Animation Palette: An Interface for Prototyping Dynamic Aerial Motions

by

Peng Zhao

B.E., Tsinghua University, 2000

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

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming to the required standard

The University of British Columbia

August 2004

© Peng Zhao, 2004

Abstract

This thesis presents an interface for the interactive design of dynamic aerial stunts, such as platform diving, freestyle aerial ski jumps and half-pipe snowboarding. This has applications for the prototyping of sports motions or for computer character animation in video games. The user selects from a library of actions, such as "stand", "pike" and "extend", in order to create a motion. Some actions define a set of target joint angles for proportional-derivative controllers, while others invoke balance controllers employing closed-loop feedback. A real-time physics based simulation determines the final motion. A motion can be created interactively in real-time, or edited off-line to allow for incremental refinement. Motion timings can be optimized automatically to produce desired outcomes. The interface can be used to explore questions such as "Is this new stunt possible?" or "What if the pike began earlier?" We demonstrate the interface using simulations of platform diving (29 unique dives), freestyle aerial ski jumps (13 unique jumps), and half-pipe snowboarding (4 unique jumps).

Contents

Abstract	ii
Contents	iii
List of Tables	vi
List of Figures	vii
Acknowledgements	ix
1 Introduction	1
1.1 Animation Methods	1
1.2 Motivation	2
1.3 An Example	4
1.4 Contribution	5
1.5 Thesis Organization	5
2 Related Work	7
2.1 Physics-based Animation	7
2.1.1 Constraint-based Approaches	8
2.1.2 Controller-based Approaches	8

	2.2	Finite	State Machine Controllers	9
	2.3	Interac	ctive Interfaces for Character Animation	10
	2.4	Summ	ary	11
3	\mathbf{Th}	e Anin	nation Palette Interface	12
	3.1	Interfa	ce and Workflow	12
	3.2	Action	Buttons	14
		3.2.1	Kinematic Actions	15
		3.2.2	Dynamic Actions	15
	3.3	Switch	ing Between Actions	16
		3.3.1	Kinematic to Kinematic Transitions	17
		3.3.2	Dynamic to Dynamic Transitions	17
		3.3.3	Kinematic to Dynamic Transitions	18
		3.3.4	Dynamic to Kinematic Transitions	19
	3.4	Summ	ary	19
4	Pro	ototypi	ng Dynamic Aerial Motions	20
	4.1	Motior	1 Stages and Stage Parameters	20
	4.2	Dynan	nic Controllers	24
		4.2.1	Proportional-derivative Controller	24
		4.2.2	Balance Controller	25
	4.3	Offline	Motion Refinement	28
		4.3.1	Manual Adjustment	28
		4.3.2	Automatic Optimization	30
	4.4	Summ	ary	32

5	5 Implementation and Results 3					
	5.1	Dynamics Simulation	33			
	5.2	Character Models	35			
	5.3	Platform Diving	36			
	5.4	Freestyle Aerial Ski Jumping	40			
	5.5	Half-pipe Snowboarding	45			
	5.6	User Experiences	46			
	5.7	Summary	47			
6	Co	nclusions and Future Work	50			
6	Co : 6.1	nclusions and Future Work Improved Dynamic Controller Actions	50 50			
6	Co: 6.1 6.2	Inclusions and Future Work Improved Dynamic Controller Actions Natural Motion	50 50 51			
6	Co 6.1 6.2 6.3	Inclusions and Future Work Improved Dynamic Controller Actions Natural Motion GamePad Interface	50505151			
6 Bi	Co 6.1 6.2 6.3 bliog	Inclusions and Future Work Improved Dynamic Controller Actions Natural Motion GamePad Interface Graphy	 50 51 51 52 			
6 Bi A _I	Co 6.1 6.2 6.3 bliog	Inclusions and Future Work Improved Dynamic Controller Actions Natural Motion GamePad Interface graphy dix A Dynamic Model Details	 50 51 51 52 56 			

List of Tables

5.1	Dimensions of kicker jumps.	41
A.1	Physical parameters of diver model.	56
A.2	Joint strength of diver model	57
A.3	Physical parameters of skier model	57
A.4	Joint strength of skier model	58
A.5	Physical parameters of snowboarder model	58
A.6	Joint strength of snowboarder model	59
B.1	Description of diving buttons	60
B.2	Description of buttons for aerial ski jump control	61
B.3	Description of buttons for snowboard control	61

List of Figures

1.1	Example of animation created online using our system	4
3.1	Interface of Animation Palette.	13
3.2	Control of kinematic motions.	16
3.3	A kinematic action to dynamic action transition	18
4.1	Simulated platform diving and ski jumping	21
4.2	5 stages for forward somersaults pike	22
4.3	Three examples of stage parameters	23
4.4	Sagittal plane balance control in standing	25
4.5	Lateral balance control in standing	26
4.6	Lateral balance control in crouching	27
4.7	Interface for offline design	29
4.8	Dive entry before and after optimization of the time for extension	31
4.9	Ski landing before and after optimization of the time for stretch. $\ .$.	31
5.1	Degrees of freedom used for diving, snowboarding and skiing models.	35
5.2	States of Platform Diving	36
5.3	Platform diving: forward $1\frac{1}{2}$ somersault pike with one twist	38
5.4	Platform diving: backward $1\frac{1}{2}$ somersault pike	38

5.5	Platform diving: reverse $1\frac{1}{2}$ somersault tuck.	39
5.6	Platform diving: bad entry for inward $1\frac{1}{2}$ somersault pike	39
5.7	States of Ski Jumping	40
5.8	Specification of ski-hill.	41
5.9	Dimensions of kicker jumps	41
5.10	Ski jump: double twisting double back flip	43
5.11	Ski jump: front pike single.	43
5.12	Ski jump: back layout-tuck-layout.	44
5.13	Ski jump: bad landing for a back-full-full	44
5.14	States of Half-pipe Snowboarding.	45
5.15	Section view of half-pipe.	46
5.16	Half-pipe snowboarding: back-grab.	48
5.17	Half-pipe snowboarding: front-grab.	48
5.18	Half-pipe snowboarding: back 360.	49
5.19	Half-pipe snowboarding: crash landing for back-grab	49

Acknowledgements

First, I would like to thank my supervisor, Michiel van de Panne, for leading me into the world of computer animation. His guidance and encouragement was invaluable. I would also like to thank Alla Sheffer and Jason Harrison for giving me useful comments to improve my thesis.

Thanks to other members in Imager lab, who I was pleasant to work with. In particular, I wanna thank Hamish Carr for his suggestions on my thesis project, thank Ken Alton for his help in my English, and thank Dave Burke for introducing me into the group of playing the Quake game.

I am grateful to my friends here, including Shuzhen Wang, Xiaojing Wu, Long Li, Pingdong Ai, Yushuang Liu, Dan Xiao, Lin Zhong, Qian Huang, Kangkang Yin, Chen Yang, Zhijin Wang and many others. They made my life in UBC a wonderful memory.

Finally, I'd like to thank my parents and my wife, Huanchun Ren, for their endless love and support.

PENG ZHAO

The University of British Columbia August 2004

Chapter 1

Introduction

Computer animation is a rapidly developing area in computer science, with applications such as feature films, digital simulators, computer games and virtual reality. Improved techniques are required to simplify and speed up the process of creating convincing animations. One key direction involves the use of physical simulation to generate physically realistic motions. This thesis presents an approach for the interactive design of dynamic aerial stunts. Before further discussing the motivation of our thesis, we will first provide a brief overview of methods for creating human figure animations.

1.1 Animation Methods

Broadly speaking, there are three widely used methods to make computer animations: keyframing, motion capture and dynamic simulation. In keyframe character animation [3], the animator defines the position of all parts of characters at specific points in time. These specified positions are known as keyframes. The computer then computes the in-between frames using an interpolation algorithm. Although keyframing is time consuming and requires significant skill, it remains the method of choice in many animated productions because of the complete control it offers the animator.

