Physical and Kinematic Controllers for Virtual Humanoid Animation

Valentina Ercolani, UCLA 2002-2003

Abstract:
The subject of the work presented in this document is the physical simulation of articulated human 3D models inside a realistic 3D environment, where forces influence the character's behaviour and maintaining balance is a not trivial task.
Our work is presented as an extension of previous techniques designed to control the virtual character and produce movements similar to human-like everyday actions.
According to literature, a virtual character can be animated in three main ways: physically, kinematically and interactively.
We implemented each kind of animation, using the theory of composite controllers and producing the movement both automatically and applying user input.
The first part of the developed project is a pose controller for a specific task that brings the virtual character from an initial prone position to lay on his back; the second part is a visual interface to allow user to have a tangible control on the character, defining motions for limbs through the combination of a background physical controller and a foreground inverse kinematic actuator; finally we propose an analytical method to solve the problem of keeping balance during shifting weight on one foot.

Introduction:

The animation of virtual anthropomorphous characters able to perform realistic actions is receiving great interest from the community of computer animators, and several presented works are proof of sensible reached successes.
We took clue for our work by methods to perform good simulations in 3D realist environment illustrated in last years animation Siggraph's papers, as a valid basis for any research in this ambit.
In particular, we followed the work of Petros Faloutsos and Joseph Laszlo about how applying motion controllers on characters and how defining the actions both automatically and manually .
Our primary goals are an easier control of the virtual character and an improvement of the motions' lifelikeness.
The creation process of controllers not involves only a good understanding of motion and timing, but it also require knowledge of how controllers are implemented.
Furthermore, working with physical animation and simulation means deriving motion by forces properly applied on the virtual performer, so we need to exactly know the amount of the physical influence on the character and the correct way to use it according to the controller structure.
For all these reasons controllers development procedure is still prohibitive for a certain percentage of users.
But if the typical user could have interaction with the character without having knowledge of the low-level controller structure, the target community of users could increase and animators and designers without any physical or computer science background could be included.
Moving towards this purpose, we pondered the possibility to derive the parameters needed by the motion controller only by user high-level directions, given by a common input device as mouse or keyboard.
In more details, the user should not think about syntactic correct instructions or work with numerical values but he could produce the planned motion simply selecting a final position on the screen and let the character reach it.
Visual interface can be useful in several other ways:
Movements directly defined could be finalized to get the correct position to activate a controller or to have a status transition. Moreover, taking advantage of the presence of background controllers, we can get different combinations of controllers with the foreground inverse kinematic actuator that manage the visual interface.
With little effort, we could replace the balance controller currently used in background and try the parallelism with any kind of motion controller.
To simplify the management of the virtual creature, we propose also a method to automatically evaluate controller parameters in order to satisfy balance constraints.
The core mathematical method is supported by sensors that can detect the state of the simulation and the position of character links in the 3D environment.
The whole project presented in this document was developed inside DANCE Framework using its simulation features and its plugin-based structure.
We actualized two controllers, one for the rolling action and the other for the visual IK interface: the balance handle system was included in the last one.

Background and Previous Works:

<u>Physically-based animation and DANCE Framework:</u>
A recent new way of thinking animation is producing motion with the application of physic laws in order to integrate

animator skills with good convincing simulations of reality.
In this ambit, our main references have been Siggraph's papers concerning motion controllers and animation of human-like models.
In 2001 Faloutsos et al. [ ] presented a complete framework for creating physically-based environments, both 2D and 3D, in which simulations and animations of anatomically coherent human models can be performed.
This architecture, named DANCE (Dynamic Animation and Control Environment) was our development platform.
DANCE has a portable an extensible structure that allows adding new features to the core system, sharing code with others users, and lets developers and animators work on a higher implementation level, providing all the routines to handle views, cameras and input devices.
We used the control composition framework implemented inside DANCE to acquire experience with control techniques and to create our motion controllers: for "controllers" we intend proportional-derivative controllers (PD) that compute the joint torques by the law T = Ks(qd-q) - Kdq'.
Each motion controller is a plugin inside DANCE and inherits his characteristics from the class "Actuator", where actuator is everything that can exert force or interact with system under simulation .
We consider controllers as black boxes able to detect the current state of the simulation, valuate the variables related to the character and perform an adequate response.
Each controller is aimed to a specific task and the pool of controllers is managed by a Supervisor Controller, that loops through the pool and return the first controller that can handle the specific situation. Supervisor Controller also evaluates the value returned by active controller to establish its success or failure.
Having full access to the information related to any system in DANCE, controllers can define arbitrary sensors in order to obtain all the necessary information concerning the character and the environment.
In our tests we have made extensive use of sensors to activate transitions or to detect validity of a state.
Finally, the model used in our simulations is an anatomically coherent 3D skeleton, with 16 links and 37 degree of freedom (DOF), six of which correspond to the global translation and rotation parameters.
This model is implemented as DANCE System: for any system under simulation we can compute forces and define states. Systems have also dynamic proprieties such as mass and moment of inertia, and we can apply particular constraints and limits to them.

