Eric Singer, Athomas Goldberg, (.us)
Ken Perlin, Clilly Castiglia, Sabrina Liao

Media Research Laboratory New York University

IMPROV: INTERACTIVE IMPROVISATIONAL ANIMATION AND MUSIC

Introduction

Improv is a system for the creation of real-time behavior-based animated ac tors. There have been several recent efforts to build network distributed a utonomous agents, but in general, these efforts do not focus on the author 's view. To create rich interac tive worlds inhabited by believable animated actors, authors need the prope r tools. Improv provides tools to create actors that respond to users and t o each other in real-time with personalities and moods consistent with the author's goals and intentions.

The character animation system in Improv consists of two subsystems. The first is an Animation Engine that uses procedural techniques to enable author s to create layered, continuous, non-repetitive motions and smooth transiti ons between them. The second i s a Behavior Engine that enables authors to create sophisticated rules gove rning how actors communicate, change and make decisions. The combined syste m provides an integrated set of tools for authoring the 'minds' and ' bodies' of interactive actors.

Recent development has added audio and musical features to the Improv system. Known as Improv Musique, this development work focuses on two areas. Interactive Virtual Musicians and Dancers; and Audio Support for Virtual Environments Features include actor speech, voice recognition, controllable music, environmental sound and user input from external devices and video cameras.

This paper begins with an overview of the Improv animation and authoring system, followed by the music and audio features, and concluding with a description of recent demonstrations and installations created with Improv.

Authoring Animation in Improv

As an authoring system, Improv must provide creative experts with tools for constructing the various aspects of an interactive application. These must be intuitive to use, allow for the creation of rich, compelling content an d produce behavior at run-time which is consistent with the author's vision and intentions Animated ac tors must be able to respond to a wide variety of user interactions, in way s that are both appropriate and non-repetitive This is complicated by the fact that in applications involv ing several characters, these actors must be able to work together while fa ithfully carrying out the author's intentions. The author needs to contro l the choices an actor makes and how the actors move their bodies.

Architecture

The model used by Improv consists of an Animation Engine which utilizes des criptions of atomic animated actions (such as Walk or Wave) to manipulate 3 D models, and a Behavior Engine which is responsible for higher-level capab ilities (such as going to the store or engaging another actor in a conversation) and decisions about whic h animations to trigger In addition, the Behavior Engine maintains the int ernal model of the actor, representing various aspects of an actor's mood s, goals and personality In a s ense, the Animation Engine represents the 'body' of the actor while the Behavior Engine constitutes its 'mind'.

Animation Engine

The Animation Engine provides tools for generating and interactively blending realistic gestures and motions. Actors are able to move from one animated motion to another in a smooth and natural fashion in real time. Motions can be layered and blended to convey different moods and personalities.

The author defines an action simply as a list of joint rotations together with a range and a time varying expression for each. Most actions are constructed by varying a few of these over time via combinations of sine, cosine and coherent noise (controlled randomness). For example, sine and cosine signals are used together within actions to impart elliptical rotations. Using coherent noise in limb movements allows authors to give the impression of naturalistic motions without needing to incorporate complex simulation models. The author can also import keyframed animation from commercial modeling systems such as Alias or SoftImage. The Improv system internally converts these into actions that specify time varying values for various joint rotations or deformations. To the restof the system, these imported actions look identical to any other action Behavior Engine

Because the user is a variable in the run-time system, Improv authors cannot create deterministic scenarios. The user's responses are always implicitly presenting the actor with a choice of what to do next. Because of this variability, the user's experience of an actor's personality and mood must be conveyed largely by that actor's probability of selecting one choice over another. The behavior engine provides several authoring tools for guiding an actor 's behavioral choices. The most basic tool is a simple parallel scripting system in which individual scripts, like actions, are organized into group s of mutually exclusive behavior. However, unlike actions, when a script within a group is selected, any othe r script that was running in the same group immediately stops. In any group at any given moment, exactly one script is running. Generally speaking, at any given moment an actor wil l be executing a number of scripts in parallel. In each of these scripts, t he most common operation is to select one item from a list of items These items are usually other scripts or actions for the actor (or for some other actor) to perform.

The author must assume that the user will be making unexpected responses. F or this reason, it is not sufficient to provide the author with a tool for scripting long linear sequences. Rather, the author must be able to create layers of choices D0 from more global and slowly changing plans to more localized and rapidly changing act ivities D0 that take into account the continuously changing state of the a ctor's environment and the unexpected behavior of the human participant.

Individual Scripts

A script is organized as a sequence of clauses. At run-time, the system run s these clauses sequentially for the selected script in each group. At any update cycle, the system may run the same clause that it ran on the previou s cycle, or it may move on to the next clause. The author is provided with tools to 'hold' clauses in response to events or timeouts.

The simplest thing an author can do within a script clause is trigger a spe cific action or script, which is useful when the author has a specific sequ ence of activities (s)he wants the actor to perform. In addition to command s that explicitly trigger spec ific actions and scripts, Improv provides a number of tools for generating the more non-deterministic behavior required for interactive non-linear app lications.

In Improv, authors can create decision rules which take information about a n actor and its environment and use this to determine the actor's tendenc ies toward certain choices over others. The author specifies what informati on is relevant to the decision a nd how this information influences the weight associated with each choice. As this information changes, the actor's tendency to make certain choices over others will change as well.

Coordination Of Multiple Actors

An author can coordinate a group of actors as if they were a single actor. We do this by enabling actors to trigger each other's scripts and actions with the same freedom with which an actor can trigger its own. If one acto r tells a joke, the author may w ant the other actors to respond, favorably or not, to the punchline. By hav ing the joke teller cue the others actors to respond, proper timing is main tained even if the individual actors make their own decisions about how exa ctly to react. In this way, an actor can give the impression of always knowing what other actors are doin g and responding immediately and appropriately in ways that fulfill the aut hor's goals.

User Interaction and Multi-Level Control Of Actor State

One important feature of Improv is the ability for the user to interact wit h the system at different levels. This means that the author can give the u ser the right kind of control for every situation. If the user requires a v ery fine control over actors' motor skills, then the author can provide direct access to the action level . On the other hand, if the user is involved in a conversation, the author might let the user specify a set of gestures for the actor to use and have the actor decide on the specif ic gestures from moment to moment.

At an even higher level, the author may want to have the user directing lar ge groups of actors, such as an acting company or an army, in which case (s)he might have the user give directions to the entire group and leave it to the individual actors to carr y out those instructions. Since any level of the actor's behavior can be made accessible to the user, the author is free to vary the level of contro l as necessary at any point in the application.

Improv Musique

Over the past year, we have been adding music, audio and user-input feature s to Improv. As a result, Improv now provides facilities for adding actor s peech with lip synching, voice recognition, ambient background sound and ef fects, controllable music sequ ence playback, singing synthesis and user input from external devices and video.

Implementation

Improv Musique is made up of several components. Features are implemented o n both Macintosh and UNIX machines which communicate across a local area ne twork. Much of the audio system is implemented in OpcodeAA MAX, a visual programming environment for the Macintosh used primarily for MIDI applications. In the Musique system, we use MAX for receiving and filtering data from input devices; playing and proc essing MIDI and digital audio files; interfacing with Macintosh voice recog nition facilities; and network communication with actors.

MAX programs (called 'patches') are the central input, output and contr ol point for Musique features. Custom MAX external objects, written in C, e nable digital audio playback, MIDI file