A straightforward approach for generating realistic motion is to use a motion capture system [18] to record the 3D motion of a human actor and then map this motion onto an animated character. The ability to blend and transition between motions is very important for motion capture based animation system in order to provide adequate flexibility to create the wide range of desired animated motions required by video games, for example. Another key technology is that of *retargeting* the pre-recorded motions to an animated character having different dimensions from the person used to capture the motion, which is a common occurrence. One disadvantage of this approach is that it is difficult to alter captured motions to realistically interact with environments different from the one in which they are obtained.

Dynamic simulation [26] is a potentially powerful tool for making physically realistic animations. It has been used to both analyze and animate many classes of motion, including diving, running, and gymnastic motions. A major challenge in creating physically-based animation is that of solving for the required control to achieve desired behaviours, especially for complex models such as humans and many animals. Typically, this necessitates a great deal of trial-and-error in the design of the controllers for any given motion.

1.2 Motivation

Presently, animators have used all three of the methods described above to produce high quality animations. For interactive applications such as video games, however, the animated characters need also respond to various environments, such as varying terrains and interaction with obstacles. In this case, the simulation approach has advantages over the keyframing and motion capture. Also, the speed of today's computers now enables us to simulate a fully dynamic-based character in real-time. Recent games incorporate "rag doll" simulations which involve no active control. We believe this is due to the difficulty of designing controllers and that of defining appropriate interfaces between the game player and the game character.

We present an interface which exploits the structured nature of acrobatic aerial movements to simplify the specification of physically-based prototypes of such motions. Specifically, we support platform diving, acrobatic ski jumping and halfpipe snowboarding based upon fully dynamic character simulations. Our system, called "Animation Palette", represents the required actions using a series of sequentially executed stages. For example, platform dives typically have the following sequence of actions: *stand*, *crouch*, *takeoff*, an aerial position such as a *tuck* or a *pike*, and *extension* prior to entry into the water. The complete diving motion can be created through the proper timing and execution of all these stages.

Our interface maps each possible action for any given stage to an on-screen two dimensional virtual button. The timing of the button press represents when the specified action begins, and the exact location of the press on the button is used to represent two additional parameters which gives greater control over the action to be executed. When the interface is used in an online fashion, the simulation runs in real-time to produce the animation. As a result, the user is able to control a 3D character simulator in a fashion that is analogous to steering a plane via a flight simulator.

The interface can also be used in an off-line fashion. For off-line design, we incorporate a timeline into the interface to represent the sequence of user inputs.



Figure 1.1: Example of animation created online using our system.

The user can refine the timing and location parameters of previous online inputs and then simulate to see whether the desired result is achieved. Also, given a metric to evaluate the quality of a motion, e.g., the water entry angle for a platform dive, our system can automatically optimize the timing of a specified input to achieve the best outcome.

1.3 An Example

Figure 1.1 shows an example of our interface for designing acrobatic aerial motions. The animator uses a mouse or stylus to choose what action the simulated character should do and how it should be performed by clicking at the appropriate position inside the virtual button. Our system then maps the timing and position parameters of these selections to controllers that compute the torque at each joint to drive the motion of the dynamic character. A physics-based simulation calculates the external forces such as gravity, and determines the final motion. Both the physical simulation and graphical rendering run in real time to provide the user with immediate feedback, displaying the animation as it is being created.

1.4 Contribution

The primary contribution of this thesis is to provide a method for the interactive control of 3D physically-based characters. Specifically, we present an interface for prototyping dynamic aerial motions such as platform diving. This interface presents the control parameters of the simulation in an understandable manner, which allows the user with little knowledge of the underlying control mechanisms to make various of dynamic aerial motions in a short time.

We develop a system that implements this interface technique and use it to design three types of stunts: platform diving, freestyle aerial ski jumps, and halfpipe snowboarding. This system demonstrates the feasibility of designing sports games based upon fully dynamic character simulations, as well as the possibility of creating useful tools for sports prototyping.

1.5 Thesis Organization

The remainder of this thesis is structured as follows. After reviewing previous related work in Chapter 2, we provide an overview of the structure of our interface in Chapter 3. Chapter 4 describes the components of our interface and the specific actions that our system currently supports. Chapter 5 demonstrates a variety of results obtained using our interface. Finally, Chapter 6 presents conclusions and future work.

Chapter 2

Related Work

The use of dynamic simulation to design animation is a topic of interest in the area of character animation. This chapter describes the work which is particularly relevant to ours. First, we provide an overview of methods for producing physics-based animation. We then discuss the work specific to controllers based on finite state machines. Finally, relevant work on interactive control for character animation is reviewed.

2.1 Physics-based Animation

Physical simulations can simplify the work required to generate realistic motions of passive objects, such as sliding blocks, bouncing balls, or clothing. However, for active objects such as humans and animals, we must also solve for the control mechanism to direct their movements. These must determine how the muscles act in order to produce desired motions. There are two different approaches to this problem, one is constraint-based and the other is controller-based.

2.1.1 Constraint-based Approaches

Constraint-based techniques [2, 4, 16, 35] attempt to generate the animation through an optimization of a specified objective function subject to the constraints. Typically, the motions are defined as a trajectory with a collection of constraints imposed at specific times, such as the starting and ending positions. The physics is also imposed as a constraint by adding the equations of motion as a constraint for each time step. The entire space-time trajectory is then optimized with respect to an objective, e.g., minimum control energy or minimum time, to achieve a unique "best" solution, thereby solving for a complete motion as a single optimization problem. The work of [8] presents an approach which can scale well to more complex characters using a set of objective functions and constraints that lead to linear time analytical first derivatives. More recently, this approach has been expanded upon in interactive techniques developed to control physical simulations of passive objects [23].

Space-time constaint techniques have several limitations. The solution derived by constraint-based techniques must be recomputed in new situations for which it was not calculated. Also, the available numerical methods often do not converge to acceptable answers because of spurious local minimum.

2.1.2 Controller-based Approaches

Another approach, that of controller-based techniques, is to synthesize a controller that activates the muscles to perform some action. For instance, the locomotion of snakes [19] and the walking motion of cockroaches [17] have been produced with periodic controllers. In many cases, controllers are designed by hand [11, 36], which requires carefully fine-tuning the control parameters. There are also several methods based on automatically synthesizing controllers [31, 32] using stochastic search strategies. These automatic synthesis techniques work well only for finding controllers for simple creatures with a small number of degrees of freedom. This is because the space of possible controllers becomes exceedingly large as the degrees of freedom increase, which makes it difficult for the stochastic search method to locate useful solutions.

One of the principle advantages of using controller-based techniques is that controllers are generally reusable and composable [6, 7, 27, 33]. A reusable controller can potentially be used to achieve a given motion within different environments. Composable controllers may be composed sequentially over time to generate a sequence of motions, which is what the user does with our system. That is, the user chooses controllers for certain actions and arranges them in particular order to create the entire animation.

2.2 Finite State Machine Controllers

Controllers can be based on finite state machines, which exploit the structured nature of certain motions, representing each phase of motion by a single state. The transitions between states can be time-based [31] that occurs after the controller has spent a prespecified time in a given state, or sensor-based [10] that occurs on a particular event. [11] created the animation of running, bicycling and vaulting using finite state motion control. These controllers are designed by hand which is difficult in that it requires specifying and testing many control parameters at each state. [31] presented an algorithm to automatically generate-and-test these parameters for relatively simple creatures and motions. The work of [14] uses a local stabilization technique upon a state machine model for the bipedal locomotion.

The animations created in our system are similar to the work of [36], which

explored the creation of three simulated 10m platform dives. The controller in that work is also based on a finite state machine model, with all the parameters tuned by hand. We provide a system for user based mid-level control, which significantly speeds up the creation of new motions by exposing all the key parameters in a readily accessible fashion, as shown in Figure 1.1. As a result, we are able to rapidly author many new motions, as demonstrated by the set of 46 motions across 3 sports that we have authored to date using the system.

