The Animation Canvas:

A Sketch-Based Visual Language for Motion Editing

by

Michael Welsman-Dinelle

B.Sc., Dalhousie University, 2007

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

December 2011

Abstract

We propose the Animation Canvas, a system for working with character animation. The canvas is an interactive two-dimensional environment similar to a sketch editor. Abstract interaction modes and controls are provided to support editing tasks. Consistent motion-as-curve and pose-as-point metaphors unify different features of the system. The metaphors and interactive elements of the system define a visual language allowing users to explore, manipulate, and create motions.

The canvas also serves as a framework for presenting interactive motion editing techniques. We have developed two techniques in order to explore possibilities for motion editing while demonstrating the flexibility of the system. The first technique is a method for interacting with motion graphs in order to explore motion connectivity and construct new blended motions from shorter clips. The second is a real-time spatial interpolation system that enables users to construct new motions or control an animated character.

Table of Contents

List of Figures

Chapter 1

Introduction

Computer-animated characters are ubiquitous in modern visual media. In pre-rendered movies, complicated animated characters with many degrees of freedom run, jump, and perform other movements. In video games, characters are controlled in real time and react in response to dynamic user input. Creating animations for these applications is important and challenging. We propose a new system, the Animation Canvas, to assist with this task.

The Animation Canvas is an interactive, functional prototype with an interface resembling a sketch editor or drawing application (see Figure 1.1). It is referred to as a “canvas” because it provides an open work surface similar to an artist’s canvas. The user can sketch static illustrations or annotations directly on the canvas surface using a mouse or stylus. The drawing metaphor is extended with tools for sketching abstract, interactive visual elements on the canvas surface.

The most important element is the *motion curve*, a stroke drawn on the canvas that is bound to a character animation instance (see Figure 1.2). A stroke represents the timeline of an animation. Unlike the timelines of traditional video editing or viewing layouts, motion curves can be positioned freely on the canvas surface. The user can also interact with the curves directly to perform motion editing operations like cutting, splicing, or keyframing. The purpose of motion curves and the other canvas elements provided is to combine simple, lightweight sketch-style interaction with powerful abstractions in order to make it easier and faster to work with motion data.

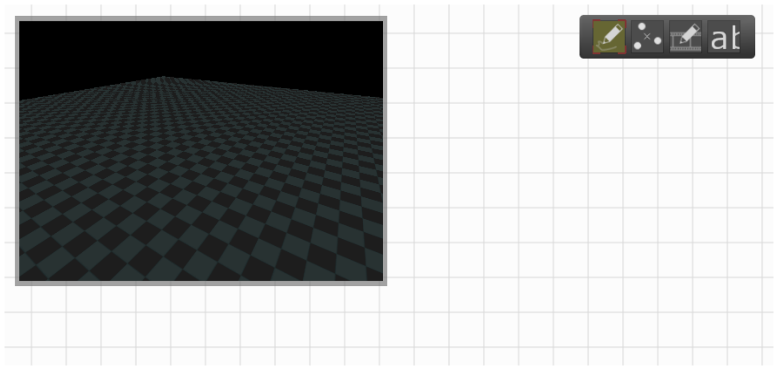


Figure 1.1: A blank canvas surface.
The Animation Canvas surface is equivalent to a canvas in a drawing program. A palette of tools is provided as well as a window for viewing animations.

The canvas system is intended to be a part of a larger workflow for creating animation content. The canvas operates on a palette of pre-existing motions. These motions might be generated with motion capture techniques or algorithms, created by an artist, or produced using a combination of methods.[1]

[1]The seed animations used for testing the canvas are from the CMU Graphics Lab Motion Capture Database at `http://mocap.cs.cmu.edu/`

After the canvas is used to create new motions, they can be exported and refined using other tools. To support this motion prototyping use case, canvas operations are intended to be simple and fast.

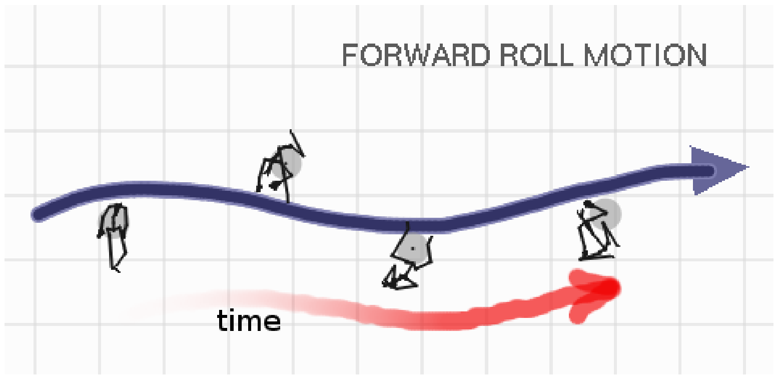


Figure 1.2: Motion curve and annotation.
The user has sketched an interactive motion curve, which appears above in blue. The arrowhead is drawn at the end where the user completed the drawing stroke. The user was able to pull character poses from the animation by clicking on part of the motion curve. The direction of the motion timeline is shown in red. Parts of the curve closer to the blue end arrow correspond to later parts of the animation. At the beginning of the curve the character is preparing to roll and by the end the roll is complete.

The Animation Canvas supports the following basic types of motion editing tasks:

- **Viewing.** Users can play back animations or view individual frames. Multiple simultaneous views are supported and the user can control the camera perspective.
- **Cutting and splicing.** A motion can be cut into smaller motions, or two motions can be combined end-to-end to create a longer motion.
- **Synthesis.** There are multiple ways to create new animation content by combining existing poses and motions.
- **Reuse.** The system maintains a dynamic working set of motions created by the user that can be used for further editing.

One goal of the Animation Canvas design is to create simple, consistent user interface metaphors and conventions that support this set of tasks. The canvas is built around the concept of a palette of imported and recently-created animations (See Figure 1.3). As a starting point, the palette is filled with existing motions captured or created by other systems. As new motions are created they are added in turn to the palette, which contains a dynamic working set of animations. The motion palette naturally complements the canvas metaphor as an analog to brushes used in a drawing application; by selecting a motion on the palette the user declares that the motion will be bound to new curves drawn onto the canvas. “Running” or “jumping” may be thought of as analogs of a “red” or “blue” pen, for example. The motion curves drawn onto the canvas are dynamic and interactive, however, not static pixels.

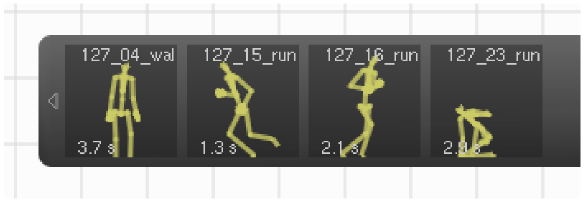


Figure 1.3: The canvas motion palette.
The motion palette is analogous to a brush palette in a drawing program. The user clicks on a motion to associate it with newly-drawn motion curves, as a colour might be chosen for drawing new lines.

Many motions can be drawn at once and can be arranged freely on the canvas surface. Motions can also be drawn end-to-end to be spliced together or can be added to interconnected webs of motion curves called *motion graphs*. Each path along a motion graph with a given topology corresponds to a new motion that can be added to the motion palette.

Another concept built into the canvas is the character pose as a building block of character motions. The canvas allows users to pull poses directly out of motion curves and to reuse them as keyframes. Alternatively, interpolation regions allow users to use pose arrangements to generate new character poses and motions or create real-time performance animation. These features are shown in a single screenshot of the system in Figure 1.4.

1.0.1 Visual Language of the Canvas

The interactive graphical elements of the Animation Canvas define a visual language. The elements can be composed together to create semantics in

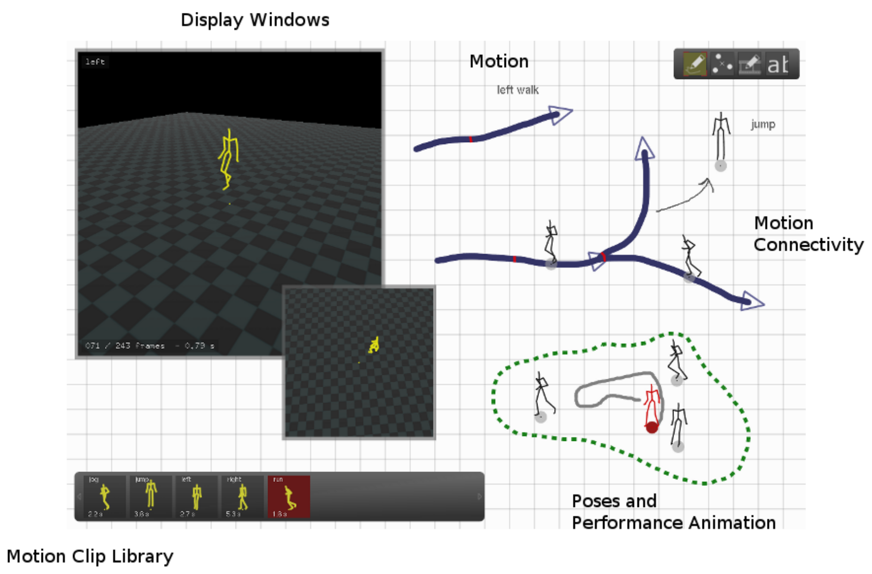


Figure 1.4: Screenshot of the system.
The canvas has palettes for selecting working motions and interaction modes. Motions, curves, and interpolation spaces are drawn directly on the canvas surface. Motions and poses are created and viewed in real time.

the form of different output poses or motions. The canvas also features a hierarchy of abstractions, shown in Figure 1.5.

Higher-level abstractions aggregate lower-level elements together while hiding some of their properties to simplify user interaction. Motion curves represent motions that are composed of a large set of poses. The user can interact with the curves directly to perform operations like blending without being concerned with the complexity of each individual pose. Motion graphs allow the user to work with sets of motions and motions spanning across multiple captured segments of animation. Spatial keyframing is another abstraction that allows the user to choose poses or motions from a large implict set specified by positioning key poses. Additional layers of abstraction are described in Section 5.1.1.

Another important aspect of the visual language of the canvas is its cyclical, closed nature. The output of operations with poses, motions, motion graphs, or spatial keyframing is always new poses and motions that are visually and functionally equivalent to the motion data loaded when the canvas is first launched. Output poses and motions can be used to iteratively create new content that diverges from the original motion set.

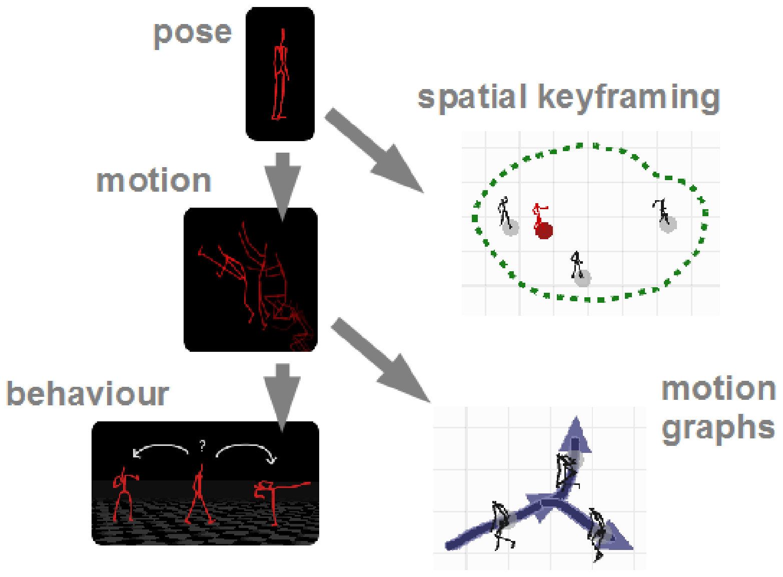


Figure 1.5: Layers of abstraction of character animation.
A single pose is a building block for a motion or can be used to specify a spatial keyframing region. Sets of motions can be used to define motion graphs or character behaviours. Abstractions simplify interaction by hiding some properties of their constituent elements.

1.1 Contributions

The primary contribution of this project is a visual language for motion creation and editing. The canvas creates explicit visual representations of motions, poses, and the relationships between them. Manipulation of motion data takes place through direct manipulation of the visual representation. This visual language is supported by the following elements:

1. **A sketch-based system for motion editing.** The Animation Canvas provides a blank canvas as an open work area. Using a stylus or mouse, the user can paint visual elements freely onto the canvas and arrange them arbitrarily within the two-dimensional canvas plane.

2. **Motion graph visualization and editing.** Motion graphs, as presented by Kovar et al. [23], are explicitly rendered on the canvas surface and paths through the graph can be played back in real time or saved by directly interacting with the motion graph. Motion clips can be interactively added to or removed from a motion graph.

3. **Real-time character control or performance animation through motion playback.** The Animation Canvas can direct a live character and play back motions in real time.

4. **Spatial keyframing as described by Igarashi et al. [21].** The canvas saves motions drawn as curves through the keyframing space. These motions can in turn be used as building blocks for other motions.

Chapter 2

Related Work

This section presents existing work and describes how it relates to the Animation Canvas. First, the process of creating and editing motions is discussed. An overview is given of motion capture and of basic techniques for transforming motion data, several of which are components of the canvas. Motion resequencing and motion graphs are of particular importance. Next. performance animation systems are introduced, including the spatial keyframing system that forms the basis of one method of motion synthesis and editing supported by the canvas.

Later sections of this chapter discuss two key concepts that have informed the design of the canvas system. A brief history of sketch-based systems is given and the usefulness of the interaction style is established. Work related to the visual language of the Animation Canvas is presented. This work underlines the importance of careful design of visualizations to aid the user in understanding underlying information about motion data. Finally, some existing animation tools are discussed and compared to the Animation Canvas.

2.1 Motion Capture

Motion capture is a way to convert real motion of a subject into motion data useful for computer animation and other applications. It provides an alternative to manual motion specification using keyframing and other methods. A survey of motion capture techniques is provided by Moeslund and Granum [28].

Motion capture systems are often built with a series of video cameras that use computer vision techniques to convert video into skeletal model poses that represent frames of animation. This transformation requires several steps. First, cameras are positioned so that they focus on a physical viewing volume in which the motions take place. Multiple cameras capturing two-dimensional video make it possible to disambiguate the three-dimensional configuration of an actor performing a motion, and make the system more robust to occlusion. Reflective markers may be used to capture joint locations on the human model. Alternatively, some capture techniques may use other image features such as the silhouettes of models presented by Vlasic et al. [41]. These reflectors become high-intensity points captured by the cameras. The points are in turn used to estimate the pose of the model's skeleton during a fitting step. The pose of the skeleton is the final output of the motion capture process.

2.2 Motion Editing

There are many reasons why it might be impractical or impossible to use pre-captured motion data in animation applications. Many applications are dynamic, requiring changes to be applied at runtime. Lasseter also points out [24] that many desirable motions are not physically correct. It is not possible to obtain these motions directly from live actor performance.

One obvious way to edit motions is to manually change the parameters of the motion data. This is tedious because of the number of degrees of freedom of character models and the frame rate required for good quality animation. For example, ten seconds of animation of a model with 50 degrees of freedom sampled at 60 frames per second results in 30,000 values describing the motion. Keyframing techniques are commonly used to allow relatively direct manual control of an animation while cutting down on the amount of data to be manipulated. With these techniques, the character pose is only specified for certain frames. Other frames are produced by blending the keyframes with existing motion data or by interpolating between keyframes.

Popović and Witkin present a method for providing keyframes as constraints that are blended into existing motions [45]. Similar methods for blending using time-varying weights are used by the Animation Canvas. Kovar and Gleicher introduce registration curves in order to make blending useful for a wider variety of motions [22].

2.2.1 Motion Retargeting

Motion data is often structured as a series of frames each consisting of a list of joint angles.The joint angles alone do not fully specify the motion; they are used to pose one particular character skeleton. Joint angles for one skeleton might not look correct when used to pose another skeleton with different segment lengths. Skeletons may not even have a similar topology or number of joints, in which case there is no obvious direct mapping of joint angles. The problem of determining a mapping is described by Gleicher [13] as *motion retargeting*. Gleicher explores retargeting for skeletons with different segment lengths and for skeletons with differing topologies. Hecker et al. [17] describe retargeting techniques for user-created characters from the game *Spore*.

No motion retargeting is implemented for the Animation Canvas. The results of operations involving multiple poses or motions with different skeletons are undefined. In practice, this problem is avoided by starting with a set of motions all based on one character skeleton. Since no canvas operations modify the character skeleton there no need for retargeting.

2.3 Motion Resequencing

There are a number of reasons why it might be impossible or impractical to create a desired motion ahead of time. In video games, for example, it is impractical to capture every complete motion that a character might be required to perform over the course of gameplay. One approach to overcoming this problem is to instead find ways to structure motion capture data, resequencing short capture segments and concatenating them to form longer motions.

Kovar et al. present motion graphs [23] as a solution to the motion resequencing problem. Arikan and Forsyth present a similar algorithm [2]. The algorithm constructs a graph structure where graph edges are motion clips and vertices correspond to blending points in the animation where there may be multiple options for subsequent clips. In order to construct the graph, animations are compared to find frames where the characters are sufficiently similar that blending can produce adequate results. Because changes in joint angles do not necessarily correspond to proportional changes in final character appearance, points sampled from the character's skeleton or a downsampled mesh skin are used for better comparison results. Animations are cut to obtain clips that can be blended together and these are used to build a dense motion graph. The motion graph is used to generate animations for constrained walking problems from generic motion capture data.

