Music-Driven Character Animation

DANIELLE SAUER
University of Alberta
and
YEE-HONG YANG
University of Alberta

Music-driven character animation extracts musical features from a song and uses them to create an animation. This paper presents a system that builds a new animation directly from musical attributes, rather than simply synchronizing it to the music like similar systems. Using a simple script that identifies the movements involved in the performance and their timing, the user can control the animation of characters easily. Another unique feature of the system is its ability to incorporate multiple characters into the same animation, both with synchronized and unsynchronized movements. A system that integrates Celtic dance movements is developed in this paper. An evaluation of the results shows that the majority of animations are found to be appealing to viewers and that altering the music can change the attractiveness of the final result.

Categories and Subject Descriptors: J.5 [**Arts and Humanities**]: Performing Arts; I.3.4 [**Computer Graphics**] Graphics Utilities - *Software support*; I.3.7 [**Computer Graphics**] Three-Dimensional Graphics and Realism - *Animation*
General Terms: Design, Experimentation, Human Factors
Additional Key Words and Phrases: Character animation, motion synthesis, music analysis, primitive movements

---

1. INTRODUCTION

Animations, whether they are in movies, television or video games, always capture the viewer's interest more if they are accompanied by music. Music has the capability to set the mood for a scene and can alter the viewer's perception of what she is seeing. The ability to tie the correct type of music in with an animation is a difficult and time-consuming process. Not only is choosing the proper type of music important, but proper synchronization of music with the events in an animation is essential when attempting to secure the attention of a viewer. An interesting animation brings with it a "wow" factor, enticing the viewer to watch and appreciate the work. This can be achieved through a good combination of interesting movements and relevant music. This paper proposes a method that attains this combination by using musical attributes such as the beat and

This research was supported by NSERC and AutoDesk.
Authors' addresses: Danielle Sauer, Department of Computing Science, The University of Alberta, Edmonton AB Canada T6G 2E8; Yee-Hong Yang, Department of Computing Science, The University of Alberta, Edmonton AB Canada T6G 2E8.

dynamics to build an animation that fits user specifications and is tailored to the music. The user is able to choose any type of music she desires and create an animation that is not only automatically synchronized to the music, but also projects key elements of the music's intent as well. Our system provides a user-friendly method for creating a high-quality character animation where the user chooses pre-designed movements to build a motion sequence. Through the use of a script file, the user can choose the order of specific movements and build a dance routine for a character, or set of characters, of her choosing. Our technique concentrates on using simple movements to create complex motion and providing maximum user control.

2. PREVIOUS WORK

Direct synchronization of an already existing animation with a piece of music is the technique most similar to our music-driven character animation method. The purpose of most synchronization methods is to take an already existing animation and synchronize it so that movement changes line up with beats in a given piece of music. Movement transition graphs are a popular technique for achieving this. Approaches proposed by Alankus et al. [2005] and Kim et al. [2003] use transition graphs to synthesize new motion sequences from motion capture data. The graph is used to choose which movements best fit with the beats of the music, as well as create a movement ordering where transitions between motions occur smoothly. Shiratori et al. also separate the original motion capture data into smaller sections for synthesizing new motion [2004, 2006]. In their method, rhythmic similarity between music and movement segments is used to synchronize the animation with the song. Some synchronization techniques use motion curves rather than motion data. Cardle et al. [2002] implemented a system that performs motion editing directly onto the keyframed motion signal. For example, motion warping can be mapped to the musical beat by adding a point to the displacement map for each beat, resulting in a jump in the signal at each displacement point.

It is our belief, along with others [Fod 2002,Woch 2004], that complex motion can be simplified into a combination of basic movements called *primitives*. Dancing is a real-world example that supports this theory. Long dance sequences can be split into routines that consist of separate dance moves. The individual dance moves are the primitives that are combined together to create dance routines and performances. Fod et al. [2002] also implemented an algorithm for automatically detecting and segmenting primitives from movement data. Another method exists in which the characterization of primitive movements is built from the kinematic theory and its deltalognormal model $\Delta\Lambda$ [Woch

2004]. These techniques support the construction of complex motion from primitive movements as performed by our system.

MIDI files are the most popular input format for songs because it is less difficult to extract musical attributes from the data [Cardle 2002, Taylor 2005]. These files are not easily accessible and imply musical knowledge by the user, so we choose to use .wav files and use signal processing techniques to extract the tempo, beat onsets, and dynamics from the music datum. Beats can be considered regular pulsations of music and determine the tempo of the music, which is the overall pace of a composition of music, e.g. fast, slow. Dynamics represent the variation of loud and soft levels in a song and establish the emotion of the music. Methods for performing beat detection vary but the use of tempo tracking is a popular technique. Both Scheirer [1998] and Dixon [2003] use tempo detection to reinforce their beat detection algorithms. Scheirer uses filterbanks and comb filters to track tempo changes and beat positions, while Dixon uses multiple agents and clustering. These techniques led to the development of our beat detection algorithm. We combine Tzanetaki et al.'s [2001] tempo detection algorithm with Goto's beat detection algorithm [1994, 1999, 2001] to create a technique that uses the tempo to help determine the beat onsets. We develop our own algorithm for dynamics extraction using the power spectrum.