2.3 Interactive Interfaces for Character Animation

There are many kinds of interactive interfaces for character animation. Most of them are based on kinematic approaches [15, 24], which copy particular joint angles from an input device, or copy things such as the positions of the hands or feet and then use inverse kinematics (IK) to compute joint angles. [21] developed a framework that used the two 6-DOF tracking devices to control a 33-DOF character by layering motions in several passes. [37] uses a foot pressure sensor pad to interactively control avatars. [28] demonstrates sketch-based interface for animating an articulated human character. The system presented in [5] presents a novel mimic-then-act interface for creating animations.

The increase of computer processor speed has made it possible to simulate the motion of articulated human models in real-time. This provides opportunities to extend the interactive control approaches used in flight and driving simulators to physically-simulated human figures. Early work on the use of interactive control for physically-based character was presented in [29, 30], which proposed the use of manual manipulation to control the motion of bipedal characters. The work of [13] mapped the 2D movements of a mouse and keystrokes to control parameters in order to design the motion of planar models. However the system did not provide support for refining inputs and there are potential scalability problems for 3-dimensional motions. Our system employs additional levels of abstraction in the interface and allows for the control of 3D character motion. The timeline component of our interface further supports the refinement of timing parameters. More recently, [22] demonstrates interaction with a partial dynamic 3D simulation for stepping motions. Our system is focussed specifically on dynamic-based aerial motions.

More recently, NaturalMotion has developed an industrial software, called endorphin [1], for dynamic motion synthesis. This software allows users to instruct characters like directors instruct actors by choosing from an extensive list of available behaviours, such as balancing, jumping, and staggering. The AI and biomechanics techniques built in this system enable the simulated character to interact realistically with the environment. For example, when characters walk over a swaying bridge, they try to negotiate the bridge's movements for keeping balance. The motions created in our system depend mostly on user-based control which exploits the user's knowledge and intuition. The system-based control is only for maintaining balance during standing and crouching.

2.4 Summary

We have presented the work from the computer animation literature that is related to our method. Our system builds on previous work in controller-based animation and interactive interfaces, as applied to 3D characters. The key ideas are embodied in our animation palette interface, which we describe in detail in the following chapter.

Chapter 3

The Animation Palette Interface

The core of our animation system consists of an *Animation Palette*, which is intended to be analogous to a painting palette. Simply put, it provides a library of actions for the user to interactively select. The user can specify both when a particular action should happen and how it should be done, such as "walk quickly in this direction" and "fall to the left." This chapter presents an overview of this system, while the following chapter elaborates on the specifics of how it is used to create dynamic aerial motions.

3.1 Interface and Workflow

Figure 3.1 shows an example interface of our system. The top part is a window displaying the animation that is being created. The inset-window in the bottom left corner of the animation window serves to show information related to the current action, such as the action name and the target pose¹. This inset-window is helpful for refining the motion offline, which is discussed in section 4.3.1.

¹This term will be defined shortly.



Figure 3.1: Interface of Animation Palette.

Below the animation window is a palette with 14 action buttons, one for each possible action. Each button defines an action and has two associated parameters that can be directly specified by the relative location of the user mouse-click or stylus-tap within the action button. The meanings of the two parameters are defined according to the attributes of the action. The specific details of the palette and actions are presented later in sections 3.2 and 4.1.

The bottom part of the interface is a timeline, which represents the sequence of user inputs. The user can adjust the timing and parameters of previous action events and then replay or resimulate to see the result. For dynamic aerial motions, our system can automatically optimize the timing of a specified input to achieve the best outcome. This process is described in further detail in section 4.3.2.

A typical workflow starts by interactively creating an animation through the use of the action buttons. This is an online process that has the user specifying actions during an ongoing physics-based simulation of the character. The user then manually fine-tunes the motion offline using the timeline provided in our interface, or can resort to using the automatic timing optimization feature.

3.2 Action Buttons

There are two basic classes of actions in our system, kinematic actions and dynamic actions. Kinematic actions produce kinematically-generated animation, i.e., it involves a direct specification of positions and orientations over time and does not involve the use of a physics-based simulation. For dynamic actions, motions are the result of physical simulation, which models of the body mass and moments of inertia, internal and external forces and torques, and interaction with the environment.

3.2.1 Kinematic Actions

Our system supports two types of kinematic actions, keyframe-based and motion capture-based. The first allows the user to specify parameterized keyframes. For instance, a "Point" action may use the x and y virtual button parameters to indicate the direction of the finger pointing, thus various pointing actions could be specified by clicking different positions of a "Point" button.

An alternate technique for kinematic animation is the use of motion capture clips. In this case, the two parameters are used to offer high level control over the motion sequence. Figure 3.2 shows two examples of this. The "Walk" button tells the character to walk, and the click position of this button controls the direction (θ) and speed ($v = k \cdot r$, where k is a constant) of walking. Likewise for the "Fall" button, the y position specifies the forward distance of the fall and the x position indicates the direction of the fall relative to the current walking direction.

3.2.2 Dynamic Actions

Dynamic character motions include both passive "rag doll" simulation and the use of active controllers. For passive simulations, the x and y parameters specify the environment properties such as the gravity and ground elasticity, or some inactive attributes of the character such as the joint stiffnesses.

In our system, the majority of action buttons for prototyping dynamic aerial motions implement active controllers. The x and y parameters of click position are used to parameterize the controllers, such as the target pose for proportionalderivative controllers, and the target lean angle for balance controllers. The details of this will be elaborated in section 4.1 and Appendix B.



Figure 3.2: Control of kinematic motions.

3.3 Switching Between Actions

When action A is in progress and a user then selects action B, we need a method for transitioning between these two types of actions. As stated above, our interface supports both kinematic actions and dynamic actions. This section discusses the four types of transitions that can occur during animation, namely kinematic to kinematic transitions, dynamic to dynamic transitions, kinematic to dynamic transitions, and dynamic to kinematic transitions.

3.3.1 Kinematic to Kinematic Transitions

Transitions between kinematic actions in our system are computed by blending from one animation to another over a period of 0.5 seconds. We blend the global position, global orientation, and joint angles using a linear interpolation method. This can work well when the new motion is sufficiently similar to the current motion. Other issues need to be solved to expand the range of motions between which we can make realistic transitions, such as the *footskate* cleanup. We do not solve this problem because the focus of our work is on generating and controlling dynamic aerial motions.

3.3.2 Dynamic to Dynamic Transitions

Transitioning between dynamic motions is straightforward because it involves a simple substitution of controllers. The continuity of the motion is guaranteed because the state of the system (positions and velocities) is preserved when the new dynamic action begins. The new controller begins control over the simulated character immediately upon the user choosing a new dynamic action. For example, the "landing" action of ski jumps happens once the user selects it and drives the character toward the pose preparing for landing. Note that the new controller does not provide a guarantee of a successful motion. Rather, it relies on the user's experience and perception to determine whether the character can successfully perform landing. Developing "automatic landing" controllers for actions which are difficult to reliably achieve is not within the scope of this thesis.



Figure 3.3: A kinematic action to dynamic action transition.

3.3.3 Kinematic to Dynamic Transitions

During kinematic to dynamic transitions, current positions and computed velocities for the degrees of freedom of the character become the initial states of the physical simulation which is initiated. Thus, the animated characters will initially preserve their linear and angular momentums immediately after the transition, which results in a natural-looking motion. Figure 3.3 shows an example of this. The character in red represents the first animation frame after the transition from a kinematic motion to the dynamic simulation. After that point the character can have physicallyrealistic interaction with the environment, such as the collisions with the block and the tree.

3.3.4 Dynamic to Kinematic Transitions

The dynamic to kinematic transition is a hard problem which requires a large library of motion clips and adaptive blending algorithms. In our system, we simply stop the physical simulation and do a straight cut to the kinematic action, which thus results in a visible discontinuity in the motion.

3.4 Summary

The animation palette interface provides a library of parameterized actions for the user to choose. These actions can be kinematic or dynamic. The user can interactively select among them to create a motion sequence, or refine them offline to improve upon a previous animation. In the next chapter, we will discuss in detail the use of dynamic actions for prototyping acrobatic aerial motions.

