

IMPROV: A System for Real-Time Animation of Behavior-Based Interactive Synthetic Actors

Athomas Goldberg
Media Research Laboratory
New York University
New York, NY 10003
athomas@mrl.nyu.edu

1 Introduction

The IMPROV Project at NYU's Media Research Lab is building the technologies to produce distributed 3D virtual environments in which human-directed avatars and computer-controlled agents interact with each other in real-time, through a combination of Procedural Animation and Behavioral Scripting techniques developed in-house. We are also exploring multi-modal interaction paradigms combining traditional forms of input (keyboard and mouse) with speech and gesture recognition in conjunction with various forms of presentation, including 2D and 3D display. The system is intended to operate over Local and Wide Area Networks using standard internet protocols, enabling anyone with access to the World Wide Web to develop or participate in fully interactive, virtual experiences.

2 “The Mind-Body Problem” – Procedural Animation and Behavioral Scripting

Using traditional computer animation techniques, variations in animated motion, and the transitions to and from these motions, must be plotted out ahead of time. A job that is generally time-consuming, therefore constraining the animator to an extremely limited set of possible behaviors. Procedurally animated characters are able to automatically generate transitions between animated motions in a smooth and natural fashion in real-time. In addition, motions can be layered and blended to convey an extremely wide range of behavior, mood and personality. Actors can be given the form of humans, animals or animated objects, and actors with common components can share sets of animated behaviors.

In IMPROV, an action is defined as a single atomic or repetitive activity, one not requiring explicit higher-level awareness or conscious decisions. For example, walking is an action, as is typing, but walking to the computer and beginning to type constitutes a sequence of several actions.

An actor is not limited to performing a single action at once though. In order to appear life-like and believable, an actor must be able to perform certain activities simultaneously, like walking and chewing gum, or typing, while talking to someone across the room. Once again, it would be impractical to suggest that the animator should create a separate animated behavior for every combination of activities.

The problem with combining actions automatically is that there are often different ways in which actions might be combined. An actor might be instructed to momentarily scratch his head while waving at someone. It would not be very realistic to have the actor try to perform both these actions at once, making vague scratching gestures toward his head

while simultaneously making awkward waving motions at the other actor. On the other hand, if we wanted the character to wave while walking downstage, it would not make much sense to have the actor stop walking each time he tried to wave.

The difference between these two examples is that in the former case we're describing actions that are mutually exclusive, whereas in the latter case, the two actions are complementary and can gracefully coexist. What's important here is that there be a clear and powerful structure in which these action relationships can be defined.

We facilitate this structure by dividing actions into Groups. Actions within a group are mutually exclusive of one another; activating one causes the action currently active to be deactivated. Actions in different groups may operate simultaneously, allowing actions involving a specific part of the body to be combined with or layered over those involving others. In general, actions involving a specific body part are generally layered over those involving a region of the body, which are in turn layered over those involving the whole body. For example: one actor may be talking to another, one group may contain stances the actor assumes, such as shifting his weight from side to side, standing straight or sitting. At the same time another group contains a set of upper-body motions like crossing his arms or putting his arms at his sides or behind his back. Still another group will contain those gestures performed with the right or left hand, like gesturing to make a point or scratching his chin.

Using this structure, these few actions can be combined to create dozens of composite animations while minimizing the risk of inadvertently creating behavior that is either unbelievable or not life-like.

It is also important that when an actor goes from performing one action to another that the actor's body does not go through physically impossible transitions. If an actor has his arms behind his back, what's to prevent him from passing his arms through his body? In the "real world", we're bound by the laws of physics, but this is not the case in the virtual environment. By allowing animators to create buffer actions, we provide an easy way to avoid unbelievable behavior. For example: if we use hands-at-sides as a buffer action for hands-behind back, we save the actor the embarrassment of accidentally passing his arms through his body, by declaring that every time the actor wants to put his arms behind his back he will first put his arms at his sides. Likewise, when the actor tries to go from having his hands behind his back to another action involving his arms, he'll once again execute the arms-at-sides gesture before continuing to the next gesture.

