Accelerometer-based User Interfaces for the Control of a Physically Simulated Character

Takaaki Shiratori   Jessica K. Hodgins
Carnegie Mellon University

Abstract

In late 2006, Nintendo released a new game controller, the Wiimote, which included a three-axis accelerometer. Since then, a large variety of novel applications for these controllers have been developed by both independent and commercial developers. We add to this growing library with three performance interfaces that allow the user to control the motion of a dynamically simulated, animated character through the motion of his or her arms, wrists, or legs. For comparison, we also implement a traditional joystick/button interface. We assess these interfaces by having users test them on a set of tracks containing turns and pits. Two of the interfaces (legs and wrists) were judged to be more immersive and were better liked than the joystick/button interface by our subjects. All three of the Wiimote interfaces provided better control than the joystick interface based on an analysis of the failures seen during the user study.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Gaming

Keywords: character animation, physical simulation, user interface, Wiimotes

1 Introduction

Mass market handheld controllers for interactive experiences and video games had remained largely unchanged for a number of years until the Wiimote controller was introduced in late 2006 [Nintendo 2006a]. With a three-axis accelerometer, distance sensing (via the Sensor Bar) and a simple BlueTooth interface, the Wiimote controller offered developers, both commercial and independent, the opportunity to create many different types of novel interfaces. They responded quickly to this challenge by using the Wiimote to create a head tracker for desktop VR [Lee 2008], an industrial robot that swings a tennis racket [USMechantronics 2008], gesture-based controllers for a humanoid robot [Android Technologies, Inc. 2008] and a music synthesizer [Youtube 2008]. These interfaces are novel and have the potential to be pervasive because they are constructed using cheap, readily available technology.

In this paper, we explore the utility of the acceleration sensing provided by the Wiimote for the control of physically simulated characters. We have designed three interfaces that use the signals from the three-axis accelerometers in two or three Wiimotes for the control of a dynamically simulated creature that can walk, run, jump, and turn (Figure 1). These are performance interfaces in that they require users to imitate motions seen in human walking, running and jumping. Because the resulting animation is physically correct and resembles the locomotion patterns we see in daily life, we believe that imitating those patterns in the interfaces may be immersive. In the first interface, the user moves his wrists rapidly to specify the gait and speed of the character. Small, out-of-phase motions command a walk, larger, faster out-of-phase motions indicate running, while in-phase motions specify a jump with the magnitude controlling the height. In the second interface, the user moves his arms back and forth as if he were walking (straight arms), running (bent arms) or jumping (bent arms moving in phase). The third interface allows the user to closely mimic the motions desired of the character. Wiimotes are attached to the lower legs of the user and the frequency and phase of the user's motion determine whether the character walks, runs, or jumps. The inclination of the head is used for turning. The final interface is similar to that used in many video games and was created to serve as a baseline comparison. It uses the joystick on the Nunchuck and a button on a Wiimote to command a jump.

E-mail: {siratori, jkh}@cs.cmu.edu

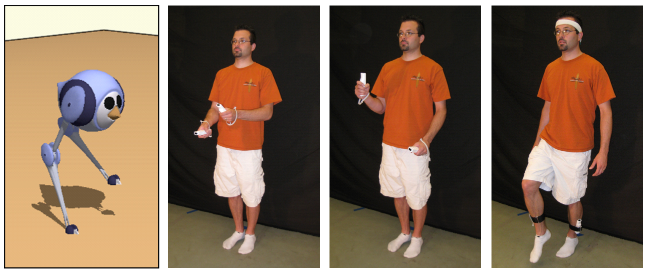


Figure 1: *A user controlling a physically simulated bird character with the three Wiimote interfaces: wrists, arms, and legs.*

We chose to use a physically simulated character in our experiments because such characters offer the potential for natural responses to rough terrain and other disturbances. Physically simulated characters are generally regarded as more difficult to control with a traditional joystick interface because the response to a commanded action cannot occur instantaneously (the character has to plant the correct foot in order to turn or push off the ground to generate the vertical velocity for a jump, for example). We are interested in exploring whether performance-based interfaces such as those that we have constructed with the Wiimote mitigate this problem.

Beyond just extending the space of possible interface designs for the Wiimote, we had two goals with this research:

- explore different ways in which the Wiimotes can be used for character control: what human actions can be measured reliably with a set of three-axis accelerometers and how can those measurements be mapped onto the control inputs of a character?
- explore whether the latency to a commanded change in be-

havior that is inherent in a dynamically simulated character degrades the user's control when the interface allows the user to move his or her limbs in a pattern that is reminiscent of the behavior being controlled. Is the mapping from user actions to character actions sufficiently strong that needed changes in behavior can be anticipated?

We performed an assessment of user performance with each interface on a set of test tracks containing turns and pits. We measured time to completion, number of failures, and the answers to a set of survey questions. Two of the interfaces (legs and wrists) were judged to be more immersive and were better liked than the joystick/button interface by our subjects. An analysis of the failures seen during the user study showed that all three of the Wiimote interfaces provided better control than the joystick interface.

In the next section, we discuss the most closely related work in interfaces for on-line control of characters and techniques for dynamically simulating the motion of characters. The following sections briefly discuss the hardware of the Wiimote, the physical simulation of the character, and then explain the measurement and control techniques behind the three interfaces. We then describe the results of our user study and an application that shows the potential advantages of a physically simulated character.

2 Prior Work

In this section, we discuss prior work on interactive control of animated characters and physical simulations of animated characters.

2.1 Interactive Control of Animated Characters

There are many examples of both research and commercial games that have explored input devices that might allow for good control of an animated character. The control problem is not easy because, in general, the animated character will have many more degrees of freedom than can be specified directly.

One common input device is the mouse. Laszlo and colleagues [2000] developed an interactive system to control physically simulated planar characters by mapping mouse movements to joint torque or to desired values for a PD servo. Users were able to control jumping robots as well as a planar human character that performed a ski jump [van de Panne and Lee 2003]. Neff and colleagues [2007] used correlation maps to allow the user to control multiple joints simultaneously. Igarashi and colleagues [2005] proposed a spatial keyframing technique. Unlike traditional keyframe animation in which keyframes are specified at particular moments in time, their users set up a correspondence between a 3D position and the character's posture. The user then dragged a 3D marker to create a performance by interpolating among the corresponding character poses.