Chapter 4

Prototyping Dynamic Aerial Motions

Interactively controlling 3D dynamic characters is challenging due to the number of degrees of freedom (DOF) and issues such as the difficulty of maintaining balance. The interface should ideally be sufficiently expressive for the user to create a large variety of motions while still being tractable to learn. The animation palette provides a generic framework for creating animations, and we shall use it to design dynamic aerial stunts. Figure 4.1 shows two examples of the types of simulated motions that can be rapidly constructed using our interface.

4.1 Motion Stages and Stage Parameters

The structured nature of acrobatic aerial motions makes it possible to represent the required actions using a series of sequentially executed stages. For example, platform dives can typically be represented by the following sequence of actions: *stand*, *crouch*, *takeoff*, an aerial position such as a *pike* or a *tuck*, and *extension*. A



Figure 4.1: Simulated platform diving and ski jumping.

user can create the complete diving motion through specifying the proper timing and execution of all of these stages. As an example, Figure 4.2 illustrates the stages for forward somersaults in a pike position for platform diving. The simplest of our presented motions have 4 stages, while the most complex have 8 stages. Throughout the remainder of the thesis, the words *stage* and *action* will be used interchangeably.

We use one action button for controlling each possible action. The timing of the button selection represents when the associated action takes over the control of the simulated character, while two action parameters are given by the x and y location of the selection within the button. These action parameters serve to parameterize the action to make a variety of motions.

The following list describes the types of action parameters used in our system.

- **Target joint angle:** Sets the desired angle of a single joint for proportionalderivative controllers discussed in section 4.2.1. For example, the x-location of the *takeoff* button specifies the desired bend angle of the waist during a takeoff jump.
- Target body position: Indicates the desired position of several joints. For



Figure 4.2: 5 stages for forward somersaults pike.

instance, the x-location of the open pike button click represents the target angle of the pike, which is a combination of the waist and hip angles.

- Stiffness of joint: Designates the stiffness of one or more joints for the proportional-derivative controllers, which is helpful to create jumping motions with different heights.
- Center of mass: Denotes a target position for the center of mass relative to the location of ankles. This is an important parameter for the standing and crouching actions and preparing the body for a somersault, as illustrated in section 4.2.2.
- **Speed of performance:** Specifies how fast the character will reach the given target pose. It is important for most aerial actions like pike or tuck and it is discussed in detail in section 4.2.1.
- Initial state: Represents the starting locations of the motions. For example, in the case of ski jumps, this parameter specifies the starting height on the hill leading towards the ski jumps.

Figure 4.3 shows three examples of action buttons used in our platform diving interface. The two key parameters for the crouch action are the lean angle and



Figure 4.3: Three examples of stage parameters.

the height of crouch, which are mapped to the x parameter and the y parameter respectively. The lean angle is determined by the position of center of mass relative to the location of ankles, and is implemented in our system by setting the target lean angle of the balance controller discussed in section 4.2.2. The height of crouch is determined by the target hip and knee angles. The important attributes for the takeoff action are the initial rotation momentum, as determined by the target waist bend angle while taking off (x), and the height as determined by the stiffness of the hips and knees while jumping (y). For the open-pike action, the controllable parameters are the target angle of pike (x), which is a combination of the waist and hip angles, and the speed to reach this pose (y), which is determined by the interpolation duration of the target angles discussed in section 4.2.1.

A complete summary of the set of buttons used for the platform diving, acrobatic ski jumping, and snowboard control is given in Appendix B.

4.2 Dynamic Controllers

In order for a given dynamic action to be realized, we need to determine low-level controllers that compute how much torque each joint should exert in order to achieve the desired result. There are two classes of controllers used in our system, one for driving the character toward target poses and another class of controller for maintaining balance. We now discuss each of these in turn.

4.2.1 Proportional-derivative Controller

The joint torques that drive joint angles toward their desired values are computed using proportional-derivative (PD) controllers:

$$\tau = k_p \cdot (\theta_{desired} - \theta_{current}) - k_d \cdot \theta,$$

where τ is the torque applied at the joint, $\theta_{desired}$ is the desired joint angle, $\theta_{current}$ is the current joint angle, $\dot{\theta}$ is the relative angular velocity of the two links connected by that joint, and k_p , k_d are proportional and derivative control constants.

The values of k_p and k_d control the strength of the actuator for each joint, which should be strong enough to achieve particular desired actions. Appendix A gives the specific control constants used for our dynamic models. For the hip, knee, and ankle joints, their control constants are different across various actions. For example, the "stand" action usually requires higher k_p and k_d values at the ankles than the "pike" action.

Linear interpolation of the target angles, i.e., $\theta_{desired}$, is used between targetangle poses in order to produce smooth transitions between the most recent pose and a newly chosen pose. The duration of the interpolation is exposed as one of the motion parameters for many actions, such as pike and tuck.



Figure 4.4: Sagittal plane balance control in standing.

4.2.2 Balance Controller

Maintaining balance is crucial for stages that preceed and follow the aerial portions of the motion. Balance control can depend on many factors, such as the distance between two feet and the placement of two feet. In our system, we focus on the balance during a regular stance with the feet being spaced at shoulder width.

According to [34], balance in the sagittal plane can be modeled as an inverted pendulum about the ankle joint, as shown in Figure 4.4. The torque due to gravity is

$$M \cdot g \cdot h \cdot \sin(\theta) \approx M \cdot g \cdot h \cdot \theta,$$

where M is the mass of the character, g is gravitational acceleration, h is the height from the ankle to the Center Of Mass (COM), and θ is the leaning angle as measured in radians. The torque exerted by the spring muscle at the ankle is $\tau = k_p \cdot \theta$. As long as $k_p > M \cdot g \cdot h$, the ankle will be sufficiently strong to maintain the equilibrium position. The addition of damping in the ankle joint torque ensures that the body



Figure 4.5: Lateral balance control in standing.

will eventually stop near the target leaning angle, which is set by the x parameter of the "Stand" and "Crouch" buttons. The body can no longer maintain balance when the lean angle is such that the projection of COM onto the ground plane falls outside the support polygon of two feet. However, such a body position is required before a jump in order to initiate angular momentum. The front-back balance controller thus does not only serve to maintain balance, but also to prepare the body for somersaults.

Using the ankles for lateral stability is not very stable because the width of foot is much smaller than the length of foot. Thus, other joints must be used to help keep lateral balance. The hip abductor and adductor moment can move the Center Of Pressure (COP) between two feet, which draws the COM back to the equilibrium position. For human during the quiescent stance, the lateral balance is dominated by this hip load-and-unload strategy [34]. Figure 4.5 shows the character in the frontal plane, in which the equilibrium position is the point midway between



Figure 4.6: Lateral balance control in crouching.

the two feet. The torques of the hip abductor and adductor are computed using a spring-damper model:

$$\tau = k_p \cdot x - k_d \cdot \dot{x}$$

where x is the distance between the COM and its equilibrium position and \dot{x} is the velocity of the COM.

In the crouched position, a knee-based strategy is used for lateral balance. Similar to the hip strategy, the knee flexor and extensor also serve to move the location of the COP in order to control the COM. For example, extending the left knee and flexing the right knee will move the COM to the right. Figure 4.6 illustrates a crouching character, using knee joints to maintain lateral balance.

Maintaining balance is a challenging part of creating dynamic character motions. The mechanisms discussed above can only solve the balance problem in some cases. For complex problems, such as the landing of the skier and the snowboarder, we need to develop more adaptive controllers. This is discussed as future work in Chapter 6.

4.3 Offline Motion Refinement

Our system supports both online and offline authoring of aerial motions. Online use of the interface enables the creation of simulated animations in real time through the use of the action buttons during an ongoing simulation. Offline adjustments can be used to refine the resulting motions, through either manual adjustment or automatic optimization.