Actions are only useful for defining and blending subtleties of physical motion. Behavioral Scripting allows you to describe behavioral "scripts" to create more complex sets of actions and behaviors. Scripts may be used to trigger other scripts and/or actions enabling virtual world designers to easily construct hierarchies of behavior, from low-level decisions about the types of gestures and actions to perform, to high level behavior, describing an actor's goals and the kinds of social activities the an actor will partake in. a "Life to Picking Your Nose" model of behavior in which there is a continuum from the most abstract description of character activity to the construction of specific motor skills. In this way, long-term goals and activities can be broken down into component behavior and further broken down until individual physical activities are described.

The smallest unit of time in IMPROV is called a frame, which is approximately 1/30th of a second or the length of a single frame of animation. A basic script generally

consists of a sequence of steps that are executed in order, one per frame, though sequencing commands can be used to cause steps to be repeated.

These sequencing commands include:

- “wait”: causes the current step to be repeated for the specified number of seconds. When it appears in a step by itself, it serves as a timed delay between steps.
- “while”: causes the current step to be repeated until the specified conditions are met.
- “loop”: causes the current step to be repeated indefinitely, or until the script is deactivated.
- “do again”: causes the current script to repeat from step 1.

Like Actions, scripts are organized into parallel-running Groups. Scripts within a given Group are mutually exclusive of each other: activating a script within a group immediately deactivates whatever script is currently active in that Group. Scripts active in different Groups operate simultaneously, enabling high-level scripts to continue as the low-level behaviors they trigger are performed. In this way, the actor can address his environment from multiple levels of abstraction, maintaining high-level goals and long-term activities while managing and carrying out the short-term behaviors necessary to fulfill these. A simple script might look something like this:

```
SCRIPT "Greeting"  
  steps  
    1. Enter room  
    2. wait 4 seconds  
    3. Turn to Camera  
    4. wait 1 second  
    5. Wave to Camera, Say Hello to Camera  
    6. wait 3 seconds  
    7. Leave room
```

In this example, the actor first activates the “Enter Room” script (which instructs the actor to enter the room). The “Enter Room” script is in a different script-group, so the “Greeting”: script continues as the “Enter Room” script is executed, waiting four seconds before activating the “turn” which tells the actor to turn and face the specified target, in this case the camera. There is another pause of 1 second, giving the “turn” script time to finish, before instructing the actor to begin the “Wave” and “Say Hello” scripts. Another 3 seconds pass as the actor waves and says hello, at which point the “Leave Room” script is activated, causing the actor to turn and leave.

Two important features of believable human behavior are timing and coordination. While it may be possible to build animated agents with sufficient intelligence to identify and react quickly to rapidly changing events and situations, success along these lines has, thus far, been limited when applied to complex coordinated social behavior. In order to create life-like social scenarios in which multiple actors play out rich interactions, it’s important to provide tools that will enable virtual world designers to not only design the behaviors of an individual actor, but also those of groups of actors.

The “cue” command enables actors to cue each other to respond in appropriate ways at the right moments. An example:

SCRIPT "Hello-Hello"

steps

1. Turn to my target
2. wait 1 second
3. Wave to my target
4. cue my target to Wave to me
5. wait 1.5 seconds
6. Turn to Camera, cue my target to Turn to Camera
7. wait 1 second.
8. Wave to Camera, cue my target to Wave to Camera

In this example the actor turns to face his or her target (the actor they are currently interested in), waits a second and then waves. The actor's target turns, and after a second, waves back. A couple of seconds later, the two turn simultaneously toward the camera and a second later wave in unison.

3 "No! Go THAT Way!" – Navigating Virtual Worlds

While games like "Street Fighter" and "Mortal Combat" give you a pretty direct control over the actions of your avatar, there are times when it may be preferable to direct your animated representative through a more intuitive set of instructions and then let the character play out the scenario in a naturalistic and dramatic fashion. It's not uncommon for new players of games like "Doom" to make several attempts before successfully passing through a doorway or turning a corner. An example of high-level instruction might involve pointing to a door, and telling the character to open it. The character would then cross to the door automatically, avoiding obstacles in the way, and open it.