Tablets have also been used to control animated characters. Thorne and colleagues [2004] used these devices to design a character and to animate it by drawing pre-defined strokes. The system had pre-defined motion primitives which were modified by properties such as the speed of the stroke. Oshita and colleagues [2005] built a pen-based interface for animation control. They used pen strokes to specify the trajectory of a character, pressure for a jumping/squatting motion, and tilt for body attitude.

Capturing the user's motion via a single camera is also an interesting approach. Freeman and colleagues [1998] used computer vision techniques such as image moment and optical flow to allow the user to interact with a virtual character, or to directly map the user's motion to the character's motion. Sony produced the EyeToy [2003] to take advantages of this class of approaches for commercial games.

A large database of motion capture data can be used as domain knowledge to allow the control of more complicated motions with low degree-of-freedom input devices. Motion graph techniques have been controlled by sketching, keystrokes and performance interfaces [Arikan and Forsyth 2002; Kovar et al. 2002; Lee et al. 2002]. Chai and Hodgins [2005] built a vision-based system to produce an animation of a human character based on two video streams and a sparse marker set worn by the user. Their system learned local models of segments of pre-recorded motion capture data that were close neighbors to the input markers. An optimization in the space of that local model was able to reconstruct the user's motion with reasonable accuracy.

Interesting systems have also been developed using unique devices. Johnson and colleagues [1999] introduced the concept of a *sympathetic interface* that used a plush toy with sensors such as an accelerometer and gyroscope inside, and synthesized animation by recognizing gestures from the sensor data. Oore and colleagues [2002] developed motion layering using magnetic sensors to determine the orientation of the input device. Dontcheva and colleagues [2003] proposed a layering method that used widgets whose location and orientation was known from real-time motion capture. Yin and colleagues [2003] developed the FootSee system that measured pressure between the feet and the ground, and used it as a query to reconstruct the user's motion based on a pre-existing database. Nintendo has recently developed the Wii Fit [Nintendo 2008b], which recognizes some elements of the user's motion based on force measurements. Slyper and Hodgins [2008] demonstrated that upper body motions could be reconstructed with input from five three-axis accelerometers and a database of appropriate motion capture data. In contrast, the commercial motion capture systems based on accelerometer technology use inertial measurement units (IMUs) that contain gyroscopes and magnetometers to allow the accurate reconstruction of orientation [Xsens Technologies].

Locomotion interfaces for virtual navigation have been developed using 6DOF magnetic sensors [Templeman et al. 1999], IMU trackers [Razzaque et al. 2002], or using a special pad that detects foot contacts [Bouguila et al. 2004]. Usoh and colleagues [1999] evaluated virtual navigation systems for *Walking* (real walking), *Walking-in-place*, and *Flying* (face direction-based control), and learned that walking was the best for human-scale spaces. These systems do not reconstruct the full motion of the character but instead, like our system, determine how it should locomote in the virtual environment.

Many companies are exploring novel uses for the Wiimote. Wii Sports [Nintendo 2006b] includes sports games such as boxing, bowling, and tennis. The human motion of the character is specified when the user performs an action such as swinging a tennis racket. Although the details of the input processing algorithms are not publicly available, it is our understanding that most of the existing Wii games either obtain static measurements by inclining the Wiimotes or dynamic measurements from detecting local min/max or zero-crossings of acceleration. The inclination interface is generally used to control the direction or the speed of the character (see, for example the mini games of Mario Party 8 [Nintendo 2007b] or Mario Kart Wii [Nintendo 2008a]). Dynamic measurements are generally used just to control the speed or to specify a jump with the Nunchuck without allowing the users to control the direction/orientation of the character (see, for example Donkey Kong Barrel Blast [Nintendo 2007a] and Rayman Raving Rabbids 2 [Ubisoft 2007]). Our interfaces provide a finer level of control over the locomotion patterns of our dynamically simulated character.

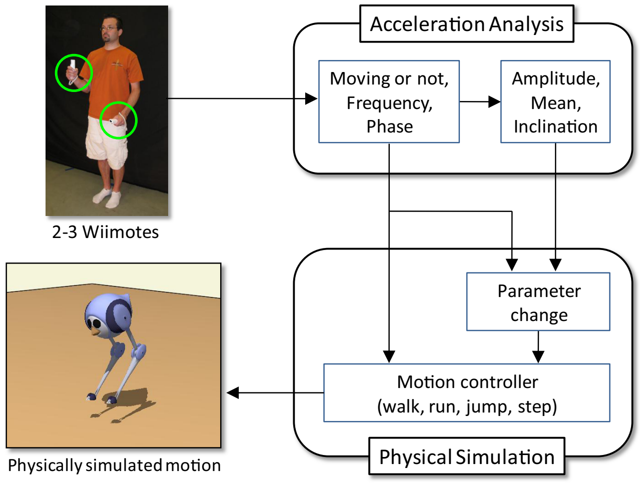

2-3 Wiimotes
Acceleration Analysis
Moving or not, Frequency, Phase
Amplitude, Mean, Inclination
Parameter change
Motion controller (walk, run, jump, step)
Physical Simulation
Physically simulated motion

Figure 2: *Overview of control for a biped character using the Wiimotes.*

2.2 Physical Simulation of Animated Motion

Controllers for the running of a two-legged character were first developed by Raibert and colleagues [Raibert 1986; Raibert and Hodgins 1991; Hodgins 1991]. Their basic idea was to construct a finite state machine (FSM) for each gait based on the contact states seen in that gait. Each state then contained control laws for each joint of the system. This strategy was first applied to a running human character by Hodgins and colleagues [1995]. SIMBICON developed by Yin and colleagues [2007] was motivated by this earlier work. They realized a simple controller to synthesize robust, physically valid motion of a human character. FSM-based controllers for biped gaits have proven to be very robust and ours follows a similar design to that used in this earlier work.