4.3.1 Manual Adjustment

Manual adjustments are useful for producing a motion through iterative refinement and is supported through the use of a timeline included at the bottom of the interface panel. Each virtual button press introduces an event that is displayed on the timeline, as shown in Figure 4.7. The timing of this event can be adjusted directly on the timeline by dragging with the mouse. Selecting any event marker on the timeline also allows for the associated action parameters to be altered, as specified by the selection point within the associated virtual button. In Figure 4.7, the red cross-hairs drawn on the "Takeoff" button illustrate the current parameter settings associated with a selected takeoff action. These parameters can be adjusted with a mouse or stylus press at the desired location within the button. Once a set of desired adjustments has been made, the user hits a "Resimulate" button in order to review the newly refined motion.

Unlike during online creation, manual adjustments offline have no real-time feedback of the animation. Nevertheless, there are several useful feedback signals



Figure 4.7: Interface for offline design.

that can help the user with the motion refinement. A simple-but-helpful cue is showing the target pose while the user adjusting the parameters associated with a button. This provides the user with immediate feedback of what kind of action is specified. As shown in Figure 4.7, the inset-window in the bottom left corner of the animation window serves this purpose.

Another useful form of visual feedback is to display the animation frames while the user drags the timeline cursor. This is especially helpful for adjusting the timing of inputs. Since some aerial motions pass by too quickly for the user to catch particular events, such as the specific time instant when the skier reaches the top of the kicker jump, offline scrolling through the animation is the best way to handle this.

4.3.2 Automatic Optimization

To further speed the creation of successful simulated aerial motions, our system can automatically optimize the timing of a specified input to achieve the best outcome. Accomplishing this requires a metric to evaluate the quality of a motion and a fast function evaluator, which in our case consists of a simulation that determines how changes in timing affect the given metric. Numerical methods which exploit derivative information will generally require the fewest function evaluations. However, this requires differentiable optimization metrics, which may be difficult to achieve in the face of non-linear effects such as joint limits and other discontinuous events that may occur during the course of a simulation. In the face of these difficulties, we perform the optimization by discretely sampling the timing parameter in question at regular intervals that bracket its current value. Function evaluations are then carried out using a dynamic simulation from that point in time forward to the time



Figure 4.8: Dive entry before and after optimization of the time for extension.



Figure 4.9: Ski landing before and after optimization of the time for stretch.

where the metric is evaluated. The previous motion is used to start each of these simulations from the appropriate dynamic state.

Our optimization metric for platform diving is computed by integrating the error of the water entry angle for the body:

$$C = \frac{\int_{t_1}^{t_2} E^2(t) \,\mathrm{d}t}{t_2 - t_1}$$

where C is the total error which needs to be minimized, t_1 is the time at which the diver first touches the water, t_2 is the time at which the upper body is fully under water, and E(t) is the error of the entry angle at time t which is defined in terms of the angular deviation from the vertical. A component measuring twist errors could also be added, although we have not experimented with this. Figure 4.8 illustrates several frames from a diving motion before and after the optimization of the timing parameter associated with the extension action, which is the last action prior to water entry.

Similarly, the optimization metric for ski jumping is calculated by integrating the error of the landing angle for the body, from the time the skier first contacts the ground to the time it reaches the end of the landing area. Figure 4.9 shows the landing motions before and after the optimization of the timing for the stretch action, which is the last action prior to landing.

4.4 Summary

In this chapter, we have presented the animation palette framework for prototyping dynamic aerial motions. The required actions are represented by a series of sequentially executed stages. We use one action button for controlling each possible action. PD-controllers compute the torque at each joint for driving the character to achieve a desired action. Balance controllers are responsible for actively maintaining the balance. Finally, offline motion refinement can serve to improve motions to achieve the desired result.

In the following chapter, we will describe implementation issues in detail and we present the resulting motions created using our system.

Chapter 5

Implementation and Results

We have implemented a system using our- interface ideas to design motions for platform diving, freestyle aerial ski jumps, and half-pipe snowboarding. The next two sections will describe the dynamic simulation and the character models we used to make the aerial motions, and the remaining sections will describe our results in detail.

5.1 Dynamics Simulation

Our system makes use of the publicly-available Open Dynamics Engine (ODE) [26] for computing and integrating the equations of motion. One advantage of this simulation library is that models can be assembled on the fly. Links and joints can be added, removed or changed during the simulation. This feature makes ODE more flexible to use than some other libraries such as SD/FAST [12] which requires pre-compiling the models before the simulation begins.

The process of simulating the motions of rigid bodies through time has two parts. During each time step, the first part uses the equations of motion to determine the instantaneous accelerations for all the rigid bodies. The second part, namely the integration, advances the current time by a given step size and adjusts the state of all the rigid bodies for the new time value. ODE uses a first order semiimplicit integrator, which includes both the implicit forces (i.e., constraint forces at the joints) and explicit forces (i.e., external forces applied by the environment and the applied joint torques). Since we use controllers to drive the character toward target poses and the external forces are explicit, the integration can become unstable when the system is stiff, i.e., when k_p and k_d are large. Reducing the simulation time step enhances the stability, but results in slower simulation speed. We limit the maximum torque that each joint can exert to avoid excessively-large control torques, which reflects the reality of limited muscle forces and also helps with the stability of the simulation. In our system, the maximum allowable torque at each joint is given by $\tau_{max} = 1.0 \cdot k_p$, where k_p is the spring constant for this joint.

For collision detection, we assign each body a simple shape (e.g. sphere or box), and use the collision detection engine supplied with ODE to determine which bodies are in contact at each time step. The average time of this collision engine for intersection testing is O(n) [26], where n is the number of objects. To simplify the control and increase the stability, we only process collisions between the character and the environment (e.g., ground), and we ignore the self-collision of different parts of the character model. The collision reaction forces in ODE are based on a penalty method and a Coulomb friction model is used to limit the tangential component of the reaction forces. This is sufficient to simulate the interaction forces between the character and the ground.

The ODE-based simulation runs in real-time on a 2.66 GHz P4 PC. The fixed time step which we use to integrate the equations of motion is 0.001s. Without



Figure 5.1: Degrees of freedom used for diving, snowboarding and skiing models.

graphical display, our diving simulation can compute 3.7 simulation seconds in 1 wall-clock second. For skiing and snowboarding, the performance numbers are 1.68 and 1.69 respectively. The diving simulation requires significantly fewer collision detection computations and is faster as a result.

5.2 Character Models

The character models are based on the anthropometric parameters used in [6]. We make changes to the original model according to the nature of each aerial motion. For example, the skier needs to bind each foot to a ski. In order to increase the speed and stability of dynamic simulation, we remove the wrist joints and the joint



Figure 5.2: States of Platform Diving.

between head and neck. These joints have little effect on the simulation results and can make the simulation unstable, due to the large disparity between lightweight body parts such as the hands and the heavier links such as the torso.

The structures of the three character models are shown in Figure 5.1. The diver model has 13 links and 26 Internal Degrees Of Freedom (IDOF). The snowboarder is the same but with an extra DOF for each ankle. Lastly, the skier has fixed ankles, giving 13 links and 22 IDOF. The details of these dynamic models are given in Appendix A, including all the physical parameters of the links (e.g., mass and moment of inertia) and joints (e.g., joint strength).

5.3 Platform Diving

Platform diving is the least difficult motion to design because it does not need to solve the problem of balance upon landing, as is the case with ski jumping and snowboarding. We represent the motion of dives with 5 stages of actions: stand, crouch, takeoff, one or more aerial positions, and extension before entry into water. Our interface has 12 action buttons for the control of all possible actions during diving. Figure 5.2 shows the actions and the possible transitions between them. Table B.1 illustrates the actions and x and y parameters associated with each button.

We have used the interface to author 29 types of 5m platform dives, including all four takeoff methods a diver can use: forward, backward, inward and reverse dives. In the diving terminology of [20], we have authored dive types 101a, 101b, 101c, 103b, 103c, 105b, 107c, 201a, 201b, 201c, 203b, 203c, 205c, 301a, 301b, 301c, 303b, 303c, 305c, 401a, 401b, 401c, 403b, 403c, 405b, 405c, 407c, 5132d. Figures 5.3, 5.4 and 5.5 show three of them, while Figure 5.6 illustrates an unsuccessful dive in which the extension action is executed too early.