Parametric motion graphs used in interactive applications are presented by Heck and Gleicher [16]. In contrast with graphs built out of motion clips, parametric graphs connect motion spaces. Each motion space represents a continuous parameterized range of related motions that can be created through blending. This approach allows for greater flexibility in output motions, making it possible for characters to interactively move or turn in a continuous manner, or bend down to pick up an object at a given location.

Motion-motif graphs, presented by Beaudoin et al. [6], are a different modified motion graph where clusters of motions are grouped together. Principal component analysis (PCA) is used to reduce the dimensionality of the motion data. Motion data is then grouped into clusters using a heuristic k-means clustering algorithm and each cluster is assigned a letter. Motions are transformed into strings of characters corresponding to the clusters of the character poses and motifs, patterns in the data, can be discovered by looking for patterns in the strings. Motions are composed by finding paths through the motion motif graph, blending motions from motif to motif.

Motion graphs are also used by Sung et al. to generate motions for crowds of characters [36]. A randomized search algorithm is used to find approximate results quickly, making it possible to generate distinct motions for a larger number of characters. Reitsma and Pollard present techniques for evaluating motion graphs in terms of how they permit a character to efficiently navigate to any point within a given environment [33].

The Animation Canvas supports the creation of motion graphs. These graphs are sketched directly onto the canvas. In contrast with the approach presented by Kovar et al. [23], the purpose of the graph is to allow a user to construct a persistent visual representation of the relationships between motion clips. In order to generate composite clips the user sketches a traversal through the graph.

2.4 Performance Animation

Performance animation is the process of interactively creating animation under the real-time direction of a human animator. The animations created through this process cannot be anticipated ahead of time and must be created through the use of an interative system. One example of a performance animation system is presented by Laszlo et al. [25]. This system allows users to control characters in a physics simulation in real time with a mouse and keyboard (see Figure 2.1). In one trial presented by the paper, mouse position is bound to desired joint angles of a lamp-shaped character. By moving the mouse the user causes the figure to apply joint torques and move about in simulation. In more complicated simulations, keyboard input was used in conjunction with finite state machines to manage the periodic cycles of locomotion and other behaviours. With this setup users were able to drive a cat-like figure, causing it to run and jump by specifying when it should transition into new motion states. Control actions can also be used to specify inverse kinematics targets and other aspects of character motion and behaviour.

A second performance animation system presented by Zhao and van de Panne [49] makes it possible for users to interactively control human characters that perform snowboarding stunts or dives in a physics simulation. In this system, joint angle targets are selected from an action palette using a gamepad. Offline editing and slow motion modes are provided.

Users of both systems are able to directly control articulated characters in a realistic physics simulation. One key advantage of these systems is that they can be used to create motions or stunts that were not necessarily conceived of at the time of system development. Allowing users to directly control aspects of character movement opens up a rich space of possible motions, but it is difficult to devise a real-time system enabling a user to fully control a complicated humanoid character. For this reason, neither system focused on the careful creation of expressive motions that rely on fine tuning of the animated characters.

Not all performance animation systems are physics simulations. Chai and Hodgins [9] provide techniques for performance animation based on a small number of control points. They implemented a simplified motion capture system, meant to be practical for home use, that obtains control points

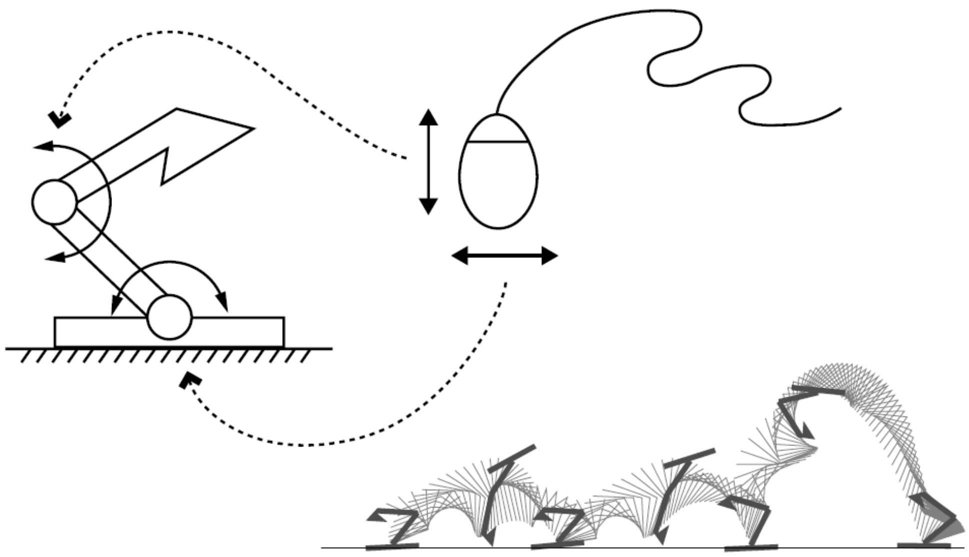


Figure 2.1: Interactive control of an articulated figure, Laszlo et al. [25]. Mouse position determines the target joint angle, causing the Luxo lamp to move.

from actors. The control points are used to search a set of existing motions and new motions are synthesized in real time to match the movement of the actor. The overall technique is an example of controlling a high-dimensional character model with a low-dimensional control signal. Often this is desirable since it can be impractical to generate a full control signal in real time. It is impractical, for example, for a user to directly control a character model with 50 degrees of freedom using a mouse and keyboard.

Another approach to low-dimensional control, *spatial keyframing*, is presented by Igarashi et al. [21]. Spatial keyframing is a technique for interactively generating kinematic character motions based on key poses. In contrast with traditional keyframing techniques, the key poses are associated with a point in space rather than a point along a timeline. Instead of playing back a timeline and generating each frame of animation with a blended character pose, the user interactively navigates through the three-dimensional blend space using a control cursor to specify which pose should be drawn at the given frame.

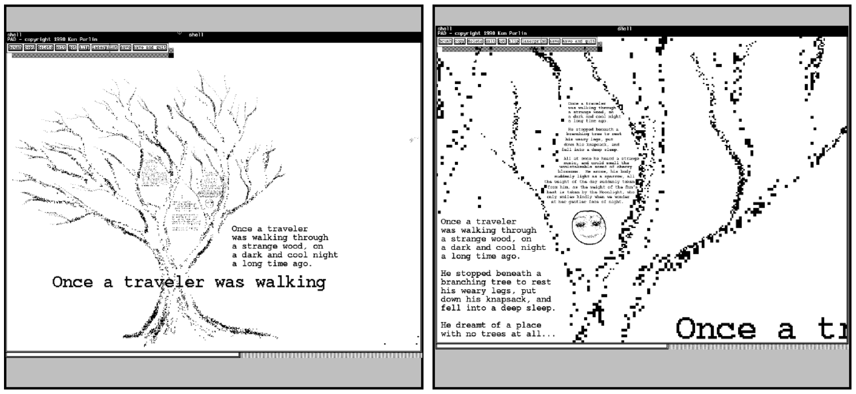
In order to create blended poses, the spatial keyframing system uses radial basis function interpolation. This interpolation method produces an interpolating function by combining a series of weighted radial functions centred at the keyframe positions. The interpolating function evaluates to the keyframes at the keyframe points and produces blended approximations at other points. A detailed description of method is given by du Toit [12].

The spatial interpolation technique is another way to tackle the problem of controlling a character model with more degrees of freedom than a user can directly manage in real time. This approach does not necessarily permit the user to independently control each part of the model, but by thoughtfully choosing character poses it is possible create a wide variety of motions. The authors demonstrate how the system can be used to generate motions for both simple and highly articulated characters. Choi et al. provide layering and other extensions to spatial keyframing [10].

The Animation Canvas includes a modified implementation of spatial keyframing (see Section 3.5).Canvas spatial keyframing is useful both for performance animation and for generating persistent poses and motions to be used in later motion editing tasks.

2.5 Sketch-Based Interfaces

The desire to create innovative visual interfaces dates back to at least the 1960s, the decade when Ivan Sutherland developed *Sketchpad*, a system enabling users to directly interact with a computer-driven display using a light pen to draw shapes and perform other tasks [37]. Later works include *Pad* [32], which introduces an interactive two-dimensional work surface that can display graphics while supporting contextual or semantic zooming. Pad uses a spatial metaphor for interface design in order to take advantage of natural spatial cognitive abilities. The system uses the dynamic display to reveal data only at useful scales. For example, on a small scale the surface might display titles while at a larger scale it may display summaries and full text, as shown in Figure 2.2. Consistent spatial arrangement allows the user to relate the current view to other views and to the overall structure of the data. *Pad++* expands upon this system [7].



Once a traveler was walking through a strange wood, on a dark and cool night a long time ago.

Once a traveler was walking

Once a traveler was walking through a strange wood, on a dark and cool night a long time ago.

He stopped beneath a branching tree to rest his weary legs, put down his knapsack, and fell into a deep sleep.

All at once he heard a strange music, and could smell the unmistakable scent of cherry blossoms. He arose, his body suddenly light as a sparrow, all the weight of the day suddenly taken from him, as the weight of the Sun's heat is taken by the Moonlight, who only smiles kindly when we wonder at her gentler face of night.

He dreamt of a place with no trees at all...

Once a tr

Figure 2.2: Screenshots from *Pad* [32].
Contextual zooming shows different content at a larger scale in the right view.

Saund and Moran proposed *PerSketch*, what they call a perceptually-supported sketch editor, in 1994 [35]. This system uses a variety of techniques, including early computer vision, to augment the functionality of simple pixel-by-pixel sketch editors. For example, selections of sketched curve segments are inferred from dragging operations, allowing users to reposition portions of sketched figures. Saund and Moran discuss the distinction between *paint*-style and *structured graphics*-style editors. Traditional paint-style editors force users to interact with a blank canvas on a pixel-by-pixel basis. Structured graphics-style editors allow users to create abstract objects, like ellipses, but restrict how the user can interact with them. The Animation Canvas combines both styles of editing and has some features that span both styles. Annotations are paint-style, but motion curves are abstract objects.

Some more recent related work uses a similar flexible two-dimensional sketch interface within more specific domains. In the case of this thesis, an open canvas is used as a framework to support animation work. *MathPad2* [26] is built upon the idea of extending handwritten math with a dynamic display capable of quickly generating graphs and other graphics derived from user-provided information. Extracting this information is a difficult problem because it relies on handwriting recognition and domain knowledge.

K-sketch is another system that provides users with a two-dimensional canvas to use to create motions [11]. K-sketch allows users to sketch out world objects which they then directly manipulate by specifying changes in position, scale, and orientation of objects. Users create animations in real time, preserving the timing of their actions, and can layer different animations together to create a complicated scene (see Figure 2.3). A fundamental difference between this system and the Animation Canvas is that, on the Animation Canvas, users sketch out many motions as timelines instead of directly drawing characters to be animated on a global timeline. The interaction metaphor of the Animation Canvas motion curve is not as direct and the end goal is to work with collected motion capture data to produce plausible motions of highly-articulated figures rather than a sketched scene.

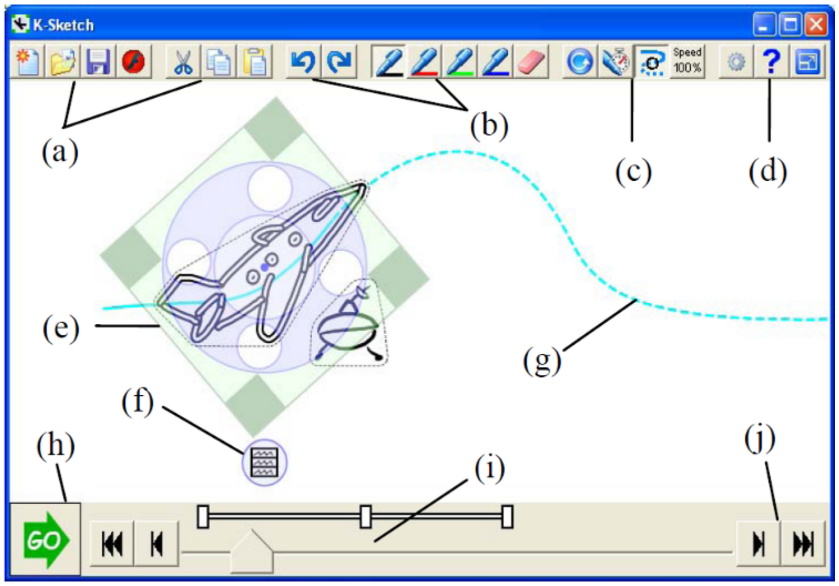


Figure 2.3: A screenshot of *K-Sketch* [11].

Motion Doodles, presented by Thorne et al., provides an interface enabling the user to directly sketch out a character that is then automatically turned into an animated, articulated figure [39]. Once a character is created, the user can sketch a path to set the character in motion. This differs from the Animation Canvas in that users directly specify characters and motions instead of working with pre-existing character models, abstract motion spaces, and abstract topologies of motion graphs. One key component of Motion Doodles is the algorithm to create a bipedal character model from a profile sketch. Another particularly relevant component of the system is the visual language used to specify sequences of character actions. By drawing loops and other patterns, the user can induce the animated figure to perform flips and other behaviours. Part of the goal of the Animation Canvas is to support a similar abstract visual vocabulary for working with character motion. Instead of sketched paths, the vocabulary of the canvas consists of motion curves and character poses.

2.6 Visualization, Language, Symbology, and Notation

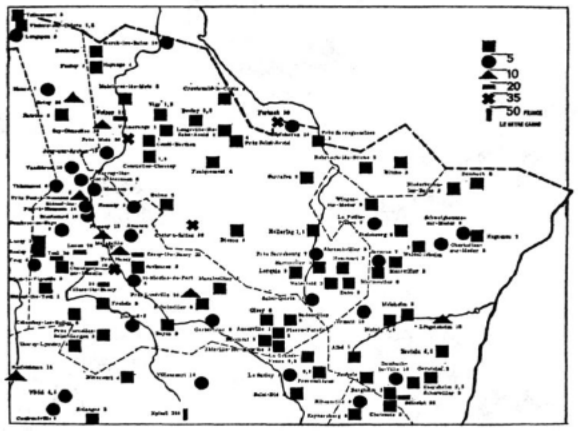
There is a large field of research dedicated to studying how to effectively present information visually. Tufte [40] discusses the visualization of quantitative information, mostly by analyzing static graphics, and provides an interesting history of the field of visualization. Ware [43] presents a survey of computer-driven information visualization (InfoVis), discussing visualization techniques and characteristics of human perception. Bertin [8] provides a rigourous evaluation of different styles of visual representation and shows how they vary in terms of effectiveness. His relatively early work focuses on static charts and graphs, but alludes to a future of common interactive, computer-driven displays.

Bertin emphasizes that the usefulness of graphical representation extends beyond aesthetics, and that graphics can serve as external memory or as tools of calculation and inference. He points out that the structure of a graphical representation can have a dramatic impact on its usefulness in accomplishing a particular task. He describes partially separable perceptual channels of the human visual system and how they related to graphical properties like size, colour, or orientation of visual elements. Each channel has different characteristics and must therefore be used differently in order to convey information as clearly as possible.

The choice of mappings from data to visual channel can have a dramatic impact on the usefulness of a graphic as a tool to explore questions about a data set. Figure 2.4 provides an example of this effect. The problem of designing visual representations for the Animation Canvas requires a careful consideration of similar alternatives.

Zhang and Norman [48] provide an evaluation of numeration systems that again demonstrates how visual representations can be valuable problem solving tools. They find that some systems are dramatically more efficient than others at supporting arithmetic operations and other related tasks. In order to explain these differences, they discuss the notion of graphical representation as external memory, suggesting that the cognitive work required to solve an arithmetic problem can be partially offloaded into a visual representation that keeps track of important working information. They claim that this effect is so strong that the invention of the Arabic number system itself is a key advancement in human history.

Figure 2.5 shows a choreography notation system that specifies synchro-


(a) Data encoded as discrete glyph forms.

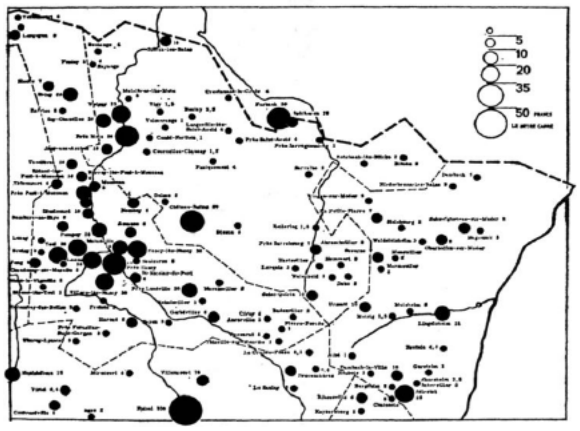

(b) Data encoded as glyph size.

Figure 2.4: Two visualizations of land prices in northeastern France [8]. Figure (a) requires a careful visual inventory of data points that must be interpreted using the legend. In figure (b) the larger dots are more visually salient; we immediately see where land values are highest.

nized music and dance together in a single two-dimensional static image. This is an example of a compact notation that provides a powerful mapping from simple stick figures to complex human motion.

The Animation Canvas has a number of features that serve as external memory for a user working with motion data. Motion graphs, for example, maintain information about how motions related to each other can be blended together. Poses can be used to highlight key parts of motions. Annotations allow the user to add more information as desired. Panning allows the user to take advantage of a canvas surface much larger than what can be displayed on the screen at one point in time.

