User Interfaces for Interactive Control of Physics-based 3D Characters

Peng Zhao Michiel van de Panne
University of British Columbia*

Abstract

We present two user interfaces for the interactive control of dynamically-simulated characters. The first interface uses an 'action palette' and targets sports prototyping applications. When used online, the user selects from a palette of actions (e.g., stand, pike, extend) during an ongoing simulation. Actions are defined in terms of a set of target joint angles for PD controllers or as feedback-based balance controllers. When used offline, the timing of the key motion events can be adjusted manually or optimized automatically to produce desired outcomes. We demonstrate the action palette interface with simulations of platform diving, freestyle aerial ski jumps, and half-pipe snowboarding. The second interface explores the feasibility of using a game-pad to control a 13-link rigid body simulation of snowboarding for game applications. Unlike traditional video game play, the stunts accessible through our interface need not be preconceived by the game author and can emerge as the product of the physics, the terrain, and the player skill. We describe the control mapping and provide a mechanism to simplify balance control. We demonstrate the system using numerous snowboarding stunts.

Keywords: Character Animation, User Interfaces, Physics-based Simulation, Control

1 Introduction

Dynamic simulation 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 physics-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.

We propose two interfaces which can be used to interactively control motions for 3D multi-link rigid body simulations of aerial motions such as diving, ski jumping, and snowboarding. Figure 1 shows an example of the type of motions that can be rapidly constructed using the interfaces.

We refer to our first interface as an "action palette" because it defines motions in terms of a series of actions that can be selected at any time. These actions are represented on screen as a series of virtual buttons. The timing of a button press initiates the specified action, such as a pike position, while the exact location of the button press is used to represent two additional parameters. These parameters help more precisely specify the desired action, such as the target angle for a pike and how quickly the character should try and reach that target pike angle. The interface can be used on-line or off-line.

*email: pzhao,van@cs.ubc.ca

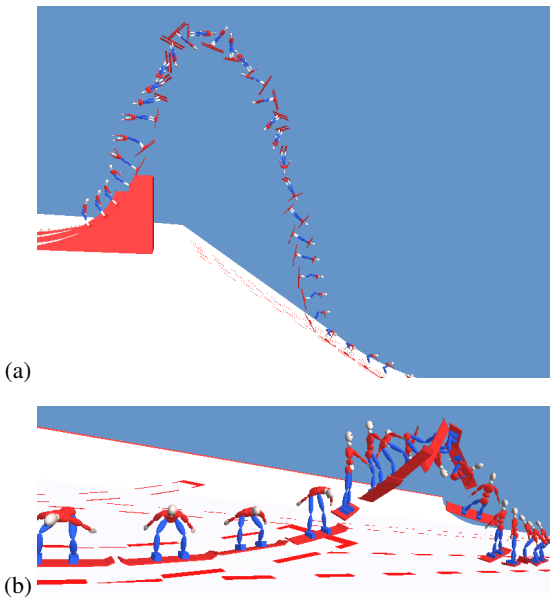


Figure 1: (a) Aerial ski jump performed using the action palette interface; (b) Snowboard stunt performed using the gamepad interface.

A second interface looks specifically at how a game-pad controller can be used to provide interactive control over a fully dynamic articulated-figure snowboarding simulation. With an appropriate interface, can one learn how to perform interesting snowboarding stunts with a simulator that isn't specifically tuned for game play but rather one that is intended to be physically-realistic?

Our principal contributions are twofold. First we present two novel interfaces for interactively exploring the space of physically-realizable motions for 3D characters, looking particularly at the design of motions with a significant aerial component. One interface targets motion prototyping while the other targets game applications. Second, we are among the first to present fully dynamic simulations of aerial ski jumping and snowboard stunts, both of which have educational and entertainment applications.

The remainder of this paper is structured as follows. Section 2 reviews related work. Section 3 describes the action palette interface and its use in simulating platform diving, ski jumping, and snowboarding motions. Section 4 presents a gamepad interface for a physically-realistic snowboarding simulation and discusses the relevant simulation and game-play issues. Finally, section 5 presents conclusion and future work.

2 Previous Work

The use of 3D dynamic simulation for modeling human motion can be traced back to the late 1980's, a good early example in computer graphics being the work of Forsey and Williams[1988]. The

technique of dynamic simulation has continued to develop as a tool for animating human motions. A long term goal in this area is to produce autonomous characters with specific skill sets with applications to both animation[Faloutsos et al. 2003] and entertainment robotics.