Figure 5.3: Platform diving: forward $1\frac{1}{2}$ some rsault pike with one twist.



Figure 5.4: Platform diving: backward $1\frac{1}{2}$ somer sault pike.



Figure 5.5: Platform diving: reverse $1\frac{1}{2}$ some rsault tuck.



Figure 5.6: Platform diving: bad entry for inward $1\frac{1}{2}$ some rsault pike.

5.4 Freestyle Aerial Ski Jumping

For skiing, we unite the lower leg, foot and ski into one rigid body segment for the simulation model. Similar to platform diving, aerial ski jumps have 6 executed stages: in-run, crouch, takeoff, aerial positions, stretch for landing and finish position. The purpose of the "finish" position is to improve the visual effect by making the character straighten and raise the arms upon a successful landing. Table B.2 describes the definition of the virtual buttons we use to control the acrobatic ski actions and Figure 5.7 shows the feasible transitions between them. For the in-run action, one of the button parameters provides control over the start position on the in-run, and hence the speed accumulated upon take-off.



Figure 5.7: States of Ski Jumping.

The kicker jumps and ski-hill in our system have been designed to match freestyle skiing competition specifications [25]. Figure 5.8 shows the specification of ski-hill, while Figure 5.9 and Table 5.1 illustrates the dimensions of kicker jumps. In reality, the snow friction coefficients are different between the forward sliding C_f and the sideway sliding C_s . We use 0.001 for C_f and 0.1 for C_s to ensure the realistic interaction of the skis and the snow.



Figure 5.8: Specification of ski-hill.



Figure 5.9: Dimensions of kicker jumps.

kicker	Α	В	С	D (1 <i>m</i>)	$\mathbf{E}(2m)$	$\mathbf{F}(2m)$	G(2m)
small	6.70	3.25	5.95	50°	30°	13°	5°
big	8.10	3.65	6.70	55°	33°	14°	6°

Table 5.1: Dimensions of kicker jumps.

The creation of successful aerial ski jumps is more difficult than diving because of the difficulty of maintaining balance upon landing. It is important to find the best time for extension in order to decrease the rotation and twist speed before landing. We decrease the stiffness of knees and hips for landing position in order to cushion the kinetic energy of the character when landing on the ground. This also makes the landing action look more natural.

A series of 12 aerial ski jumps were created using our interface: bFF, bL, bLLT, bLTL, bPP, bTT, fF, fL, fPP, fT, fTT, and fTTT, where 'f' and 'b' indicate front and back flips respectively, 'F' indicates a flip with a full twist, and 'L', 'P', and 'T' indicate layout, pike, and tuck positions, respectively. Figures 5.10, 5.11 and 5.12 show three of them and Figure 5.13 illustrates a bad landing for a back-full-full jump. We have also experimented with simulating landings in water, as is typical of summer training.



Figure 5.10: Ski jump: double twisting double back flip.



Figure 5.11: Ski jump: front pike single.



Figure 5.12: Ski jump: back layout-tuck-layout.



Figure 5.13: Ski jump: bad landing for a back-full-full.

5.5 Half-pipe Snowboarding

Half-pipe snowboarding was the most difficult motion to produce using our system, the principal reason being the difficulty of maintaining balance at a variety of points in the half-pipe. Landing diagonally on the transition between the wall and ground was found to be much harder to control than on the landing hill of ski jumps. Also, the snowboarder needs to be well balanced upon approaching the wall of the halfpipe in the lead up to the jump in order to successfully initiate a jump. The terrain anticipation of a real snowboarder during the rapid transition from the horizontal to the near-vertical edge of the half-pipe is not implemented in our system, making this challenging to properly execute.



Figure 5.14: States of Half-pipe Snowboarding.

From our experience with platform dives and ski jumps, motions with twists were generally found to be more difficult to reconstruct than motions without twists. Most half-pipe stunts involve a twist and this thus makes the snowboarding motions more challenging to control. We have thus far created 4 types of half-pipe snowboarding stunts, including a back 360, back 360 with grab, a front grab, and a back grab. Figure 5.16, 5.17, 5.18 and 5.19 illustrate some successful or unsuccessful motions we have made. Figures 5.14 and Table B.3 explains the virtual buttons for this stunt. The specification of the half-pipe comes from [9]. Figure 5.15 illustrates the section view of the half-pipe, which has an inclination with respect to the horizontal of 15° in the dimension going into the page.



Figure 5.15: Section view of half-pipe.

5.6 User Experiences

Our system has been used by several users who have little knowledge of control mechanisms. After 10 minutes tutorial, all of them can create relatively simple diving motions in 5 minutes, such as forward $1\frac{1}{2}$ somersaults in pike position, and backward $2\frac{1}{2}$ somersaults in tuck position. For the dives which involve more somersaults, these users need some basic knowledge of platform diving and about half an hour guided practise before they can produce such motions within 10 minutes. For the creation of ski jumps, these users usually spend most of time on finding the appropriate time of takeoff and the proper pose for landing. This makes the time of creating successful ski jumps three times as much as that of creating successful platform dives. As we expected, the most difficult part is the creation of half-pipe snowboarding or motions which involve a twist. Making these motions may take several hours to fine-tune the inputs.

5.7 Summary

Our animation system provides a method for interactively controlling 3D simulated characters. It allows for dynamic aerial motions to be produced in a short time by a novice user. A variety of motions and variations can be created as demonstrated in this chapter. Since we expose the key parameters of each action in a readily accessible fashion, this interface can work in different environments with no modification. For example, the virtual buttons for 5m platform diving can also be used to produce 10m platform diving without any changes to the animation palette.

The main weakness of this system lies in the balance controllers, especially for motions with twist and landing actions. More robust and adaptive control strategies are required to make the interface more tractable for creating successful examples of these types of motions.



Figure 5.16: Half-pipe snowboarding: back-grab.



Figure 5.17: Half-pipe snowboarding: front-grab.



Figure 5.18: Half-pipe snowboarding: back 360.



Figure 5.19: Half-pipe snowboarding: crash landing for back-grab.

Chapter 6

Conclusions and Future Work

We have presented an interface for prototyping dynamic aerial motions. This technique demonstrates the future feasibility of designing sports games based upon fully dynamic character simulations. It allows a user to exploit his knowledge and intuition to design a variety of motions. Our interface can also help the user to understand and learn these acrobatic stunts.

There are several directions which require further research to make the interface easier to use and the resulting motions more realistic.

6.1 Improved Dynamic Controller Actions

We plan to investigate how controllers specific to the execution of twists and landing actions can be constructed. Additional work on "automatic landing" controllers and user control is required in order to make the required user skill more manageable for use in a game scenario. Our current automatic optimization tool uses a simple metric and optimization technique. We wish to explore the use of other metrics that allow for more abstract specifications of a desired motion, as well as an analysis of the timing requirements of the various stages of the motion.

6.2 Natural Motion

We wish to seek feedback from coaches and athletes with regard to using this type of tool. In order to become an accurate prototyping tool, better strength models will likely be required. It would also be interesting to use this system in conjunction with a vision-based motion tracking system. This would support the asking of "what if" questions in the context of exploring changes to a recently executed motion.

6.3 GamePad Interface

Another possible direction for future work is toward the using of gamepad as the input device. One of the potential applications of our research is sports genre video games. These are most commonly played on video game consoles, which are equipped with a gamepad rather than a mouse. Although a gamepad has a large number of inputs, namely dual thumb sticks and about ten buttons, it is difficult to map our parameterized action button interface onto such a device. We need to find other methods to map the gamepad inputs to the control parameters.