3 Approach

We chose to use an indirect mapping from the motion of the Wiimotes to the control inputs for the simulated character. The interface measures the frequency of the user's motion, the phase difference between two Wiimotes, and in some cases the inclination of the Wiimote or the magnitude of the oscillation. These actions are then used to select the gait and the velocity and turning rate with which it should be performed (Figure 2). These commands are implemented via a control system which is similar in spirit to that presented by Raibert and Hodgins [1991]. We now briefly describe the hardware in the Wiimote and the physical simulation of the character before describing the input that we obtain from the Wiimote and how that information is used to control the character's locomotion pattern.

4 Wiimote

The Wiimote has eight buttons, a three-axis accelerometer, and an IR receiver to measure distance from the Sensor Bar. The attachable Nunchuck has a continuous directional controller (joystick) and also a three-axis accelerometer for input. For output, the Wiimote has a speaker, a vibration generator and four LEDs. A connection between the Wiimote and the host computer is established via BlueTooth. The price per unit is around \$40, and approximately 30 million copies have been sold worldwide (as of June 2008).

Our interfaces rely on the three-axis accelerometer in each Wiimote. The x-, y-, and z-axes of the accelerometer point toward the left, front, and downwards (Figure 3). The accelerometer can capture the net force acting on the Wiimote between $-3g$ to $3g$, where g is gravitational acceleration. The three acceleration values are quantized to 8 bits.

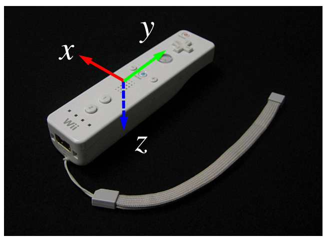
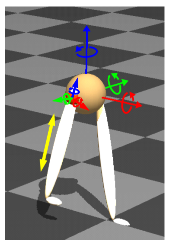


Figure 3: *The Wiimote. Arrows show local coordinates of the three-axis accelerometer.*



Segment	Mass [kg]	Moment of inertia [kgm^2]
body	9	$\begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1.2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
hip	1	$\begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.001 \end{bmatrix}$
knee	1	$\begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.001 \end{bmatrix}$

Figure 4: *Our hopper model has fourteen DOFs. The knees telescope rather than bending.*

5 Physical Control of a Biped Character

The three-dimensional (3D) robot model for the physical simulation (Figure 4) consists of a body that is free to rotate or translate (six uncontrolled DOFs), two hip joints with three DOFs, and two telescoping joints with one translational DOF. Unlike the standard biological kinematic structure with rotary knees, this model has telescoping knees. The character's leg length is about 1 m so it is similar in size to a human and therefore has a similar running and walking pace [McMahon 1984].

The biped control system consists of four gaits and the transitions between them: stepping in place (stopping), walking, running, and jumping with both legs. We control these motions with finite state machines (FSM) which determine the desired angle for each controlled joint for each state in each gait. Force or torque is then applied to each joint using a proportional-derivative (PD) controller $\tau = k_p(\theta_d - \theta) - k_d\dot{\theta}$, where θ_d is the desired joint position and θ and $\dot{\theta}$ are the current joint angle or joint length and angular or linear velocity respectively. The gains for the PD servos are tuned by hand initially and then fine tuned via optimization once a working controller is obtained.

Walking Controller: As shown in Figure 5, the walking controller has three states for each leg: *Double Support*, *Rise* (the first half of swing/single support), and *Fall* (the second half of swing). The transitions between states occur based on changes in contact or the movement of the rear leg.

During Rise, the controller applies a torque about the hip joint of the supporting leg to control the pitch (nose up and down) and roll (side-to-side) attitude of the body. The desired angle, ϕ_d^W for pitch

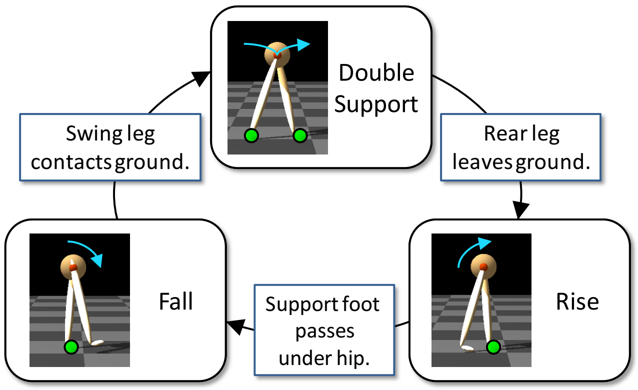


Figure 5: *State machine for walking.*

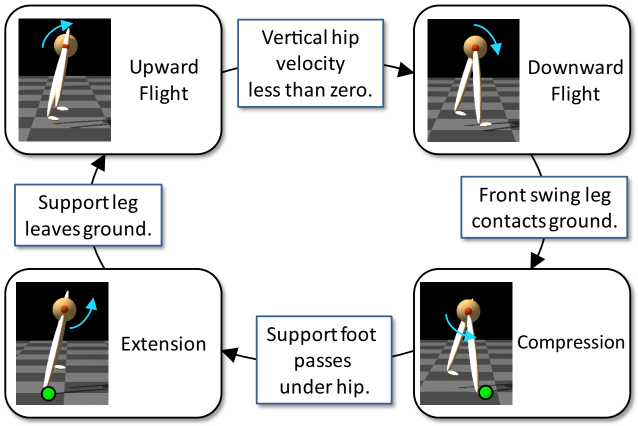

Upward Flight
Vertical hip velocity less than zero.
Downward Flight
Support leg leaves ground.
Front swing leg contacts ground.
Extension
Support foot passes under hip.
Compression

Figure 6: *State machine for running.*

and roll in walking is calculated from the current body velocity v:

$$\phi_d^W = \phi_o + k^W(v_d - v), \tag{1}$$

where v_d is the desired body velocity, ϕ_o is an offset angle and k^W is a feedback coefficient for walking. The rear leg is swung forward to reach ϕ_d during this state and held short to clear the ground. The stance leg tries to maintain the touchdown length.

