Dynamically Simulated Characters in Virtual Environments

David C. Brogan, Ronald A. Metoyer, and Jessica K. Hodgins
Georgia Institute of Technology

Animated characters can play the role of teachers or guides, teammates or competitors, or just provide a source of interesting motion in virtual environments. Characters in a compelling virtual environment must have a variety of complex and interesting behaviors, and be responsive to the user's actions. The difficulty of constructing such synthetic characters currently hinders the development of these environments, particularly when realism is required. In this article, we present one approach to populating virtual environments—using dynamic simulation to generate the motion of characters. We explore this approach's effectiveness with two virtual environments: the Border collie environment, in which the user acts as a Border collie to herd robots into a corral, and the Olympic bicycle race environment, in which the user participates in a bicycle race with synthetic competitors (see Figure 1).

Border collie and Olympic bicycle race environments test one approach to populating virtual worlds using dynamic simulation to generate characters' motions.

Motion for characters in virtual environments can be generated with keyframing, motion capture, or dynamic simulation. All three approaches require a tradeoff between the level of control given to the animator and the automatic nature of the process. Animators require detailed control when creating subtle movements that are unique or highly stylized. Generating expressive facial animations usually requires this low level of control. Automatic methods are beneficial because they can interactively produce motion for characters based on the continuously changing state of the user and other characters in the virtual environment.

Keyframing requires that the animator specify critical, or key, positions for the animated objects. The computer then fills in the missing frames by smoothly interpolating between those positions. The specification of keyframes for some objects can be partially automated with techniques like inverse kinematics. However, keyframing still requires that the animator possess a detailed understanding of how moving objects should behave over time as well as the talent to express that information through the character's configuration. A library of many keyframed animations can be generated offline and subsequently accessed in an interactive environment to provide the motion for a character that interacts with the user.

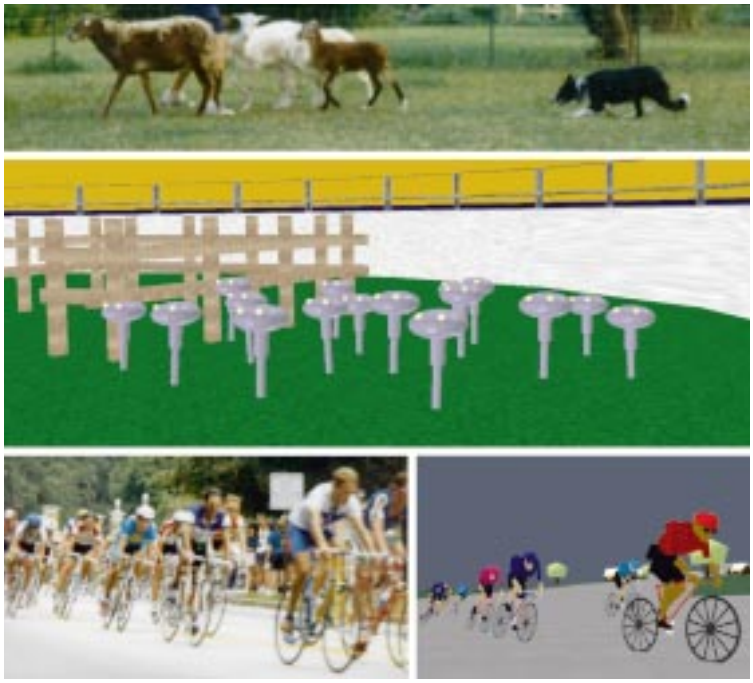
In motion capture, one of the most commonly used animation techniques, magnetic or vision-based sensors placed on an actor record the positions of body parts or joint angles as the actor performs a desired action. This recorded motion is then played back through a graphical character. Motion capture is growing in popularity because of the relative ease of recording many human actions. In particular, sports video games often use motion capture to generate the stylistic movements of athletes in an interactive environment. However, a number of problems prevent motion capture from being an ideal solution for all applications. As with keyframing, recorded motion capture sequences must be blended together skillfully to create realistic movements that change in response to the user's actions. Discrepancies between the shapes or dimensions of the motion capture subject and the graphical character also can lead to problems. If, for example, the subject was recorded touching a real table, the hands of a shorter graphical character might appear to intersect the table.

Current motion capture technology also makes it difficult to record certain movements. Magnetic systems often require connecting the subject to a computer by cables that restrict the range of motion. These systems also produce noisy data when used near metal objects like treadmills. Optical systems have problems with occlusion caused by one body part blocking another from view. Motion capture will become easier to use in

0272-1716/98/$10.00 © 1998 IEEE

interactive environments as researchers develop automatic techniques that reuse motion captured segments to animate graphical characters of many shapes and sizes and increase the variety of character actions by blending two motion captured movements with a smooth transition.

Unlike keyframing and motion capture, simulation uses the laws of physics to generate the motion of figures and other objects. Virtual characters are usually represented as a hierarchy of rigid body parts connected by telescoping and rotary joints. The equations of motion that simulate these body parts calculate the movements that result from acceleration due to gravity, forces caused by the ground during collisions, and torques applied at a joint. Each simulation also contains control algorithms that calculate the appropriate torques at each joint to accomplish such desired behaviors as hopping, riding, and balancing. Higher level algorithms can use these control algorithms to direct a group of simulations to move as a herd or to navigate along a narrow path.



1 Scenes from the two virtual environments and the real world scenarios that inspired their creation: a Border collie herding sheep and the 1996 Olympic bicycle race.

Dynamic simulation offers two potential advantages over other sources of motion for synthetic characters in virtual environments. First, simulation generates physically realistic motion that may be difficult to create using keyframing. While not all environments need or even benefit from physical realism, a growing set of applications like sports training, task training, and team-oriented games require it. Second, because their motion is computed on the fly, dynamically simulated characters offer a more precise form of interactivity than characters animated with a fixed library of precomputed or recorded motion. For example, in football video games, the motion resulting from a collision between opposing players is a function of the magnitude and direction of their velocities as well as their body configurations at the time of impact. Because of the very large number of initial conditions, it's difficult to model this interaction accurately with a library of fixed motions.

Computational cost is one disadvantage of dynamic simulation. For the two examples presented here, we used multiple processors with either virtual or physical shared memory to obtain the required performance. Dynamic simulation also imposes some limitations on the behavior of synthetic characters. Simulated characters are less maneuverable than those modeled as point-mass systems and those that move along paths specified by animators. For example, although a point-mass model can change direction instantaneously, a legged system can change direction only with a foot planted on the ground. If the desired direction of travel changes abruptly, the legged system may lose its balance and fall. These limitations, although physically realistic and therefore intuitive to the user, make it more difficult to design robust algorithms for group behaviors, obstacle avoidance, and path following.

To illustrate the use of dynamically simulated characters, we built the Border collie and Olympic bicycle race environments. Each includes a geometric description of the terrain and scenery, a user interface in the form of a bicycle, interactive characters and their associated behaviors, and the system architecture necessary for real-time simulation of multiple characters.