In the same way that scripts may be triggered from other scripts, they may also be activated by human participants from the user-interface. The same behaviors displayed in the "Greeting" example, might also have been effected by a human participant directing the avatar to enter the room, waiting a few seconds and then directing the avatar to turn to the camera. Likewise the participant might have activated the "Greeting" script as part of a more complex situation, in which the exact physical motions of the actor are not what's important, but rather, that a specific point (saying hello) is made.

The level of instructions you give the avatar may vary depending on the situation, ensuring the most effective interaction with the environment at all times. The layered structure of IMPROV enables the human participant to direct the actor at whatever level of control is most appropriate from moment to moment.

4 "The Indigenous Peoples of Cyberspace" – Autonomy for Autonomous Agents

While it may be possible to create virtual worlds entirely populated by human-directed avatars, it's often useful, especially in the case of interactive fiction environments, to have a supporting cast of characters whose purpose is to keep the drama moving forward, provoke conflict, introduce new situations and stimuli and otherwise maintain the level of excitement in the scenario. In addition to the techniques described above for performing the appropriate animation for any given activity, these actors must also be able to choose an appropriate course of action from the set of myriad possible actions available at any given time. Once again, we could try to envision all the possible

situations the agents might find themselves in, and write appropriate action scripts, but the amount of work involved in ensuring the characters always did the right thing without becoming repetitive would rapidly become prohibitive. On the other hand, randomly choosing from a set of possible actions available to an actor is not particularly believable. The decisions a character makes should somehow reflect that character's role or personality; and certain actions will seem more appropriate to one character than to others. Without going too deeply into a character's deep motivations and the "hard" problems of true human-like cognition, we can describe a character's personality in terms of their tendency to behave in a certain manner that's consistent from one situation to the next. When confronted with a new situation, reckless characters will tend to choose reckless actions, whereas cautious characters will be likely to choose cautious behaviors. At the same time, to remain believable, autonomous agents must be able to appear at least as unpredictable as their human-directed counterparts. We can accomplish this through the use of what we call "tunable statistics." Characters make weighted decisions where certain options have a higher chance of being chosen than others.

Take the "Fidget" script for example:

```
SCRIPT "Fidget"  
  steps  
    1. Choose from  
      (Twiddle Thumbs .7, Scratch Head .3, Pick Nose .2)  
    2. wait 3 seconds  
    3. do again
```

In this example: we give an actor a choice of three actions to perform, say, twiddling his thumbs, scratching his head, or picking his nose, and to each of these we assign a weight between 0 and 1, so that twiddling is given a weight of .7, scratching is given a weight of .3 and picking is given a weight of .2. In this case, there is a 7 in 12 chance the actor will choose to twiddle, a 3 in 12 (or 1 in 4) chance the actor will choose to scratch and a 2 in 12 (or 1 in 6) chance the actor will pick.

As we'll discover later, these weights can be determined procedurally, through the combination of one or more actors' personalities with the objects and events in a scenario. For example, an agent entering a social situation, might be most likely to talk with agents or avatars they are familiar with and have a high sympathy for. This does not mean that the actor will always choose the person they have the highest sympathy for, nor does it mean that the character will never talk to characters they have a low sympathy for, but, like "real" people, will generally be drawn to certain characters over others. These techniques allow the agents to always make decisions that are "in character" without ever becoming repetitive or inhumanly predictable. The way these attributes that shape the decision-making process are defined is described in the next section.

5 "I Feel Like a Number" Personality Definition for Agents and Avatars

In addition to the scripts an actor can perform, each actor is also assigned a set of behavioral attributes used in giving the impression of a unique personality. Some examples might include: strength, coordination and health which would govern how a character performed certain physical activities, and intelligence aggressiveness and amiability which would affect how the character interacts with other characters.