The contributions of this paper are as follows:

- We develop a system that builds a new animation directly from musical attributes, rather than synchronizing an already existing animation to music.
- We present a signal processing-based beat detection algorithm based on Goto's beat onset method and Tzanetakis' tempo recognition method, as well as a novel dynamics extraction algorithm.
- We present a script file that allows for the animation of several characters and the ability to specify and build different movement routines for each character.

3. PROPOSED METHOD

Our character animation system is made up of two principal components. The first is music analysis, where the musical attributes used by the system are extracted from the input music file. All music analysis work is performed in Matlab 7.1. The second component is character animation in the form of Celtic dance moves. This portion of the system controls the motion of the character in the scene, including the timing and expression of the movements. The animation system is built as a C++ plug-in for Autodesk's Maya, where Maya's interface is used to create the character, background and lighting, and the plug-in is called to perform the movement. The result of these two components is a unique animation developed from movements chosen by the user and timed according to the music.

3.1 Music Analysis

Music analysis involves analyzing an input signal and extracting specific musical features. The features extracted by this system include the tempo, beat positions and dynamics. These attributes are the most recognizable aspects of a piece of music, especially to the untrained ear. Analysis on the song is performed by combining two different algorithms: Tzanetakis et al's [2001] tempo detection method and Goto's beat tracking method [1994, 1999, 2001]. Tzanetakis' method was faithfully followed in the implementation, but several changes have been made to Goto's method in order to make it work better for our purposes.

3.1.1 Beat Detection

The process behind beat detection is analyzing a musical signal and finding the positions of all the beats. Goto's original algorithm uses drum patterns and chord change information to make the system more robust, but we have not included these features in our implementation. Instead, we rely on Tzanetakis' tempo algorithm to give us more accuracy in determining a beat onset. Our system does not require the precision that Goto's algorithm strives for so we fashioned a simpler version of his system that runs in close to real-time and does not require additional musical knowledge. Our changes to his algorithm are discussed below.

Goto's algorithm uses a power spectrum to extract the beat onsets from the signal. He divides the beat onsets into seven frequency bands for further analysis. This can result in a large number of possible onsets. To narrow down the range of possible beat positions, a threshold is used to remove onsets with the smallest amplitudes. This is based on the assumption that beat sounds are fairly high in amplitude compared to other musical features. The threshold is computed by multiplying the maximum value of each frequency band with a percentage value. The percentage value usually ranges from 80-90% of the signal's amplitude, meaning the onset components that fall within the highest 80-90% of the signal's amplitude are retained and the rest are discarded. It is important to note that the amplitude of the beat is dependent on the amplitude of the signal. If the dynamics of the signal at a point in time are soft, then the amplitude of the beat will be low to match this, as will the amplitudes of the other musical features. This detail is the reasoning behind the choice of the percentage value. If the percentage value does not cover the softer ranges of music then the beats are not detected in those time intervals. The percentage value that works best for the threshold changes from song to song and is manually set based on experimentation.

The estimated onset times are then compared across frequency bands and only the positions where an onset has been detected in two or more bands are stored. The tempo detected by Tzanetakis' algorithm is then used to calculate an inter-beat-interval (IBI). An IBI is the distance between two beats and can be approximated from the tempo. A direct relationship occurs between the speed of the song and the distance between beats and this relationship is used to compute the IBI directly from the tempo. The first estimated beat is stored as the first true onset of the signal and used as a comparison point for the next estimated beat in the list. The distance between this first actual onset and the next estimated beat is calculated. If the distance is greater than the *IBI-error*, where the error value is 5 frame-times, then it is stored as the next actual onset in the signal. This distance threshold check ensures that the final beat onsets are not too close together, as can be the case when the algorithm detects weak beat positions. Weak beats are the beat sounds that occur between the actual beats of a song. They are generally found at twice the tempo rate and half the distance between two actual beats and can be mistaken by beat detection algorithms as real beats. Tracking of these beats is avoided by using the IBI to ensure only beat positions that occur around or further than the known interval are chosen. This procedure is followed for all the estimated beats in the list and the end result is a vector of actual beat onsets for the entire song.

3.1.2 Testing and Results

Testing of the beat detection algorithm has to be performed manually in order to assure accuracy. Both visual data and audio data are used to compare the generated results with the true beat positions in the musical signals. Visual data is used for the synthesized signals where the beat positions are discernable. Table I displays the results for testing the beat detection algorithm on the ten synthetic signals with increasing amounts of noise. The algorithm uses a threshold value of 90% to obtain the majority of the results seen in Table I. The superscripts in the noise level row (the second row) denote that different threshold values were used to obtain these results than the threshold used in the first six columns. The threshold used by the algorithm in column 7 ([1]) is 72% while the threshold used in column 8 ([2]) is 65% and the threshold for column 9 ([3]) is 25%. This indicates that the new algorithm is quite robust because its threshold value can be altered to reflect the signal. The results are consistently very good until the last noise level is reached, at which they drop off considerably.