Simulating human motion in sports is an ongoing endaevor in both biomechanics and computer graphics. To limit our scope, we shall primarily mention the relevant computer animation work here. Hodgins et al.[1995] simulate running, bicycling, and handspring vaults. Yang et al.[2004] look at interactive dynamic control of swimming. Previous work in both biomechanics and graphics has looked at the offline planning of acrobatic motions[Yeadon 1990; Yeadon 1997; Huang and van de Panne 1996; Liu and Popovic 2002; Fang and Pollard 2003].

More closely related to our work, Wootens and Hodgins[1996] simulated three 10m platform dives using a state machine model. While we use similar types of motion control primitives, our work differs in several respects. Instead of employing off-line manual tuning, we develop interactive interfaces. As a result, we are able to rapidly author new motions, as demonstrated by the set of 46 motions across 3 sports that we have authored to date using our action palette interface. To our knowledge, we provide the first fully dynamic articulated figure simulations of aerial ski jumping and snowboarding stunts. Lastly, we explore the potential of using gamepad controllers for the control of fully-dynamic snowboard stunt simulations.

Previous work on the interactive control of physically-based character animation[Troy and Vanderploeg 1995; Laszlo et al. 2000] is also closely related. Mouse movements or keystrokes are mapped to control parameters in order to interactively guide the motion of planar character models. The interfaces we present differ in that we allow control over more parameters, are demonstrated to control 3D character motion, and examine gamepad-based control. The timeline component of our interface further supports refinement of timing parameters. Oore et al.[2002] use hybrid kinematic/dynamic simulations for expressive leg and arm motions.

Lastly, video games such as SSX[Electronic Arts. 2003] and Amped[Microsoft. 2003] (both snowboarding games) employ some use of physics in determining the character motion and provide the user with well-tested gamepad-based interfaces. The physics models used in these games are not publically documented, although it is reasonable to speculate that the snowboarder is simulated as a single body rather than a fully dynamic simulation of a multi-link articulated figure. One of our contributions is thus to examine the feasibility of using multi-link rigid body dynamics to animate characters in sports game applications, beyond their more conventional use for rag-doll effects. Our interfaces and underlying articulated figure simulation allow for the prototyping of new stunts rather than having these be predefined during the game authoring process. Our action palette interface provides much finer control than can be achieved using a gamepad interface.

3 Action Palette Interface

The first interface we present is shown in Figure 2 and is intended for motion prototyping applications. In the following description, we motivate the structure of the interface, describe the authoring process, and then illustrate results obtained with this interface.

3.1 Motion Stages

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, aerial position, and extension. A user can create the complete diving motion through specifying the proper timing and execution of all these stages. The simplest of the motions we work with has 4 stages, while the most complex has 8 stages. We shall use the words *stage* and *action* interchangeably.

Our interface consists of a panel with a family of virtual buttons, one for each possible action, as shown in Figure 2. Each button defines an action and has two associated parameters which can be directly specified by the relative location of the user mouse-click or stylus-tap within the virtual button. The meanings of the two parameters vary according to the action. A summary of the set of buttons used for the platform diving, acrobatic ski jumping, and snowboad control is given in Tables 1, 2, and 3.

Our system uses proportional-derivative (PD) controllers to drive each joint toward a target position. Actions consist of either controllers or target poses. Controller actions are used for maintaining balance for stages that preceed and follow the aerial portions of a motion. These operate by looking at a projection of the center of mass onto the ground plane and determining the error of this projection with respect to a target center-of-mass projection. The target angles of the ankles are then set as a linear function of this error.

Linear interpolation is used between target-angle 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.

3.2 Authoring

Our system supports both online and offline authoring of aerial motions. The authoring process begins with an online attempt at authoring the desired motion. The user initiates an interactive simulation which may run in real-time, it can also be useful to run the simulation in slow motion, thereby giving more time to think about the desired sequence of actions. The user authors the simulated animation with appropriately-timed selection of the desired actions. A record of the selected actions appears on the timeline at the bottom of the inteface panel.

The simplest form of editing consists of retiming the existing events, which is supported using a *retiming* simulation. The user acts out the motion using a sequence of timed spacebar presses. Each spacebar press then specifies the revised timing of the next event, with the events always preserving their original ordering. This is a useful mode to quickly explore multiple variations of the same motion being executed with slightly different timings.

Offline adjustments are useful for producing a motion through iterative refinement. The timing of each event can be adjusted directly on the timeline. Selecting an event marker on the timeline allows for the associated action parameters to be altered, as specified by the selection point within the associated virtual button. In Figure 2, The red cross-hairs drawn on the *pike* button illustrate the current parameter settings associated with a selected pike action. These parameters can be adjusted with a mouse press at the desired location within the button. Once a set of desired adjustments has been made, the user resimulates in order to observe the newly refined motion.