In the Border collie environment, the user navigates by steering and pedaling a stationary bicycle and, like a Border collie, attempts to herd a group of dynamically simulated one-legged robots into a corral. Each robot uses knowledge of the locations of the other robots and the user to reactively avoid collisions. The Border collie environment has served as a testbed for the insertion of dynamically simulated characters into virtual environments. Further refinement could transform this environment into an intriguing game that might entice amateur athletes into a longer or harder workout.

The Olympic bicycle race environment lets the user experience the 13-kilometer road race from the 1996 Olympics. The user interacts with synthetic cyclists while riding a stationary racing bicycle mounted on a platform. The platform pitches to match the slope of the racing course, provides pedaling resistance, and measures the steering angle of the front fork. Each dynamically simulated synthetic cyclist rides in a realistic fashion around the racecourse. We implemented a distributed system that lets us simulate multiple bicycles in real time and display the graphical environment at 30 frames per second (fps).

While we intended the Border collie environment as entertainment, we envision that environments like the Olympic bicycle race may someday prove valuable to avid cyclists or professional racers limited by time, weather, or insufficient situational training. For exam-

ple, racers at the Olympics are allowed only limited training time on the course without traffic. The athletes would benefit from additional practice time, allowing them to better tune their racing strategy for a particular course or field of competitors.

Background

Over the past decade, many researchers have explored the problems inherent in creating autonomous, believable characters for virtual environments. These problems include the control of individual characters, multiagent behavioral control, virtual environment interfaces, and system design.

Control of individual characters

The first problem is the control of individual characters. Many recent results from computer animation relate to this problem, but we briefly review just the work of those who have applied their results to virtual environments. Their solutions include fully autonomous characters as well as characters controlled at various levels by the user. An appropriate level of autonomy depends, in part, on the application domain. Users provide explicit direction to their graphical representations—called avatars—and in many training applications, the characters must be responsive to directions from a supervisor or coach to create the desired training situation. Single-player video games, on the other hand, need fully autonomous characters to serve as opponents or companions.

Blumberg developed autonomous characters using a layered approach for behaviors, motor skills, and geometry.[1] Two additional layers provide some abstraction or generalization between characters with different functionality. This architecture was used to create an animated responsive dog in the ALIVE system.

Perlin and Goldberg developed the Improv system to facilitate the creation of autonomous interactive characters.[2] Like Blumberg's system, Improv uses a layered architecture with a behavior engine for selecting among higher level behaviors. It also has an animation engine, which uses high-level descriptions to move the characters, and a geometry layer. Improv allows direction at several levels and has been used to design a virtual environment with a variety of interacting characters that exhibit distinct personalities.

The Jack system, developed at the University of Pennsylvania, facilitates the animation of human characters in virtual environments by providing autonomous walks and other behaviors.[3] A real-time behavioral controller generates paths that guide Jack through an environment, while reactive navigation controllers avoid obstacles and compute footstep placements. In this environment, Jack can walk along the city's streets and sidewalks while observing pedestrian crossing signals. Other research groups have used the Jack system extensively to develop medical and military training scenarios in virtual environments.

The Mira Lab at the University of Geneva and the Computer Graphics Laboratory at the Swiss Federal Institute of Technology have a rich tradition of research in animating human motion. Recently, researchers in those labs have focused on the development of synthetic humans for use in virtual environments. In particular, they have explored controlling avatars with many degrees of freedom, generating autonomous walking and grasping motions, and controlling animated crowds.[4] They have also developed systems to facilitate creating networked virtual environments.[5]

Multiagent behavioral control

Reynolds was one of the first graphics researchers to animate group behaviors.[6] Actors in his system are bird-like objects similar to the point masses used in particle systems except that each bird has an orientation, and the model includes important dynamic features such as gravity, lift, and banking. The birds maintain position and orientation in the flock by balancing their desire to avoid collisions with neighbors, match the velocity of nearby neighbors, and move toward the center of the flock. Each bird uses only information about nearby neighbors. This localization of information simulates one aspect of perception and reaction in biological systems and allows for proper balancing of the three flocking tendencies. Reynolds' work convincingly demonstrates that applying simple rules to determine the behaviors of the individuals in a flock can create realistic animations of group formations.

Brogan and Hodgins expanded on Reynolds' work by applying similar control algorithms to dynamically simulated characters.[7] They explored the algorithm's performance with a herd of 105 hopping robots and a group of 18 cyclists for a test suite of three problems: steady-state motion, turning, and avoiding obstacles. Both the legged robots and the cyclists must control balance, facing direction, and forward speed as well as movement within the group. These limitations on the maneuverability of the individuals in the group make it more difficult to design robust control algorithms for herding.

Tu and Terzopoulos populated a virtual marine world with fish that hunt, flee, mate, and wander.[8] To create fully autonomous artificial creatures, they modeled the physics of the animal and environment, locomotion style, perception, and higher level behaviors. To make the interactions more interesting, they modeled fish that differed not only in shape and color, but also in behavior by including predators, prey, and pacifists. The fish were modeled as spring-mass systems with sinusoidal patterns actuating the springs and propelling the fish through the water. Their layered architecture consists of an intention generator that creates goal-directed behavior, a motor system that implements higher level motion primitives such as "swim forward" or "turn left," and motion controllers that translate low level control parameters such as speed and direction into muscle actions. Yu and Terzopoulos adapted this system for real-time performance by replacing the fishes' simulated motion with kinematic motion derived from a prerecorded database of systematically simulated fish maneuvers.[9]

Virtual environment interfaces

Researchers have explored many different naviga-

tional interfaces for virtual environments. Flying with a six degree-of-freedom (DOF) input device and locomoting with a 2D treadmill or stationary bicycle are among the most common. Bicycling provides a particularly intuitive interface because the virtual camera's motions correspond to familiar motions from the real world. Although bicycling is not a good interface for all applications, restricting the user's motion to riding makes it easy to measure all of the user's actions and to have an accurate representation of the user's avatar in the virtual environment.

Bülthoff and Distler created a virtual environment to investigate cognition and visual perception in complex environments.[10] The user navigates by riding a stationary bicycle while looking at a flat screen display. This system allowed Bülthoff and Distler to conduct experiments in visual attention, effects of cognitive load on peripheral vision, object recognition in 3D scenes, navigation, and optical flow and time-to-collision in virtual environments.

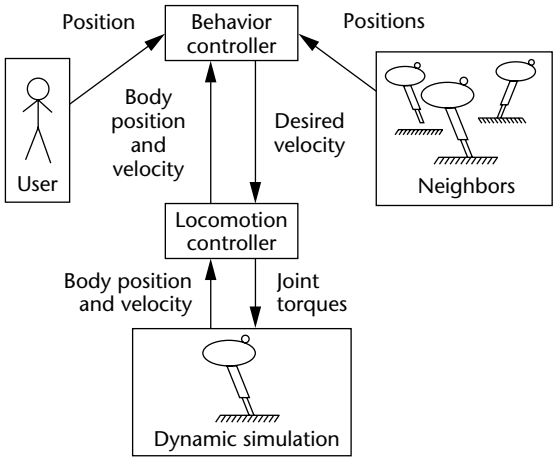
Researchers at Bell Laboratories created a virtual Peloton to explore virtual reality systems for interaction and collaboration over the World Wide Web.[11] Users navigate by riding a stationary bicycle as the system applies resistive torques to the wheel to simulate hill riding.