Table I: The results from performing beat detection with the new beat detection algorithm on ten synthetic signals with a tempo of 153.3682. Each signal was created with a different random seed and eight noise levels were used, ranging from 1/100 to ½ of the beep's amplitude.

Signal #	**Number of beats detected (/58) for each noise level**							
	1/100	***1/50***	***1/25***	***1/16***	***1/8***	***1/6*** [1]	***1/4*** [2]	***1/2*** [3]
1	58	58	58	58	58	58	58	27
2	58	58	58	58	58	58	58	22
3	58	58	58	58	58	58	58	18
4	58	58	58	58	58	58	58	22
5	58	58	58	58	58	58	58	17
6	58	58	58	58	58	58	57	11
7	58	58	58	58	58	58	58	19
8	58	58	58	58	58	58	58	17
9	58	58	58	58	58	58	58	19
10	58	58	58	58	58	58	58	16
Avg # Beats	58 **58**	58 **58**	58 **58**	58 **58**	58 **58**	58 **58**	57.9 **58**	18.8 **19**

3.1.3 Dynamics Extraction

Dynamics consist of the louds and softs of the music, including transitions between the two that are also known as crescendos (soft to loud) and decrescendos (loud to soft). The dynamics levels are extracted because they are useful in creating corresponding movements. The purpose of this extraction algorithm is to detect the 50 positions in the music where the dynamic level is highest and 50 positions where the dynamic level is lowest. These positions represent the loud and soft dynamics respectively. Dynamics are extracted by using a moving window with a size of 44100 samples to compute the power spectrum of the music signal. The power spectrum is performed on the information in each window. The inverse FFT is performed on the outcome. Since the signal is symmetric, the second half of the signal is removed and the algorithm proceeds to calculate the absolute values for the signal's first half. The regional maxima are detected from the remaining values and the highest peak and the lowest peak are added to a list before the window is moved. This technique is performed for each window until the entire signal has been analyzed, with the resulting list being comprised of the highest and lowest values from each window. Finally, the algorithm determines the 50 highest and lowest values in the temporary list and stores them as the dynamic positions. The system detects 50 of the highest and 50 of the lowest values because we believe that 100 dynamic positions are enough to build a complete representation of the dynamic structure of the song. Crescendos and decrescendos can also be represented by the dynamic positions. A transition over time from a high dynamic value to a low dynamic value signifies a decrescendo while a transition from a low dynamic value to a high dynamic value signifies a crescendo.

3.2 Celtic Animation System

This animation system integrates a unique music-driven approach to character animation. It generates an animation that looks like Celtic dancing, but is a unique variation of existing performances. Celtic dancing was chosen because it is an interesting and exciting dance where the movements are performed almost entirely by the legs. Using only three major body parts (two legs and the torso) simplifies the system and allows us to concentrate on the main movements. The system is provided with knowledge of Celtic dancing, including several preprogrammed primitive movements and routines.

Producing high-quality character animation has proven to be difficult for inexperienced users. Animation systems such as Autodesk's Maya are intimidating for a new user because of the enormous amount of features they provide. Setting up and animating a character is extremely time consuming and it generally takes practice and experience for a user to satisfactorily manipulate a human body. This system provides a user-friendly method for creating a high-quality character animation where the user chooses pre-built movements to build a motion sequence. Through the use of a script file, the user can choose the order of specific movements and build a dance routine for a character, or set of characters, of her choosing. This ensures that she does not have to struggle with positioning character joints in order to achieve a specific motion. The system also gives the user the chance to experiment with different types of characters by supporting interchangeable characters. The user can change the appearance of the characters in the animation and easily use different characters in the same motion sequence. Maximum user control is provided by this system without relying on the user for the key components of the animation.

3.2.1 Script Files

Script files are utilized to give the user control over what occurs in the animation. They are simple text files that list Celtic primitives and routines that the user wants performed in the resulting animation. The system reads the script file using a specially designed parser and records each movement in the system as it is read in the script file. The script file is an easy and user-friendly method of allowing the user to create her own animation through a combination of built-in primitives, built-in routines and user-designed routines. The script file is also designed to allow for multiple characters in a scene. There is no limit to the number of characters that can be specified by the user. The system is designed so that multiple characters can use the same script file to perform the same sequence of movements or they can use different script files to perform different animation sequences. There are two script files that are used to define the animation. The first script file is the master script and it defines the characters and which secondary script file each one uses. The secondary script file is used to define the animation by listing the movements in the order they should be performed.