To further speed the creation of successful aerial motions, our system can automatically optimize the timing of a specified input to achieve the best outcome. This requires a cost function to evaluate the quality of a motion as well as a means for optimizing this cost as a function of the timing parameters. Optimization methods that exploit derivative information will generally require the fewest function evaluations and thus these are used by spacetime-constraint methods for related trajectory optimization problems[Popovic et al. 2000; Liu and Popovic 2002; Fang and Pollard 2003]. However, this assumes differentiable optimization metrics, which may be difficult to achieve in the face of non-linear effects such as joint limits

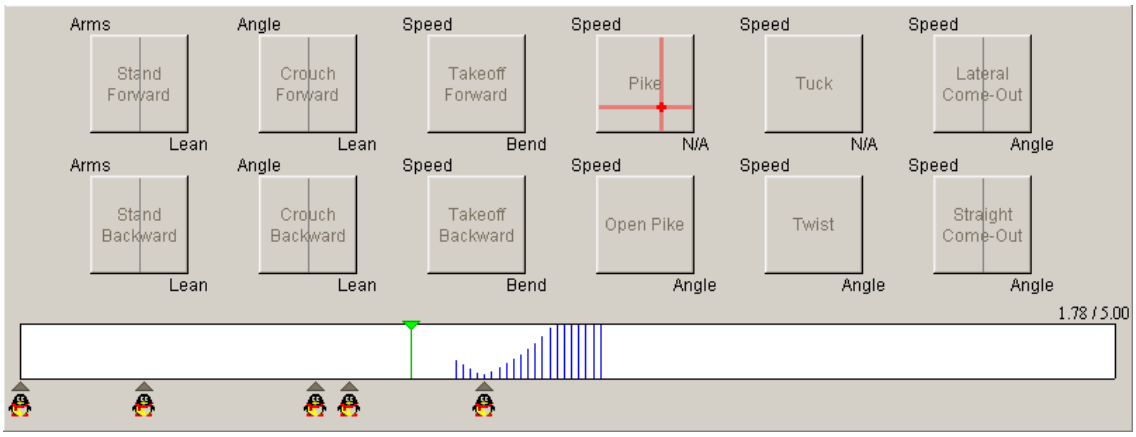


Figure 2: Interface panel for platform diving. The virtual buttons provides access to a set 12 actions. Each action has a 2-dimensional parameterization, with the action parameters being set according to the (x, y) coordinates of the press within the button frame. The bottom of the panel contains a time line. Each button press leaves an event marker on the timeline – there are five of these shown in this example. The series of blue line segments in the timeline are a visualization of an optimization cost function computed while varying the timing of the last event of a dive (see text).

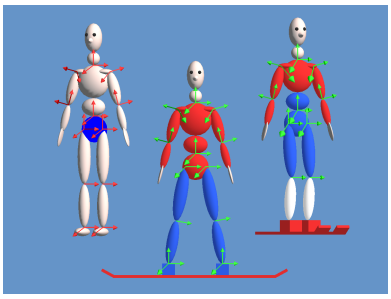


Figure 3: Degrees of freedom used for diving, snowboarding, and skiing models.

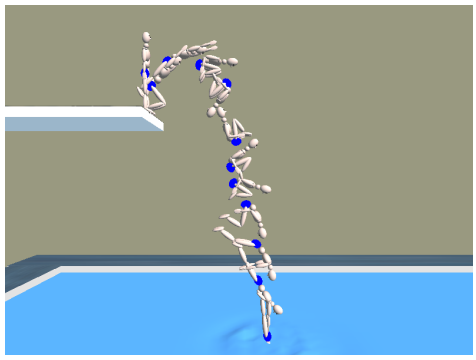


Figure 4: A $5m$ platform dive.

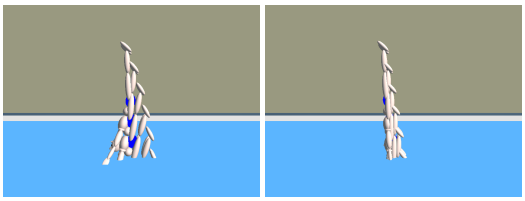


Figure 5: Dive entry before and after optimization of the time for start of extension.

or arbitrarily defined cost functions. We treat the optimization as a parameter search process and discretely sample the timing parameter in question at regular intervals that bracket its current value. Cost function evaluations are then carried out using a dynamic simulation from that point in time forward to the time where the metric is evaluated. The previous motion is used to start each of these simulations from the appropriate dynamic state.