Several commercially available systems for sports training use bicycling as an interface. The Computrainer provides a load generator to simulate pedaling resistance on hills and monitors energy expenditure. Using an 8-bit Nintendo Entertainment System, the user can view an animated cyclist riding along the 2D course. Tectrix markets a stationary exercise bicycle called a VRBike that lets the user pedal and steer through a 3D environment while experiencing changes in resistance for hills and water. Several VRBikes can be connected together with a local network for group competition.

System design

Virtual environments are often supported by a network of computers that provide the graphics processing power for multiple users and the computational power for calculating the motion of synthetic actors. One such network-based system, Spline (Scalable Platform for Large Interactive Networked Environments),[12] provides a framework for creating networked virtual environments that allows multiple users in an environment to experience spoken interaction, 3D sound, and many forms of motion generation. A demonstration environment called Diamond Park let users ride around a park on modified Tectrix VRBikes and converse with other users. The park was populated with vehicle simulations, autonomous characters, and figures driven by motion capture data.

NPSNet-IV is a 3D virtual environment for multiplayer participation over the Internet.[13] The system supports networked virtual environments that include large scale communication, networked multimedia for sound and video, and autonomous agents. Our motivation is similar in that we would also like to create training environments for situations either too expensive or dangerous to create in the real world.



2 The input and output of the behavior and locomotion controllers.

Dynamically simulated characters

We constructed the Border collie and Olympic bicycle race environments to explore the use of interactive, dynamically simulated characters in virtual environments. In these two systems, every character is a physically realistic, rigid-body simulation of either a hopping one-legged robot or a bicycling human. Mass, moment of inertia, and a polygonal model define the individual character's body parts. Telescoping joints or rotational joints with one, two, or three DOF connect the bodies. We use a commercially available system, SD-Fast (Symbolic Dynamics), to compute the equations of motion based on these parameters. Integrating these equations determines a character's configuration at a given moment in time.

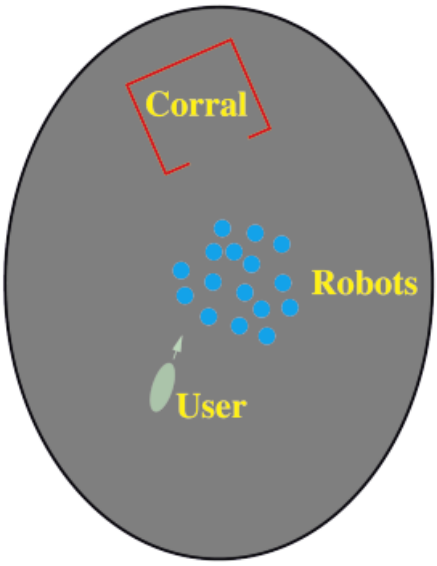
Because both cyclists and legged robots are active systems with simulated motors or muscles that provide an internal source of energy, the characters need control systems for locomotion. The control systems for bicycling and hopping take a desired velocity as input and compute the joint torques that will control speed and direction of travel while maintaining balance.

Characters in virtual environments must not only locomote in a natural manner, they must also interact with the user and other characters in an intelligent fashion. This interaction is accomplished through a behavioral controller that allows the characters to move as a group, avoid obstacles, and follow paths on the terrain. The behavioral controller computes a desired velocity for the locomotion controller based on the positions of the character, its neighbors, and the user. Figure 2 shows the layout of these controllers.

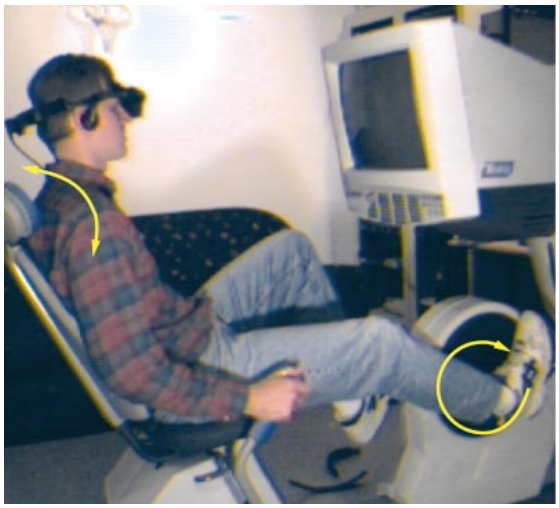
A virtual environment that includes dynamically simulated characters must have the following four components: a user interface, a graphical description of the world, dynamically simulated characters with associated behaviors, and a systems architecture that allows real-time simulation of multiple characters. We now present each of these components in more detail for the two virtual environments.

Border collie environment

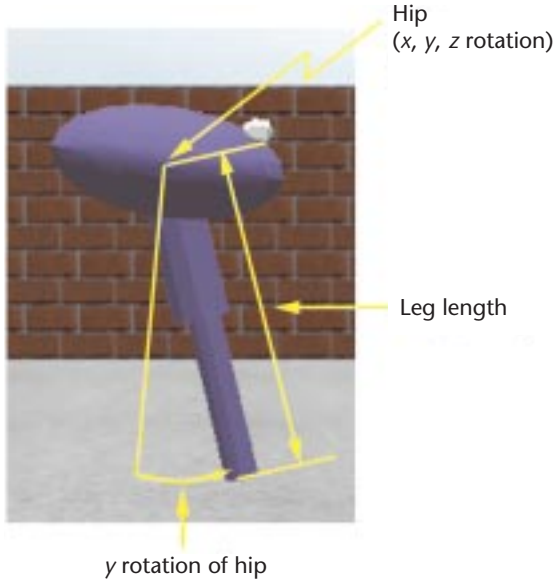
In the Border collie environment, the user pedals and steers around a playing field while attempting to



3 The layout of the playing field in the Border collie environment.



4 The Tectrix stationary bicycle that serves as the interface to the Border collie environment.



5 The controlled degrees of freedom of the one-legged robot.

herd a group of 16 dynamically simulated, one-legged robots into an open corral. The robots maintain a group formation as they roam the playing field and reactively flee from the user. The computation of the 16 robots is distributed across a cluster of workstations that communicate the position of each robot via distributed, shared memory.

The Border collie environment occupies the flat, oval infield of the polygonal velodrome model extracted from Diamond Park.[12] At one end of the infield is a 10-meter-square corral with a 6-meter opening in one side (see Figure 3). The environment, consisting of 11,000 polygons, and the robots, consisting of 500 polygons, are rendered using the IRIS Performer Graphics API at 30 fps.