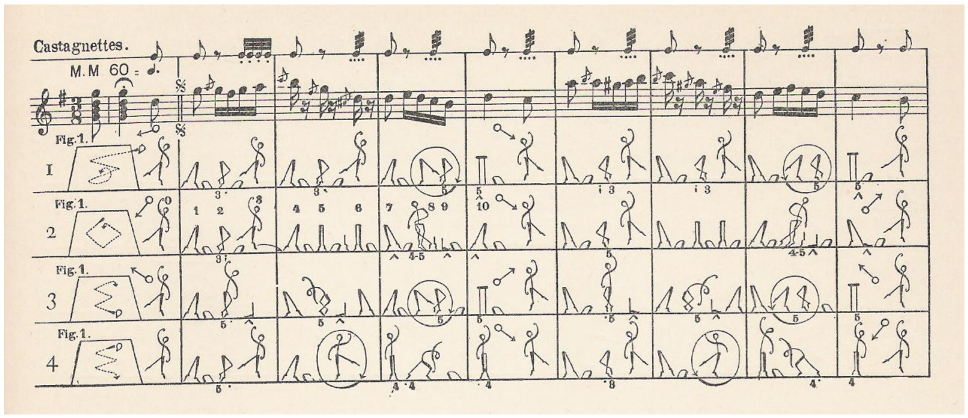


Figure 2.5: Friedrich Albert Zorn's representation of music and motion on paper.
The movements of *La Cachucha* are described along with synchronized music. Image available under GNU Free Documentation license from `http://en.wikipedia.org/wiki/File:Zorn_Cachucha.jpg`.

2.6.1 Visual Programming Languages

Visual programming languages are also related to the system of interactive visualizations presented by the Animation Canvas. In contrast with conventional textual programming languages, visual programming languages use visual representations of programming constructs to simplify the task of developing software. Whitley uses empirical investigation to evaluate visual programming languages [44]. The importance of semantic groupings and meaningful visualizations are described. In some cases, visual programming languages are demonstrated to be an improvement over other methods.

Several visual programming languages have become common software tools. In most cases, they are limited to more specific domains than general-purpose text-based programming languages; this is unsurprising given the tradeoff between simplicity and flexibility. One example of a commercial visual programming language is Apple's *Quartz Composer* [1]. Another common commercial tool is National Instruments *LabView* [29], used to create circuit models. The logic of circuit design is closely related to text-based programming.

2.6.2 Visualizing Animation

The tradeoffs involved in displaying three-dimensional character animations on a two-dimensional screen are complicated and have been studied in their own right. Reitsma and Pollard show that human sensitivities to different characteristics, such as acceleration or deceleration, vary [34]. Hodgins et al. show that the way an animated character is rendered has a significant impact on how the underlying motion is perceived; users are more capable of discerning changes in a skinned model than in a stick figure model [19].

One problem with character animation is that direct playback is slow. Searching for one particular character pose in a motion, for example, requires viewing every frame, unless the viewer has some general idea of the characteristics of the motion. Many attempts have been made to provide a useful synopsis of animations with two-dimensional, often static images. Assa et al. provide a technique for automatically choosing key poses that attempt to convey character motion with only a few frames [3]. The key poses are superimposed in one or more images to provide a motion synopsis. Transparency is used to make it possible to see poses that would otherwise be occluded (See Figure 2.6).

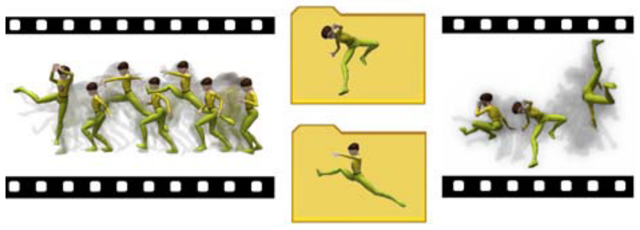


Figure 2.6: *Action Synopsis* [3].
Key poses are selected from a motion and superimposed to produce a representative image. Transparency is used to de-emphasize less distinctive poses that are thought to be less important for conveying the original motion.

Yasuda et al. present *Motion Belts*, a technique that both selects key poses and selects optimal orientations for poses projected down to the two-dimensional image plane [46]. Motion belts are shown in Figure 2.7. Hu et al. describe *Motion Track* [20], which uses an alternative pose selection algorithm and a completely different kind of visualization. Motion tracks embed key poses in a two-dimensional reference space, mapping pose characteristics to location in the two-dimensional plane. A curve is drawn on top to show the transitiion between poses (See Figure 2.8). This approach is reminiscent of the spatial keyframing techniques discussed in Section 2.4.

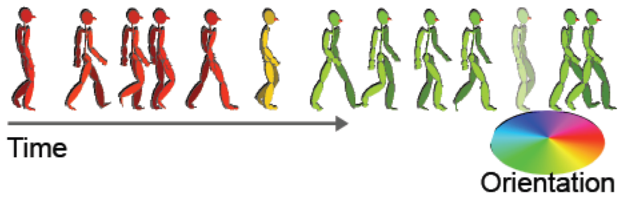


Figure 2.7: *Motion Belts* [46].
A series of key poses is drawn. Each pose is oriented so that it can easily be distinguished from the character's rest pose. In this example, several different pose orientations are used.

The Animation Canvas displays two-dimensional representations of motions as curves. No attempt is made to automatically show key poses, but the user is able to double-click on motion curves to permanently display any desired pose. Automated pose selection could be a future enhancement.

2.7 Animation Tools

A variety of commercial tools are available that allow users to work with motion data. Autodesk *3ds Max* [4] and *Maya* [5] are popular tools used by professional artists to create high-quality animations. These tools are more complicated than the Animation Canvas and have user interfaces for specifying model parameters at a higher level of detail. They are also built

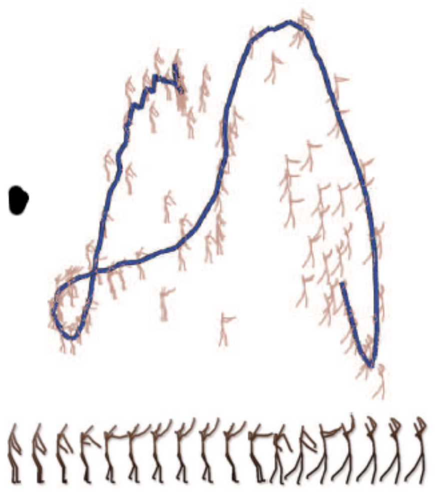
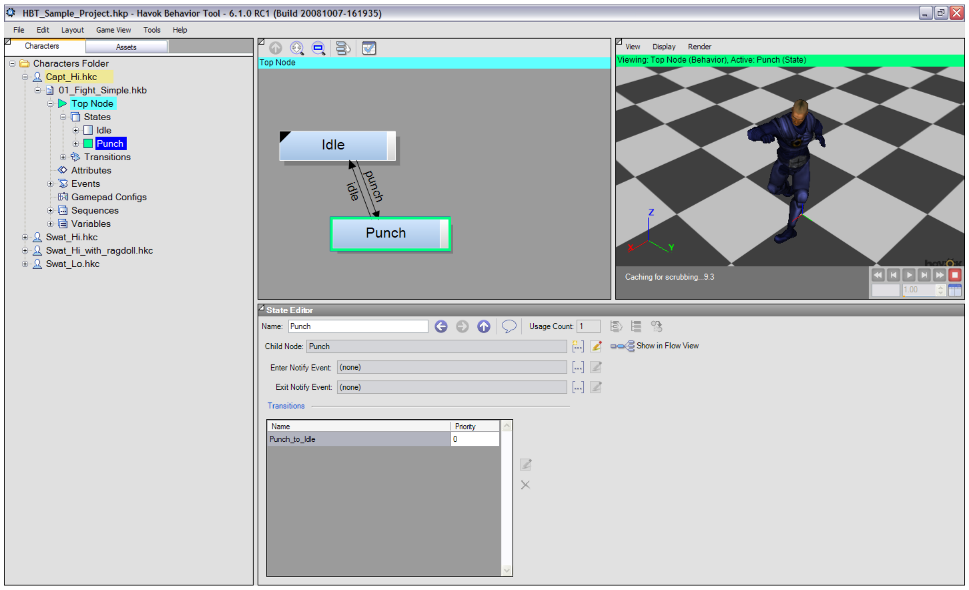


Figure 2.8: *Motion Track* [20].
High-dimensional pose data is reduced to two dimensions and poses are embedded at the appropriate location in the space. A curve is traced through the space to show the transitions from pose to pose that make up a given motion.

around presenting single animations with linear timelines, in contrast with the more abstract motion curves drawn on the Animation Canvas.

Many other tools, such as NaturalMotion *Morpheme* [31], have emerged to specifically support the creation of video game content. The *Havok Behavior Tool*, shown in Figure 2.9, allows users to construct motion graphs to define transitions between character states and animations [30]. These motion graphs can be tied to game logic. For example, the character might transition to a "Punch" state when a button is pressed during gameplay, and then return to a resting "Idle" state once the punching animation has been fully played.

Interactive visualizations of motion graphs can also be created using the Animation Canvas. One important distinction between graph creation using the two systems, however, is the representation of motions and transitions as either graph edges or vertices. In Havok, motions are vertices and transitions are graph edges. On the canvas, motions are edges and connect to each other at vertices. The canvas approach is consistent with a motion-as-timeline metaphor and allows for motions of arbitrary length that can easily be selected and manipulated. The Havok approach emphasizes connectivity and simplifies the view of motions in an interface where they cannot be altered.



Name: Punch

Usage Count: 1

Child Node: Punch

Enter Notify Event: (none)

Exit Notify Event: (none)

Name	Priority
Punch_to_Idle	0

Figure 2.9: A screenshot of the *Havok Behavior Tool* [30]. Motion are represented as boxes in a state machine-like representation of a motion graph.

Chapter 3

The Animation Canvas

The Animation Canvas is an environment for working with motion data. The main interactive part of the system is a two-dimensional open canvas region supporting direct user interaction (see Figure 3.1). Users sketch motions onto the canvas surface and, using the visual language of the canvas, explore these motions on the canvas or generate new content to be used in subsequent sketches or to be exported to some other medium.

This chapter provides technical details to explain how the components of the Animation Canvas work together. The next chapter provides usage scenarios to demonstrate how these components are used to produce animations.

3.1 Sketching on the Interactive Canvas Surface

The Animation Canvas is a primarily two-dimensional interactive environment. The word *canvas* is meant to evoke an artist's canvas, an interaction metaphor frequently used in drawing programs. The canvas is an open environment that allows users to position controls freely within the two-dimensional canvas space, arranging them in whatever way is most desirable. Users can also draw or sketch directly on the work surface as they

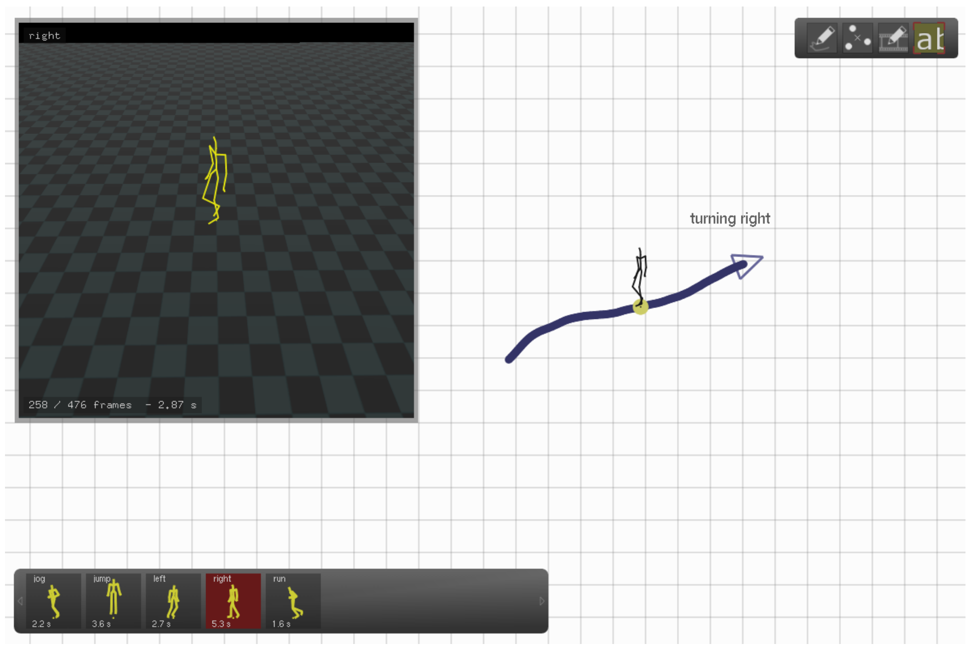


Figure 3.1: A motion curve drawn on the Animation Canvas.

would in a sketch editor. On the Animation Canvas, some strokes become abstract representations of motions. After a motion curve is drawn, the user can interact with it to work with the underlying motion.

The user interacts with the canvas by using a mouse or stylus and tablet. Most operations are performed through a combination of clicking, double clicking, hovering, or dragging actions. One exception is textual annotation on the canvas, which uses keyboard input. Users may also choose to annotate by drawing directly on the canvas surface. Annotation creates static marks or text on the canvas but many other user operations generate dynamic interface controls that, once drawn, respond to user interaction by changing in appearance.

3.1.1 Interacting with Canvas Elements

The user draws elements such as annotation or motion curves onto the canvas by clicking and dragging on the white canvas surface. Palettes for changing input modes and viewing windows for displaying animations exist in a second layer in front of the canvas surface and can be repositioned.

The canvas implements a global event system that propagates mouse and stylus input to canvas controls. Events are first propagated by layer, then by drawing order. If a click or drag event affects a palette in the front layer, the event is not passed along to controls on the canvas work surface. The most recently drawn canvas elements are drawn in front of older canvas elements and are given priority for interaction. In other words, with few exceptions, the user interacts with whatever is immediately visible under the cursor. Simple bounding box checks keep this system responsive even with hundreds of canvas elements.

Deletion of elements has not been implemented for this prototype but could exist as other input modes. For example, a standard eraser tool could be used to remove annotations. A splicing tool could be used to cut motion curves. Another possibility explored was a tool for dragging existing elements that would be combined with a ”trash can”. Elements dragged to the trash can would disappear from the canvas. See Section 4.2.5 for more details.

3.1.2 Canvas Palettes and Toolbars

In order to support mode switches between motion curve drawing, annotation, and other features, a floating toolbar is provided (Figure 3.2). Another toolbar contains the palette of available motions. Motions can be thought of as “brushes” that affect the properties of new motion curves drawn onto the canvas, as a brush in a sketch editor might specify that a new stroke will be thick, thin, or dotted. The user clicks on a motion to make it the active brush (Figure 3.3). Any motion curve drawn represents an instance of the active motion chosen in the palette.

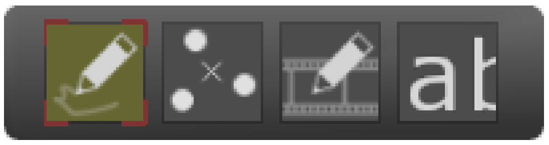


Figure 3.2: Canvas mode toolbar.
The canvas mode toolbar supports switching between four modes: freehand annotation, spatial keyframing region construction, motion curve drawing, and text annotation.

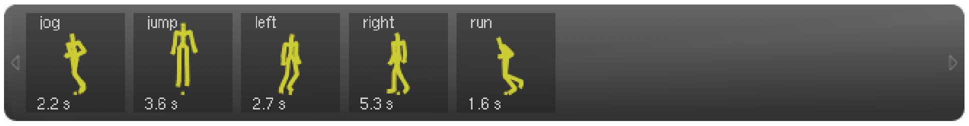


Figure 3.3: Canvas motion palette.
The motion palette shows a list of motions that can be used when drawing motion curves. Each motion is shown with a name and runtime. Users can hover their cursors over a motion to play it. When new motions are created, they appear on this toolbar.

Note that the selection of a motion from the motion palette is only meaningful for drawing motion curves. It has no effect on other tools such as annotation. The motion drawing tool is therefore automatically selected from the toolbar when a motion is selected; the user can click on a motion in the palette then draw immediately onto the canvas. If another tool is selected the motion palette maintains its current selection. If the user reselects the motion curve drawing tool the original motion is still selected from the palette.

The motion palette is designed to be central to the workflow of the canvas. Animation tasks begin by selecting motions from the palette, which may initially be populated with imported motions from motion capture databases or other sources. As new motions are created they can be saved back to the motion palette, enlarging the set of motions available for the user to work with. Ideally, this evolving palette of available motions would allow the user to diverge from the original set of input motions and create a wide variety of desirable synthesized motions.

3.1.3 Annotation

Annotations are inactive elements that can be added to the canvas. The user may draw freehand annotations; these are particularly useful for highlighting regions of the canvas or showing links between canvas elements. Typed annotations are also supported in order to allow the user to add text notes that are embedded in the canvas along with motion curves and other elements. Examples of annotation are shown in Figure 3.4.

Annotations are drawn in a dark grey colour to create contrast with the canvas background while differentiating inanimate strokes and text from dynamic controls drawn onto the canvas. Motion curves, for example, are blue and yellow and much thicker than annotation curves.