Attributes can also be used to describe the relationships between actors, objects and behaviors. A character's behavior toward another character may be influenced by their sympathy (or hatred!) for that character. Attributes may be used to describe an actor's knowledge of a given subject or skill at performing certain activities. Some examples of actor attributes might include:

```
ACTOR "Gregor"
  attribute          value
  Intelligence       .99
  Amiability        .5
  Strength          .33
  Coordination       .3

  Sympathy-toward Otto      .15
  Sympathy-toward Gabrielle .9

  Knowledge-of Ancient Greece .75

  Skill-at Public-Speaking .9
  Skill-at Driving:Cars    .25
  ...
```

Attribute values are generally in the range of 0 to 1. Here we see that Gregor is extremely intelligent, of average amiability, but not very strong or coordinated. Gregor dislikes Otto, but is extremely fond of Gabrielle, has a pretty good knowledge of Ancient Greece and while being a fine public speaker, is only a mediocre driver. These attributes are all dynamic and may be assigned and modified at any time, and the animations an actor performs will reflect these changes. When a character encounters a prop for the first time, the actor's attribute list is searched for the relevant skill. For example, Gabrielle gets on a motorcycle for the first time. In this case, we look for the "Skill-at Driving:Motorcycles" attribute. If the attribute is not found it is assigned to the actor at some minimal level. This starting level may be modified based on other characteristics. (A character's Intelligence, Coordination, and/or skill in driving other types of vehicles may influence how well the character drives a motorcycle the first time on it.) As the character spends time riding the bike, the motorcycle driving skill value will increase and the animation will reflect the increased proficiency. Next time the character comes across a motorcycle, the actor will again be searched for the "Skill-at Driving:Motorcycles" attribute and this time the value is found. She now appears to drive like an experienced biker.

We can also use these mechanisms to simulate the effects of the actor's behavior on the rest of the world. For example:

```
SCRIPT "Oh... Excuse Me!"
  steps
  1. Belch
  2. wait 1 second
  3. set other-actors Sympathy-toward me to .01
  4. ...
```

In this crude example, the actor makes an obnoxious noise, which has an almost immediate and profound effect on the other actors' attitudes toward the actor, which will, no doubt, be demonstrated in their future behavior toward the actor.

In addition, attributes can be attached to the scripts a character follows which are used to determine how and when an actor will perform that script. These properties might include: prerequisites for performing the script, level of difficulty, or the formality/informality of the behavior. For example:

```
SCRIPT "Ride Bicycle"
  attribute      value
  Activity-Type  (transportation, recreation)
  Prerequisites  (have bicycle)
  Difficulty     .32
  Physicality    .73

  steps
  1. Get on bike
  2. ...
```

In this example, "Ride Bicycle" is a script that may be used for transportation or recreation. Executing the script requires that the actor has a bicycle. The difficulty attribute indicates that riding requires a certain amount of skill, and the physicality attribute shows us that riding is a somewhat strenuous activity. When the actor executes this script, a comparison between these attributes and the attributes of the actor will be used to determine how the animation for riding the bicycle will be executed.

In addition, scripts can be used to monitor and effect other actor states and properties, allowing the behavior of the actor, and the effects of that behavior to influence the actor's personality which in turn may influence the actor's behavior under varying social conditions and changing circumstances. In this way, designers can carefully orchestrate the ways in which an actor will appear to grow and change over time due to their actions and the effects of their actions.

6 "What to Do, What to Do..." – Decision Making in Autonomous Agents

Much of what makes a character believable comes from the choices they make, and how much these choices reflect that character's personality, goals, tendencies, likes, dislikes, etc. As we described earlier, we can control an actor's tendencies toward certain choices in a decision using weighted lists. For example, in the script "Go to Store" the actor is going shopping and has to decide how he's going to get there. We could write it something like this:

```
SCRIPT "Go to Store"
  steps
  1. Exit Home
  2. Choose from
     (Walk .1, Take Bus .5, Take Cab .4, Ride Bike .2)
  3. Enter Store
  4. ...
```

In this example, the actor executing this script is most likely to take the bus or a cab, but will occasionally ride his bike, and on rare occasions walk. The problem with this is that every actor executing this script is going to have the exact same tendencies toward each of these modes of transportation, even though in the real world, a lazy person might *never* walk or ride a bike, an athletic person might prefer riding or walking and a person on a strict budget might avoid taking a cab, if at all possible. It would be impractical to try and create a separate “Go to Store” script for every actor, just to reflect that individual’s personality, since there may be thousands of different actors in the virtual world and “Go to Store” is only one of hundreds of activities an actor might engage in.