The user navigates through the environment by riding a modified Tectrix recumbent bicycle. The user pedals slower or faster to control speed and leans left or right to steer (see Figure 4). Sensor measurements of these actions are combined with a simple model of bicycle dynamics to compute the user's position in the virtual environment. This interface provides the user with fairly natural controls for general navigation. However, the herding task is difficult because the bicycle has a large turning radius and cannot easily reverse direction.

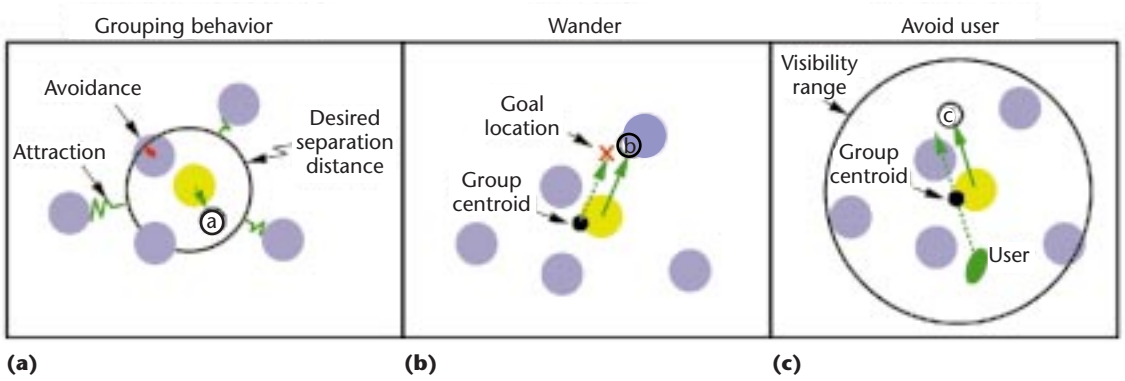
The user wears an i-glasses head-mounted display (HMD) that tracks the roll, pitch, and yaw of the head. This information, along with the user's location, determines the world view presented through the display. Wearing an HMD lets the user ride in one direction while looking in another. Although some studies have found that users do not always take advantage of this feature, we have found it necessary in this setting when the user attempts a flanking maneuver around the side of the robot group.

Hopping one-legged robot. The Border collie environment is populated by a herd of one-legged robots. Each robot is modeled by a physically realistic, rigid-body simulation (see Figure 5). A telescoping joint connects the upper and lower cylindrical legs, and a three-DOF hip joint connects the upper leg to the ellipsoidal body.

The control system for hopping takes a desired velocity as input and computes the foot position at touchdown that will achieve this desired velocity by the next liftoff.[14] Flight duration is controlled by extending the telescoping leg during stance to make up for losses in the system. Exerting a torque between the body and the leg during stance controls the body attitude.

We believe the dynamically simulated motion of the one-legged robots is intuitive to users because it mimics the movements of real legged systems. For example, to turn left quickly, a legged robot must plant its foot out to the right in much the same way football players cut to avoid an opponent. The user can anticipate the change in velocity by watching the angle of the robot's leg during flight.

Behavioral controller. We modeled the behavioral controller of the one-legged robots to mimic the actions of sheep moving as a herd while avoiding a Border collie. Like grazing sheep, the robots maintain a tight



6 The behavior controller for a one-legged robot produces a desired velocity based on its position relative to (a) its neighbors, (b) the goal location for the wander behavior, and (c) the user.

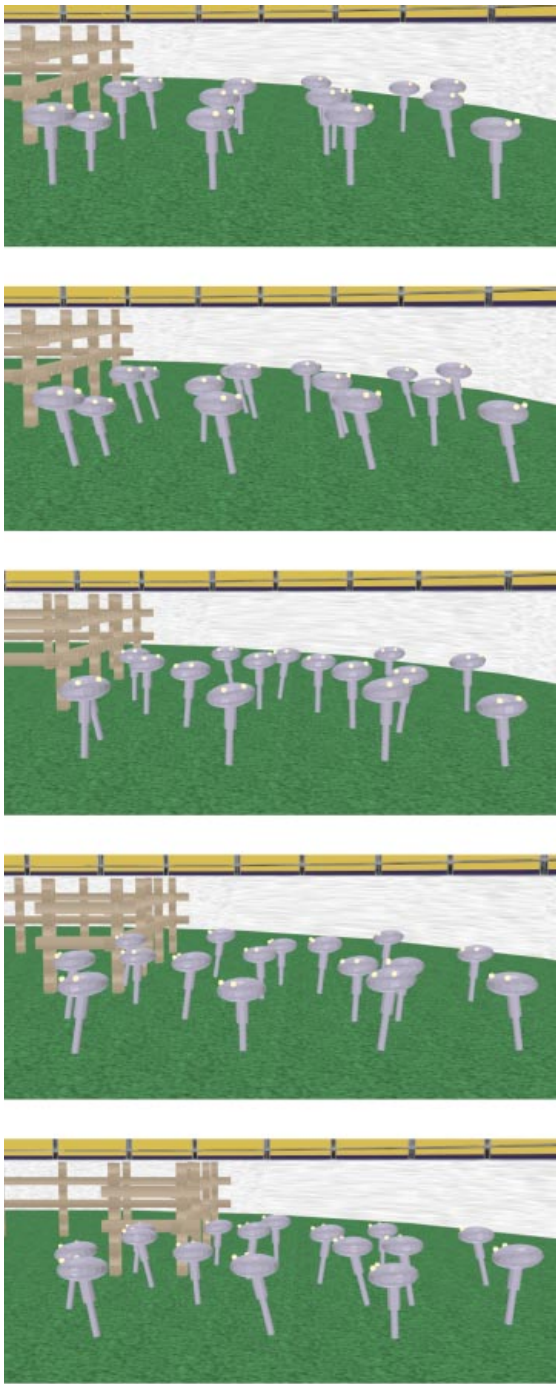
group formation and wander through the environment together when the user is not nearby. When the user moves close to the group, the robots reactively avoid the user just as sheep will move away from a Border collie. The behavioral controller must generate a desired velocity that satisfies the grouping, wandering, and fleeing goals of each robot. To maintain the sense of presence in the virtual environment, the autonomous behaviors of the robots must accomplish these three goals without causing the robot to fall down or allowing the behaviors to become repetitive.