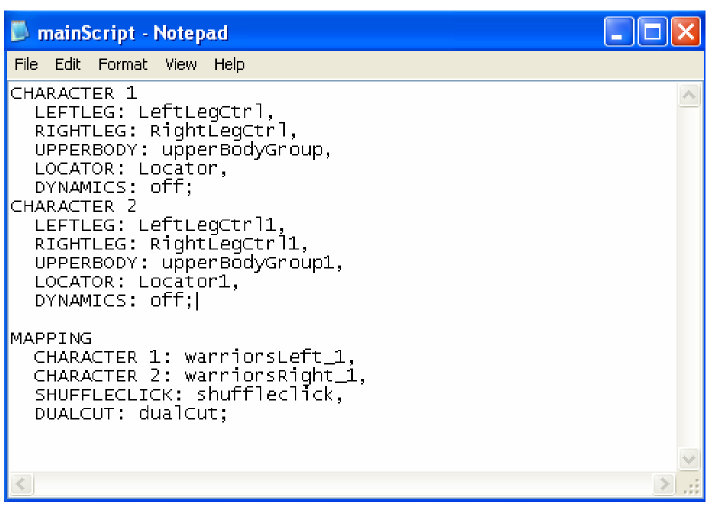

Figure 1: An example of a master script file using two characters that are each mapped to different secondary script files.

The system uses three main body parts: left leg, right leg and upper body (torso). In order to manipulate a character, the system needs to be able to choose those body parts from the scene. At the beginning of the main script file the user needs to define the name of the object in the scene that corresponds to each of the main body parts in the system. An example of this is

LEFTLEG: leftLegCtrl

where *leftLegCtrl* denotes the name of the character's left leg in the scene. It is the user's responsibility to ensure that she is mapping the correct scene object to its corresponding system body part. The object will be picked out of the scene and connected to the system so they can share information. Dynamics are one of the musical attributes that are mapped to movements. The mapping of this attribute can be turned on and off through the master script file. This gives the user the choice to allow dynamics to alter the movements or to use a constant dynamic range through the animation. A complete example of a character definition in the main script file is found in Figure 1 under the title CHARACTER 1. Increasing the number of the previous declaration creates different character definitions. This results in definitions that range from CHARACTER 1 to CHARACTER *n*.

The secondary script file provides a blueprint of how the animation will look. This script file defines the dance by using primitive movements, built-in routines and user-designed routines. The user can construct her motion sequence simply by listing the movements she wants included in her final animation. Each primitive movement and routine has a corresponding name that needs to be specified in order to execute the motion. The user must stick to these naming conventions when creating the secondary script file or the correct movement will not be called.

In some cases it may be necessary to start the motion sequence at a certain frame or divide the sequence up into large intervals of time. The system is implemented so that the user can choose a start time for each segment using the keyword *START*. When a user adds this keyword into the secondary script file, the system will perform the first movement at the frame number specified directly after START.

A movement or routine can be performed several times in a row by specifying the name of the movement and then the number of times it should be performed directly after it in the script line. There is no limit to the number of times a movement can be looped through. An example of this can be found in the second line of Figure 2. Along with animation timing, the system provides the ability to control the timing of individual movements. The user can influence the timing of the movements through the application of *brackets* and *rests*. Brackets are used to indicate that more than one movement is performed at the same time. In many of the built-in routines, several movements are performed at the same time to create a realistic motion. The user can copy this by putting brackets around the movements occurring in the same time interval. The first movement of the interval is specified normally and the remaining movements are placed within brackets. An example is

```
LIFTLEG: RightLeg;
(HOP);
(LIFTLEG: LeftLeg);
```

Rests are used to stagger the starting and stopping times for movements. The concept is adopted from music, where rests denote breaks between musical tones. The rest is specified in the system by the '^' character. Each rest is worth 1/8 of a beat, which means that the length of a rest will change from one piece of music to the next. The faster the song is, the shorter a rest will be. Rests can be placed before or after a movement name and one movement can use several rests. If the rest is placed before a movement, the movement will wait ⅛ of a beat before beginning. If the rest is placed behind a movement, it will end ⅛ of a beat earlier. Examples include

^HOP;

CROSS^;

^^STAMP^;

An example of a secondary script file is shown in Figure 2.

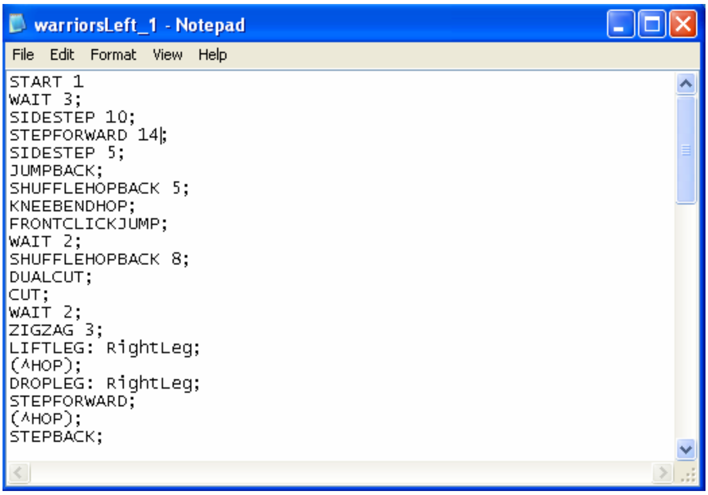

Figure 2: An example of a secondary script file in which the user designs her motion sequence.