3.3 Implementation

We have implemented an action palette interface for platform diving, freestyle aerial ski jumps, and half-pipe snowboarding. This system makes use of the publicly-available Open Dynamics Engine[Smith] for computing and integrating the equations of motion. The character models are based on the anthropometric parameters used in [Wooten and Hodgins 1996] and are structured as shown in Figure 3. The diver model has 13 links and 32 degrees of freedom (DOF). The snowboarder is the same but with an extra DOF for each ankle. Lastly, the skier has fixed ankles, giving 13 links and 28 DOF. Without graphical display, the diving simulation can compute 3.7 animation seconds in 1 wall-clock second. For skiing and snowboarding, the simulation speed numbers are 1.68 and 1.69 respectively. The diving simulation requires significantly fewer collision detection computations, and is faster as a result. These performance numbers are for a 2.66 GHz P4 PC.

3.4 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 aerial ski jumps and snowboarding stunts. 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 virtual buttons for the control of all possible actions during diving and is shown in Figure 2. Table 1 describes the actions and x and y parameters associated with each button.

We have used the interface to author 29 types of $5m$ platform dives[O'Brien 2003], as demonstrated in the video that accompanies this paper. In diving terminology, we have authored dive types 101a, 101b, 101c, 103b, 103c, 105b, 107c, 201a, 201b, 201c, 203b,

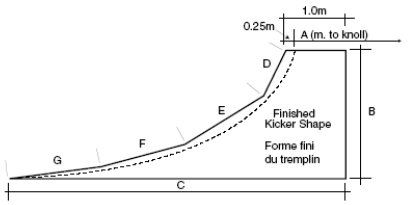
203c, 205c, 301a, 301b, 301c, 303b, 303c, 305c, 401a, 401b, 401c, 403b, 403c, 405b, 405c, 407c, 5132d. Two of them are shown in the top two rows of Figure 11.

An offline optimization can be used to automatically optimize the timing of the action events. We compute a cost function by integrating the error of the entry angle for the body, beginning at the time the diver first touches the water through to the time that the upper body is fully under water. The error of the entry angle 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 5 illustrates several frames from a diving motion before and after the optimization of the timing parameter associated with the extension immediately prior to water entry.

3.5 Aerial Freestyle Skiing

For skiing, we unite the lower leg, foot and ski into one 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. We decrease the stiffness of the knees and hips in preparation for landing in order to cushion the kinetic energy of the character when landing on the ground. 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 2 describes the definition of the virtual buttons we use to control the acrobatic ski actions. For the in-run action, one of the button parameters provides control over the starting position on the in-run, and hence provides control over the amount of speed accumulated at take-off.

The geometry of the 'kicker' jumps and ski-hill have been designed to match freestlye skiing aerial site specifications[CFSA]. Figure 6 shows the kicker specification. We have also experimented with simulating landings in water, as typical of summer training.



Nº	Kicker	A	B	C	D (1m)	E (2m)	F (2m)	G (2m)
1	Small Back	4.00	2.30	5.50	40°	17°	6°	N/A
2	Med. Back	6.70	3.25	5.95	50°	30°	13°	5°
3	Med. Back	7.00	3.30	5.90	52°	31°	13°	4°
4*	Big I	8.00	3.45	6.00	52°	33°	13°	4°
5*	Big II	8.10	3.60	6.70	55°	33°	14°	6°
6*	Big III	8.10	3.65	6.70	55°	33°	14°	6°

Figure 6: Geometric specifications for kicker jumps.

We have experimented with a series of 12 aerial ski jumps 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. Figure 1(a) and Figure 11(c)–(e) illustrate several of these jumps. We have also successfully applied the timing optimization for non-twisting jumps.

The creation of successful aerial ski jumps is more difficult than diving because of the difficulty of maintaining balance upon landing. In particular, we found it important to find the best time for extension in order to decrease the rotation and twist speed before landing. We note that highly skilled aerialists have the apparent capability to make mid-flight body pose adjustments in response to their perceived position and orientation relative to the ground. With our simulation, it is the user who must perform such adjustments, if any. The development of controllers that could correct for small errors that accumulate during a high-degree-of-difficulty jump remains an exciting area of future work.

3.6 Half Pipe Snowboarding

We have used the action palette interface to produce 4 types of half-pipe snowboarding stunts, including a back-360, back-360 with grab, a front grab, and a back grab. Figures 7, 11(f) and 11(g) illustrate three of these and Table 3 explains the virtual buttons for this action.

Half-pipe snowboarding was the most difficult motion to produce using the action palette system. The principal difficulty is that 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 half-pipe in the lead up to the jump in order to successfully initiate a jump. The terrain anticipation used by a real snowboarder to maintain balance during the rapid transition from horizontal to the near-vertical edge of the half-pipe is not implemented in our system, making this challenging to properly execute.

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.