Interactively Animation:
We implemented a visual interface to allow some puppetry-like features combined to the physical simulation.
The goal is to give the users an intuitive interaction with the character, similar to a puppeteer with his marionette.
In computer animation this idea may correspond to perform a mapping from character DOF to input devices DOF and define motion as key-frame animation.
Our starting point was paper presented by Lazslo et al. [ ] at Siggraph 2000 that gives an interesting overview about this kind of mapping, and proposes some techniques to build interfaces for interactively-controlled animated characters.
As explained in this paper, to build an interface with those kind of features, we have to find a way to manage a large number of DOF with a normal input device (i.e. mouse, keyboard,..).
Assuming that user can not manage the large number of DOF associated to human models, we reduced the complexity of the problem considering only arms and legs movements and we used inverse kinematic techniques to calculate them: the interaction with the whole body can be derived by extension and with the aid of suitable devices.
Lazslo also advises about the simulation rate: we want user interaction so the simulation rate must be adequate. If it is too slow, user may lose the connection to the motion and he could miss the sense of timing associated to it; on the other hand, if the motion is too fast, it become impossible to interact efficiently with the virtual creature.
We also focused on the use that Lazslo made of keystrokes and inverse kinematic (IK) techniques in his work.
An interface built using IK to establish desired joint angles is more user-friendly than the classical controller implementation method where users are forced to deal with numerical instructions, directly given inside the controller structure.

Implementation:

1. TurnOnBackController
We implemented a new controller named TurnOnBackController, and we added it to DANCE controllers pool.
This new controller is aimed to emulate the action of a person that is lying faced-down and wants to rolling-over to a supine position.
TurnOnBackController is designed as a DANCE PoseController composed by five successive poses strictly depending each other, so that one pose can occur only once, during the controller time of activity.
Pose controllers work on a finite state machine, in our specific case without any cycle, and normally with time transitions between states; anyway, developers can also let controllers change state according to particular information related to the character and returned by sensors.
We chose PoseController structure because it is pretty useful in modelling complex actions with composite nature, such

as walking, and in general it is helpful to simplify the understanding of intricate time-depending motions.
So we splitted the whole action of reaching the supine position in five poses overall: the first pose brings arms closer to body to be suitable for rolling; the second pose performs the proper effort to rise from ground and to start turning the trunk; the third pose performs the rolling on the hip, while the fourth moves towards the supine position; finally the fifth pose refines the final face-up position placing arms and legs in a natural rest state.
There is also an extra pose, not included in the pose list, that is called as first one and is responsible for placing links into the correct initial position if they are in a not too different one.
Transitions between states are normally accomplished setting a time limit to perform motion currently active.
The only exception is established by the transition between pose one and pose two: we require that both arms are in suitable location to perform the rolling and left forearm is on the ground. Anyway, even in this case a time value is set and the transition occurs only if arms condition is met or the time limit is exceed.

2. IKInteractController

DANCE provides a plugin which makes use of inverse kinematics to derive joint angles for a desired position of character limbs. This plugin is named IKActuatorForce and was conceived and implemented by Gordon Cook. IKActuatorForce is the core of the interface we developed: it solves the inverse kinematic for our angles and its methods are called to handles our IK chains.

IKInteractController is designed to let user control character limbs and move them towards a target selected by mouse.
In more details, we defined four IK chains, two both for arms and legs, with each couple sharing the same root: character trunk is arms root while hip is legs one.
In a future extension we may consider different chains, and let user change interactively their roots and endeffectors. We could also try to manage IK chains defined on more than three links.
Anyway, even if chains and roots are automatically set when the controller become active, user has to choose which endeffector he wants to move: each root can have only one endeffector active at time.
Keystrokes are used to perform endeffector selection and to handle the interface:

- q: right hand selected
- w: left hand selected
- o: right foot selected
- p: left foot selected
- a: to actuator mode
- x: to command mode
- t: add stiffness to the system
- d: print all the control strings

3. Shifting body weight

Finally, to allow the character to perform steps, it is necessary shifting the body weight on one foot and leave the other endeffector free to move.
Considering all the character links position in space, the goal is meeting the balance condition: the centre of mass projection has to lay inside the portion of ground delimited by feet (Support Polygon), at every simulation instant.
We can calculate the centre of mass position as weighted average of the origins of all the articulated character links; that means it changes every time the character moves in space.
Given that, what we achieved is a mathematical method to keep the centre of mass projection on the line between feet (with an acceptable error), for all the time needed to perform the weight shifting, and ideally reach the centre of the target foot at the end of the simulation.

Conclusions and Future Works

The work presented is easily extendable in different ways: in particular the interaction between physic and kinematic controllers, presented with IKInteractController, can be tested more in depth, considering different behaviours and enhancing the user level of interaction.
Replacing the keyboard interface with a more interactive device, such as joy-pads or game controllers, would give more flexibility to the system and it would be possible testing the system on combined behaviour of both upper and lower body at the same time.