3.2.2 Mappings

The main purpose of our animation system is to use music as the prime vehicle to drive an animation. Musical attributes such as the beat are mapped to Celtic movements and used to alter the motion based on the music. This system does not simply synchronize an already existing animation with a piece of music, but it actually builds the animation according to details extracted from the input song. Unlike synchronization methods, the movements in our system change along with the music. We create a final animation that is tailored to fit the music chosen by the user while providing an interesting and entertaining sequence of motion.

The timing of the movements is based almost entirely on the tempo of the music, where the faster the song, the faster the movements are performed. The position of the beats are inputted into the animation component by the music analysis component and used to determine the length of each movement's time interval. Celtic dancing is a fairly high-speed dance where several movements occur in the space of one beat. In this system two primitive movements are performed for each one beat. This rule applies to routines with multiple primitives as well. Several routines use three or four primitive movements and result in taking 1.5 or 2 beats to finish. Rather than performing each routine in a single beat, we choose to map two primitives to one beat because it provides smoother motion and better transitions between primitives.

The dynamics extracted from the music can also be used to affect the movements used to dance to a particular song. In real life, small and timid motions are not used on a song that is loud, and large extreme motions are not used on a song that is consistently soft. This system is designed to take this into account by creating a dynamics range that is used to alter the movement of certain motions so that they correspond better to the music. The dynamics levels in the system range from 1 to 5, where 1 denotes soft dynamics and 5 denotes loud ones. There exist several primitive movements where the dynamics level affects the distance moved by a body part or the height of a jump or kick. The higher the dynamic level, the higher the height or the longer the distance will be. As the dynamics in the song change, the dynamics level corresponding to the current frame will change as well. If a movement is currently in progress, the system will not change the dynamics level. This is to prevent jerky motions during the course of a movement. A change in dynamics level will only be incorporated at the beginning of a primitive movement. Incorporating dynamics levels into the performance helps to make the resulting movement more tailored to the input music.

3.2.3 Constraints

Foot position is an extremely important aspect of Celtic dance. It can help to determine the next movement in a motion sequence or the direction the character moves in around the stage. In many cases, the front foot is used as the starting foot for a routine or movement. This is the main reason that the system keeps track of which foot is in front and which is behind at each frame. We incorporate this Celtic knowledge into the system through the implementation of constraints. These constraints are used in some primitive movements and all built-in routines. Their purpose is to ensure that a primitive or routine is being performed by the correct body part according to the rules of Celtic dance. For example, in a built-in routine the constraints ensure that the primitives are performed by the correct body parts and in the correct order according to the Celtic routine it simulates. The constraints incorporate system knowledge of the positions of the character's feet with Celtic knowledge of how movements and routines should be performed. The use of constraints in a movement or routine is decided entirely by the animator and cannot be altered by the user. Constraints are used to enforce the integrity of Celtic dance and make it easier for the user to put together realistic motion.

3.2.4 Routines

The dance routines implemented in this system are more complex dance steps than those provided by the primitive movements. In many cases, Celtic dance has a dance step that consists of several primitive motions, but it is referred to by the name of the dance step rather than the primitives individually. Combining several primitive movements allows for these complex routines to be created and used by the system. The user can use these routines by specifying them by name. The routine will automatically call the appropriate primitive movements to create the movement. The system handles two different types of routines. The first is the built-in routine, as programmed by the animator, and the second is a user-designed routine.

The built-in routine is implemented directly into the system by the animator. It makes use of several primitive movements and controls the timing of them to create an actual Celtic dance step. The purpose of a routine is to make the animating process easier for the user. Rather than having the user continuously specify small primitive movements in the same order, she can call a routine that does the same thing. They save the user time and frustration because the animator has already worked out the timing of the primitive movements so that the routine is correct. This makes it easier for the user to create an entire Celtic dance based on known Celtic movements. These routines are similar to how a person would learn to Celtic dance and are taken directly from [Dunne 1996]. The built-in routines are as follows:

- ClickZigZag
- Cut
- CutBack
- FrontClickJump
- JumpBack
- KneeBendHop
- Shufflehopback
- SideStep
- SlidingStep
- Turn
- ZigZag

In some cases the user may want to use routines that are not implemented in the Celtic system. The system allows for user-designed routines in which the user can define her own routines through text files. The user can create her own dance moves by specifying primitives or built-in routines and their order. There is no maximum length limit for a routine, so the user is free to use as many primitives as necessary. The user-designed routine makes it easier to create an animation sequence because the user can define routines with combinations of movements that are used continuously in the animation. For example, if the user finds that she is constantly using three primitives in the same order in several places in her animation, she can put them into a routine. Rather than specifying the three primitives each time she wants that specific combination, she can specify her specially designed routine instead. The system will retrieve the routine as input and perform the primitives found in that routine. Once the routine files are designed they can be reused in any animation and changed easily by the user.