An alternative to this is to specify a set of criteria an actor or set of actors will use in making a decision. These are, in effect, the rules governing how a specific decision gets made, which each actor will interpret differently based on their individual personalities. Using criteria-based decision-making, our “Go to Store” script might look something like this:

```
SCRIPT "Go to Store"
  steps
  1. Exit Home
  2. Choose from (Walk, Take Bus, Take Cab, Ride Bike)
     using criteria "best way to get there"
  3. Enter Store
  4. ...

CRITERIA "best way to get there"
  criteria                                     weight
  1. my energy compared-to its physicality     .7
  2. my wealth compared-to its cost           .3
```

In this example, energy and wealth are attributes of the actor, while physicality and cost are attributes assigned to the various options. In this case, each option is assigned a weight based on how close these attributes match. The value under `weight` represents how much influence each of these criteria has on the final decision, in this case the amount of energy required by each choice is the most significant factor, but how much it costs also plays a small role. In this example, an actor with a high energy attribute will tend toward the more physical activities like walking and cycling, especially if they are low on cash, whereas lazy actors will choose the less physically demanding choices, taking a cab if they have lots of money, taking the bus if they do not.

We might also want to be more abstract about the scripts we’re choosing from. In this case, the actor might not necessarily be concerned with choosing one of these specific activities, but rather is only interested in finding an appropriate mode of transportation. Here we accomplish this by choosing from the list of scripts that have some attribute in common (they all have transportation as one of their activity-types)

```
SCRIPT "Go to Store"
  steps
  1. Exit Home
  2. Choose from
     (scripts with Activity-type: transportation)
     using criteria "best way to get there"
```

3. Enter Store
4. ...

CRITERIA "best way to get there"

criteria	weight
1. my energy compared-to its physicality	.7
2. my wealth compared to its cost	.3

This way, should we later wish to add an "in-line skate" script, it will automatically be included in the decision as long as we make "transportation" one of its activity-types.

We can already begin to see how we can create a wide variety of interpretations of the "Go to Store" script, without greatly increasing the amount of work necessary to accomplish this. Still, in this example each actor will assign the same importance to physicality and cost, which while reflecting an actor's current state, does not really tell us anything about the actor's values, another important aspect of behavior.

If we rewrite the "best way to get there" criteria as follows:

CRITERIA "best way to get there"

criteria	weight
1. my energy compared to its physicality	my importance-of physical-enjoyment
2. my wealth compared to its cost	my importance-of money

we enable the actor to decide how important each of these factors is. In this case, actors who place a lot of importance on physical enjoyment will tend toward activities that best suit their current level of energy, whereas those that place more importance on money matters will tend toward those activities that meet their current means.

In the same way attributes may be assigned to actors and scripts, attributes can also be assigned to the criteria an actor uses to make a decision, enabling the virtual world designer to tailor a script to an even wider variety of personalities and interpretations. Perhaps our athlete does not care at all about the cost of getting there, he just wants the exercise. Meanwhile, individuals that are more decadent refuse to do anything that does not display their wealth or requires too much effort. In this case, our script might look something like this:

SCRIPT "Go to Store"

- steps
 1. Exit Home
 2. Choose from (scripts with Activity-type: transportation)
using criteria (Choose from (Criteria-type: attitude)
using criteria "Personality")
 3. Enter Store
 4. ...