To maintain a tight group formation, the behavioral controller computes a velocity, v_{herd}, that should move each robot away from nearby neighbors and toward distant ones (see Figure 6). When every robot in a group exhibits this behavior, the robots settle into a circular pattern with roughly equal spacing.[7]

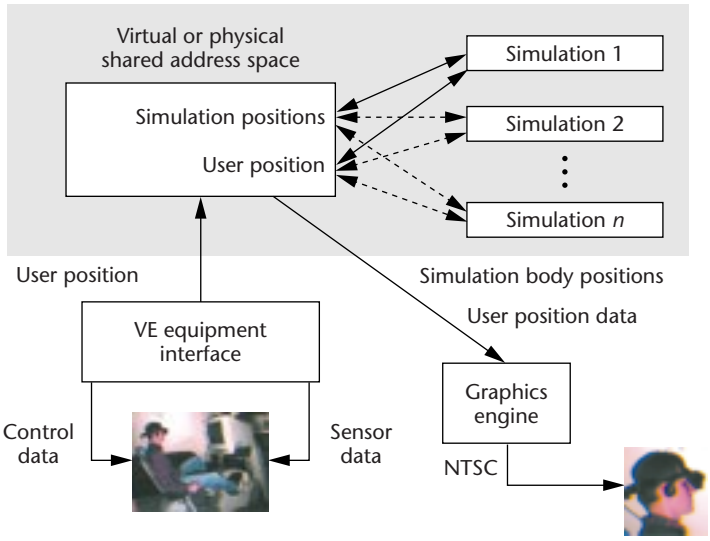
We generated a wander behavior to make the robots' movement through the environment more interesting. The behavioral control algorithm calculates the centroid of the robot group and computes a vector between this centroid and a random goal position on the terrain. The rate at which the robots move toward the goal, v_{goal}, is proportional to the distance to the goal. When the centroid of the group reaches the vicinity of the goal, the behavioral controller selects a new goal position. The alert user can capitalize on a herd that wanders too close to the corral.

The group of robots abandons its wandering behavior and reactively flees from a user who comes within 15 meters of the group's centroid. The behavioral controller computes a desired position for the group's centroid just beyond this threshold on the extended line between the user and the centroid. The controller then creates a velocity, $v_{user,}$ that moves the group towards this desired position. As with the wander behavior, v_{user} is proportional to the robot's proximity to the user. We combined these three behaviors by setting the desired velocity for each robot to $v = 0.5\ (v_{herd} + v_{user})$ when the user comes within 15 meters of the robot group, and $v = 0.5\ (v_{herd} + v_{goal})$ otherwise.

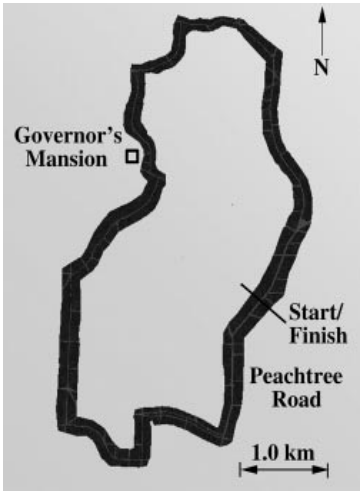
The interaction of these three goals causes the robots to hop out of phase after just a few seconds of simulation. When the user moves close to the herd, the robots' response is also staggered in time because each robot must have a foot in contact with the ground in order to change its velocity (see Figure 7). The visual complexi-



7 The group of 16 one-legged robots changes direction abruptly to avoid the user (0.5 second intervals between snapshots).



8 System components that enable using dynamically simulated characters in the virtual environment.



9 The racecourse model for the Olympic bicycle race environment.

ty created by the variation in hopping and reacting to the user makes the virtual environment more interesting. The herd of robots' behavior remains intuitive because the individual robots move as legged systems do in the real world.

System architecture. The diverse computational requirements of the Border collie environment led us to use multiple machines, each specifically suited to perform a particular function of the overall system (see Figure 8). Five machines are dedicated to computing the motion of the simulated characters, one manages the bicycle user interface, and one generates the rendered images.

Simulating the one-legged robots consumes the majority of the computational resources. A 250-MHz Sun UltraSparc can simulate three one-legged robots in real time. In the context of these virtual environments, real time means the time required to compute one second of simulated motion equals one second of wall-clock time. The rendering process produces new images at a rate of about 30 fps, or every 0.033 seconds. To produce accurate robot positions for each new frame, the simulation must execute until 0.033 seconds of simulated time has passed. If a machine computes the simulated passage of 0.033 seconds before an equivalent amount of wall-clock time has passed, the simulation must pause to prevent rendering an image where the robot positions are sampled from inconsistent moments in time.

We use a system called Beehive to provide synchronization protocols for the cluster of networked Sun UltraSparcs that execute the simulations.[15] Beehive provides a software barrier to enforce global time synchronization between the simulations. The software barrier also triggers a process that obtains the body geometry transformations from each simulation and transmits a UDP datagram stream to the computer rendering the user's view of the graphical world.

We selected the frequency of this barrier to provide the rendering process with accurate robot body transformations 30 times per second. The amount of time that a simulation process blocks due to the barrier depends on the capabilities of each processor performing the simulation computations. Because we use a homogeneous set of processors during periods of low load, we expect all robot simulations to proceed at similar rates. Therefore, a simulation process does not spend much time waiting for other processes to complete.

In addition to providing synchronization tools, Beehive also provides virtual shared memory access across the cluster of networked simulation servers. To conserve communication bandwidth between the cluster machines that compute the physical simulations, each robot simulation accesses the positions of its neighbors and the user only when the behavioral controller needs that information. Because the new desired velocity is calculated during flight, the position data is requested as the robot's foot leaves the ground. The virtual shared memory access provided by Beehive causes very little delay in the acquisition of neighboring robot position data.

The Tectrix recumbent bicycle we use in the Border collie environment outputs the lean angle and pedal rates to a 200-MHz, R4400 SGI Indy. The Indy processes and integrates this data with a simple model of bicycle dynamics to compute the user's position and body orientation in the virtual environment. The Indy also interfaces with an i-glasses HMD that includes a head tracker to monitor head roll, pitch, and yaw. Every 0.033 seconds the Indy transmits the user's current position to the simulations on the UltraSparc cluster. It also sends the user's position and head orientation to the machine rendering the scene.

The Border collie environment uses a dual-processor 194-MHz, R10000 SGI Onyx2 with InfiniteReality graphics to perform the rendering. Three processes run on this machine and communicate through shared memory. The rendering process uses the IRIS Performer Graphics API to create, illuminate, and render images

of the environment. The positions of the one-legged robots in the environment are obtained by a process that monitors a network port for data packets sent from the simulation cluster . The camera's position is generated by a process that receives data packets from the Indy interfacing the user's bicycle.

Olympic bicycle race environment

In the Olympic bicycle race environment, the user rides with 16 simulated cyclists through the streets of the 1996 Olympic road racecourse in Atlanta. In this environment, an athlete can experience the course both visually and physically by wearing an HMD and riding a bicycle that tilts and applies an adjustable load to the rear wheel. The behavioral controller creates a dynamic formation of simulated cyclists that ride at varying speeds through the course. Computing the simulated motion for the 16 cyclists proceeds in real time on a multiprocessor that provides shared memory for communication between simulations.

The virtual environment accurately models the roads and terrain of the racecourse. Using topographic maps obtained from the Atlanta Water Works, we digitized and modeled the racecourse, side streets, and surrounding terrain of the 13-kilometer course, preserving dimensions and height information. The counter-clockwise course begins at the southeast corner and has a maximum incline of 13 percent and decline of 4 percent (see Figure 9). The road surface is modeled with one nonuniform rational B-spline (NURBS) surface and the surrounding terrain with a second. After the surfaces have been triangulated, the environment has 26,000 triangles (including a sparse layout of houses and trees), and each cyclist consists of about 300 polygons.