In Fall, the supporting leg begins to extend to restore forward velocity lost during the stance phase. The body attitude is controlled as in the Rise state. The swing leg is extended to make contact with the ground and the hip is positioned for touchdown.

In Double Support, both of the hip joints apply torques to control the body attitude. The front supporting leg maintains its touchdown length while the rear supporting leg extends to add energy.

Stepping Controller: The controller for stepping in place is only a minor modification of the walking controller. Because the body is not moving forward, the rear leg is explicitly picked up off the ground after a certain amount of time has elapsed in double support rather than being pulled up off the ground by the forward movement of the body. The transition from Rise to Fall occurs when the foot of the swing leg reaches a specified height and the sideways velocity of the hip is greater than a threshold.

Running Controller: The running controller has four states for each leg: *Upward Flight*, *Downward Flight*, *Compression*, *Extension* (Figure 6). The transitions between states occur based on changes in contact, the vertical velocity of the body, or the movement of the stance leg as shown in Figure 6.

During Upward Flight, the controller positions the front leg for touchdown and shortens the rear leg that has just left the ground. The desired angle ϕ_d^R for pitch and roll in running is calculated as:

$$\phi_d^R = \frac{vT}{2} + k^R(v_d - v), \tag{2}$$

where T is the predicted duration of the next support period and k^R is a feedback coefficient for running. During Downward Flight, the front leg is lengthened for landing. The rear leg mirrors the motion of the front leg to reduce disturbances to the body. Compression and Extension both use the stance leg to maintain the body attitude (using the same equation as in walking). In Extension the stance leg is extended to add energy to the system. In both states, the idle leg maintains its length and orientation.

Jumping Controller: Jumping is similar to running except that now the two legs must operate in phase. A single copy of the state machine for running controls both legs in jumping. If one leg reaches the ground before the other, the second leg is extended to make contact. Similarly, if one leg leaves the ground first, the second leg is retracted to initiate flight. These control laws are an implementation of Raibert's concept of virtual legs [Raibert et al. 1986].

Optimization: The parameters for each physical controller are tuned through a gradient descent optimization. The objective function evaluates the time until failure, the length of the flight or swing and stance durations, the stride length, and the error in forward speed. The optimization tunes the spring and damper coefficients for the PD servos, the constant values used in the equation (1), and the landing length of the telescoping legs. Once parameter sets for slow and fast speeds of walking and running are obtained, parameters for intermediate speeds can be obtained by interpolating the parameter sets. This is similar to Yin and colleagues' work [2008].

Gait Transitions: Gait transitions occur when the user changes his or her command via the user interface and the model subsequently reaches a state where it can effect the gait transition. Changing from walking to running or jumping requires that the control system add energy to create a flight phase by extending the support leg during Fall. The controller is changed from walking to running or jumping when the support leg leaves the ground. This strategy is similar to Hodgins's work [1991]. The same is true for changing from running to jumping.

During running, the horizontal velocity is much faster than during walking and a gait transition from running to walking or stepping requires reducing the speed as well as eliminating the flight phase. This reduction in energy is accomplished by extending the support leg during Compression. The controller is also changed from running to the walking or stepping controller during that state.

Jumping has a slower forward velocity than running but energy must still be absorbed in a transition to walking or stepping. The support legs shorten to reduce the vertical velocity and keep the feet on the ground. The controller is then changed to Double Support of stepping. If the user commands walking, the controller is immediately changed from stepping to walking. This strategy is also similar to the approach used by Hodgins [1991]. There is no transition from jumping to running implemented.

Graphical Display: Because telescoping legs evoke a robot rather than a biological creature, we chose a creature with a kinematic structure for the legs that mimic those of a bird to display the simulated motion. The ankle joints on the character bend backward as

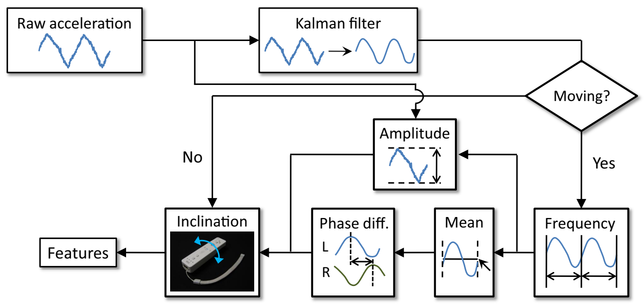


Figure 7: *Acceleration features extraction.*

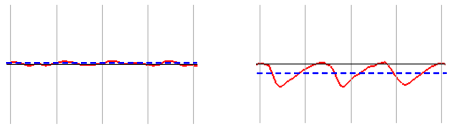


Figure 8: *x component of the Wiimote acceleration during vertical periodic movement without inclination (left) and with inclination (right). The horizontal blue dotted lines represent the mean of the x component during one cycle.*

they do on a bird and kinematically replace the telescoping joints that were dynamically simulated. The forces that would be exerted by such a structure do not match the actual forces applied to the body after this kinematic change but the motion remains physically plausible.

6 Wiimote Acceleration Analysis

We use a Kalman filter to reduce noise [Rasco 2008], and extract a set of features from two or three Wiimotes: 1) whether the Wiimote is moving or not, 2) frequency, 3) phase difference between two Wiimotes, 4) amplitude, and 5) direction of inclination. Figure 7 illustrates the process of extracting features from the acceleration data.

The first step of the analysis is to determine whether the Wiimote is moving. We make this determination by thresholding the variance of the recent acceleration measurements.

Because the user's basic action is swinging the Wiimotes with the arms or legs, the acceleration sequences are periodic and we can measure the frequency with an auto-correlation, R_a:

$$R_a(\tau) = \sum_{t=T-t_p}^{T} W(t)\,(\mathbf{a}(t)\cdot\mathbf{a}(t-\tau))\,, \tag{3}$$

where T is a current time, t_p is the period for auto-correlation, $\mathbf{a}(t)$ is the acceleration at time t and $W(t) = 1-(T-t)/t_p$ is a windowing function that reduces the effect of older data points. By picking the peak of the auto-correlation, we can estimate the frequency of the Wiimote movement.