4 Gamepad Interface

Our second class of interface uses gamepads for the control of multi-link rigid body character simulations with game scenarios in mind. We have experimented with gamepad interfaces for platform diving, aerial ski jumping, and snowboarding. The description below is restricted to the snowboarding interface because it is the one with which we currently have the most experience.

Excluding the use of rag-doll simulators, our interface is, to our knowledge, one the first explorations of a gamepad-based control interface for a dynamically-simulated multi-link articulated figure for real-time sports game applications. Snowboarding games such as *SSX*[Electronic Arts. 2003] and *Amped*[Microsoft. 2003] are an obvious point of comparision for our interface. The exact extent of the physics model used in these games is not publicly documented. However, we speculate that these games have physics models that primarily treat the character's motion in terms of a single rigid body rather than the physics of a multi-link articulated figure.

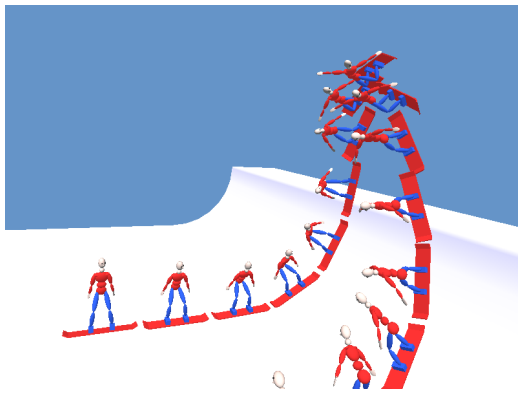


Figure 7: A simulated snowboard 'back grab' jump on a quarter-pipe.

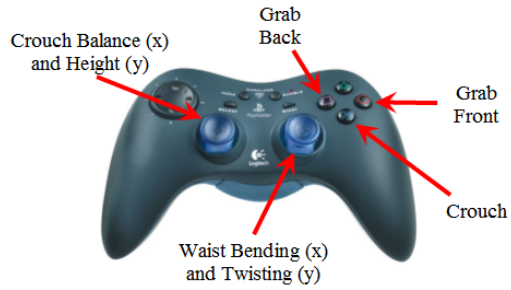


Figure 8: Gamepad interface for snowboarding control. The joysticks are used to control crouch balance and waist angles. Grab actions are mapped to buttons; multiple-presses result in faster grab actions.

Another key difference of our work with respect to existing game interfaces is that the set of possible stunts in a game is largely decided in advance by the game creators. For our interface, the possible stunts are constrained only by the articulated-character physics, the terrain, and the players skill. This is clearly a double-edged sword, however. The constrained set of possible motions in existing game interfaces greatly simplifies the control required of the user. The detailed physical simulation we use makes the control of balance an issue, even during relatively simple gliding and steering motions. We address this with the use of stabilization, as will be described shortly.

4.1 Control Mapping and Simulation Details

Figure 8 shows the mapping of the gamepad controls that we employ. The interface operates by setting a desired pose for the character. With one exception, this is expressed in terms of a set of target angles for PD-controllers. The x-axis of the left joystick controls the balance on the snowboard by setting a target point for the projection of the center of mass onto the plane of the snowboard. A feedback controller drives the pitch of the ankle joints so as to drive the character's center of mass to lie over this target point. Because the snowboard is underactuated with respect to pitching about it's principal axis, the desired balance target may not always be achievable in practise. The y-axis controls the height of the crouch and can thus be used to first anticipate and then initiate a vertical jump. This operates by altering the target joint angles for the hips, knees, and ankles.

The right joystick controls the waist bend and twist. This is used to initiate twisting jumps as well as to turn while gliding. A fast turn of the snowboard requires a slight unweighting of the snowboard (accomplished with a rapid, moderate crouch) and accompanying this with a waist twist. The rest of the interface consists of 3 buttons corresponding to back and front grabs, and a third button for returning to crouch mode. Crouching in the air will decrease the inertia in order to perform effective somersaults, while minimal inertia for spins is achieved with the body fully extended. The motion of the arms is coordinated with the crouching action – for an upright position folds the arms down beside the body, while a crouched position is coordinated with the arms extending laterally.

Proper simulation of the interaction of the snowboard with the snow is necessary to achieve a realistic simulated behavior. With the board placed on edge, we assign a high coefficient of friction to resist lateral motion, as well as modeling a modest amount of side-cut, which causes a moderate turning behavior to occur. With the board placed flat, the board can slide laterally, but with more overall friction than for sliding along the principal axis. As in real life, sliding laterally and then "catching an edge" will lead to a fall.

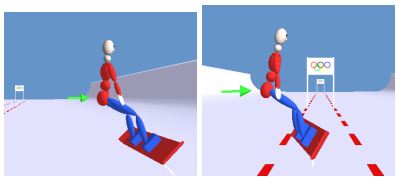


Figure 9: An external force (shown in green) is applied to aid with balancing during regular gliding and turns.