The bicyclist simulation interfaces with the triangulated road surface in two ways. To provide the simulated cyclist's perceptual system with knowledge about the road's path, we manually created a set of points representing the centerline of the road and fit a Catmull-Rom spline to the points. The behavioral controller uses this spline to guide the cyclist down the road. The dynamic simulation of the cyclist and the simpler simulation of the avatar's movement also require height information for the road surface. We implemented a quadtree spatial subdivision of the road surface to efficiently determine the road's height at a particular location.

We have experimented with the placement of additional polygonal objects to improve the user's perception of motion in the environment. Light posts, fire hydrants, trees, and houses along the side of the course not only provide the sensation of rapidly moving objects in the user's field of view, but also provide the visual cues of parallax. Simple texture maps on the road and grass surfaces contribute to the sensation of motion (see Figure 10).

The user rides a racing bicycle that measures control inputs and provides feedback (see Figure 11 next page). The user can steer in a ±20-degree range, brake, and change gears while pedaling through the environment. The bicycle is instrumented to measure the speed of the rear wheel and the turning angle of the front wheel. These measurements are combined with the simple model of bicycle dynamics used in the Border collie environment to compute the user's position in the virtual environment. The bicycle is mounted on a motion platform that pitches fore and aft by ±12 degrees to provide the user with feedback about the road's slope. The bicycle's rear wheel connects to a generator and flywheel to allow limited freewheeling and to match the wheel load to the terrain angle. The user wears an i-glasses HMD that provides head tracking and lets the user selectively focus on particular neighbors or look ahead to plan a path through the next corner.

10 **Sixteen cyclists round a corner (1.0 second intervals between snapshots).**

The bicyclist and riding controller. A 12-segment, rigid-body model connected by rotary joints with 17 controlled DOF models the simulated bicycle

11 The user rides a racing bicycle mounted on a platform.

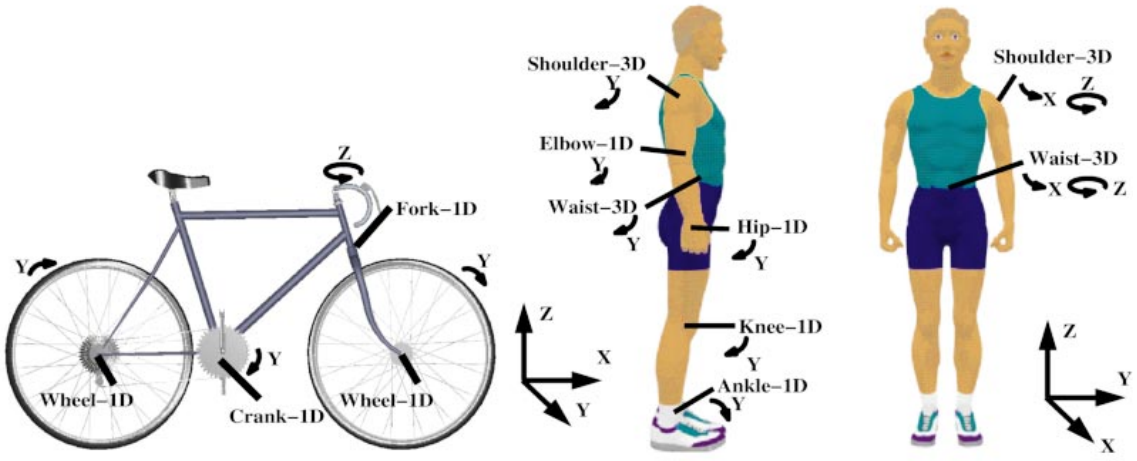
rear wheel. The connecting springs are two-sided, and the cyclist can pull up on the pedals as if the bicycle were equipped with toe-clips. The details of the bicycling control system appear in Hodgins et al.[16]

The motion of the dynamically simulated cyclist exhibits some of the subtle details present in real bicycling. For example, to complete a right turn, the simulated cyclist must first steer to the left slightly and shift the center of mass to the right side of the bicycle. Then the cyclist steers to the right until the desired turn is completed. Because the rate at which the center of mass shifts to the right is a function of the system dynamics, the cyclist has a limited turning rate.

Behavioral controller. The behavioral controller for the Olympic bicycle race environment lets each cyclist ride around the racecourse alone or in a group including other simulated cyclists. Like the behavioral controller for the one-legged robots, this controller combines three goals to compute a desired velocity: remain on the road, ride as a group, and ride with the user.

To prevent the cyclists from riding off the road, the behavioral controller computes a desired direction of travel for each bicycle and multiplies this direction by a predetermined training speed to obtain a desired velocity, v_{road}. The goal position that lies a specified distance beyond the cyclist's current location determines the direction of travel for each cyclist (see Figure 14). The goal position is computed by first finding the point on the centerline spline closest to the cyclist's current position, then by calculating a point a given distance further along the spline. If the cyclist is not currently riding on the centerline, then the goal position is moved out on a line perpendicular to the tangent to the spline. This prerider. Some joints, like the knee, are modeled as a single-axis pin joint. Other joints, like the waist and shoulder, are modeled by three-axis gimbal joints (see Figure 12). A detailed polygonal model is used to compute the mass and moment of inertia for each body part (see Table 1). We obtained a simple polygonal version of the cyclist from Diamond Park (see Figure 13) and used this model as the graphical representation of the cyclist in the environment to preserve interactive rendering rates.

The cyclist must maintain balance while attempting to match the desired velocity specified by the behavioral controller. To steer, the simulated human rider applies forces to the handlebars with the hands. To provide propulsion, the simulated cyclist applies forces to the pedals through joint torques at the hips and knees. Spring and damper systems connect the hands to the handlebars, the feet to the pedals, and the crank to the



12 The controlled DOF of the bicycle and human models. The human model has 11 joints; the DOF at each joint are shown, as are the 4 DOF of the bicycle model. A pivot joint connects the rider's pelvis to the bicycle seat. The polygonal models were purchased from Viewpoint Datalabs.

Table 1. Parameters of the rigid-body model of a human. The moments of inertia are computed about each link's center of mass.