Whether the two Wiimotes are in phase or not is also an important feature for our interfaces. We calculate the phase difference between two Wiimotes using cross-correlation, R_c:

$$R_c(\tau) = \sum_{t=T-t_p}^{T} W(t)\,((\mathbf{a}_L(t)-\bar{\mathbf{a}}_L)\cdot(\mathbf{a}_R(t-\tau)-\bar{\mathbf{a}}_R))\,, \tag{4}$$

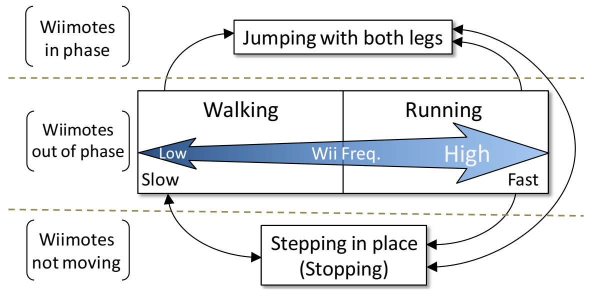


Figure 9: *Motion controller transitions with the Wiimotes. Arrows indicate a transition between controllers. During walking/running, the speed changes based on the frequency of the Wiimotes.*

where $\mathbf{a}_L$ and $\mathbf{a}_R$ denote the acceleration vectors of the Wiimotes on the left and right hands, and $\bar{\mathbf{a}}_L$ and $\bar{\mathbf{a}}_R$ denote the mean acceleration vectors in one cycle, respectively. Because the phase difference should be within $[0, 2\pi]$, we search for the peak of the cross-correlation using the inverse of the estimated frequency as the cross-correlation period.

We calculate the amplitude of the acceleration from one cycle. Spikes that are caused by rapid motion will be smoothed by the Kalman filter, and we therefore use raw acceleration data to compute the amplitude.

If an accelerometer is not moving, we can easily calculate the pitch and roll from the direction of gravity. For example, we calculate the roll angle θ_R of the Wiimote: $\theta_R = \mathrm{asin}(a_x/g)$ where a_x is the x component of the acceleration, and g is the gravitational acceleration represented in Wiimote coordinates.

If the Wiimote is moving, we cannot extract the exact orientation of the accelerometer because of inertia forces. However, as long as the motion is cyclic and the Wiimote is not moving too fast, we can estimate the inclination from the mean acceleration in one cycle. Figure 8 shows an example sequence of acceleration with/without inclination. While mean acceleration without inclination is nearly zero, mean acceleration with inclination is non-zero.

7 Mapping of Wiimote Features to Physical Simulation

Our goal in the design of these interfaces was to mimic the motions seen in human walking, running, and jumping. Our hope was that this correspondence would make users feel a connection between their actions and that of the character. For example, the motion of the feet in running is faster than in walking and the legs move out-of-phase in walking and running but in-phase for jumping. These observations led us to design the gait transitions based on frequency and phase of motion of the Wiimotes (Figure 9). When the Wiimotes are moving in phase, the character jumps. When the Wiimote motions are out-of-phase, the character either walks or runs depending on whether the frequency is low or high. Within a gait, the desired speed of locomotion is also determined by the frequency. If the Wiimotes are not moving, the stepping controller is used as a default behavior. The jump height is controlled by the amplitude of the acceleration. We use the inclination (lean) of the Wiimotes to control the rate of turning. The character changes direction with a constant increment of 5° for each step using a PD servo on the yaw of the body.

Through informal user studies, we obtained the comfortable range of actions (frequency, for example) for the interfaces. Similarly, the

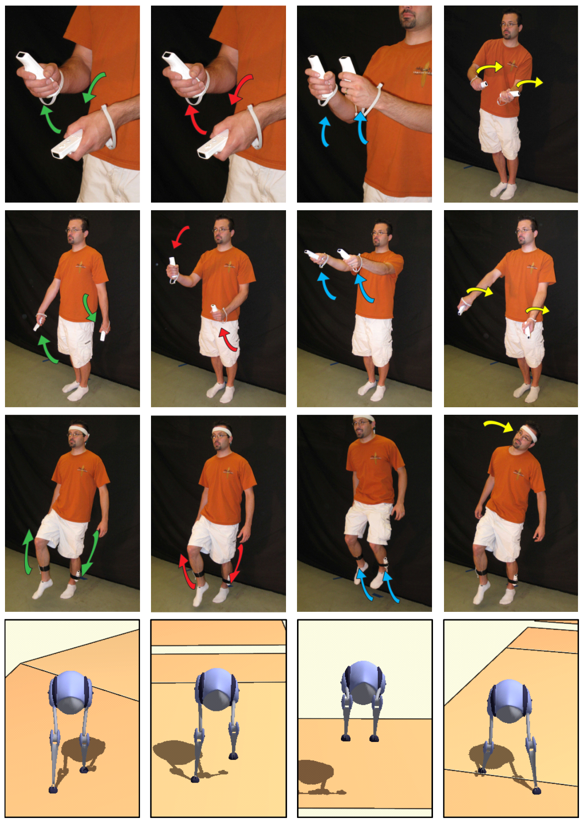


Figure 10: *The wrist, arm, and leg interfaces and an animated character (top to bottom). Each column shows the gesture of each interface: walking, running, jumping and turning (left to right).*

stable range of speeds and jumping heights were determined experimentally for the biped simulation. We then computed a set of constant scaling factors to map the comfortable range of the user interface to the stable range for the simulation. No further calibration of the Wiimotes or of the interfaces was necessary.

Wrist Interface: In the wrist interface, the user imitates the foot motions seen in locomotion by swinging two Wiimotes with his or her wrists. The motions used in this interface are illustrated in the first row of Figure 10. For walking and running, the user swings the Wiimotes out-of-phase. As the frequency increases, the interface commands first a higher desired walking speed, then a gait change from walking to running at 2 Hz and then a faster running speed. When the user swings the Wiimotes in phase, the character jumps. The user commands the character to turn by twisting the wrists and inclining the Wiimotes.