4.2 Stabilization

During regular gliding and turning motions, we apply an external force to the body in order to simplify maintaining balance. The force is computed using a PD-controller applied to the body lean angle and is applied to the character pelvis. The lean angle is defined as the angle between an *up*-vector and the vertical plane that embeds both of the ankle joints. The up-vector is defined by a line passing from the midpoint of the left and right ankles through to the character's center of mass. The magnitude of the external balancing force is limited so that the character can still fall if sufficiently unbalanced. The external force is only applied when part of the snowboard is in contact with the ground. Because the force acts in a plane orthogonal to the principal axis of the snowboard, it does not affect the forward sliding motion of the character.

Figure 9 shows two illustrations of the external force in operation. In practise we find that this simple strategy works well as a means to simplify balance control, while still allowing for the character to fall if badly off balance.

4.3 Results

We have experimented with our gamepad snowboarding interface on three types of terrain: a shallow slope with a series of flags to steer around (see Figure 10); a shallow slope with a series of small jump ramps (see Figure 1) and a shallow slope leading to a drop-off into a steeper slope (see Figure 10). All three of these terrains are bounded on their left and right edges by quarter-pipes similar to that shown in Figure 7.

Two example motions are illustrated in Figure 10. We note that our interactive gamepad control is more difficult to use than a typical game-pad interface for snowboarding games. This is not surprising, given that the physics-based simulation reflects more of the real difficulties of such a sport than the game versions. Our interface is an exploration of what can be done with a detailed physics-based simulation of a sport such as snowboarding, in contrast to simplified game physics. The anecdotal observations of users of our interface show that the challenge of experimenting with a detailed physics model is both appealing and addictive. This is also supported by previous experience with simulations such as

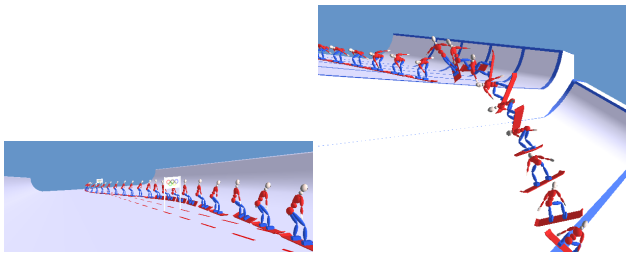


Figure 10: Example stunts performed using the gamepad interface. Please see the the submitted videos for animations of additional results.

Ski Stunt Simulator (SSS)[van de Panne and Lee 2003]. SSS is a Java-applet of a planar 4-link dynamically simulated skier whose actions (crouch/stand, forwards/backwards lean) are controlled interactively using a mouse and has averaged approximately 200,000 plays times per day over the past three years.

We speculate that our current gamepad interface and simulation-based animation is best used in a game as an additional game-play mode that supplements more traditional constrained game-play that is easier to learn and control. The dynamic simulation mode could be engaged or disengaged by a game player, or it could also be automatically engaged based upon the specifics of the terrain or the game level being played. In a hybrid mode, one could use traditional joystick navigation to reach interesting jumps and cliffs and then engage the full physics-based simulation and associated gamepad interface to perform stunts in a fashion that is constrained only by physics and skill of the player.

5 Conclusions

We have presented two interfaces for the interactive control of phyics-based 3D character simulations. The action palette interface is designed for prototyping dynamic aerial motions. 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. Questions such as "Is this new stunt possible?" or "What if the pike began earlier?" can be explored by using this system. The gamepad interface demonstrates the feasibility of designing sports games based on multi-link rigid body simulations for character motion. To our knowledge, this work is among the first to demonstrate detailed dynamic simulations of aerial ski jumping and snowboarding. This is in large part feasible because of particular interfaces we have developed.

The interfaces we present are exploratory in nature and have targetted motions with an aerial component. It is unclear that they would generalize well to expressive motions and motions where balance plays a continuous role. Human motion is an incredibly rich phenomena to model, however. Indeed, it is unlikely that any single motion authoring language or interface can succeed at modeling all human motions, nor that a single interface could support the very different requirements of animators, athletes, and game players. Our interfaces thus fill particular niches in this space.

There are several directions which require further research. We plan to investigate how controllers specific to the execution of twists can be constructed. Similarly, "automatic landing" controllers would be a useful addition to both types of interface. Our current automatic optimization tool uses a simple cost function and optimization technique. We wish to explore the use of other cost functions 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.

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 need to be incorporated. Lastly, we envision a scenario where an athlete might perform a dive whose motion is then reconstructed using a vision-based motion tracking system. The offline version of our action palette interface could then be used to hypothesize adjustments to the motion and to immediately view the simulated outcome of those adjustments.