Link	Density (g/cm³)	Mass (kg)	Moment of Inertia(x,y,z kgm²)		
Upper Body	1.00	34.24	1.390	1.298	0.337
Pelvis	1.03	16.61	0.23	0.18	0.16
Upper Leg	1.04	8.35	0.15	0.16	0.025
Lower Leg	1.08	4.16	0.055	0.056	0.007
Foot	1.07	1.34	0.002	0.008	0.007
Upper Arm	1.07	2.79	0.025	0.025	0.0050
Lower Arm	1.10	1.78	0.0218	0.0230	0.0023

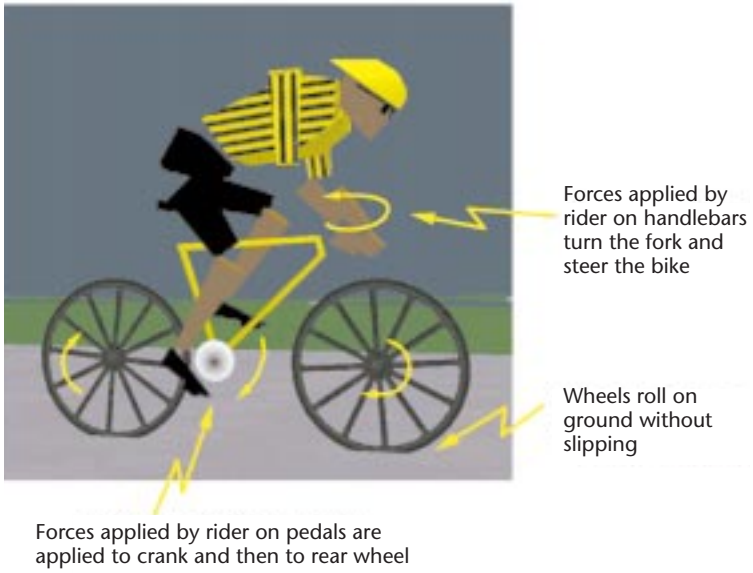
serves the cyclist's relationship to the centerline. We empirically determined that when the goal position lies 30 meters ahead on the road, the cyclists could complete the entire 13-kilometer course. This heuristic for road following places the cyclists near the center of the road on the straight sections while smoothing their path through tight corners, but it does not necessarily calculate optimal paths through corners.

Cyclists routinely ride in groups because the middle and rear riders expend 30 to 40 percent less energy than the leading edge of the pack.[17] The behavioral controller achieves similar grouping behaviors by computing a goal position relative to each cyclist that moves it closer to distant neighbors and further from close neighbors. A desired position, p_i, is computed relative to each of the n neighbors within a given visibility range. A second desired position, p_{user}, is computed relative to the user's position. The goal position is a weighted average of these desired positions, represented by

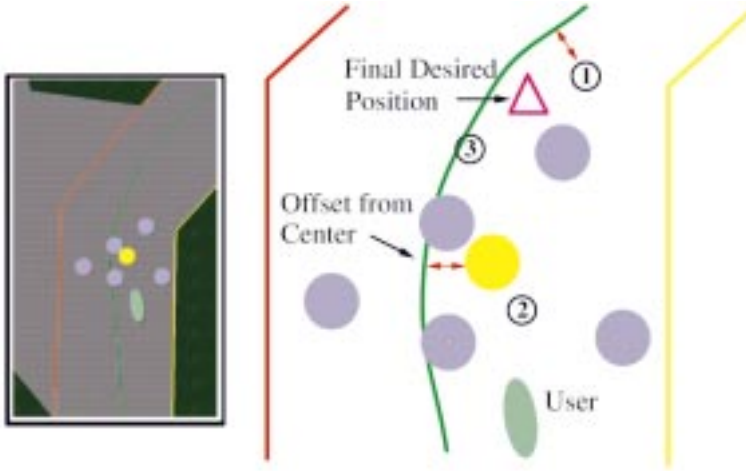
$$p_{goal} = \frac{\sum_{i=1}^{n} \frac{1}{d_i} p_i + \frac{1}{d_{user}} p_{user}}{\sum_{i=1}^{n} \frac{1}{d_i} + \frac{1}{d_{user}}}$$

where d_i is the distance between the cyclist and the ith neighbor and d_{user} is the distance to the user. The error between the current position of the cyclist, p, and the goal position is $e = p - p_{goal}$. To eliminate this error, a new desired velocity is computed using a spring and damper system: $v = v_{road} + k_d e + k_v \dot{e}$, where $\dot{e}$ is the rate of change of the error in desired position. For the experiments reported here, $k_d = 0.1$ and $k_v = 0.01$.

System architecture. Like the Border collie environment, the Olympic bicycle race environment requires significant distributed computational power to render the environment and simulated characters, interface with the bicycle hardware, and compute the motion of the simulated characters. Each dynamically simulated cyclist runs in real time as a separate process on an SGI Power Challenge with 16 195-MHz R10000 processors. The simulations communicate position information to one another through shared memory. The shared memory and synchronization protocols are implemented in hardware on the Power Challenge, but their functionality is identical to the Beehive system used for the Border



13 The stylized polygonal model of the simulated cyclist.



14 The behavior controller for a cyclist produces a desired velocity based on its position relative to a lookahead distance along the center of the road, 1; its neighbors, 2; and the user, 3. The desired velocity is computed from the weighted average of these three relative positions (represented by the triangle).

collie environment. The synchronization ensures that all the simulations compute at identical rates and triggers the transmission of body geometry transformations to the graphics engine. Another process receives the user position from the graphics engine and places this data in the shared memory region.

A 200-MHz, R4400 SGI Indy interfaces with the bicycle hardware—a Motorola 68332—through a serial connection. Sensors on the bike platform measure the rear wheel's rotation rate and the front wheel's steering angle. The Indy uses this data to drive a simple model of the bicycle dynamics on the polygonal terrain to compute the user's position in the virtual world. The slope of the terrain at the user's current position determines the pitch angle of the bicycle platform and the resistance on the rear wheel. This information is sent to the Motorola 68332, which controls the motors on the bike platform that adjust the pitch angle and rear wheel rolling resistance.

The Indy sends the head orientation obtained from the i-glasses along with the rotation rate of the rear wheel and the steering direction of the front wheel to the graphics engine. The graphics engine integrates the simple model of bicycle dynamics to calculate the exact camera position and orientation at every new frame. The graphics engine also renders the image and sends the bike position to the shared memory region of the machine computing the simulations.

Discussion

These two virtual environments represent a first step toward populating interactive environments with dynamically simulated characters that move and behave realistically. While previous research has described how dynamic simulations enhance the realism of animated motion, researchers are just beginning to investigate how to effectively use dynamically simulated characters in virtual environments. Our preliminary results indicate that dynamic simulations allow a variety of behaviors while realistically portraying a character's physical capabilities in an intuitive fashion.

We believe that the realistic nature of dynamic simulation makes the characters' movement intuitive to the user. The visual cues provided by the dynamic simulations help create a feeling of immersion in the synthetic environments. For example, the simulated cyclists usually slow when riding up hills because they have to work harder. As the cyclists approach a turn, they swing out in preparation for the lean into the turn. The herd of one-legged robots exhibits similar behavior. If the herd is hopping along in a constant direction at a constant speed, the motion will be fairly static with little leg motion. When the player approaches the herd in an attempt to guide them into the corral, the motion suddenly becomes much more dynamic as the robots begin to change direction. The direction change requires a robot to kick its leg out to the side during the flight phase, much like the cyclist preparing for a turn or even a human preparing to make a sudden change of direction.

The disadvantage of these realistic responses is that in extreme situations, the character may lose its balance and fall down. Either the higher level behaviors must be sufficiently conservative that failures do not occur or falling down must be presented in a realistic fashion as part of the virtual environment's story line. These physically correct reactions to the environment and user provide a degree of realism not yet reached with other animation methods for real-time motion generation.