Bibliography

- [1] Nick Alexander. Endorphin. http://www.naturalmotion.com/.
- [2] Lynne Shapiro Brotman and Arun N. Netravali. Motion interpolation by optimal control. In SIGGRAPH 88, Computer Graphics Proceedings, pages 309– 315. ACM Press, 1988.
- [3] Kyle Clark. Inspired 3D Character Animation. Premier Press, 2002.
- [4] Michael F. Cohen. Interactive spacetime control for animation. In SIGGRAPH 92, Computer Graphics Proceedings, pages 293–302. ACM Press, 1992.
- [5] Mira Dontcheva, Gary Yngve, and Zoran Popovic. Layered acting for character animation. ACM Trans. Graph., 22(3):409–416, 2003.
- [6] Petros Faloutsos. Composable Controllers for Physics-Based Character Animation. PhD thesis, Department of Computer Science, University of Toronto, 2002.
- [7] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Composable controllers for physics-based character animation. In SIGGRAPH 2001, Computer Graphics Proceedings, pages 251–260. ACM Press / ACM SIGGRAPH, 2001.
- [8] Anthony C. Fang and Nancy S. Pollard. Efficient synthesis of physically valid human motion. ACM Trans. Graph., 22(3):417–426, 2003.
- [9] Frank Gille. Snowboarding: make a perfect start. Meyer & Meyer Sports, 2002.
- [10] Jessica K. Hodgins, Paula K. Sweeney, and David G. Lawrence. Generating natural-looking motion for computer animation. In *Proceedings of the conference on Graphics interface '92*, pages 265–272. Morgan Kaufmann Publishers Inc., 1992.

- [11] Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. Animating human athletics. *Proceedings of SIGGRAPH 95*, pages 71–78, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California.
- [12] Michael G. Hollars, Dan E. Rosenthal, and Michael A. Sherman. Sd/fast, 1991.
- [13] Joseph Laszlo, Michiel van de Panne, and Eugene Fiume. Interactive control for physically-based animation. In Kurt Akeley, editor, SIGGRAPH 2000, Computer Graphics Proceedings, pages 201–208. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.
- [14] Joseph F. Laszlo, Michiel van de Panne, and Eugene Fiume. Limit cycle control and its application to the animation of balancing and walking. *Proceedings of* SIGGRAPH 96, pages 155–162, August 1996.
- [15] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. In *SIGGRAPH 2002, Computer Graphics Proceedings*, pages 491–500. ACM Press, 2002.
- [16] Zicheng Liu, Steven J. Gortler, and Michael F. Cohen. Hierarchical spacetime control. In SIGGRAPH 94, Computer Graphics Proceedings, pages 35–42. ACM Press, 1994.
- [17] Michael McKenna and David Zeltzer. Dynamic simulation of autonomous legged locomotion. In SIGGRAPH 90, Computer Graphics Proceedings, pages 29–38. ACM Press, 1990.
- [18] Alberto Menache. Understanding Motion Capture for Computer Animation and Video Games. Morgan Kaufmann, 1999.
- [19] Gavin S. P. Miller. The motion dynamics of snakes and worms. In SIGGRAPH 88, Computer Graphics Proceedings, pages 169–173. ACM Press, 1988.
- [20] Ronald F. O'Brien. Springboard & Platform Diving: a complete guided for divers and coaches. Human Kinetics, 2003.
- [21] S. Oore, D. Terzopoulos, and G. Hinton. A desktop input device and interface for interactive 3d character animation. In *Proceedings of Graphics Interface* 2002, pages 133–140, 2002.
- [22] S. Oore, D. Terzopoulos, and G. Hinton. Local physical models for interactive character animation. *Computer Graphics Forum*, 21(3):337–346, 2002. ISSN 1067-7055.

- [23] Jovan Popovic, Steven M. Seitz, Michael Erdmann, Zoran Popovic, and Andrew Witkin. Interactive manipulation of rigid body simulations. In SIGGRAPH 2000, Computer Graphics Proceedings, pages 209–217. ACM Press/Addison-Wesley Publishing Co., 2000.
- [24] Hyun Joon Shin, Jehee Lee, Sung Yong Shin, and Michael Gleicher. Computer puppetry: An importance-based approach. ACM Trans. Graph., 20(2):67–94, 2001.
- [25] Pat Smith. Canadian freestyle ski association. http://www.freestyleski.com/.
- [26] Russell Smith. Open dynamics engine. http://q12.org/ode/.
- [27] A. James Stewart and James F. Cremer. Beyond keyframing: an algorithmic approach to animation. In *Proceedings of the conference on Graphics interface* '92, pages 273–281. Morgan Kaufmann Publishers Inc., 1992.
- [28] Matthew E. Thorne. Motion doodles: A sketch-based interface for character animation. Master's thesis, Department of Computer Science, University of British Columbia, 2003.
- [29] J. Troy. Dynamic Balance and Walking Control of Biped Mechanisms. PhD thesis, Department of Mechanical Engineering, Iowa State University, 1995.
- [30] J. Troy and M. Vanderploeg. Interactive simulation and control of planar biped walking devices. In Workshop on Simulation and Interaction in Virtual Environments, pages 220–224, July 1995.
- [31] Michael van de Panne, Ryan Kim, and Eugene Fiume. Virtual wind-up toys for animation. In *Proceedings of Graphics Interface '94*, pages 208–215, Banff, Alberta, Canada, 1994.
- [32] Michiel van de Panne. Sensor-actuator networks. In SIGGRAPH 93, Computer Graphics Proceedings, pages 335–342. ACM Press, 1993.
- [33] Michiel van de Panne, Eugene Fiume, and Zvonko Vranesic. Reusable motion synthesis using state-space controllers. In SIGGRAPH 90, Computer Graphics Proceedings, pages 225–234. ACM Press, 1990.
- [34] David A. Winter, Aftab E. Patla, Francois Prince, Milad Ishac, and Krystyna Gielo-Perczak. Stiffness control of balance in quiet standing. *Journal of Neu*rophysiology, 80:1211–1221, 1998.

- [35] Andrew Witkin and Michael Kass. Spacetime constraints. Computer Graphics, 22(4):159–168, 1988.
- [36] Wayne L. Wooten and Jessica K. Hodgins. Animation of human diving. Computer Graphics Forum, 15(1):3–14, 1996. ISSN 0167-7055.
- [37] KangKang Yin and Dinesh K. Pai. Footsee: an interactive animation system. In Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pages 329–338. Eurographics Association, 2003.

Appendix A

Dynamic Model Details

Below are the details of three dynamic models shown in Figure 5.1: diver, skier and snowboarder. For each model, the COM of hip is placed at the origin, with the character facing the positive z-axis. We present only the information of left arm, hand, leg and foot; the properties of corresponding right parts are symmetrical. For some joints, the strength constants depend on whether the character is in *stance*, *takeoff, flight* or *landing* state. In these tables we use 's', 't', 'f' and 'l' to represent them.

link	mass	center of mass	moment of inertia
	(kg)	$(\mathrm{x},\mathrm{y},\mathrm{z}\ m)$	$(x,y,z \ kgm^2)$
hip	16.61	0.000 0.000 0.000	$0.1147 \ 0.1147 \ 0.1445$
thigh	8.35	0.086 - 0.237 0.025	$0.1654 \ 0.0176 \ 0.1662$
shin	4.16	0.086 -0.663 0.025	0.0522 0.0069 0.0522
foot	1.34	0.086 -0.887 0.075	0.0076 0.0084 0.0020
chest	29.27	0.000 0.335 0.025	$0.5282 \ \ 0.1645 \ \ 0.5829$
neck	1.00	0.000 0.611 0.025	0.0021 0.0021 0.0021
head	5.89	0.000 0.792 0.025	0.0457 0.0261 0.0417
upper arm	2.79	0.300 0.498 0.025	0.0047 0.0229 0.0229
lower arm	1.21	0.568 0.498 0.025	0.0009 0.0061 0.0062
hand	0.55	0.732 0.498 0.025	0.0004 0.0018 0.0015

Table A.1: Physical parameters of diver model.

name	location	axis	$\mathbf{k_p}$	$\mathbf{k}_{\mathbf{d}}$
	$(\mathbf{x}, \mathbf{y}, \mathbf{z} \mid m)$			
		x	1275	60
hip	0.086 0.000 0.025	y	200	10
		z	1275	60
knee	$0.086 - 0.475 \ 0.025$	x	1275	60
		z	80	5
ankle	0.086 - 0.850 0.025	x (s)	1275	60
		x (t, f)	120	12
		x	1275	60
waist	0.000 0.114 0.025	z	1275	120
		y	800	40
		z	100	10
neck	0.000 0.555 0.025	y	20	2
		x	200	10
shoulder	0.152 0.498 0.025	y	100	5
shoulder	0.102 0.430 0.020	z	100	10
elhow	0.440 0.408 0.025	\overline{x}	20	1
ernow	0.449 0.490 0.020	y	100	5

