional runtime data such as the position of the ball. For example when transitioning from a run to a kick, allowing the ball contact position to be specified gives the programmer greater control over how the transition is to look.

Currently the framework supports modification of the data using inverse kinematics (IK) and motion warping [2, 10]. To use this technique the animator must supply additional data about the how the IK and warping are to be performed.

Motion warping can also be used to correct constraint violations introduced by the motion blending such as the feet penetrating the floor. Automatically detecting all possible violations is extremely expensive. As part of the transition specification process the animator could flag which violations can potentially occur for a given transition thus greatly reducing the detection costs.

Some of the main transition techniques and parameters have been discussed here but there are too many to go into detail for all of them so the reader is referred to [6] for a complete description of all of the parameters.

3.2. Interactive Editor

If the time taken to define a transition is excessive then the animators will not have the time to fully fine-tune the transitions to the point that they desire. The resulting transitions will simply be made “good enough” because of the time constraints. This then diminishes the benefit of having the animators specify the transitions in the first place. As well, waiting until the transitions are incorporated into the game to view them simply takes too long. The process of editing and testing the transition should be a tight interactive cycle.

A prototype transition editor was developed to ease the specification process. The editor gives the user a graphical way of setting transitions and interactively playing them back for testing. The editor supports a variety of techniques including the parameters outlined above. The system also allows for multiple transitions to be viewed concurrently to allow for comparisons.

Figure 2 shows the diagrammatic representation of a transition used by the editor. The bar on the left with the arrows coming from it is the source animation. The column of bars on the right are the possible target animations. The length of the bars corresponds to the length of the animations. The source animation is divided into 3 ranges. A single set of

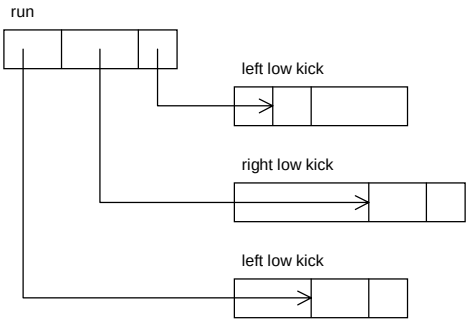

Figure 2: *Diagrammatic representation of a motion transition used by the editor.*

transition parameters may not apply to the entire animation. The user can divide the animation into ranges with differing parameters. A different target animation can also be specified for a range. In most cases 3 or fewer ranges is adequate.

The line at the tip of the arrow shows the frame at which to start the blend. The user can drag this line to change the frame or have it automatically set according to the leg phase. The second line determines the length of the blend and it can be dragged as well. The user composes the transition by dragging and dropping animations and can test various transition times and interactively playback the resulting motion.

The design of the editor was made extendable so that support for new transition techniques could be added later. At the start of a new project, there may be a decision to support a new set of techniques. The editor would then need to be modified at this time.

4. Evaluation

An informal user evaluation was done by letting a group of 3 animators use the system off and on for a week. After the evaluation period a discussion was held about what they thought of its usefulness and what were possible improvements. In general they saw great potential in such a system. The animators liked the idea of having full control of how the motion is presented in the game. They also saw immediate use for such a system as a means to visualize transitions and show programmers how they wanted them to look. The notion of setting parameters for a range of frames was a new concept. Animators are more accustomed to sequencing together animations at a fixed frame but learning this new concept was not difficult for them.

The comments about improvements did not center around specific transition techniques but were more related to user interface issues and ways to better visualize the motion. The users suggested that they wanted iconic representations of the ball position and the player's current direction of travel. It was also the animators that requested a way of showing the player's pose at various frames with "stickmen" in the same way they are presented in Figure 3.

The animators' lack of comments regarding the actual transition techniques is attributed to the informal nature of the evaluation. Given a longer evaluation period and some real production work to do, it is more likely that the animators would have more to say about specific transition techniques and the usefulness of certain parameters and the desire for others.

5. Conclusion and Future Work

As part of this work a successful prototype framework was developed for data driven transitions. The framework allows animators to specify the transitions with an interactive editor and the programmers to access and apply the data through a simple API. The feedback given by the animators for the prototype system was encouraging and shows that pursuing such a system is worth while. Some necessary improvements have been identified before it can be used in a full production environment and are detailed in [6].

It also remains to be seen how such a data driven approach to transitions could be used in conjunction with a higher level motion system. For example, the motion in the game could be a hybrid of motion generated by a dynamic system and playback of recorded motion capture clips. The dynamic system could be used to control collisions between characters. Once the collision between characters has ended a transition could be made into an animation clip.

Another example of a high level system is one where motion blending and warping are used outside of transitions. Walks and runs at different paces can be blended to produce a large variation in tempos. The blended motions can be cycled and a transition can be made from the blended motion into an existing clip or another blended motion.