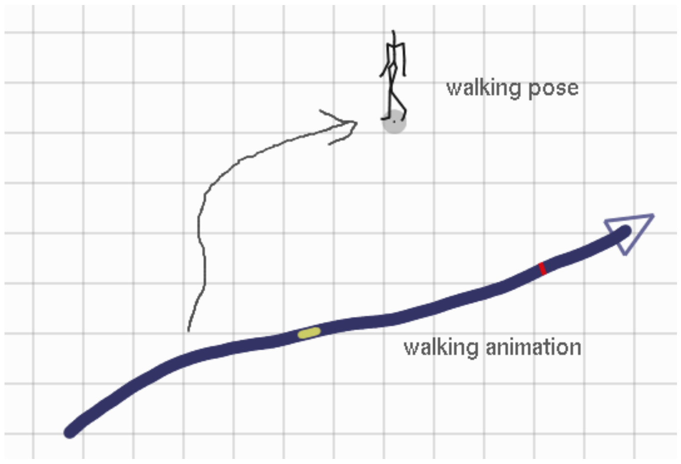


Figure 3.4: Typing and drawing on the canvas.
Two kinds of annotation are supported by the Animation Canvas. One tool behaves like a pen, allowing the user to draw static pixels directly onto the canvas surface. A second tool allows the user to place a cursor on the canvas and type out text that would be difficult to draw with the pen.

3.1.4 Interactive Canvas Viewing Windows

Viewing windows or viewports are a part of the animation canvas that allow the user to see motions rendered in a three-dimensional world (see Figure 3.5). When the canvas is first opened it has a single viewing window. Windows can be resized, cloned, or closed. Each window has its own camera that can be repositioned by the user. Multiple linked views can easily be set up to show the same figure from different angles, giving a more complete perspective on the three dimensional world.

Windows views are tied to motion curves or connectivity shown in the canvas with free-form curve segments or poses shown as points on the canvas. When the user hovers the cursor over a point on a motion curve the character is positioned appropriately and rendered in all viewports on the canvas. The rendering includes a checkerboard ground plane, the character root, and slight atmospheric attenuation to give a sense of scene depth. Fully rendering the skeletal model in a three-dimensional scene gives the user a more accurate sense of what the animation looks like than the two-dimensional projected poses shown on the canvas.

The canvas supports multiple linked viewports, created from the main viewport by double-tapping with the stylus. Once created, the viewports can be dragged to a new location or can be resized. Another feature of the viewports is that each has an independent camera that can be moved by dragging the cursor. Moving the viewpoint in multiple viewports allows the user to see a character motion from multiple perspectives, providing a more complete sense of the three-dimensional motion than would be possible by rendering directly onto the two-dimensional canvas surface.

The camera position is determined by three quantities. First, a radius r specifies the distance from the camera position to the root of the character being displayed or the centre of the screen. Next, θ specifies the rotation about the y-axis, parallel to the floor plane. The final quantity, ϕ, is an angle on the interval $[-\pi/2, \pi/2]$ specifying the deviation from the floor plane. Using these three quantities as inputs a function was implemented to derive the camera position and orientation. The camera always points toward either the origin of the world coordinate frame or the root of the character. If a motion is being played, the camera follows the character and the motion is always displayed in the viewing region.

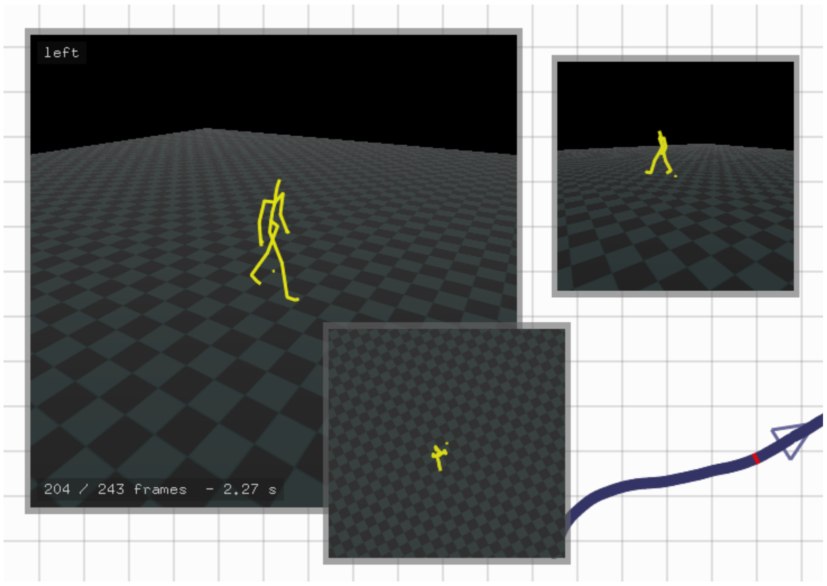


Figure 3.5: Canvas viewing windows.
Multiple linked canvas viewports show different views of the same animation in order to give an accurate sense of the motion of the skeletal character in its three-dimensional environment.

3.1.5 Stateful Character Control and Playback Modes

There are two ways to play back animations in canvas viewports. The first mode of playback directly renders an animation; each pose of the animation is rendered in sequence with respect to the origin of the motion in world coordinates. This mode is useful for directly viewing existing motions in isolation, but makes it difficult to see how a motion might be applied to a persistent character moving about in a three-dimensional world. The second playback mode, stateful playback, uses motions to interactively drive a persistent, video game-style character. In this playback mode, playing a running motion causes the character to run from its initial location to a second location. Playing a subsequent jumping motion would cause the character to jump from their second location to a third location.

In the stateful playback mode the character has its own persistent position and skeletal configuration. When new motions are triggered, they create relative changes in the character's state instead of directly specifying the character's absolute location and position. In order to make motions fit together, the character can be given a rest pose to return to after each motion. Motions are reproduced relative to the character's origin rather than the starting position of the motion in world coordinates. Similarly, motions are oriented in the X-Z ground plane relative to the character; a straight walking path will move the character forward in whatever direction it was facing, but turning motions can change the character's orientation.

The stateful playback mode makes it possible to control a character as a player would in a video game. This feature makes the canvas a potential prototyping tool for testing the suitability of motions to real character control in a virtual world which could be extended with any level of game logic, rendering, etc. Stateful playback also makes the canvas a tool capable of supporting performance animation, or direct real-time character control. Finally, stateful playback also highlights the use of the canvas as a manipulable controller; users can build and modify motion graphs and other constructs on the canvas that are themselves used as character controllers.

3.2 Poses on the Canvas

A character pose is the set of translations and joint angles that fully describe the position of the character's skeleton and the configuration of its articulated components. Poses can be considered samples of continuous motion. Conversely, an animation can be produced by rendering poses in sequence.

Poses are considered building blocks for motions on the Animation Canvas. Poses are shown as points on the canvas. Each point embedded in a motion curve corresponds to an instant in time and a pose sampled from the motion. To display a pose, the user can tap on any point along a motion curve. The pose is projected to two dimensions and is always rendered with the vertical and horizontal animation world axes are aligned with the canvas axes. This means that the character will be upright in a neutral "T-pose" but would, for example, remain upside down in the middle of a rolling animation.

One a pose is displayed it can be repositioned by dragging. At that point it becomes an independent element on the canvas surface. Poses can be used for annotation or can be dropped back into motion curves to set key frames (see Figure 3.6). Poses can also be used to populate interpolation regions, making it possible to synthesize new motions and poses (see Section 3.5).

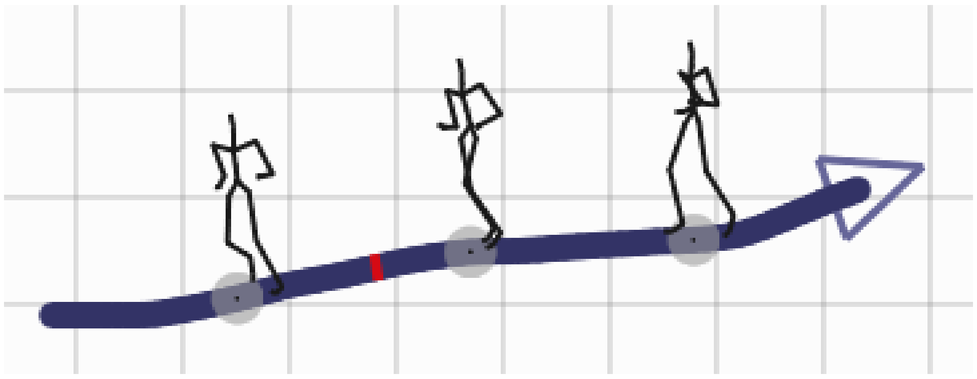


Figure 3.6: Poses selected from a jogging motion curve.

Poses are rendered by recursively traversing the character skeletal model's hierarchy, applying transformations and drawing bones as required. This process is described in Appendix A.

3.3 Motions on the Canvas

The Animation Canvas displays motion timelines as free-form curves drawn in the two dimensional canvas plane. The user draws a motion curve by first selecting a motion from the palette of available motions and then drawing out a curve on the canvas surface using the mouse or stylus. The results are shown in Figure 3.7. Motion curves can be drawn anywhere that is not occluded by a viewing window. Motion curves can cut across other motion curves; a halo around the curve makes it clear which intersecting segment is in front to avoid ambiguity.

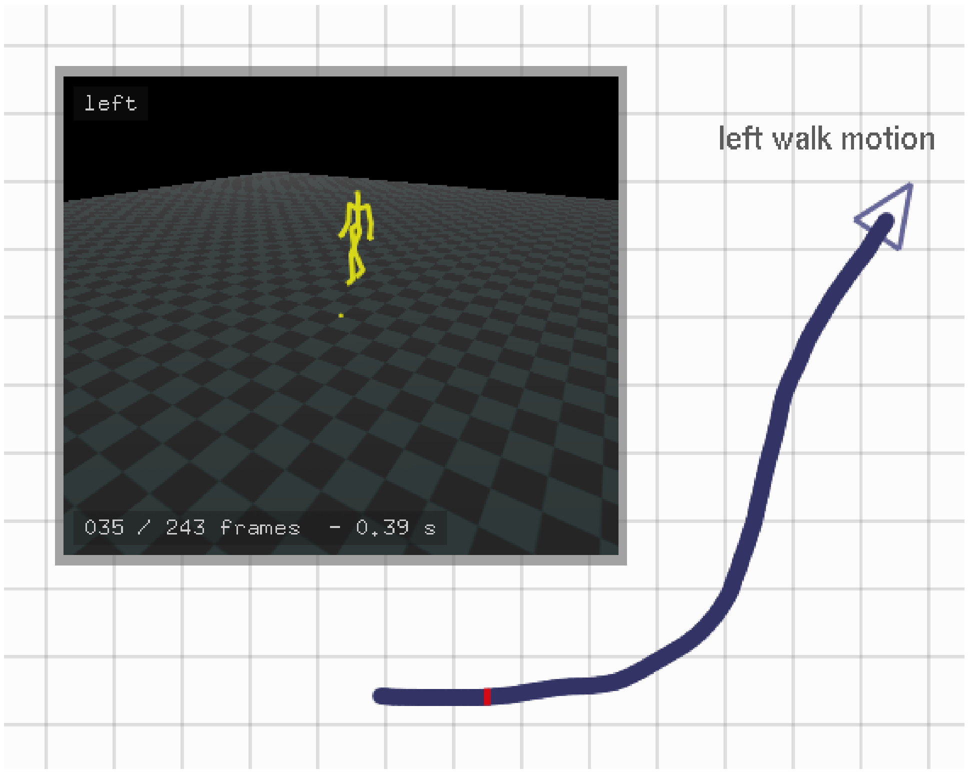


Figure 3.7: Canvas motion curves.
A motion curve appears as a blue curve on the canvas with an arrow indicating the direction of the animation. The location of the hovering cursor is shown with a red bar that corresponds to a frame of the animation that is played in all canvas viewing windows.

Once the curve is drawn, the user can tap on a point on the curve to view an animation frame or play through the motion and view a rendered 3D animation demonstrating the character motion. The correspondence of curve points and motion frames is calculated relative to the length of the motion curve drawn (See Figure 3.8). Because the orientation and length of the curve are not significant, the user can arrange motion curves in many different ways and easily connect motion curve segments. A more detailed explanation of this design decision is given in Section 4.2.2.

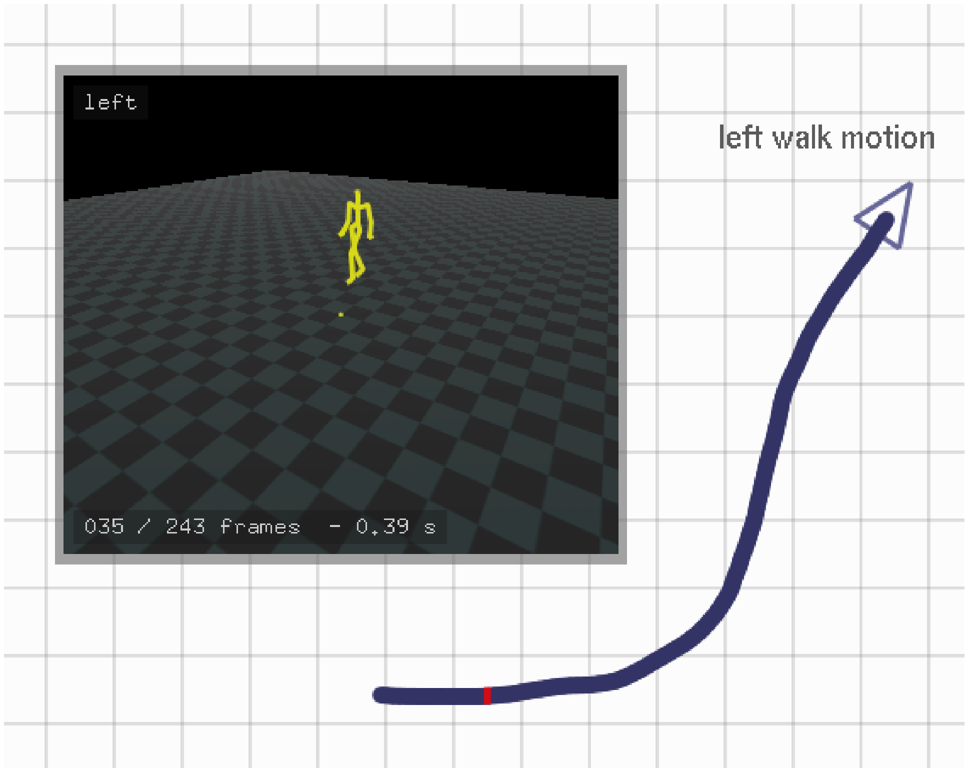


Figure 3.8: Canvas motion curve length, position and orientation. A motion curve appears as it is drawn on the canvas. The animation progresses along the path that was drawn by the user; an arrow provides a cue to indicate the end of the motion. The orientation of the motion curve is not significant. The position of the motion curve is only significant if an endpoint connects to one or more other curves, in which case the curve is part of a motion graph.

In order to cut motions into smaller parts, the user can highlight a portion of the motion curve with a drag-style selection technique. In cases where motion curves overlap, the selection range follows along the curve segment closest to the original selection point. If the user starts by selecting one curve and crosses another curve, the second curve is not selected. This behaviour ensures that users select a continuous motion segment.

An alternative to drag selection would be to support single clicks or stylus taps at each endpoint of a motion selection. The downside of this approach is that more complicated motion graphs (see Section 3.4) often have more than one path connecting two points, and selecting across motion boundaries is a key feature of the system. A shortest path algorithm might be a way to provide a useful starting point with these selections, which could then be refined by moving the path around. A similar system is in use for Google Maps, which selects travel routes automatically [14].

The selected portion of the motion curve corresponds to a motion clip. A single tap plays back the selected motion clip and double-tapping adds the motion clip to the motion palette so that it can be used in the future as a new, independent motion. Free-form motion curve drawing also allows the user to define relationships between motions.

Motion curves are drawn as a path, a series of two-dimensional points on the canvas sampled over time through mouse events. Once the path is drawn, it is smoothed by averaging each point's location with its neighbours in order to reduce noise and produce a more attractive curved appearance. Endpoints along a path are not changed during the process because this could affect connectivity with nearby graphs on the canvas and would be inconsistent with the input provided by the user. For more technical details about interaction with motion curves see Appendix C.

3.3.1 Motion Blending and Transitions

One of the primary functions of the Animation Canvas is motion synthesis. One very simple way to create new motions is to bring together two existing motions, playing the second motion after the first. If the end of the first motion seamlessly matches the beginning of the second, the two clips can be combined end-to-end as a new motion and no more work is required. However, there are often discontinuities in position, velocity, and higher order changes in the articulated skeleton when two existing motions are combined together. To overcome this problem, both motion clips must be modified near the joining point.

In order to achieve continuity, the canvas applies sinusoidal blending when two motion clips are combined together. In formal terms, we can imagine two motions p and q. Each motion is an array of n channels sampled over time t. We can think of the motions as n functions over time: $m_{1\ldots n}(t)$.

In order to create a new motion r, we must choose a blending interval T that determines the length of the time that overlapping motions p and q will be blended together. For times before the blend, we take the values of the first motion, p. After the blend, we take the values of the second motion, q. During the blend, we combine the two values using a weighting function $w(t)$:

$$r_{1\ldots n}(t) = w(t)p_{1\ldots n}(t_{start} + t) + (1 - w(t))q_{1\ldots n}(t), 0 <= t <= T \qquad (3.1)$$

The constant t_{start} is added for the first motion since we begin blending near the end of the clip. In other words, t_{start} is the total running time of the motion p minus the blending interval T.