Table A.2: Joint strength of diver model.

link	mass	center of mass	moment of inertia
	(kg)	$(\mathrm{x},\mathrm{y},\mathrm{z}\ m)$	$(x,y,z \ kgm^2)$
hip	16.61	0.000 0.000 0.000	0.1034 0.1034 0.1445
thigh	8.35	$0.086 - 0.237 \ 0.025$	$0.1654 \ 0.0176 \ 0.1662$
shin	4.16	0.086 -0.663 0.025	0.0522 0.0069 0.0522
foot	2.34	0.086 - 0.891 0.083	$0.0141 \ \ 0.0147 \ \ 0.0044$
ski	1.80	$0.086 - 0.952 \ 0.094$	0.3376 0.3394 0.0020
chest	29.27	0.000 0.335 0.025	0.5282 0.1645 0.5829
neck	1.00	0.000 0.611 0.025	0.0021 0.0021 0.0021
head	5.89	0.000 0.792 0.025	0.0457 0.0261 0.0417
upper arm	2.79	0.300 0.498 0.025	0.0047 0.0229 0.0229
lower arm	1.21	0.568 0.498 0.025	0.0009 0.0061 0.0062
hand	0.55	0.732 0.498 0.025	0.0004 0.0018 0.0015

Table A.3: Physical parameters of skier model.

name	location	axis	$\mathbf{k_{p}}$	k _d
	$(\mathbf{x}, \mathbf{y}, \mathbf{z} \mid m)$			
		z	900	30
		y	1275	60
hip	0.086 0.000 0.025	x (s)	4000	60
		x (t, f)	1275	60
		x (l)	400	60
		x (s)	4000	60
knee	0.086 - 0.475 0.025	x (t, f)	1275	60
		x (l)	400	60
		x	2000	100
waist	0.000 0.114 0.025	z	2000	100
Waist		y (s, f, l)	2000	100
		y (t)	3000	100
		z	100	10
neck	0.000 0.555 0.025	y	20	2
		x	200	20
shoulder	0 152 0 498 0 025	y	100	5
SHOULUEL	0.102 0.430 0.020	z	100	10
elbow	0.449 0.498 0.025	x	20	1
CIDOM	0.449 0.498 0.020	y	100	5

Table A.4: Joint strength of skier model.

link	mass	center of mass	moment of inertia
	(kg)	$(\mathbf{x}, \mathbf{y}, \mathbf{z} \mid m)$	$({ m x,y,z} \ kgm^2)$
hip	16.61	0.000 0.000 0.000	$0.1147 \ 0.1147 \ 0.1445$
thigh	8.35	0.127 -0.234 0.025	$0.1654 \ 0.0176 \ 0.1662$
shin	4.16	0.201 -0.652 0.025	0.0522 0.0069 0.0522
foot	2.34	0.233 -0.887 0.075	$0.0141 \ \ 0.0147 \ \ 0.0044$
snowboard	3.80	$0.000 - 0.950 \ 0.075$	0.0390 0.6595 0.6209
chest	29.27	0.000 0.335 0.025	$0.5282 \ \ 0.1645 \ \ 0.5829$
neck	1.00	0.000 0.611 0.025	0.0021 0.0021 0.0021
head	5.89	0.000 0.792 0.025	0.0457 0.0261 0.0417
upper arm	2.79	0.300 0.498 0.025	0.0047 0.0229 0.0229
lower arm	1.21	0.568 0.498 0.025	0.0009 0.0061 0.0062
hand	0.55	0.732 0.498 0.025	0.0004 0.0018 0.0015

Table A.5: Physical parameters of snowboarder model.

name	location		axis	$\mathbf{k}_{\mathbf{p}}$	$\mathbf{k_d}$
	$(\mathrm{x},\mathrm{y},\mathrm{z}\ m)$				
).000 0.025	x (s, f)	1275	60
			x (t)	200	60
hip	0.086 0.000 (x (l)	400	60
			z (s)	4000	60
			z (t, f, l)	900	30
knee		8 0.025	x (s, f)	1275	60
	0.168 -0.468 (x (t)	200	60
			x (l)	400	60
ankle		0.025	x	170	17
	0.233 -0.837 (y	50	5
			z	80	8
waist		0.025	z	2000	100
	0.000 0.114 (x	2000	100
	0.000 0.114 (y (s, f, l)	2000	100
			y (t)	4000	100
neck		0.025	x	200	20
	0.000 0.555 0		z	100	10
			y	20	2
shoulder	0.152 0.408 0	8 0.025	y	100	5
	0.152 0.498 (z	100	10
elbow	0.440 0.408 0	0.025	x	20	1
	0.449 0.498 (y	100	5

Table A.6: Joint strength of snowboarder model.

Appendix B

Description of Virtual Buttons

name	axis	definition	note	
stand forward	х	target c.o.m. position	stand facing water	
stand for ward	У	arm position	stand lacing water	
stand backward	х	target c.o.m. position	stand back to water	
Stand Dackward	У	arm position	Stand back to water	
arouch arms up	х	target c.o.m. position	crouch, raise arms	
crouch arms-up	У	height of crouch	for forward takeoff	
	х	target c.o.m. position	crouch, lower arms	
crouch arms-down	У	height of crouch	for backward takeoff	
takeoff forward	х	waist bend	extend hips, knees	
	У	jump height	and ankles for jump	
	х	waist bend	extend hips and ankles	
takeon backward	У	jump height	swing arms back	
pike	У	time to reach pose		
open pile	х	target pike angle		
open pike	У	time to reach pose		
tuck	У	time to reach pose		
twist	х	twist of waist	right hand to head,	
	У	time to reach pose	left hand to chest	
lateral extend	х	waist bend	bring arms laterally	
	У	time to reach pose	to over head	
straight ortand	х	waist bend	bring arms straight	
Straight extend	У	time to reach pose	over head	

Table B.1: Description of diving buttons.

name	axis	definition	note	
start big	х	start position	start for the big kicker	
start big	У	angle of arm raise		
start small	х	start position	start for the small kicker	
Start Sman	У	angle of arm raise		
crouch arms-up	х	hip bend	bring arms to front	
crouch arms-up	У	knee bend	for forward takeoff	
crouch arms-down	х	hip bend	swing arms down	
	У	knee bend	for backward takeoff	
takeoff forward	х	waist bend	extend hips, knees	
	У	strength of jump	and ankles for jump	
takeoff backward	х	waist bend	extend hips and ankles	
	У	strength of jump	for jump, swing arms back	
takeoff twist fwd	х	waist twist	extend hips, knees, and	
takeon twist two	У	strength of jump	ankles while twisting waist	
takeoff twist bwd	х	waist twist	as above, but swing arms back	
takeon twist bwu	У	strength of jump	and apply less knee extension	
pike	У	time to reach pose		
tuck	У	time to reach pose		
layout	х	waist bend	ortand arms	
	У	time to reach pose	extend arms	
twist	У	time to reach pose	bring arms to body	
landing	х	hip bend	alter stiffness of hips	
	У	knee bend	and knees for landing	
finish	х	arm position	stand up and raise arms	
11111511	У	time to reach pose		

Table B.2: Description of buttons for aerial ski jump control.

name	axis	definition	note	
crouch left	х	waist twist	crouch down facing left	
	У	front/back balance		
crouch right	х	waist twist	crouch down facing right	
	У	front/back balance		
grab front	У	time to reach pose	grab front side of snowboard	
grab back	У	time to reach pose	grab back side of snowboard	
takeoff	х	waist bend		
	У	time to reach pose		
landing	х	waist bend		
	У	lateral balance position		

Table B.3: Description of buttons for snowboard control.