References

1. Matt Brown. Electronic Arts Confidential Information.
2. Armin Bruderlin and Lance Williams. Motion signal processing, *Computer Graphics (SIGGRAPH '95 Proceedings)*, 97–104, 1995.
3. Michael Gleicher. Motion editing with spacetime constraints. *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, 139-148, 1997.
4. Michael Gleicher and Peter Litwinowicz. Constraint-based motion adaptation. *The Journal of Visualization and Computer Animation*, 65–94, 1998.
5. Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like

characters. *Computer Graphics (SIGGRAPH '99 Proceedings)*, 39–48, 1999.

6. Mark Mizuguchi. Customizing human animation transitions for gaming environments. Master's Thesis, Simon Fraser University, 2000.
7. Ken Perlin. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics*, 175–204, 1995.
8. Zoran Popovic and Andy Witkin. Physically based motion transformation. *Computer Graphics (SIGGRAPH '99 Proceedings)*, 11–20, 1999.
9. Charles Rose and Brian Guenter and Bobby Bodenheimer and Michael F. Cohen. Efficient generation of motion transitions using spacetime constraints. *Computer Graphics (SIGGRAPH '96 Proceedings)*, 147–154, 1996.
10. Andrew Witkin and Zoran Popovic. Motion warping. *Computer Graphics (SIGGRAPH '95 Proceedings)*, 105–108, 1995.

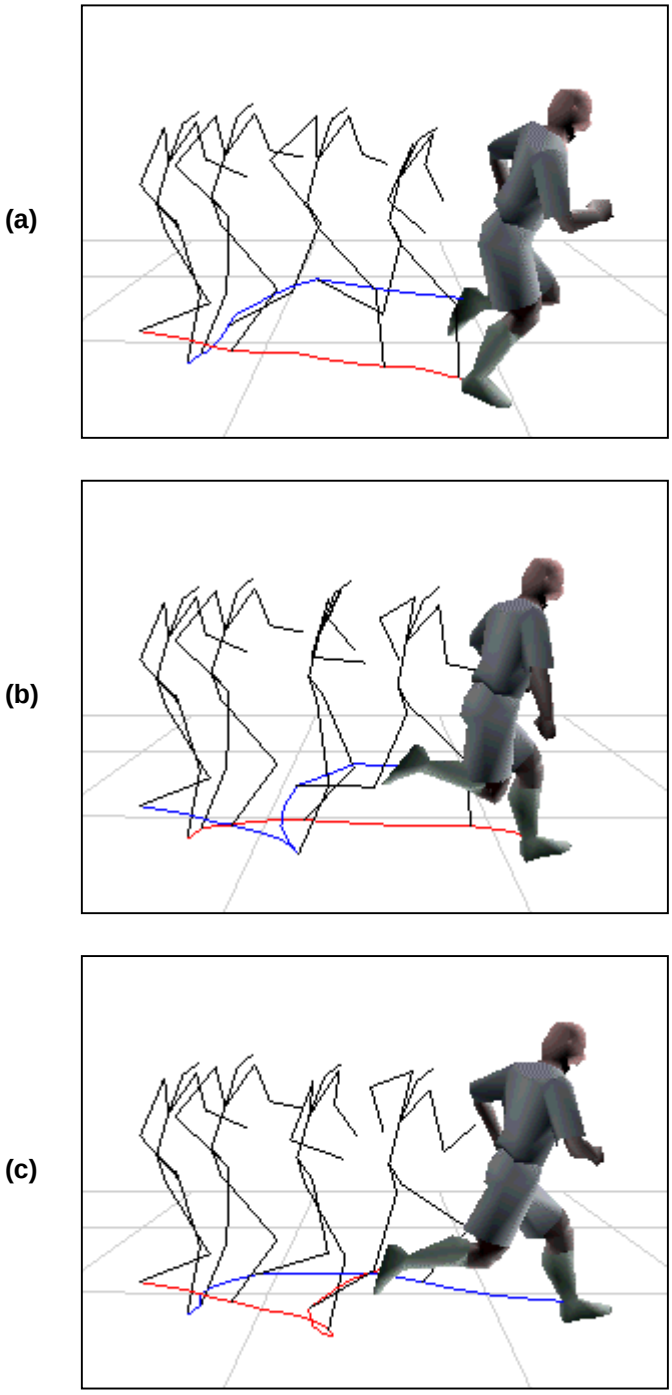

Figure 3: *Three transitions of jog to run with varying degrees of phase correspondence: (a) Good - There is a good correspondence causing the feet to strike the ground properly and the legs to move in a proper cycle, (b) Poor - The start frame has a different support leg causing a hop, (c) Moderate - The correspondence is improved over (b) but the right foot loops at one of the contact points.*