3.2.5 Primitive Movements

It is our belief that small primitive movements can be combined to create more interesting and complex motion. One of the main purposes of the Celtic system is to demonstrate that any type of primitive movement can be combined with other primitives to create an interesting sequence of motion. The primitive movements implemented in this system were determined by studying videos of Celtic dancing and establishing the simple movements that make up the larger routines [Dunne 1996]. A total of twenty-four primitive movements have been implemented into the system. They can be used in different combinations to create routines. The primitive movements are listed as follows:

- ClickHeelsIn
- ClickHeelsOut
- Cross
- CutBend
- DropLegBehind
- DropLeg
- HeelsUp
- HeelsDown
- Hop
- HopForward
- LongStep
- ShortHop
- SlideBehindStep
- Stamp
- StepForward
- StampDown
- StepBack
- SwingHeelsIn
- SwingHeelsOut
- TapOut

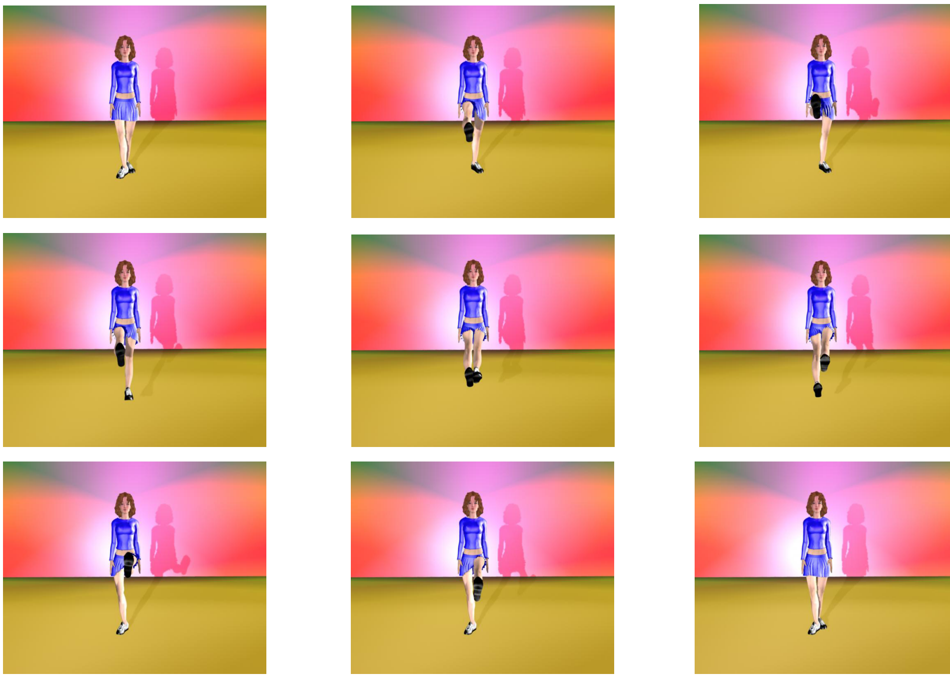

Figure 4: Sequential images displaying the different positions involved in the "FrontClickJump" Celtic routine.

- KneeBend
- LiftLeg
- TapBack
- Wait

4. RESULTS AND EVALUATION

The main purpose of this animation system is to create a unique animation with the structure of a Celtic dance but that is tailored to suit the chosen music. The resulting animation needs to be interesting, exciting and expressive of the corresponding music. Our results show that primitive movements can be grouped in different combinations to create appealing motion. Figure 4 displays the "FrontClickJump" routine, which is built from three primitive movements: "LiftLeg", "DropLeg" and "Hop". The use of these three primitives on different body parts and in a certain order creates one of the most interesting Celtic routines.

Some animations were created where multiple characters are involved in the dance performance. The Celtic system supports two types of multiple character movement: synchronized and unsynchronized. Synchronized movement involves all the characters performing the same movement at the same time. Figure 5 demonstrates unsynchronized movements of six dancers. The first and sixth characters are performing a "Sidestep" movement in all the images, while the second and fifth characters are performing a

“Shufflehopback” routine. The third and fourth characters are performing a “Cut” motion. Each group of characters is performing at the same time as the other groups but their movements are not the same, resulting in an unsynchronized performance. These results demonstrate how different characters can possess different personalities and yet still fit into the overall presentation.

4.1 Evaluation

The evaluation of a piece of music or a dance performance is generally subjective and extremely dependent on the preferences of the listener or viewer. This makes it exceptionally difficult to quantitatively determine if an animation is good or not. A qualitative evaluation was designed to assess the success of the Celtic system. There are two objectives in performing this evaluation. The first is to determine if the approach taken by the Celtic system is successful in creating appealing animations. The second is to establish if changing the music can also create appealing animations.

The evaluation involves 3 groups of 6 users per group. Each group represents a different user background. The first group incorporates users with dancing experience. These users apply their knowledge of movement to determine if an animation is good or not. The second group includes users with computer programming experience. This group of users has a technical background and will view the animations less artistically than the previous group. They will be able to focus on how well the parts fit together rather than concentrating on how accurate the movements are. The third group consists of users with neither dancing nor programming experience. These users can view the animations without any previous prejudices or expectations and are representative of an inexperienced user who may find the system useful.