Arm Interface: In the arm interface, the user imitates the arm motion seen in walking and running (second row of Figure 10). For walking, the user swings his or her arms forward and backward while keeping the arms straight, much like the arm motions seen in natural walking. The arms are bent to indicate running and the transition between walking and running is based on a change in orientation of the swinging arms rather than frequency as was used in the wrist interface. For jumping, the user swings the arms in phase. For turning while walking, the user twists both arms, while continuing to swing them forward and backward. For turning while running, the user inclines the Wiimotes sideways and swings them diagonally.

Leg Interface: In the leg interface, the user attaches the Wiimotes to both legs with socks or wrist bands, and steps in place (the third row of Figure 10). Slow stepping commands the character to walk, and faster stepping commands a run. We use 2 Hz as the threshold for the gait transition as in the wrist interface. An additional Wiimote is attached to the user's head with a head band and used to control turning through head inclination. We experimented with other methods for specifying direction changes. We tried changing the foot height in stepping, but found that it created a different rhythm for each leg which did not seem natural. We tried attaching a Wiimote to the body, and inclining the body for direction changes but this also caused an asymmetry and also did not feel natural. Of these three options, head inclination was judged the most natural in our informal user studies.

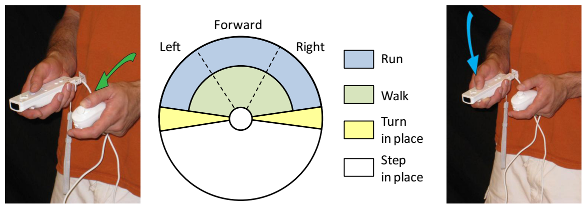


Figure 11: *The joystick interface. The half circle is used for controlling the character, and is divided into three regions to indicate turning left, right and proceeding straight. The radius of the area for walking is set to 0.6 (the maximum radius is 1.0). A button on the Wiimote is used to specify jumping (right).*

Joystick Interface: This interface is similar to a traditional game controller with running specified by a larger forward angle of the joystick and turning specified by pushing the joystick to the side (Figure 11). A button on the Wiimote is used to specify jumping.

8 Experimental Results

We used a process of iterative refinement and informal user studies to refine the design of the user interfaces. After the performance of all four interfaces was judged to be good in these informal experiments, we ran a full user study using tests of each specific behavior (walking straight and a gait change from walking to running, for example) and a set of four test tracks with curves and pits. To further demonstrate the power of these interfaces for controlling a simulated character, we constructed a competitive game in which one user attempted to trip up the simulation by constructing terrain with slopes, curves, and steps while a second user attempted to control the biped as it walked, ran, and jumped on the terrain immediately after it was constructed.

8.1 Results of User Study

We recruited fifteen volunteer subjects from the university community who were not associated with the research project. Seven of the fifteen subjects were studying computer science. Each of the fifteen subjects operated each user interface on a set of straight tracks and one of the four test tracks shown in Figure 12. The straight tracks contained a line and when the biped crossed that line, the subjects commanded it to jump, to stop or to switch from walking to running (or vice versa). For each interface, we demonstrated all the commands to the subjects, and they practiced the interfaces for five minutes before the test. Then, we asked the subjects to complete the straight track task twice and the test track task once. The order in which the subjects saw the interfaces was randomized as was the order of the straight track tasks and the test tracks.

We recorded the number of failures that occurred for each inter-

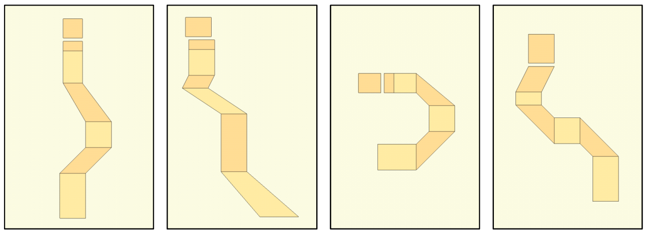


Figure 12: *The four test tracks used in the user study.*

face. When a failure occurred, the character was placed back on the track a few meters behind its previous location and the user was allowed to start again. All users successfully completed all the straight tracks and their randomly assigned test track with each interface.

During the user test, we logged the input from the Wiimotes, the resulting motion of the character, and the number of failures. After using all four interfaces, the subjects also completed a survey which required them to rank each interface with a score from one to five for "fun", "ease of use", "stress", "familiarity", "immersion" and "like", and to list good and bad aspects of each interface.

Straight Track Completion: Users experienced two types of failures on the straight tracks. On the first half of the track, they sometimes failed to maintain straight running and walking and the biped would fall off the track. The second failure occurred when users attempted to change gaits (from walking to running, from running to walking, and from running or walking to jumping). A non-parametric multiple-comparisons with Steel's test considering the joystick as a control group[1] [Hoppe and Dunnett 1993] showed that all of our wrist, arm, and leg interfaces had significantly fewer failures for straight walking on the first part of the track than the joystick interface (all $p<0.05$). A multiple-comparisons with Dunnett's test [Hoppe and Dunnett 1993] showed that the oscillation of the character's orientation with our wrist, arm, leg interfaces was significantly smaller than with the joystick interface (all $p<0.001$).

There was no significant difference in the number of failures for straight running or gait changes.

Test Track Completion: On the four test tracks, we saw three different kinds of failures. Subjects lost control of the character when the simulation controller was not robust enough to handle their commands. For example, the subject might command a transition from running to walking when the character was running very quickly and the controller might not be able to successfully accomplish that transition, causing the character to fall down. The second type of failure occurred when a subject did not negotiate a turn correctly. The tracks contained gaps that the character must jump over and a badly timed jump resulted in the third type of failure.

According to a multiple-comparisons with Steel's test considering the joystick as a control group, there was a significant difference in the ability of the subjects to negotiate curves with the four different interfaces. The leg interface had fewer failures than the joystick interface ($p<0.05$). There were no significant differences in the number of simulation and jump failures. We also analyzed the time to completion for the track and the mean time to failure with Steel's test, but found no significant differences.