References

CFSA. Canadian freestyle ski association, aerial site specifications. http://www.freestyleski.com.

ELECTRONIC ARTS. 2003. SSX 3. Game.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 2003. Composable controllers for physics-based character animation. In *SIGGRAPH 2001*, 251–260.

FANG, A. C., AND POLLARD, N. S. 2003. Efficient synthesis of physically valid human motion. *ACM Trans. Graph. 22*, 3, 417–426.

FORSEY, D., AND WILHELMS, J. 1988. Techniques for interactive manipulation of articulated bodies using dynamic analysis. In *Proceedings of Graphics Interface '88*, 8–15.

HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. In *Proceedings of SIGGRAPH 95*, 71–78.

HUANG, P. S., AND VAN DE PANNE, M. 1996. A search algorithm for planning dynamic motions. In *Eurographics Workshop on Computer Animation and Simulation*, 169–182.

LASZLO, J., VAN DE PANNE, M., AND FIUME, E. 2000. Interactive control for physically-based animation. In *SIGGRAPH 2000*, K. Akeley, Ed., 201–208.

LIU, C. K., AND POPOVIC, Z. 2002. Synthesis of complex dynamic character motion from simple animations. In *ACM SIGGRAPH 2002*, 408–416.

MICROSOFT. 2003. Amped 2. Game.

O'BRIEN, R. F. 2003. *Springboard & Platform Diving: a complete guided for divers and coaches*. Human Kinetics.

OORE, S., TERZOPOULOS, D., AND HINTON, G. 2002. Local physical models for interactive character animation. *Computer Graphics Forum 21*, 3, 337–346. ISSN 1067-7055.

POPOVIC, J., SEITZ, S. M., ERDMANN, M., POPOVIC, Z., AND WITKIN, A. 2000. Interactive manipulation of rigid body simulations. In *Proceedings of ACM SIGGRAPH*, 209–217.

SMITH, R. Open dynamics engine. http://q12.org/ode/.

TROY, J., AND VANDERPLOEG, M. 1995. Interactive simulation and control of planar biped walking devices. In *Workshop on Simulation and Interaction in Virtual Environments*, 220–224.

VAN DE PANNE, M., AND LEE, C. 2003. Ski stunt simulator: Experiments with interactive dynamics. In *Proceedings of the 14th Western Computer Graphics Symposium*.

WOOTEN, W. L., AND HODGINS, J. K. 1996. Animation of human diving. *Computer Graphics Forum 15*, 1, 3–14.

YANG, P., LASZLO, J., AND SINGH, K. 2004. Layered dynamic control for interactive character swimming. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

YEADON, M. R. 1990. The simulation of aerial movement- iv. a computer simulation model. *Journal of Biomechanics 23*, 1, 85–89.

YEADON, M. R. 1997. The biomechanics of the human in flight. *The American Journal of Sports Medicine 25*, 4, 575–580.

name	x	y	note
stand forward	target c.o.m. position	arm position	stand facing water
stand backward	target c.o.m. position	arm position	stand back to water
crouch forward	target c.o.m. position	height of crouch	crouch, raise arms
crouch backward	target c.o.m. position	height of crouch	crouch, lower arms
takeoff forward	waist bend	jump height	extend hips, knees, and ankles for jump
takeoff backward	waist bend	jump height	extend hips and ankles, swing arms back
pike	–	time to reach pose	
open pike	target pike angle	time to reach pose	
tuck	–	time to reach pose	
twist	–	time to reach pose	right hand to head, left hand to chest
lateral extend	waist bend	time to reach pose	bring arms laterally to over head
straight extend	waist bend	time to reach pose	bring arms straight over head

Table 1: Description of diving buttons.

name	x	y	note
start big	start position	angle of arm raise	start for the big kicker
start small	start position	angle of arm raise	start for the small kicker
crouch forward	hip bend	knee bend	bring arms to front for forward takeoff
crouch backward	hip bend	knee bend	swing arms down for backward takeoff
take-off forward	waist bend	time to reach pose	extend hips, knees, and ankles for jump
take-off backward	waist bend	time to reach pose	extend hips and ankles for jump, swing arms back
take-off twist forward	waist twist	time to reach pose	extend hips, knees, and ankles while twisting waist
take-off twist backward	waist twist	time to reach pose	as above, but swing arms back and apply less knee extension
pike	–	time to reach pose	
tuck	–	time to reach pose	
layout	waist bend	time to reach pose	extend arms
twist	–	time to reach pose	bring arms to body speed the twist
landing	hip bend	knee bend	alter stiffness of hips and knees for landing
finish	arm position	time to reach pose	stand up and raise arms

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

name	x	y	note
crouch left	waist twist	front/back balance	crouch down facing left
crouch right	waist twist	front/back balance	crouch down facing right
grab front	–	time to reach pose	grab front side of snowboard
grab back	–	time to reach pose	grab back side of snowboard
take-off	waist bend	time to reach pose	
landing	waist bend	lateral balance position	

Table 3: Description of buttons for snowboard control.