The canvas uses a sinusoidal weighting function for blending motions. The weighting function w at time t over the blending interval $[0, T]$ is given by:

$$w(t) = \frac{sin(\pi t + \pi/2) + 1}{2} \tag{3.2}$$

Adding $1/2$ to t translates the values of the *sin* function so that it varies from -1 to 1. The total must be increased by one and divided by 2 in order to produce the desired weights from 0 to 1. The desirable characteristic of this weighting function is that it ensures continuity of character positions as well as higher-order continuities of changes and acceleration of joint angles. The character will move continuously and avoids introducing jerkiness, although there is no guarantee that the blended motion will appear plausible.

The blending interval used is defined by a constant and can be easily modified. By default, it is set to 0.3 seconds, a value suggested by Wang and Bodenheimer [42].

3.4 Motion Connectivity on the Canvas

A graph is a data structure consisting of a set of vertices and a set of edges defining topological links between those vertices. The graph structure provides a good way to encode relationships between character motions. In the context of the Animation Canvas, a motion graph is constructed using motion curves as graph edges (see Figure 3.9). End points of the motion curves are the vertices of the motion graph. Two or more motion curves may be joined at a single vertex; in this case, transitions can be created between these motions. Vertices of a degree higher than one create topologies linking different motions and defining which motions are closely related.

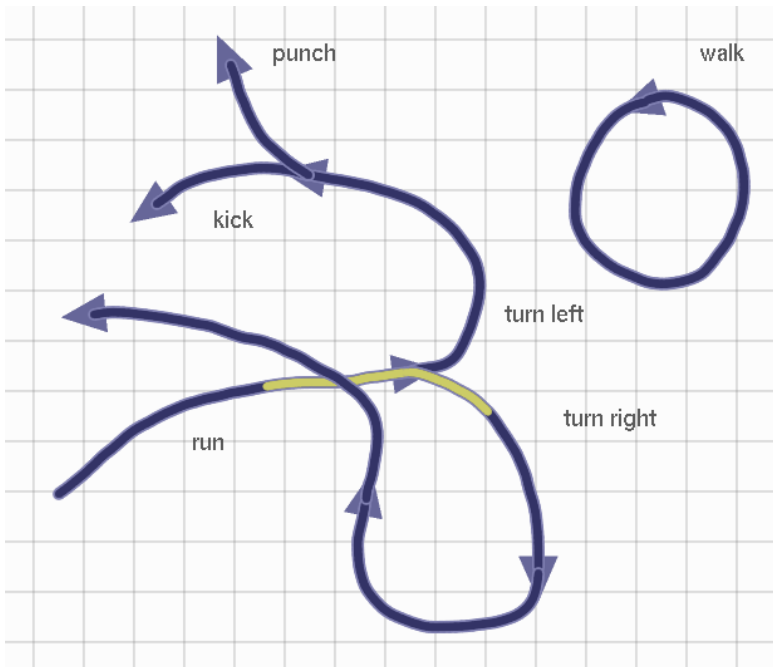


Figure 3.9: Canvas motion graphs.
A motion graph on the canvas showing connectivity between motions. This graph is non-planar, has two connected components, and one cycle. A selection is shown in yellow that crosses under another uninvolved motion curve.

Motion graph data structures are built in real time as the user sketches motion curves onto the canvas. When the user begins to draw a motion curve it either starts by connecting to an existing curve's ending arrow or is an isolated curve on the canvas. When the cursor hovers over a motion curve end arrow it turns yellow to signify that any new curves drawn will connect to the existing curve. A new curve drawn by tapping on a highlighted yellow arrow snaps to the ending coordinate of the existing motion curve. Instead of moving the cursor, a motion curve line segment is drawn to connect the cursor location to the existing curve. In this way the user can seamlessly connect curves without having to perfectly locate the endpoints of existing curves.

Each curve maintains a list of pointers to both the motion curves that precede it and the curves that follow it in the graph. When the user begins sketching a new curve near an existing graph vertex, the neighbour pointers of these graphs are updated to build the motion graph. It is possible to draw a motion curve over another existing motion curve, so motion graphs drawn with motion curves need not be planar graphs.

A path through the graph corresponds naturally to the motion sequence that arises from concatenating motions from all curves along that path. To allow for greater flexibility, subsections of motion curves can be selected at the front and tail of a path in order to include partial motion clips. This is consistent with simpler single motion curve partial selection.

The selection path is built using the motion curve neighbour pointers that make up the motion graph and is combined with information about the specific motion frames that form the beginning and end of the selection. Path selection begins when the user taps or clicks on a motion curve, at which point dragged selection is enabled for all adjacent motion curves. No other motion curves can be selected at this point because it would result in a disconnected path along the motion graph. As the user drags along a curve, the start or end frames of the selection are modified. When the user transitions across a vertex to a new motion curve, the entirety of the last curve is selected and the user can continue to drag along the new curve. In the case of selection of overlapping motion curves, the selection only extends across motion curves that are connected in the graph structure. In the case of cycles, a given segment of a motion curve can only be selected once, but the full cycle can be played back in a loop. The exact mechanics of motion curve selection are described in Section 3.3.

Once a path is selected, the composite motion represented by the path is generated. A list of motion clips is created from each of the motion curves involved in the selection. If the user only selected part of the motion curves at either end of the path, motion clips are generated from the motion curves that correspond to the selected components. A final step is to apply a blending function to provide a smoother transition between motions; this is explained in detail in Section 3.3.1. At this point, the complete motion corresponding to the selected path on the motion graph is available.

Once a new motion is generated from a path selected along the motion graph it is played back for the user in the canvas viewports. At this point, the user can tap on the selection to permanently add the new motion to the corpus of motions permanently available on the motion palette.

Motion graphs provide the user with a way to generate complex composite motions from a series of motion clips. Because of the speed of selecting paths through the graph, the motion graph on the canvas is also a good way to explore the compatibility of existing motions for composition and blending. The motions themselves can also be modified on the fly to attain better compatibility since they are present as manipulable motion curves on the canvas.

Screenshots of the motion graph construction process are presented in Section 4.1.6.

3.5 Motion Interpolation Regions

Motion interpolation regions provide another way to create new motions. To take advantage of this feature, a lasso-style tool is used to demarcate a polygonal region of the canvas defined by the selection path (see Figure 3.10). A stippled green border is created to signify that the region behaves differently from the rest of the canvas. In contrast with the normal canvas environment, poses and motions positioned within the region serve as a basis for the creation of new motions.

Each point in a motion interpolation region is implicitly tied to a specific character pose that depends on other poses and motions dropped into the region. By tapping on an empty part of the region, the user can generate a new pose that can be used in other parts of the canvas. The user can also drag a path through the region that naturally corresponds to a motion created from the series of poses along the path. This new motion can be added to the palette of permanently available canvas motions with a single tap, making it available for later use anywhere on the canvas.

While the poses in a motion are determined by the position of the path, the timing of the motion is determined by the timing of the path's creation. A path P drawn into a motion interpolation region is a series of n

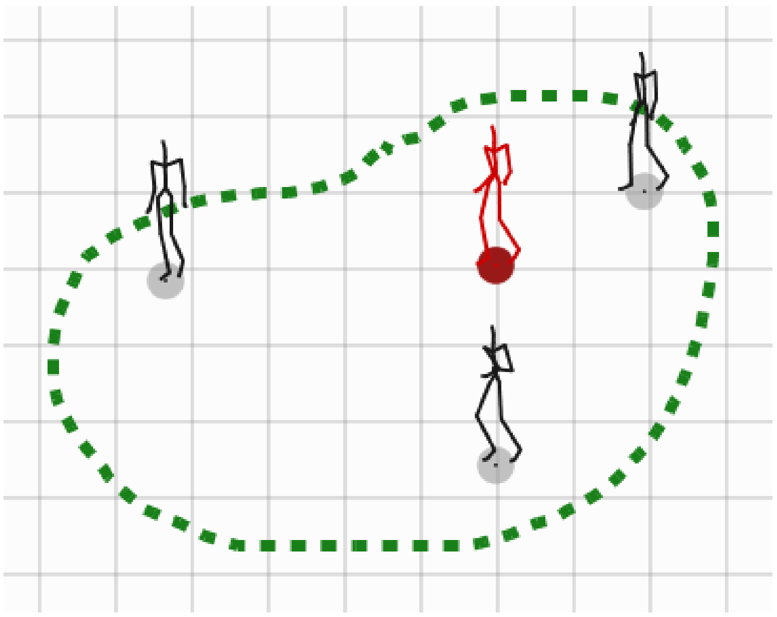


Figure 3.10: Canvas interpolation regions.
The green boundary signifies an interpolation region; these are the only regions of the canvas where position is significant. Black poses were dropped in from canvas motions by the user. The user clicks to generate a new red pose. The red pose is generated by blending black poses, with closer poses having a higher blending weight. Position does not play the same role outside of interpolation regions.

timestamped two-dimensional points, with each position corresponding to a character pose. The running time of a motion is equal to the time taken to draw the path; in other words, the difference between the timings of the first and last points on the path. Poses are sampled from the path at regular time intervals so that the output motion has a consistent frame rate. A linear interpolation scheme is used to interpolate each joint angle for the pose (see Figure 3.11).

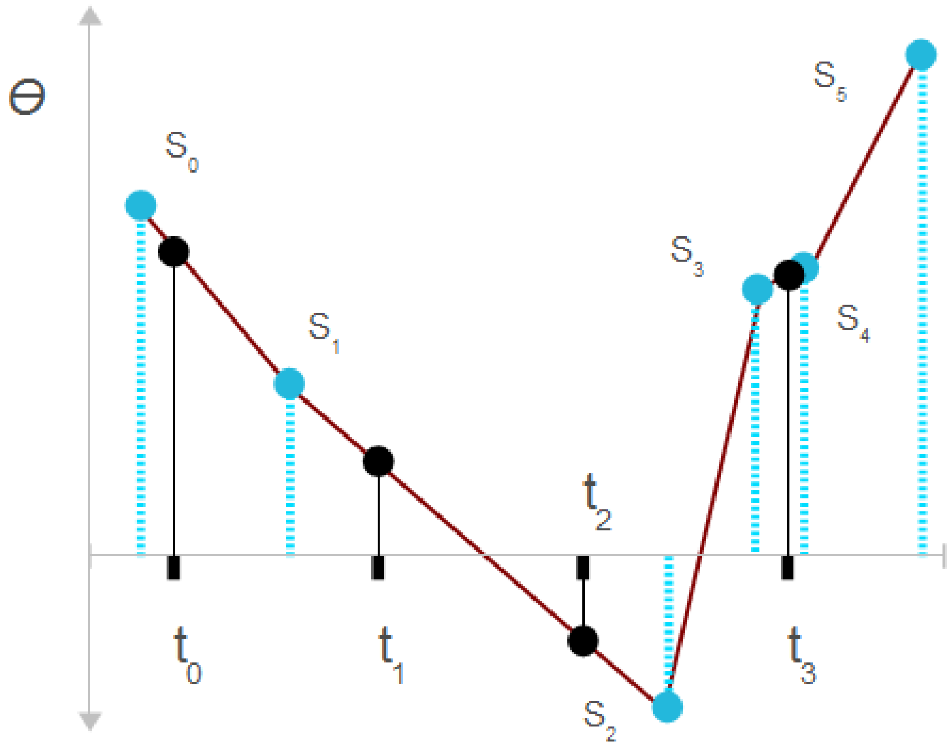


Figure 3.11: Resampling from pose joint angles
$S_0 ... S_5$ are resampled at regular time intervals $t_0 ... t_3$ using linear interpolation.

A final problem is the calculation of the poses defined by each point on the path. Poses on the path are created by combining poses that the user has added to the region. A weighted average is used to combine region poses such that their influence is inversely proportional to their distance from a point selected in the region. The distance value is squared to exaggerate the attenuation of farther poses and could be raised to higher powers to increase this effect.

3.6 Implementation Notes

The Animation Canvas is a C++ project developed using Visual Studio 2008. The project consists of approximately 100 source files and 10,000 lines of code.

The DevIL image library was used to load images for textures (`http://openil.sourceforge.net/`). Armadillo was used for linear algebra (`http://arma.sourceforge.net/`). GLUT was used for cross-platform windowing and interaction support (`http://www.opengl.org/resources/libraries/glut/`).

Figure 3.12 contains a diagram outlining the relationship between the software components of the Animation Canvas as described in this chapter.

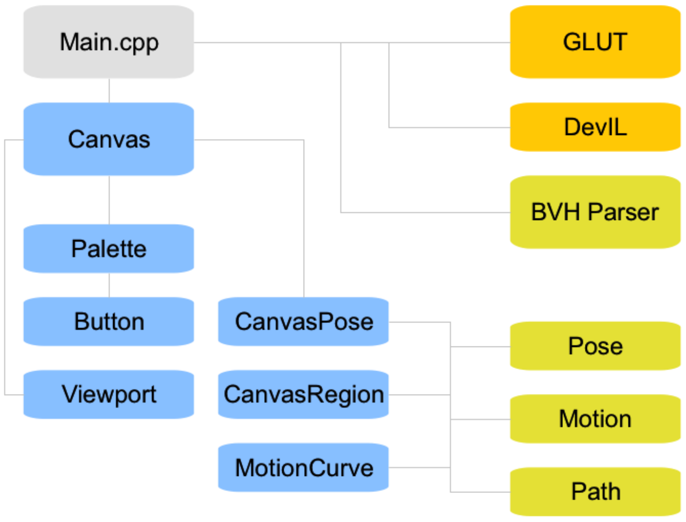


Figure 3.12: High-level diagram of the design of the Animation Canvas.

Chapter 4

Results

The primary result of this project is an interactive prototype. This section presents usage scenarios with screenshots of the prototype in action, followed by a discussion of design decisions and related observations.

4.1 Example Scenarios

The following scenarios describe the steps required to accomplish different tasks using the Animation Canvas interface.

4.1.1 Setting Up and Manipulating Viewing Windows

The canvas has floating windows that show views of an animated character that demonstrates motions as they are being worked with. The user can play back a pose or motion. If a motion is played, the current frame, total frame count, motion name, and current runtime are displayed. If the cursor is hovered over a motion curve the corrsponding frame is shown. When the canvas is launched a single viewing window is displayed.

Any viewing window can be resized and has an interactive camera for changing the view of the world. Any window can be cloned to create other windows with independent sizes and cameras. All viewing windows are displayed in front of elements drawn onto the canvas. The user can move viewing windows around to show occluded regions of the canvas. Another alternative is to pan the canvas since the location of viewing windows on the screen is unaffected by panning.

Interaction scenarios with viewing windows are shown in figures 4.1 through 4.3.

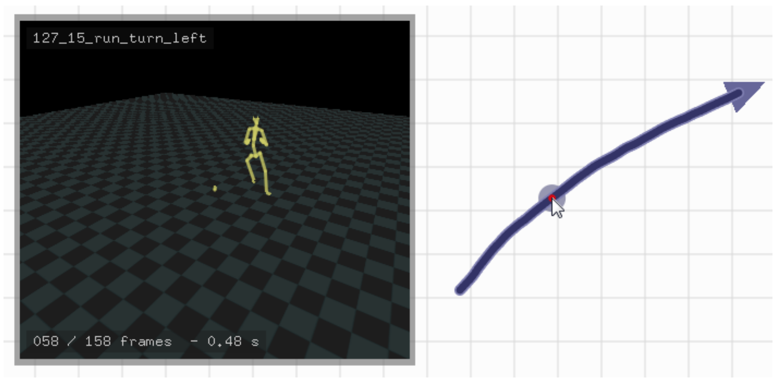
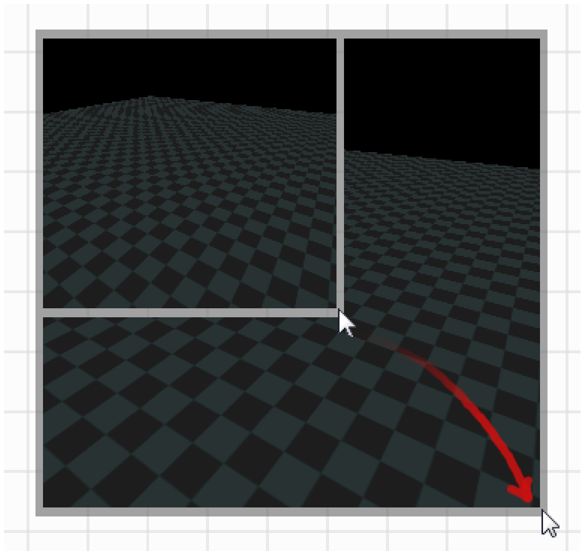
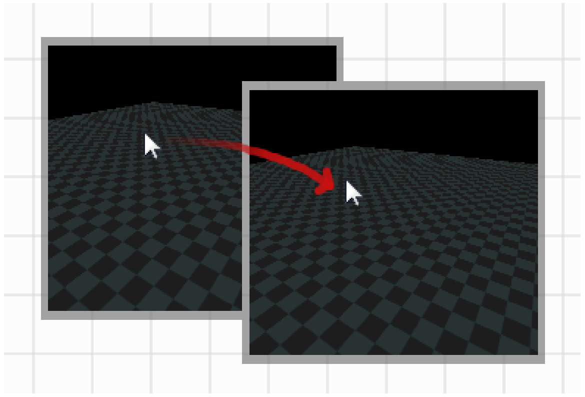


Figure 4.1: Interacting with viewing windows.
When the user hovers the cursor over a motion on the palette its first frame is shown. If the cursor is hovered over a motion curve the corrsponding frame is shown. If the cursor is however over a pose the pose is shown.