[1] In general, count data are analyzed using a Poisson-based model or χ^2 test. In our experiment, however, each failure case occurs independently, and the count increases until the subject reaches the goal. The failures are uncorrelated. Therefore, we can consider the failure count as a measure of the difficulty of each task, and analyze the count data with a non-parametric multiple-comparisons.

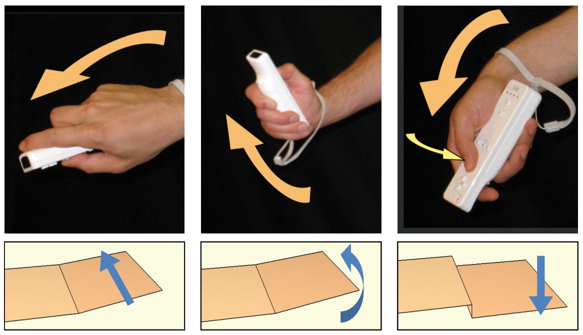


Figure 13: *Terrain editing with the Wiimote. Inclining the Wiimote sideways specifies a curve, vertically without button A creates a slope, and vertically with button A makes a step (left to right).*

Rating Score: According to a multiple-comparisons with Steel's test considering the joystick as a control group, the wrist and leg interfaces had significantly better scores for "immersion" than the joystick interface (all $p<0.05$). Steel's test also found that the wrist and leg interfaces had significantly better scores for "like" than the joystick interface (all $p<0.05$). There were no significant differences for "fun", "ease of use", "stress" and "familiarity".

8.2 Application

One potential advantage to using a physically simulated character is that it can respond naturally to external disturbances such as rough terrain. To test this idea, we designed a competitive game in which one player constructs the terrain and the second player controls the character as it runs across the newly constructed terrain. Figure 13 illustrates how a player can use the terrain editor to interactively create a slope by inclining the Wiimote vertically, a curve by inclining the Wiimote sideways, and a step by pushing button A and inclining the Wiimote vertically. To avoid making the terrain too difficult, we made the terrain width $4\,m$, and limited the slope to $\pm5^\circ$, curves to $\pm15^\circ$, and the height of a step to $10\,cm$. An example of two people playing this game is shown in the accompanying video. The user controlling the character successfully went over the terrain made by the other user.

9 Discussion

We designed and tested three intuitive interfaces for the control of a physically simulated character using input from the accelerometers in the commercially available Wiimote. We evaluated these interfaces with a user study and found several metrics under which the interfaces outperformed a traditional joystick interface and none in which the joystick was superior.

The statistical analysis of the straight track completion showed that all three Wiimote-based interfaces provided easier control of straight walking than the joystick interface. Straight walking with the joystick was achieved by keeping the joystick in the middle of the green area of Figure 11. Although this interface is an implementation of one that has been used in many commercial games, perhaps this fine level of control with the joystick was too difficult. In contrast, the turning commands for our interfaces seemed intuitive and do not require the precise manipulation of the Wiimotes. The significant difference of the curve negotiation failures in the test track completion likely resulted from these differences in the interfaces.

According to the survey results, the leg interface was more immersive and better liked than the joystick interface. We hypothesize that this was because this interface was the closest to a performance interface in that the user's actions were quite similar to actual human locomotion and closely corresponded to the resulting motion of the character. A few subjects commented that they felt that there was not a tight coupling between the character's running speed and their stepping speed with the leg interface, although the parameters of the running controller are the same as other interfaces and the stepping frequency of the running command is almost the same as that of the character.

Some subjects said that turning was easiest with the leg interface because a separate Wiimote (on the head) was used to command a turn. All three subjects who rated the leg interface as the least fun said that inclining the head also inclined their view point, making play more difficult.

The wrist interface was also rated as more immersive and better liked than the joystick. Many subjects commented that specifying walking and running with this interface was very easy. Perhaps this comment came from the narrower range of motion required with this interface. The arm and leg interfaces require approximately as big a range of motion as actual locomotion, but the wrist interface requires only relatively small wrist motions.

Although a few subjects said that the arm interface was very easy and intuitive, most subjects said that this interface was difficult to control. We believe that this problem arose because some subjects used a different grasp for the Wiimote and perhaps because the swinging motion for the arms was less dynamically constrained than the wrist or leg motion Refining the amplitude/frequency/inclination measurements to be more robust to the style of an individual user might help with this problem.

Some subjects pointed out that the leg interface was tiring to operate. This feature might be an advantage for a system such as the Wii Fit that is aimed at exercise but is probably not advantageous for hours of consecutive game play.

Our initial hypothesis was that the greater physical immersion created by the user's actions might mitigate the effect of the delay between a command and the character's response. This delay is caused by two factors. Because the character in these experiments was physically simulated, it cannot respond to a jump or a turn until the next time that the feet are in contact with the ground (unlike most game characters that do not necessarily conserve angular momentum during flight). Delay also occurs because of the time needed to interpret the accelerometer data. We are using auto/cross-correlation functions for frequency and phase estimation, and they both require taking measurements over a window of time. The joystick interface does not need the acceleration analysis phase and therefore has a shorter delay. However, most subjects did not complain about the lag in the Wiimote-based interfaces. A few subjects complained about the lag of all the interfaces including the joystick indicating that the lag in the actions of the physical simulation was likely the bigger factor. Our leg and wrist interfaces were rated as more immersive than the joystick, providing some support for our hypothesis that immersive, performance interfaces might mitigate the effect of the delay in response because the user intuitively understands delays in the actions that the character can perform when he or she is simultaneously performing similar actions.

Acknowledgement

We would like to thank Takayuki Nakamura, Moriyuki Shirazawa, Takateru Urakubo, and Yutaka Yamada for their help with the informal user studies, and all the subjects for their participation in the user study. We would also like to thank Eakta Jain for her advices on statistical analysis, Moshe Mahler for modelling the bird character and video editing, and Justin Macey for his help with the demonstration. This work was supported in part by NSF grant IIS-0326322.