However, the visual cues provided by the simulations are only as realistic as the level of physical modeling. For the simulations described here, we included plausible dimensions, mass, and moment of inertia values. Our simulations could be improved however, as we do not include friction or rough terrain in the ground contact model, the reduced aerodynamic drag from riding behind the leader of a pack, or the reaction to collisions between characters.

Repeating behaviors are commonly regarded as one of the quickest ways to destroy the sense of presence in an immersive environment. This phenomenon rarely occurs in the real world because actors adjust their actions slightly in response to changes in goals and the environment. This variety proves harder to achieve in virtual environments depicting simplistic scenarios that usually lack intricate details in the environment.

We believe that because the interactions between the user and the group of cyclists in the Olympic bicycle race environment are complex, the behaviors of the simulated cyclists will not quickly seem repetitive. In part, this lack of repetition is easily achieved because the characters respond to the user in a continuous fashion rather than by selecting from a discrete set of actions. If the user approaches the group of cyclists from a 45-degree angle, their response will differ slightly from their response to a 40-degree-angle approach.

As we continue to develop the Olympic bicycle race environment, the behavioral controller will include more discrete behaviors like initiating a breakaway, rotating out of the front position in the pack, and climbing while standing out of the saddle. With these discrete changes in actions and goals, the behavioral controller will no longer be able to average the goals of the system and achieve reasonable performance. We will then have to address such problems as the recognizable repetition of discrete actions and rapid dithering between behaviors. The developers of environments with keyframed and motion capture driven characters have already explored these issues, and we should be able to build on their results. ■

Acknowledgments

We thank H. Benjamin Brown, Jr. for building the bike platform, Stewart Brann for modeling the bicycle race course and scenery, Jeremy Heiner for an early implementation of the communication protocol, and the researchers at Mitsubishi Electric Research Laboratory for use of the Diamond Park environment and the graphical model for the simulated cyclist. This project was supported in part by NSF NYI Grant No. IRI-9457621, Mitsubishi Electric Research Laboratory, and a Packard Fellowship.

References

1. B.M. Blumberg and T.A. Galyean, "Multilevel Direction of Autonomous Creatures for Real-time Virtual Environments," *Proc. Siggraph 95*, R. Cook, ed., ACM Press, New York, 1995, pp. 47-54.
2. K. Perlin and A. Goldberg, "Improv: A System for Scripting Interactive Actors in Virtual Worlds," *Proc. Siggraph 96*, H. Rushmeier, ed., ACM Press, New York, 1996, pp. 205-216.
3. J.P. Granieri et al., "Behavioral Control for Real-time Simulated Human Agent," *1995 Symp. Interactive 3D Graphics*, P. Hanrahan and J. Winget, eds., ACM Press, New York, 1995, pp. 173-180.
4. S.R. Musse and D. Thalmann, "A Model of Human Crowd Behavior," *Computer Animation and Simulation 97,* Springer-Verlag, Heidelberg, Germany, 1996, pp. 39-51.
5. T.K. Capin et al., "Virtual Human Representation and Communication in VLNet," *IEEE Computer Graphics & Applications*, Vol. 17, No. 2, 1997, pp. 42-54.
6. C.W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Computer Graphics* (Proc. Siggraph 87), Vol. 21, M.C. Stone, ed., ACM Press, New York, 1987, pp. 25-34.
7. D.C. Brogan and J.K. Hodgins, "Group Behaviors for Systems with Significant Dynamics," *Autonomous Robots*, Vol. 4, No. 1, 1997, pp. 137-153.
8. X. Tu and D. Terzopoulos, "Artificial Fishes: Physics, Locomotion, Perception, Behavior," *Proc. Siggraph 94,* ACM Press, New York, 1994, pp. 43-50.
9. Q. Yu and D. Terzopoulos, "Synthetic Motion Capture For Interactive Virtual Worlds," *Proc. Computer Animation,* IEEE Computer Society Press, Los Alamitos, Calif., 1998, pp. 2-10.
10. H. Distler and H.H. Bülthoff, "Psychophysical Experiments and Virtual Environments," *Virtual Reality World 96,* Computerwoche Verlag AG, München, Germany, 1996.
11. J.R. Ensor and G.U. Carraro, "Peloton: A VRML-based Bicycling Simulator," *Visual Proc. Siggraph 97*, ACM Press, New York, 1997, p. 198.
12. R. Waters et al., "Diamond Park and Spline: Social Virtual Reality with 3D Animation, Spoken Interaction, and Runtime Extendability," *Presence: Teleoperators and Virtual Environments*, Vol. 6, No. 4, MIT Press, Cambtidge, Mass., August 1997, pp. 461–481.
13. M.R. Macedonia et al., "NPSNet: A MultiPlayer 3D Virtual Environment Over The Internet," *1995 Symp. Interactive 3D Graphics*, P. Hanrahan and J. Winget, eds., ACM Press, New York, 1995, pp. 93–94.
14. M.H. Raibert, *Legged Robots That Balance,* MIT Press, Cambridge, Mass., 1986.
15. A. Singla, U. Ramachandran, and J.K. Hodgins, "Temporal Notions of Synchronization and Consistency in Beehive," *Proc. 9th Annual ACM Symp. Parallel Algorithms and Architectures (SPAA),* ACM Press, New York, 1997, pp. 211-220.
16. J.K. Hodgins et al., "Animating Human Athletics," *Computer Graphics* (Proc. Siggraph 95*)*, R. Cook, ed., ACM Press, New York, 1995, pp. 71–78.
17. E.R. Burke, *Serious Cycling,* Human Kinetics, Champaign, Ill., 1995, pp. 192-200.

David Brogan *is a PhD student in the Graphics, Visualization, and Usability Center and the College of Computing at Georgia Institute of Technology. He received a BA in mathematics from the University of Virginia. His research interests include simulating dynamic physical systems and modeling high-level behaviors in interactive multiagent environments.*

Ronald Metoyer *is a PhD student in the Graphics, Visualization, and Usability Center in the College of Computing at the Georgia Institute of Technology. He received his BS in computer engineering from the University of California at Los Angeles. His research interests include simulating dynamic physical systems, programming high-level behaviors for interactive characters, and using virtual environments for task training.*

Jessica Hodgins *is currently an associate professor in the College of Computing at Georgia Institute of Technology and a member of the Graphics, Visualization and Usability Center. She received her PhD from the Computer Science Department at Carnegie Mellon University in 1989. Her research explores techniques that may someday allow robots and animated creatures to plan and control their actions in complex and unpredictable environments. In 1994 she received an NSF Young Investigator Award and was awarded a Packard Fellowship. In 1995 she received a Sloan Foundation Fellowship. She is on the editorial board of the Journal of Autonomous Robots, the IEEE Magazine on Robotics and Automation, and ACM Transactions on Graphics.*

Readers may contact Brogan at the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280, e-mail dbrogan@cc.gatech.edu.