Figure 4.2: Manipulating viewing windows.

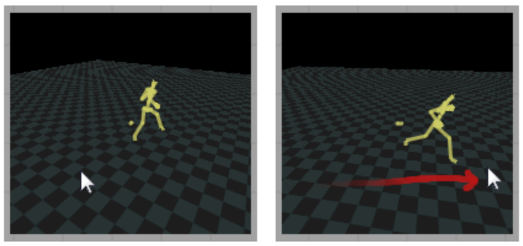


(a) Any viewing window can be resized by clicking and dragging on its border.

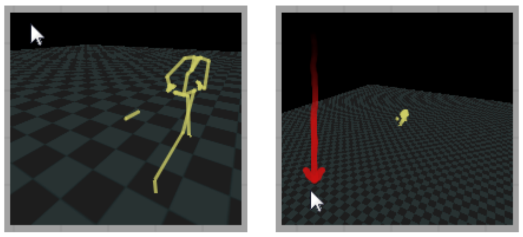


(b) Any viewing window can be dragged around by its central region.

Figure 4.3: Controlling the viewing window camera.



(a) Second and third mouse buttons or stylus modes are mapped to camera distance and position. Clicking the second button and dragging up moves the camera closer to the character. Dragging down moves the camera farther away. Clicking the third button and dragging left or right orbits the camera about the character in the X-Z plane.



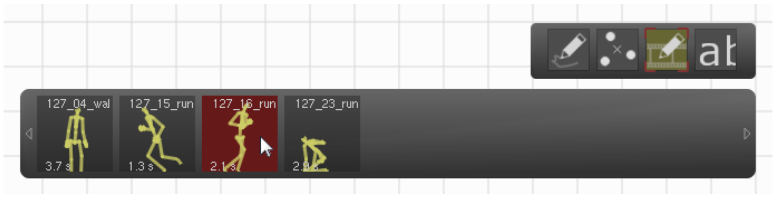
(b) Clicking and dragging the centre moves the viewpoint closer to or farther away from the animated character.

4.1.2 Drawing a Motion Curve

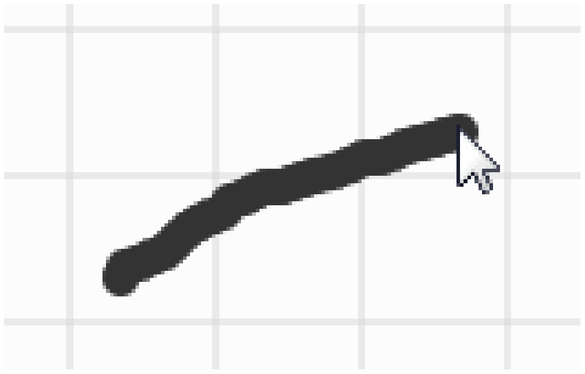
A motion curve is a visual representation of a motion on the canvas. A motion curve drawing tool is provided to allow the user to draw curves onto a canvas. A motion palette allows the user to choose which motion to bind to new curves that are drawn. The curves can be used for motion playback, segmentation, or splicing and blending. Motion curves are also a source of character poses used for other canvas features (See Section 4.1.5).

Interaction with motion curves is shown in figures 4.4 through 4.6.

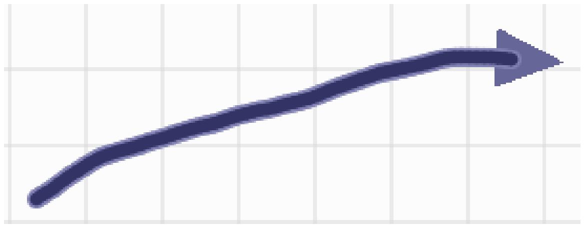
Figure 4.4: How to draw motion curves.



(a) The user selects a motion from the motion palette. The user can right-click the motion to play it in the viewing windows. The motion curve drawing tool is automatically selected from the tool palette.

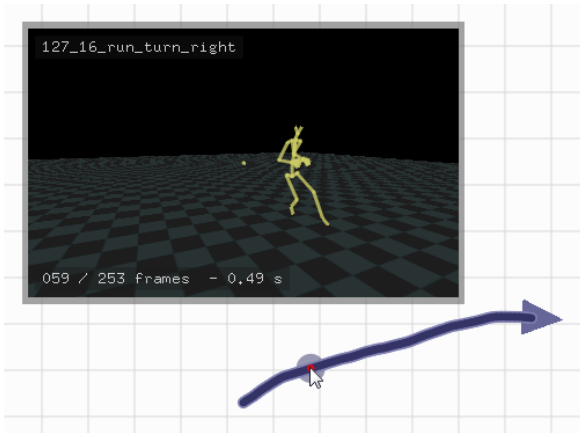


(b) The user begins sketching out a freehand curve on the canvas of any desired shape.

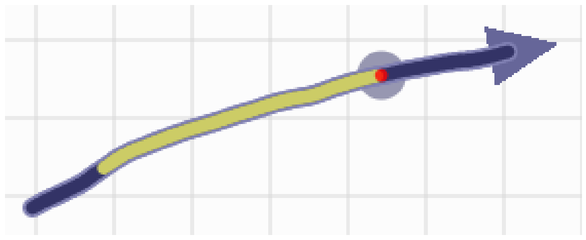


(c) The user completes the curve, which then becomes an interactive control on the canvas.

Figure 4.5: Selecting parts of a motion curve.

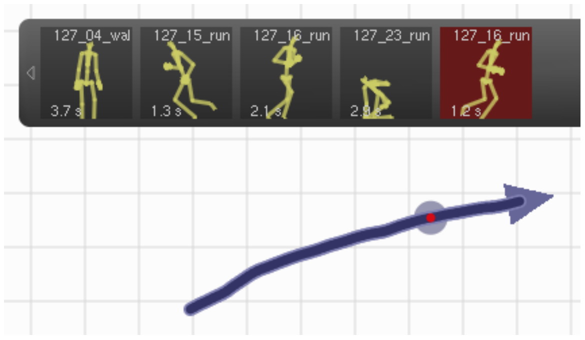


(a) The user can hover over a part of the curve to show the animation at that frame.

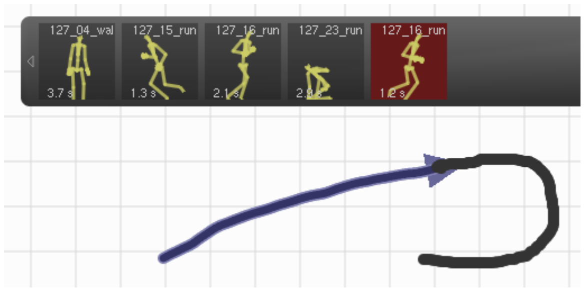


(b) The user can select a range of frames from the animation.

Figure 4.6: Using motion curve selections as new motions.



(a) By tapping or clicking on a selection, the user can add the current animation selection to the motion palette.

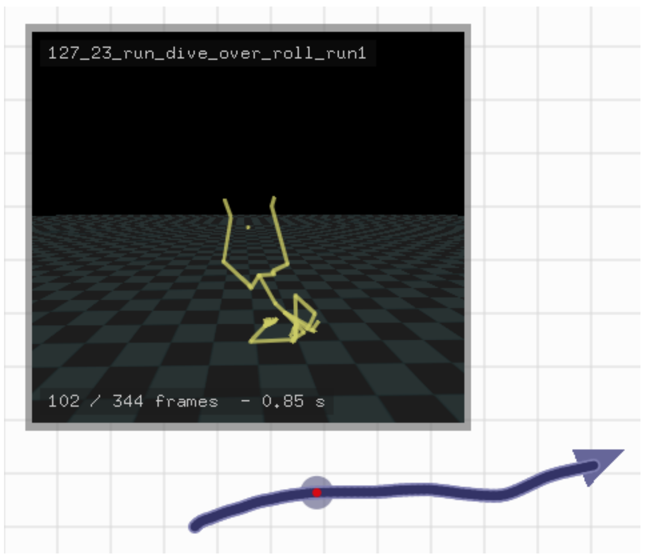


(b) The user can select the new animation from the palette and use it in subsequent sketches. In this manner, the user can cut sub-components of long animations out and use them to build other animations. For example, a single jump could be extracted from a long animation and appended to the end of a walking animation.

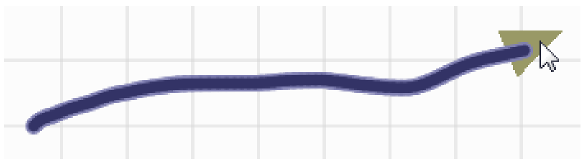
4.1.3 Combining Motion Curves

The user can combine two animations by dragging across two connected motion curves, as shown below in figures 4.7 through 4.8.

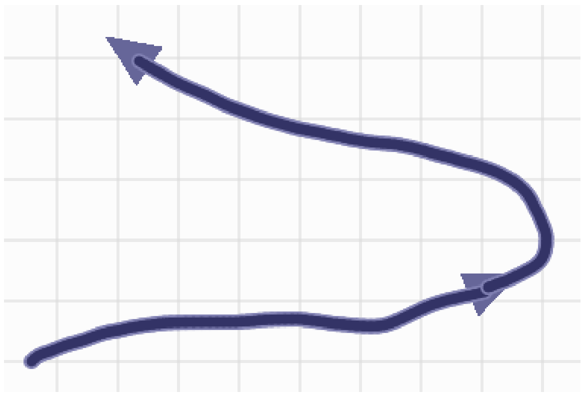
Figure 4.7: Drawing connected motion curves.



(a) A motion curve is drawn on the canvas.

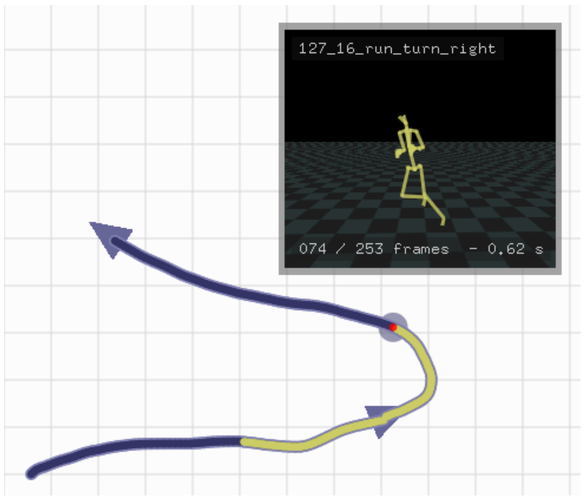


(b) When the user hovers the cursor over the end arrow of the curve it turns yellow, indicating that any new motion curve started at that point will be joined to the first.

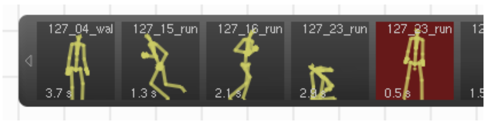


(c) The user clicks and drags to draw a new curve corresponding to a second motion. The second curve is started exactly at the end point of the first curve. The two curves are considered connected and form a small motion graph.

Figure 4.8: Generating combined motions by selecting across connected motion curves.



(a) Because the curves are connected it is possible to select across them as if they were a single motion curve. The user clicks on a point along the first curve and drags to a point on the second curve.

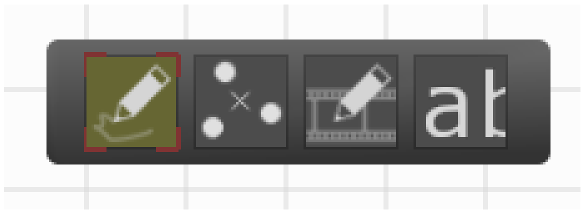


(b) A new motion is created by concatenating the two existing motions and blending through the transition, as described in Section 3.3.1. The new motion is added to the palette of motions and can be used in future operations.

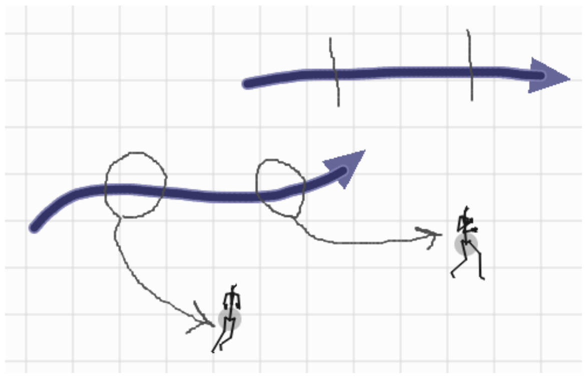
4.1.4 Annotation and Panning

Users can type text or sketch directly onto the canvas surface. It is also possible to pan the canvas view, so a large canvas area can be used even if it does not fit on the screen. Details are shown below in figures 4.9 through 4.10.

Figure 4.9: Drawing freehand annotations.

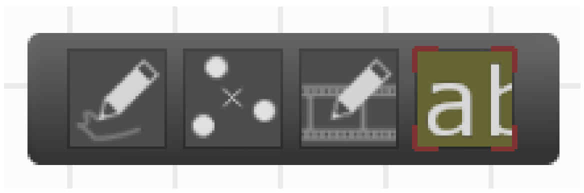


(a) The user can select a freehand annotation tool to draw directly on the canvas.

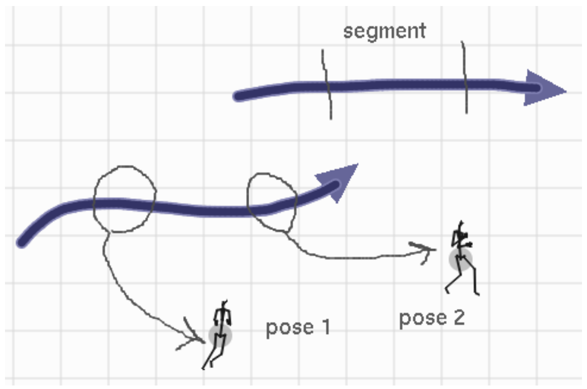


(b) When the freehand annotation tool is selected, the user can draw on any blank canvas surface. If freehand annotations cross canvas elements like motion curves they are drawn underneath. Annotations do not change the way the user interacts with canvas elements.

Figure 4.10: Annotating with text.



(a) The user can also select a text annotation tool to type out text onto the canvas.

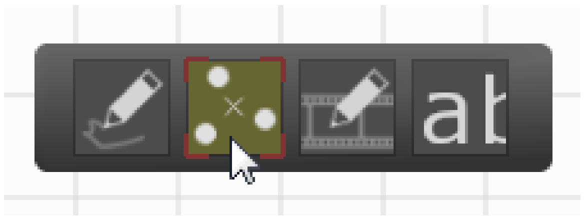


(b) Typing text is an alternative to freehand writing with the annotation tool. The user can click once using the text annotation tool to bring up a cursor on the canvas surface. The cursor moves if the user clicks somewhere else and disappears if the user hits enter or changes tools.

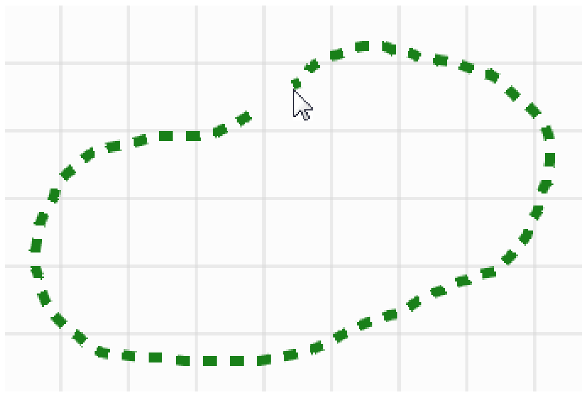
4.1.5 Interpolation with Spatial Keyframing

Spatial keyframing interpolation regions make it possible to generate new character poses and animations. The user sets up a region by drawing its bounds and then dropping in poses to be used in the blending algorithm, discussed in Section 3.5. Interaction with spatial keyframing regions is shown below in Figure 4.11.

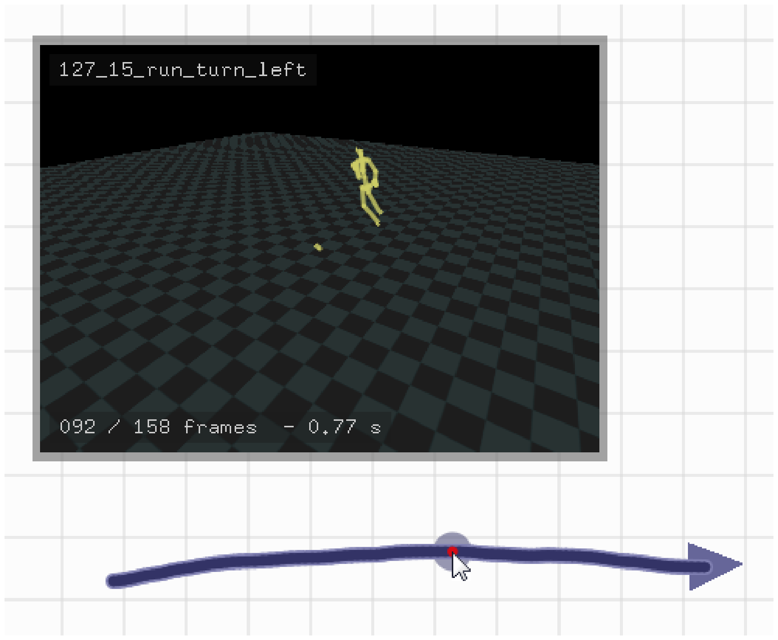
Figure 4.11: Drawing a motion interpolation region.