References

ANDROID TECHNOLOGIES, INC., 2008. Robosapien gesture controller. http://www.robodance.com/.

ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion generation from examples. *ACM Trans. on Graphics 21*, 3, 483–490.

BOUGUILA, L., EVEQUOZ, F., COURANT, M., AND HIRSBRUNNER, B. 2004. Walking-pad: a step-in-place locomotion interface for virtual environments. In *Proc. ACM Int'l Conf. on Multimodal Interfaces*, 77–81.

CHAI, J., AND HODGINS, J. K. 2005. Performance animation from low-dimensional control signals. *ACM Trans. on Graphics 24*, 3, 686–696.

DONTCHEVA, M., YNGVE, G., AND POPOVIĆ, Z. 2003. Layered acting for character animation. *ACM Trans. on Graphics 22*, 3, 409–416.

FREEMAN, W. T., ANDERSON, D. B., BEARDSLEY, P. A., DODGE, C. N., ROTH, M., WEISSMAN, C. D., YERAZUNIS, W. S., KAGE, H., KYUMA, K., MIYAKE, Y., AND ICHI TANAKA, K. 1998. Computer vision for interactive computer graphics. *IEEE Computer Graphics and Applications 18*, 3, 42–53.

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

HODGINS, J. K. 1991. Biped gait transitions. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2092–2097.

HOPPE, F. M., AND DUNNETT, C. W. 1993. *Multiple Comparisons, Selection, and Applications in Biometry*. CRC Press.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. Spatial keyframing for performance-driven animation. In *Proc. ACM SIGGRAPH/Eurographics Symp. on Computer Animation*, 107–115.

JOHNSON, M. P., WILSON, A., BLUMBERG, B., KLINE, C., AND BOBICK, A. 1999. Sympathetic interfaces: Using a plush toy to direct synthetic characters. In *Proc. ACM Conf. on Human Factors in Computing Systems*, 152–158.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. *ACM Trans. on Graphics 21*, 3, 473–482.

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

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. *ACM Trans. on Graphics 21*, 3, 491–500.

LEE, J., 2008. Head tracking for desktop VR displays. http://www.cs.cmu.edu/~johnny/projects/wii/.

MCMAHON, T. A. 1984. *Muscles, Reflexes, and Locomotion*. Princeton University Press, Princeton.

NEFF, M., ALBRECHT, I., AND SEIDEL, H.-P. 2007. Layered performance animation with correlation maps. *Computer Graphics Forum 26*, 3, 675–684,.

NINTENDO, 2006. Wii. http://www.wii.com/.

NINTENDO, 2006. Wii Sports. http://wii.nintendo.com/software_wiisports.jsp.

NINTENDO, 2007. Donkey Kong Barrel Blast. http://www.nintendo.com/sites/dkbb/.

NINTENDO, 2007. Mario Party 8. http://wii.nintendo.com/site/mp8/.

NINTENDO, 2008. Mario Kart Wii. http://www.mariokart.com/wii/launch/.

NINTENDO, 2008. Wii Fit. http://www.nintendo.com/wiifit/launch/.

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

OSHITA, M. 2005. Pen-to-mime: A pen-based interactive control of a human figure. *Computer & Graphics 29*, 6, 931–945.

RAIBERT, M. H., AND HODGINS, J. K. 1991. Animation of dynamic legged locomotion. In *Proc. ACM SIGGRAPH 1991*, 349–358.

RAIBERT, M. H., CHEPPONIS, M., AND BROWN, H. B. 1986. Running on four legs as though they were one. *IEEE Journal of Robotics and Automation RA-2*, 2, 70–82.

RAIBERT, M. H. 1986. *Legged Robots That Balance*. MIT Press.

RASCO, B., 2008. Where's the Wiimote? Using Kalman filtering to extract accelerometer data. http://www.gamasutra.com/.

RAZZAQUE, S., SWAPP, D., SLATER, M., WHITTON, M. C., AND STEED, A. 2002. Redirected walking in place. In *Proc. Eurographics Workshop on Virtual Environments*, 123–130.

SLYPER, R. Y., AND HODGINS, J. K. 2008. Action capture with accelerometers. In *Proc. SIGGRAPH/Eurographics Symp. on Computer Animation*.

SONY, 2003. Eyetoy. http://www.eyetoy.com/.

TEMPLEMAN, J. N., DENBROOK, P. S., AND SIBERT, L. E. 1999. Virtual locomotion: Walking in place through virtual environments. *Presence 8*, 6, 598–617.

THORNE, M., BURKE, D., AND VAN DE PANNE, M. 2004. Motion doodles: An interface for sketching character animation. *ACM Trans. on Graphics 23*, 3, 424–431.

UBISOFT, 2007. Rayman Raving Rabbids 2. http://ravingrabbids2.us.ubi.com/.

USMECHANTRONICS, 2008. WiiBot: How to build a sword-wielding, tennis-playing, Wiimote-controlled, friendly robot. http://www.usmgarage.com/usmgarage/WiiBot.html.

USOH, M., ARTHUR, K., WHITTON, M. C., BASTOS, R., STEED, A., SLATER, M., AND FREDERICK P. BROOKS, J. 1999. Walking > walking-in-place > flying, in virtual environments. In *Proc. ACM SIGGRAPH 1999*, 359–364.

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

XSENS TECHNOLOGIES. Moven–inertial motion capturing. http://www.xsens.com/moven.

YIN, K., AND PAI, D. K. 2003. FootSee: an interactive animation system. In *Proc. ACM SIGGRAPH/Eurographics Symp. on Computer Animation*, 329–338.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. SIMBICON: Simple biped locomotion control. *ACM Trans. on Graphics 26*, 3.

YIN, K., COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2008. Continuation methods for adapting simulated skills. *ACM Trans. on Graphics 27*, 3.

YOUTUBE, 2008. Kyma X demonstrated with Nintendo Wiimote. http://www.youtube.com/watch?v=ESDzYYl0__s.