The evaluation involves 8 animation videos with a single dancer in each. One of our objectives is to determine how different music affects the end result, so a different piece of music is used for each animation. The music types used include celtic, hip-hop, rap, rock, country and classical. The tempos range from 67 bpm to 171 bpm. The evaluator is asked to specify for each animation whether or not she liked the animation. The answer choices are a simple “yes” or “no.” She is then asked to state reasons for her answer. The reasons can give us a good idea of how a user’s background affects her opinion. The evaluation document requests that the user form an opinion based solely on the merits of a single animation, without comparison to other animations. The evaluation

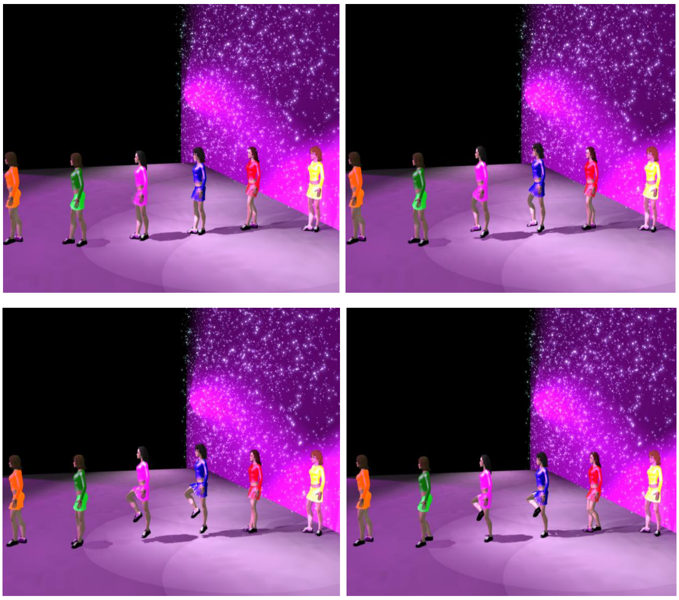

Figure 5: Results from six characters performing unsynchronized movement. The characters are split into three groups of two, with each group performing a routine different from the other groups.

concentrates on determining how successful our approach is by observing how the changing system parameters affect the user's opinion. Tables II and III display the results of the evaluation according to the responses of all 18 participants.

Table II: Overall results of the evaluation, taking into account the responses of all 18 people involved in the assessment of the animations.

Animation	**Number of 'yes' responses**	**Number of 'no' responses**	**Percentage of people who liked the animation**
BrownEyedGirl	12	6	67%
Eminem	9	9	50%
FieryNights	17	1	94%
Finale	13	5	72%
GetItStarted	14	4	78%
Nutcracker	7	11	39%
Warriors	16	2	89%
WideOpenSpaces	14	4	78%

The two animations with the highest number of 'yes' answers are both animations using Celtic music. As noted in Table II, the FieryNights animation was found appealing by 94% of the evaluators, while the Warriors animation was appreciated by 89% of the evaluators. It is interesting to note that the animations with the highest tempo (Eminem at 171 bpm) and the lowest tempo (Nutcracker at 67 bpm) are the animations found the least appealing by the majority of evaluators. The Eminem animation was only enjoyed by 50% of the evaluators, while the Nutcracker animation was liked by only 39%. These songs, however, also belong to musical types that do not typically suit dancing. Both rap and classical are difficult styles for an average person to dance to, so it makes sense that most people would feel that the dancing does not suit the music. The majority of respondents enjoy the remaining four animations, all of which correspond to music types that are traditionally easy to dance to. GetItStarted and WideOpenSpaces were appealing to 78% of evaluators, 72% of participants enjoyed the Finale animation, while BrownEyedGirl was appreciated by 65% of those involved.

Table III: Results of the evaluation split up by group into evaluators with dancing experience, evaluators with computer programming experience and evaluators with experience in neither.

Animation	**Dancing Experience**		**Computer Programming Experience**		**Neither**	
	Yes	**No**	**Yes**	**No**	**Yes**	**No**
BrownEyedGirl	5	1	4	2	3	3
Eminem	4	2	1	5	4	2
FieryNights	5	1	6	0	6	0
Finale	4	2	4	2	5	1
GetItStarted	4	2	6	0	4	2
Nutcracker	3	3	1	5	3	3
Warriors	6	0	4	2	6	0
WideOpenSpaces	4	2	6	0	4	2

The results from Table III have been divided based on their respective evaluator groupings. Several animations exist where all members of a group have found the result appealing. Participants with previous dancing experience enjoy Warriors best, with FieryNights and BrownEyedGirl tied for second. Those with computer programming experience enjoy FieryNights, GetItStarted and WideOpenSpaces the most of all the animations. Evaluators with no experience like FieryNights and Warriors the best, with Finale a close second. It is interesting to note that the animations liked best by the programming group all fall within the tempo range of 90-110 bpm. The participants with no experience overwhelmingly enjoy the animations with Celtic style music the most. The group of dancers also seems to enjoy the animations with Celtic style, as two of the top three animations were paired with Celtic music.

5. CONCLUSIONS AND FUTURE WORK

This paper presents a new music-driven character animation system that supports data-driven mappings of musical features to movements. The system helps users of all experience levels to produce appealing animations based on input music of any type and primitive dance moves and routines. One of the major contributions of this work to the area of character animation is its ability to build a motion sequence directly from extracted musical features. Unlike synchronization-based methods that simply alter an existing animation's timing in accordance to the musical beat, this system creates movements based on the musical beats and dynamics. The movements can easily change to reflect the mood and timing of the music, a feature that is not possible in systems similar to ours.