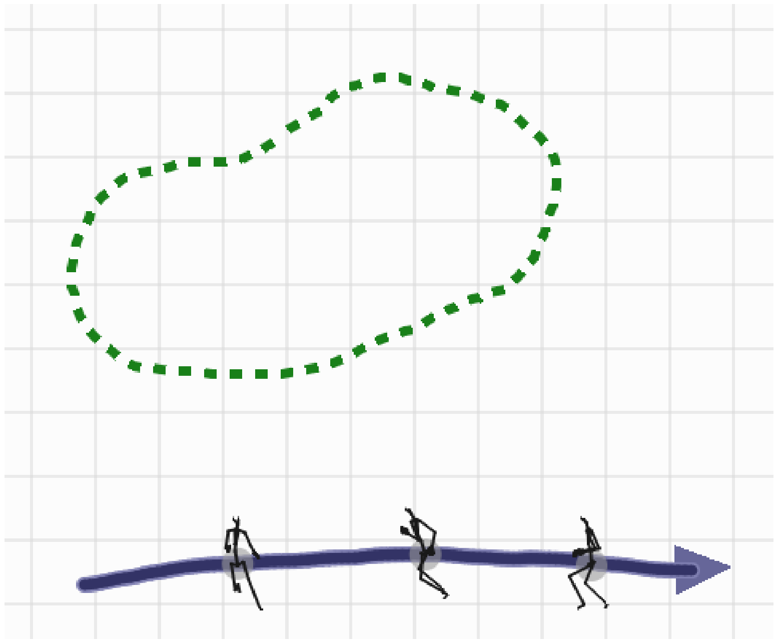
(a) Interpolation regions can be created by using the region tool from the tool palette.



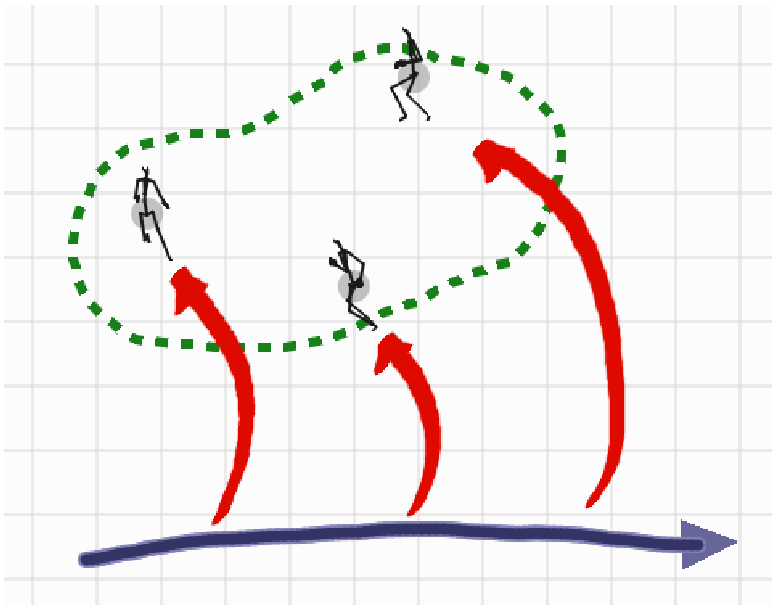
(b) The user constructs a spatial keyframing region by drawing a freehand shape. A dotted line appears as the user drags the cursor. The shape is closed when dragging stops. From then on the enclosed region behaves like an interpolation region rather than a normal part of the canvas.

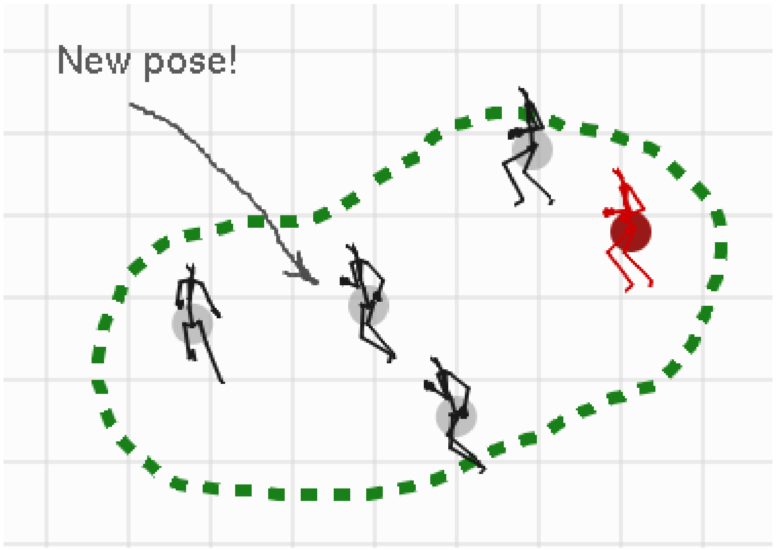


(c) A motion curve is drawn. The user can view poses from the motion by hovering the cursor over the curve.

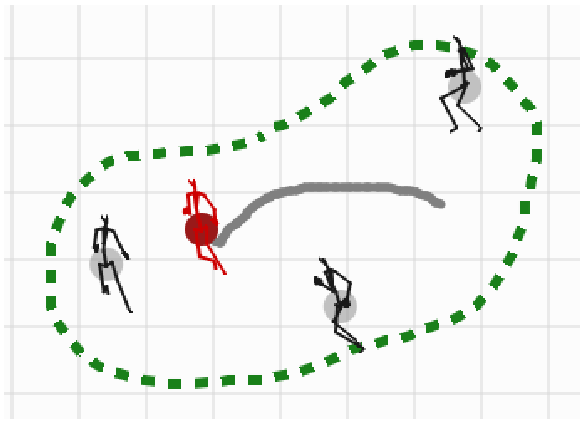


(d) Interesting poses can be spawned off of the motion curve by double clicking. These spawned poses are independent canvas elements that can be repositioned anywhere on the canvas, inside or outside of an interpolation region.

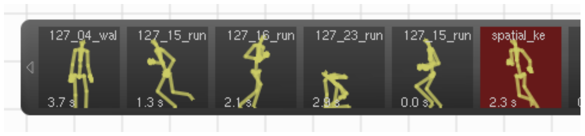

(e) The poses are dragged onto the spatial keyframing interpolation region, which can now be used to create new motions and poses.



(f) New poses can be created inside the interpolation region. The red pose is a temporary pose that is displayed after a single click. If the user clicks twice a new pose element is created. The “New pose!” shown here is a generated pose and it could be moved anywhere on the canvas. The annotation showed here was created with the canvas annotation tools.



(g) After the red highlighted pose is shown by single clicking it can be dragged. As the highlighted pose is dragged it changes to reflect the current interpolation position. It also leaves behind a temporary motion curve representing a motion that is constructed by sampling interpolated poses.



(h) The user can click on the highlighted pose or the temporary motion curve to add the newly-generated motion to the palette of available motions.

4.1.6 Construction of Motion Graphs for Motion Topologies

Many motion curves can be combined to form a larger motion graph (see Figure 4.12). The motion graph is a visual representation of the different motions and transitions available for a character. The user can define new motions across an arbitrary number of curves by clicking and dragging, as with the two-motion example in Section 4.1.3. Poses can be used to show important aspects of the animations in the motion graph. Text and freehand annotations can be used to indicate parts of the graph or label multiple graphs on the canvas.

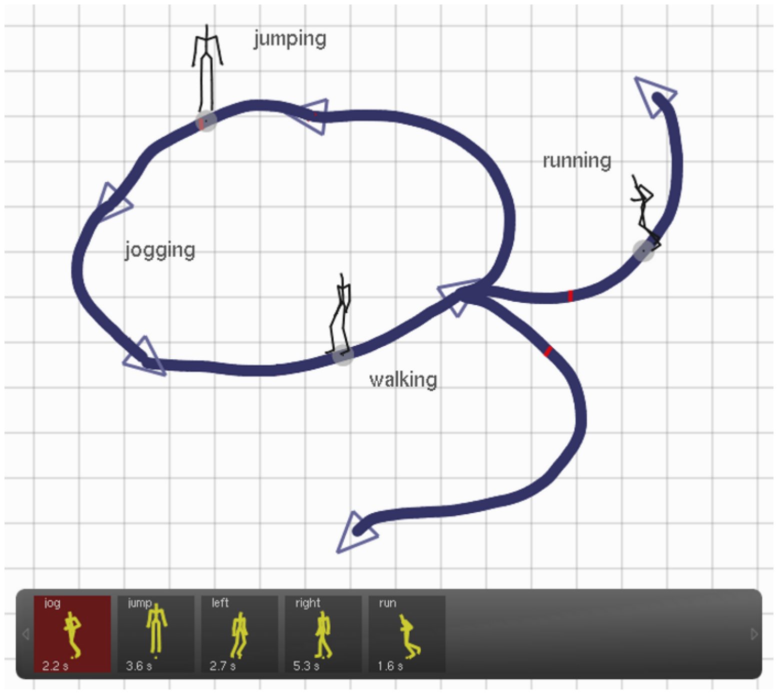


Figure 4.12: A motion graph sketched onto the canvas.

4.2 Discussion

The following section discusses the Animation Canvas system as it relates to the given example scenarios. Design tradeoffs of different components of the system are also presented.

4.2.1 A Language for Motion Editing

The Animation Canvas presents a language for motion editing. Some demonstrated strengths of the system are as follows:

Coherent visual language for consistent interaction

The canvas presents a language built around the concept of manipulable visualizations. Another key component is the consistent hierarchy of character poses as points, motions as curves, and higher level abstractions like motion connectivity or spatial keyframing regions used to create new motions. The suitability of these interaction metaphors to motion editing are demonstrated by the fact that many different motion editing techniques can naturally be translated so that they are consistent with the overall interaction style of the canvas. For example, motion curves naturally become explicit motion graph visualizations. Two-dimensional spatial keyframing regions can naturally be embedded into a two-dimensional canvas and poses can be positioned in these regions as on the rest of the canvas. Dropping poses onto motion curves also corresponds intuitively to standard keyframing techniques.

The design of interactive components of the canvas is also well-suited to sketch-style interaction. Motion curves are analogs to drawn pen strokes while motions themselves are analogous to brushes.

Simple interaction techniques for fast motion editing

A variety of interaction examples are presented in Section 4.1. All of these examples can be performed during a time frame of minutes, so the user can quickly discard undesirable results and try again. An undo feature would also be built into a more complete version of the system to enable users to quickly perform operations without worrying about errors. The system is not suited to fine-grained motion editing, but motions synthesized using the canvas could be exported to other editors.

Explicit visualization of motion data

The canvas provides explicit manipulable visualizations of all working motion data. Poses are drawn directly onto the canvas and the full length of any motion clip being used is available as an interactive motion curve. Playback features are available at all times to review motions, motion graphs, and spatial keyframing regions.

Support for multiple editing techniques

The canvas can be used to cut and resequence motions, two operations of established usefulness. Keyframing techniques have also been used extensively and are supported by the canvas.

4.2.2 Motion Curves

The design of motion curves involved several interesting decisions and trade-offs. The first was the question of using colour encodings or symbols near the curve on the canvas. For example, suitable blending points between motions could be indicated on the curves to facilitate the construction of motions graphs.

Another question was how to handle the significance of length of the curve on the canvas. Initially, length of motion curves was to be proportional to the length of the motion clip represented by the curve. This approach had the serious drawback of requiring a method to generate a fixed size of curve from variable-length paths drawn by the user. An early version of the canvas truncated or extended paths to generate correctly-sized motion curves. It also provided feedback to the user, showing a moving character while the path was drawn as well as the percentage of the clip corresponding to the current length of the path drawn. Despite the feedback, it was still very difficult to draw a curve terminating at the endpoint of another curve, which is one way of drawing a motion graph. The fixed ratio of curve length of animation length was removed; curve connectedness for defining motion graphs is a much more important and meaningful feature of the canvas.

4.2.3 Spatial Keyframing

Spatial keyframing fits naturally into the Animation Canvas interaction paradigm in a variety of ways. First, the spatial embedding of poses and motions works well with the open spatial organization of the canvas environment. Next, the core interaction metaphors present in the canvas are easily translated for use with spatial keyframing regions; poses become points in the region and motions can be extracted from the region by drawing curves, just like everywhere else in the canvas environment.

One extension considered for spatial keyframing was motions as points within the region. This was rejected for a variety of reasons. First, it is not consistent with the motion-as-curve paradigm on the rest of the canvas. Next, ambiguous timing is difficult to resolve; if the motion is represented as a point, how is motion timing set and how does the user know which part of the motion is currently being blended to create an output animation? No satisfactory solutions were found to these questions.

4.2.4 Playback

The canvas supports two modes of playback; direct playback and stateful playback. Stateful playback is particularly interesting because it shows how the canvas can be used as a character controller. Because the user creates interactive interface elements on the canvas, it is possible to create manipulable custom controllers that drive characters in interesting ways. It is easy to imagine running a game engine with the canvas and using it to interactively explore different possibilities for character control in the game. The canvas would be used as a visual programming language to change motions without the need for conventional programming or changes to scripts.

Rather than showing the game character in an empty world with a ground plane it would be possible to embed the character in a rich game world with objects and collision detection. Furthermore, it would be possible to explore the use of a physics simulation rather than purely kinematic animation. In such a simulation motion playback would signal desired joint angles which would be limited by the character's joint torque. In this world the character could manipulate objects and would respond to them appropriately.

4.2.5 Repositioning and Deleting Elements

The current prototype version of the canvas does not support the removal of elements after they are drawn. An obvious way to remove this limitation would be to add a new "eraser" tool to remove elements that fall within a certain radius. This would work well for annotation but might not be ideal for other elements like motion graphs or interpolation regions. Another problem is that an eraser tool would normally only work for the canvas. What about motions in the motion palette?

One solution attempted for motion curves was to allow repositioning after they were created. If a dragging feature were implemented then it could be tied to a "trash can" region of the canvas where curves could be eliminated. Motion curves were roughly modeled as inelastic strings subject to friction on the canvas surface that were constrained to connect to the dragging cursor. In order to avoid unintentionally breaking motion graphs the curves also had to be constrained to connect with any adjacent curves. This implied that many motions in practice would be highly constrained and could not be repositioned. This feature would have required more work to be useful and was abandoned because of its complexity.

A simple solution for the motion palette would be to support drag and drop for motion icons. This would make it possible to change the positions of the different motions on the palette and a palette-specific trash can could be added. One advantage of this approach is that there is no current motion palette feature that uses any form of dragging on the palette itself.

Chapter 5

Conclusions

The creation and refinement of character animation are challenging tasks and have been the subject of significant research inquiry. Motions must often appear plausible, recreating the effects of physical forces on complex moving bodies in the real world. It is difficult and time-consuming for an artist to create a three-dimensional animation for a highly articulated character with many degrees of freedom. Motion capture techniques are commonly used to capture real-world animations, but capture is limited to what can be performed by an actor and what can be anticipated at capture time.

The Animation Canvas gives users control over character animations without either forcing them to resort to a tedious amount of manual intervention or limiting their creativity. To this end, the canvas encourages users to work with a more developed motion vocabulary consisting of poses, motion clips, and motion topologies. By working with more abstract representations of motion, a greater breadth of information can be visualized on a single two-dimensional canvas.

The overall structure of the visualizations and metaphors built into the canvas define a visual language for working with motions. A hierarchy of poses, motions, and motion topology is introduced through the system. The canvas brings together a set of manipulable controls based on this hierarchy to allow users to perform useful operations and create new motions.

The Animation Canvas provides the following interface elements and functionality:

- **Introduction of a visual language for character motions and consistent metaphors for interaction.** Users draw motions directly onto the canvas and work with motion curves instead of lower-level motion data. Operations like concatenation of motion clips or cutting into subsegments fit naturally within the metaphor of motions as a curve or pen stroke. Poses naturally correspond to points on the canvas which can be extracted or inserted into motion curves. Motion curves and poses can be used in higher level constructs to perform operations like motion blending.
- **Pose, motion, and motion topology representation on a single canvas.** The animation canvas allows users to simultaneously view motions, poses extracted or used to build those motions, and the relationships between motions arranged into a graph topology.
- **Flexible spatial organization of work area.** Because the system is not built around a timeline and because all motion elements have visual representations that can be drawn onto the work area, users have more flexibility in how they arrange data. A panning feature is provided, much like in a drawing program; many intuitions developed from using tools like Photoshop or Illustrator also apply to the canvas.

- **Visual representation of motion topologies.** Motion curves drawn onto the canvas can be linked together and the topology formed is meaningful. By constructing a visual representation of a graph structure, users construct a real motion graph; users can generate walks through the graph to create new motions. Exploring motion connectivity is also a useful way to quickly discover which motions can be easily combined and which cannot.

- **Interactive motion blending spaces.** The canvas allows users to demarcate regions where the mode of operation is interpolation rather than direct drawing of motions. Poses or motions are arranged spatially and interpolation takes place in real time based on the user's tap or drag actions. A curve in a motion blending space becomes a motion that can be used to draw in the normal canvas space.

- **Online motion blending and character control.** Motions can be used within blending regions to generate new related motions or to control a character in real time.

- **Free-floating linked views.** The primarily two-dimensional canvas provides necessary views into the world of the three dimensional world of animated characters. Users are free to move these views about on the canvas and independently orient camera views in order to unambiguously view the animated character space.