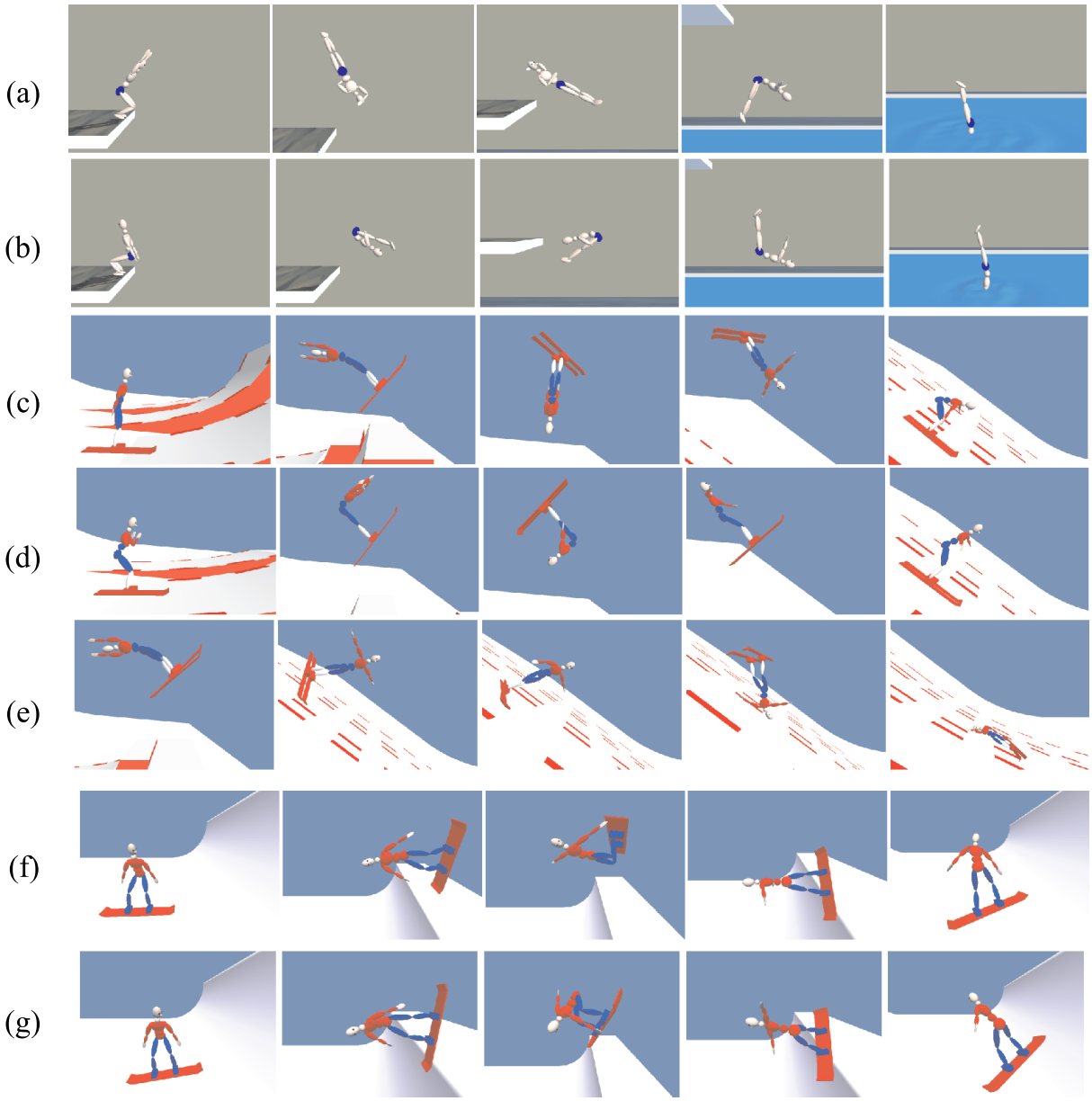


Figure 11: Results. Top to bottom: (a) single twisting front one-and-a-half dive; (b) back one-and-a-half pike; (c) back-full-full: double twisting double back flip; (d) front pike single; (e) bad landing for a back-full-full; (f) back-grab half-pipe jump; (g) front-grab half-pipe jump.