Another feature that is not supported in other systems is the ability to control multiple characters with different personalities in an animation. The user can build and easily integrate a troupe of dancers into the system. The dancers are not limited to performing the same movements, as the Celtic system is set up so that each character can use its own script file. Synchronization between characters is encouraged, but individuality makes the animation less mundane.

Our system is designed to be flexible for both the user and the animator. The system is set up to support extra primitive movements, as well as more dance types than just Celtic. The addition of other types of movements will encourage experimentation between dance structures, allowing a choreographer to easily mix moves from across different dance categories. Flexibility for the user is provided through both the script file and the musical input. Any type of music with noticeable beats can be used by the system to generate a specifically tailored animation that expresses the music. The script file gives the user a high level of control over the final animation and results that reflect her style and preference.

Figure 6: The system is easily able to accommodate multiple characters in the same scene, as demonstrated in the picture above. Sixteen girls are utilized in this particular performance.

Future work is planned for both the musical analysis section and the animation component. The occasional inaccuracy of the beat detection algorithm needs to be addressed, as well as its need for manual tweaking. We plan to design an automatic algorithm that is more accurate, as well as extract more musical features, such as the note pitch and melody, from the input signal.

We also intend to add more primitives and routines into the system in order to more faithfully represent Celtic dance. The system does not need to be limited to Celtic motion, however. Different types of dances can be added to future versions in order to increase the scope of the system and encourage experimentation between styles. Ballroom dances such as the Waltz or culture-based dances such as the Spanish Flamenco are among the possible dance types that could be incorporated into the Celtic system.

Lastly, the ability to randomly generate sections of a dance, or even an entire dance, automatically is a concept that should be included in the Celtic system. This function can be used to demonstrate the system to new users or fill in movements when a user has run out of ideas. It would increase the flexibility of the system and provide extra help for users with little experience or only a short amount of time.

REFERENCES

ALANKUS, G., BAYAZIT, A.A., AND BAYAZIT, B. 2005. Automated Motion Synthesis for Dancing Characters. *Computer Animation and Virtual Worlds16:3-4,* 259-271.

CARDLE, M., BARTHE, L., BROOKS, S., AND ROBINSON, P. 2002. Music-Driven Motion Editing: Local Motion Transformation Guided by Music Analysis. In *Proceedings of Eurographics UK,* Leicester, UK, June 2002, 38-44.

DIXON, S. 2003. On the Analysis of Musical Expression in Audio Signals. *Storage and Retrieval for Media Databases 5021,* 122-132.

DUNNE, C. 1996. *Celtic Feet*. ISBN: 156938147X. Acorn Media.

FOD, A., MATARIC, M.J., AND JENKINS.O. 2002. Automated Derivation of Primitives for Movement Classification. *Autonomous Robots 12:1,* 39-54.

GOTO, M. 2001. An Audio-based Real-time Beat Tracking System for Music With or Without Drum-sounds. *Journal of New Music Research 30:2,* 159-171.

GOTO, M. AND MURAOKA, Y. 1994. A Beat Tracking System for Acoustic Signals of Music. In *Proceedings of ACM Multimedia*, San Francisco, USA, October 1994, 365-372.

GOTO, M. AND MURAOKA, Y. 1999. Real-time Beat Tracking for Drumless Audio Signals: Chord Change Detection for Musical Decisions. *Speech Communication 27:3-4,* 311-335.

KIM, T., IL PARK, S., AND SHIN, S.Y. 2003. Rhythmic-Motion Synthesis Based on Motion-Beat Analysis. *ACM Transactions on Graphics 22:3*, 392-401.

SCHEIRER, E. 1998. . Tempo and Beat Analysis of Acoustic Musical Signals. *The Journal of the Acoustical Society of America 103:1,* 588-601.

SHIRATORI, T., NAKAZAWA, A., AND IKEUCHI, K. 2004. Detecting Dance Motion Structure through Music Analysis. In *Proceedings of International Conference on Face and Gesture Recognition*, Seoul, Korea, May 2004, 857-862.

SHIRATORI, T., NAKAZAWA, A., AND IKEUCHI, K. 2006. Dancing-to-Music Character Animation. *Eurographics 25*:*3*.

TAYLOR, R., TORRES, D., AND BOULANGER, P. 2005. Using Music to Interact with a Virtual Character. In *Proceedings of New Interfaces for Musical Expressions*, Munich, Germany, August 2005, 220-223.

TZANETAKIS, G., ESSL, G., AND COOK, P. 2001. Audio Analysis using the Discrete Wavelet Transform. In *Proceedings of WSES International Conference on Acoustics and Music: Theory and Applications,* Skiathos, Greece, September, 2001.

WOCH, A. AND PLAMONDON, R. 2004. Using the Framework of the Kinematic Theory for the Definition of a Movement Primitive. *Motor Control 8*:*4*, 547-557.