5.1 Future Work

This section presents a sketch of an extension for specifying character behaviours within the framework of the canvas. A list of smaller improvements is also presented, followed by a plan for formally evaluating the level of usability of the system.

5.1.1 Specifying Character Behaviour

The Animation Canvas features simple abstractions like motion curves and more complicated abstractions like motion graphs and spatial keyframing. These abstractions allow the user to manually drive an animated character by playing motions in sequence. Another layer of abstraction could be added to allow the user to specify character behaviours to drive a character automatically. Character behaviours could be determined by controllers built by the user. More complicated worlds could be rendered in viewing windows with multiple characters that interact based on their behaviour controllers. In addition to being a tool for creating animations, the canvas would become a tool for creating characters and simulations.

Extended motion graphs with behavioural annotation would be one way to determine behaviour animation starting points and transitions between animations. The behaviour controllers would walk through their respective motion graphs, choosing transitions at forks in the motion graph with probability distributions or more sophisticated rules. Each character behaviour could be mapped to one or more persistent characters rendered in a viewing window scene.

5.1.2 Other Improvements

Other potentially useful and interesting additions include:

- **Additional blending schemes and interactive blending functions.** Animation blending schemes have been studied as a research topic [42] and there are many different schemes that could be added to the system. Another interesting extension might be to provide users with a way to interactively create their own blending functions when two motion clips are joined by directly sketching out a function on a two-dimensional graph.

- **Exploring higher dimensional spaces with spatial keyframing.** Possible character poses represent a high-dimensional space. Time may be thought of as another dimension. The motion blending techniques presented as part of the animation canvas interpolate between existing poses and so only capture a subspace within the space of overall poses. It would be interesting to explore visualizations of the full pose space and interpolation by presenting the user with different projections, linked views, and smooth transitions.

- **Tap retiming.** The timing of strokes used to draw motion curves is not currently used by the canvas but could be used to time motions. Alternatively, key points in a motion could be timed using tap events on the canvas, an approach explored by Terra and Metoyer [38].

- **Locomotion from spatial keyframing.** A simple extension to spatial keyframing is to consider character footfalls and other metrics and to allow for character locomotion.

- **Support for non-trivial environments.** The system could be extended to support complicated environments with obstacles. Support could be added for collision detection. With these improvements it would be possible to evaluate a wider range of motion applications. It is easy to imagine, for example, simulating a game environment and using the Animation Canvas to create new motions and quickly preview how the motions can be used to drive a character in the game world. Techniques like those used by Reitsma and Pollard [33] could be used in conjunction with the canvas interface to interactively evaluate the suitability of a motion set to a given environment.

- **Animations for part of a character.** Currently all motions are considered full body motions, but motions could be limited to a subset of the full character. For example, the lower body component of a running animation could be blended with a second animation.

- **Tying viewports in with different animations.** All viewports in the canvas are associated with the motion currently being played back or interacted with, but there could be a way for users to associate viewports with motion curves or sets of motion curves on the canvas. It would also be possible to draw multiple motions in a viewport at one time.

- **Classic keyframing from poses dropped onto motion curves.** Pose extraction from motion curves is supported, but inserting poses to allow for keyframing was not implemented. This would be a natural fit with the existing motion curve and pose metaphors.

5.1.3 Prototyping and Evaluation

The Animation Canvas was conceived as a sketch-based system for viewing, editing, and synthesizing new animations. The first aspects of the system to be designed were the motion-as-curve metaphor, pose-as-point metaphor, and spatial keyframing features. A prototype of the system was created to demonstrate that these design characteristics can come together to form a coherent interface well-adapted to the tasks included in Section 4.1. The demo video, screenshots, and analysis demonstrate that the system can be used to complete these tasks with relatively simple operations.

Formal evaluation and user testing were avoided prior to implementation of the prototype due to concerns about biasing the interface design toward interaction styles that are easy to evaluate or already in use. The early design of the system changed frequently and lacked the level of specification necessary to formulate well-defined, practical questions to study. It would have been challenging to come up with narrow design tradeoffs to evaluate during the early design cycle, and any evaluation would have taken up time and effort that could have instead been devoted to research and development. Greenberg and Buxton agree that evaluation can be harmful in some cases if applied too early in the design cycle of a new system [15]. Evaluation remains a part of future work and will require careful consideration if it is to provide meaningful information.

Liebermann [27] and Zhai [47] provide interesting and opposed views on how evaluation should be handled for a system like the Animation Canvas. Because of the complexity of the system, its novelty, and the lack of polish of the prototype relative to existing software, it seems unlikely that general questions about the user interface would provide good information. Narrower questions about specific design tradeoffs may be more likely to produce answers that can be applied to design decisions.

Some questions that might be good subjects for formal evaluation in the future are:

- Is the user interface self-revealing? How long does it take new users to understand the interface metaphors? Is there a difference between novice users and expert users who are familiar with other animation tools? One reason for using a sketch editor surface for the Animation Canvas was to borrow from ubiquitous knowledge of an existing well-used and well-understood user interface metaphor.

- How do users interact with different parts of the canvas? How do they use panning? Is zooming useful? It would be useful to evaluate the way the canvas surface is used and how successful the motion palette-based workflow is in terms of facilitating iterative changes to motions.

- Should it be possible to reposition motion curves on the canvas? If so, how should motion curves be moved on the canvas?

- Is a drag and drop interface for motions desirable?

- Is there valuable information that can be added to motion curves with colour encodings? Motion curve segment colour encodings are built into the canvas prototype but were not used because no metric worth the added visual clutter was developed. One frame-by-frame metric tested was the change in character joint angles. This did not highlight key parts of the motion as it was intended to, and was abandoned. Developing useful motion metrics that can be mapped onto motion curve segment colour would require more consideration.

- Is it better to guarantee that motion curve length is proportional to time, or is it better to let users choose an arbitrary length for each motion curve? The current system allows the user to choose an arbitrary length so that it is trivial to join any two points on the canvas with a single stroke.

These are all useful and interesting questions and answers to any of them could be directly applied to the design of the Animation Canvas.

Bibliography

[1] Apple Inc. Quartz Composer. `http://www.apple.com/`, 2011.

[2] O. Arikan and D. Forsyth. Interactive Motion Generation from Examples. *ACM Trans. Graphics*, 21(3):483–490, 2002.

[3] J. Assa, Y. Caspi, and D. Cohen-Or. Action Synopsis: Pose Selection and Illustration. In *ACM SIGGRAPH*, pages 667–676. ACM, 2005.

[4] Autodesk, Inc. 3DS Max. `http://www.3dsmax.com/`, 2011.

[5] Autodesk, Inc. Maya. `http://www.maya.com/`, 2011.

[6] P. Beaudoin, S. Coros, M. van de Panne, and P. Poulin. Motion-Motif Graphs. *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pages 117–126, 2008.

[7] B.B. Bederson, J.D. Hollan, K. Perlin, J. Meyer, D. Bacon, and G. Furnas. Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics. *Journal Visual Languages and Computing*, 7(1):3–32, 1996.

[8] J. Bertin. La graphique. *Communications*, 15(1):169–185, 1970. Published by Persée. Available at `http://www.persee.fr/web/revues/home/prescript/article/comm_0588-8018_1970_num_15_1_1221`.

[9] J. Chai and J. K. Hodgins. Performance animation from low-dimensional control signals. In *ACM SIGGRAPH*, pages 686–696, 2005.

[10] B. Choi, M. You, and J. Noh. Extended Spatial Keyframing for Complex Character Animation. *Computer Animation Virtual Worlds*, 19(3-4):175–188, 2008.

[11] R.C. Davis, B. Colwell, and J.A. Landay. K-Sketch: a 'Kinetic' Sketch Pad for Novice Animators. *Proc. SIGCHI Conf. Human Factors Computing Systems*, pages 413–422, 2008.

[12] W. Du Toit. Radial Basis Function Interpolation. University of Stellenbosch, 2008.

[13] M. Gleicher. Retargetting Motion to New Characters. In *Proc. Conf. Computer Graphics and Interactive Techniques*, pages 33–42. ACM, 1998.

[14] Google Inc. Google Maps. `http://maps.google.com/`.

[15] S. Greenberg and B. Buxton. Usability Evaluation Considered Harmful (Some of the Time). In *Proc. SIGCHI Conf. Human Factors Computing Systems*, pages 111–120, 2008.

[16] R. Heck and M. Gleicher. Parametric Motion Graphs. In *Proc. Symp. Interactive 3D Graphics and Games*, pages 129–136, 2007.

[17] C. Hecker, B. Raabe, R.W. Enslow, J. DeWeese, J. Maynard, and K. van Prooijen. Real-time motion retargeting to highly varied user-created morphologies. *ACM Trans. Graphics*, 27(3):1–11, 2008.

[18] I. Herman, G. Melançon, and M.S. Marshall. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Trans. Visualization and Computer Graphics*, 6(1):24–43, 2000.

[19] J.K. Hodgins, J.F. O'Brien, and J. Tumblin. Perception of Human Motion with Different Geometric Models. *IEEE Trans. Visualization and Computer Graphics*, 4(4):307–316, 1998.

[20] Y. Hu, S. Wu, S. Xia, J. Fu, and W. Chen. Motion Track: Visualizing Variations of Human Motion Data. In *IEEE Pacific Symp. Visualization(PacificVis)*, pages 153–160. IEEE, 2010.

[21] T. Igarashi, T. Moscovich, and J. F. Hughes. Spatial Keyframing for Performance-Driven Animation. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pages 107–115, 2005.

[22] L. Kovar and M. Gleicher. Flexible Automatic Motion Blending with Registration Curves. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pages 214–224, 2003.

[23] L. Kovar, M. Gleicher, and F. Pighin. Motion Graphs. In *SIGGRAPH: Proc. Conf. Computer Graphics and Interactive Techniques*, pages 473–482, 2002.

[24] J. Lasseter. Principles of Traditional Animation Applied to 3D Computer Animation. In *SIGGRAPH: Proc. Conf. Computer Graphics and Interactive Techniques*, pages 35–44, 1987.

[25] J. Laszlo, M. van de Panne, and E. Fiume. Interactive Control for Physically-based Animation. In *ACM Proc. Conf. Computer Graphics and Interactive Techniques*, pages 201–208, 2000.

[26] J. J. LaViola, Jr. and R. C. Zeleznik. MathPad2: a System for the Creation and Exploration of Mathematical Sketches. *ACM Trans. Graphics*, 23(3):432–440, 2004.

[27] H. Liebermann. The Tyranny of Evaluation. `http://web.media.mit.edu/~lieber/Misc/Tyranny-Evaluation.html`.

[28] T.B. Moeslund and E. Granum. A survey of Computer Vision-based Human Motion Capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[29] National Instruments Corporation. Labview. `http://www.ni.com/labview/`, 2011.

[30] NaturalMotion. Havok. `http://www.naturalmotion.com/`, 2011.

[31] NaturalMotion. Morpheme. `http://www.naturalmotion.com/`, 2011.

[32] K. Perlin and D. Fox. Pad: an Alternative Approach to the Computer Interface. In *Proc. Conf. Computer Graphics and Interactive Techniques*, pages 57–64. ACM, 1993.

[33] P. S. A. Reitsma and N. S. Pollard. Evaluating Motion Graphs for Character Navigation. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pages 89–98, 2004.

[34] P.S.A. Reitsma and N.S. Pollard. Perceptual Metrics for Character Animation: Sensitivity to Errors in Ballistic Motion. *ACM Trans. Graphics (TOG)*, 22(3):537–542, 2003.

[35] E. Saund and T. P. Moran. A Perceptually-Supported Sketch Editor. In *Proc. ACM Symp. User Interface Software and Technology*, pages 175–184, 1994.

[36] M. Sung, L. Kovar, and M. Gleicher. Fast and Accurate Goal-Directed Motion Synthesis for Crowds. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pages 291–300, 2005.

[37] I.E. Sutherland. Sketchpad: a Man-Machine Graphical Communication System. *Simulation*, 2(5), 1964.

[38] S. C. L. Terra and R. A. Metoyer. Performance Timing for Keyframe Animation. In *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, pages 253–258, 2004.

[39] M. Thorne, D. Burke, and M. van de Panne. Motion Doodles: an Interface for Sketching Character Motion. *ACM Trans. Graph.*, 23(3):424–431, 2004.

[40] E.R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, CT, 1983.

[41] D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated Mesh Animation from Multi-view Silhouettes. In *ACM SIGGRAPH*, pages 1–9, 2008.

[42] J. Wang and B. Bodenheimer. Synthesis and Evaluation of Linear Motion Transitions. *ACM Trans. Graphics*, 27(1):1–15, 2008.

[43] C. Ware. *Information Visualization, Second Edition: Perception for Design (Interactive Technologies).* Morgan Kaufmann, 2004.

[44] K.N. Whitley. Visual Programming Languages and the Empirical Evidence For and Against. *Journal Visual Languages and Computing*, 8(1):109–142, 1997.

[45] A. Witkin and Z. Popovic. Motion Warping. In *Proc. Conf. Computer Graphics and Interactive Techniques*, pages 105–108, 1995.

[46] H. Yasuda, R. Kaihara, S. Saito, and M. Nakajima. Motion Belts. In *ACM SIGGRAPH*, pages 5–9, 2007.

[47] S. Zhai. Evaluation is the worst form of HCI research except all those other forms that have been tried. `http://www.almaden.ibm.com/u/zhai/papers/EvaluationDemocracy.htm`.

[48] J. Zhang and D.A. Norman. A Representational Analysis of Numeration Systems. *Cognition*, 57(3):271–295, 1995.

[49] P. Zhao and M. van de Panne. User Interfaces for Interactive Control of Physics-based 3D Characters. In *Proc. Symp. Interactive 3D Graphics and Games*, pages 87–94, 2005.

Appendix A

Rendering Motions

The motion palette makes a number of motions available to the user. Each of these motions is associated with a block of motion data. The motion data consists of a series of frames that describe the state of the channels, or parameters, of the motion over time. These parameters specify the configuration of the skeleton of the character shown in the motion. By posing the skeleton over time, it is possible to reconstruct and render a motion. By modifying the motion data it is possible to create new motions using the same character.

Initial motion data for the Animation Canvas comes from Biovision Hierarchy (BVH) files created using motion capture techniques. The first step in using these files is to parse the skeletal information that describes the topology of the character. The skeleton is described recursively in the BVH file as follows:

```
HIERARCHY
ROOT Hips {
  OFFSET 48.0276 81.5603 -18.4323
  CHANNELS 6 Xposition Yposition Zposition Zrotation
```

```
              Xrotation Yrotation

    JOINT Chest {
      OFFSET 1.23234e-007 14.8421 1.27319e-009
      CHANNELS 6 Xposition Yposition Zposition Zrotation
                Xrotation Yrotation

      JOINT RightCollar { ... }

      JOINT LeftCollar { ... }
    }
    JOINT LeftUpLeg { ... }
    JOINT RightUpLeg { ... }
  }
```

From the nested curly braces and the names of the joints in the character we can determine its structure. The root of the character, the point defined in relation to the origin of the global coordinate system, is the skeleton's hips. The chest, left upper leg, and right upper leg are connected to the hips. The right and left collar joints are connected to the chest, and if the hierarchy were completed through the ellipses we would see the remainder of the skeleton's arm and hand joints.

The hierarchical definition of the skeleton is closely related to desired functionality when posing the skeleton. The skeleton consists of a series of posable joints, and when manipulating a joint all of the child joints must be moved accordingly. For example, when rotating the upper arm joint we would like the lower arm and hand to remain connected and move appropriately. This is accomplished by composing matrix rotations and translations according to the skeletal hierarchy; a stack of matrix transformations is created that corresponds to the hierarchy of the model.

The rotations applied at the skeleton's joints are the motion data that specify the motion. The rotations are specified with joint angles sampled at regular intervals that are contained within the BVH files. Each joint has a series of permitted rotations described after the *CHANNEL* keyword. To render a motion, the system iterates over the provided joint angles and renders the skeleton at each frame. When working with motions, the system manipulates the joint angle data while the specification of the model's skeleton remains constant.

Appendix B

Supporting Interaction with Motion Curves

As with any other canvas item, events are handled by first culling mouse input not contained within the bounding box of the motion curve. A more complicated scheme is required to determine whether or not a point is contained within the motion curve and, if so, which frame of the curve's motion is best represented by the current position of the cursor.

Motion curves are rendered as a series of points connected by lines. The points are drawn with the same width as the lines in order to fill in gaps created by line segments intersecting at an angle. These points also create neater looking rounded edges for motion curves. The point and curve setup suggests one way to consider intersections with a motion curve; a point is on the motion curve if its distance from either a point or a line segment is less than the shared point and line segment width.

In order to test for collision with points on the motion curve, it is sufficient simply to compare the radius of the point with the input event's distance from the point. For line segments, it is necessary to project the input point down onto the line segment. The algorithm to determine where an input point falls on the curve therefore consists of iterating over the line segments in the curve and seeing where the projected input point falls. The algorithm terminates when the projected point is within the line segment. Using this approach, it is also possible to construct the distance along the curve of the input point; this distance is the sum of the line segment lengths plus the partial length along the line segment containing the projected input point. By dividing the distance by the precalculated length of the curve we can obtain a fractional value indicating how far the projected input point is along the total length of the curve. Multiplying this quantity by the number of total frames in the motion associated with the curve yields the frame selected by the input point.

Appendix C

Demo Video

A video of the Animation Canvas prototype is included as an accompanying DVD. The video shows the user interface elements of the system and demonstrates how to sketch motion curves, build motion graphs, and use spatial keyframing regions.