CRITERIA "Personality"

criteria	weight
1. my decadence compared-to its decadence-level	.5
2. my athleticism compared-to its athletic-level	.5

CRITERIA "Decadent"

attribute	value
Criteria-type	(attitude)
decadence-level	.9
athletic-level	.1
criteria	
1. its physicality is very-low	.7
2. my wealth compared-to its cost	.3
CRITERIA "Athletic"	
attribute	value
Criteria-type	(attitude)
decadence-level	.2
athletic-level	.99
criteria	
1. its physicality is high	1

Once again, we've enabled actors with different personalities to play out the same basic activity (going to the store) with their own interpretations. Here, actors at the extremes of Athleticism and Decadence, may almost always prefer certain forms of transportation over others, but because the choice of criteria is based on a weighted list as well, actors whose personalities fall somewhere in the middle will occasionally exhibit different reasons for choosing certain scripts over others. Perhaps today they are feeling decadent while tomorrow they feel more athletic. The frequency with which they choose certain criteria over others will also fit with that actor's personality.

I've presented a very simple example here, but it's clear that as we add more options, and more detailed criteria, we can very quickly develop a rich variety of interpretations of various activities, tailored to each actor's individual personality, goals and values.

So far I've described a broad range of scripted behavior, from linear sequences of scripts and actions, to completely personality-based decision-making. It's important to note that these are in no way exclusive of each other. Fully autonomous agents may interact freely with avatars following human direction and carefully scripted behavior, while avatars receiving high-level instruction may make these kinds of decisions about their low-level behavior. In addition, there are times when an autonomous agent might be required to perform some predefined set of actions, as in a dance, or when repeating something learned by rote, or when called for to effect certain events in an interactive drama. IMPROV gives the virtual world designer the freedom to mix and match these types of behavior and levels of control in order to create the desired effect.

7 "The Tail that Wags the Dog" – Inverse Causality for Animation

Animated actors in dynamic real-time environments are likely to encounter new situations, especially if we allow for worlds that are continuously being developed and characters that persist from one scenario to the next. As circumstances change, it is important that these characters continue to react appropriately even when confronted with new stimuli. In cases where this means encountering some truly foreign object or situation, it may be okay if the actor does not know what to do next. More often, though, these are things that the human directing the avatar is familiar with that are introduced to

the world after its creation. In these situations we want the character to appear to know what they are doing, even though they have, in fact, never been in the situation before. A traditional AI approach might suggest that we build mechanisms into the characters to enable them to analyze new situations and make judgments as to the appropriate course of action to take (in other words, the right kind of animation to perform.) This tends to be compute-intensive and while the number of trial-and-error iterations that are generally involved in this approach may produce a convincing simulation of learning to do something, we do not want the characters to look like their learning how to drink a beer, we just want them to drink. Alternately, we could try to take into account everything the character might possibly encounter and program him/her to perform the appropriate animations. The problem with this is that eventually we'll want to introduce new props and situations that we had not foreseen at the time the character was created. At this point it's too late to go back and rewrite every character to account for the new additions.

Take the following example: An actor walks into a bar... (No, this is not the start of some obscene VR joke!) The virtual world designer has just introduced alcohol into the world, and so our actor encounters beer for the first time. He may have already learned the appropriate animations for drinking from a bottle, but say he has not; how do we get the actor to do the right thing? What if, instead of teaching the character how to drink from a beer bottle, we encode the instructions for performing the appropriate animation into the bottle itself? Each item the character encountered would have all the instructions for its use built into it, requiring only that the user direct his or her attention to it in order to enable its use. In one interface scenario, the user points to the bottle with the cursor and is then presented with a number of options for what to do with it (ie. Drink, Throw, Break, etc). The user chooses one of these options and then the animation attached to the bottle drives the character to perform the action. The cause and effect relationship is reversed, but from the user's perspective, the action appears natural.

The advantage of this is that we can continually introduce new elements into the environment, because the animations for the elements' uses are included in their design. This method enables us to have an infinitely expandable variety of possible actions/animations available to the character while reducing the individual character data to a physical description of the character and a set of behavioral modifiers that are used to alter the flavor of the animations, giving each character a unique body language and personality.

In addition to increasing the degree of diversity for animated behavior, the technique also adds a great deal of efficiency by allowing us to automatically filter out all the inappropriate animations for any given set of circumstances. While there may be millions of actions possible for a single character, we only want to deal with those that are relevant to the given situation. When the user directs the character into a new environment, a search is performed to determine which objects are available to the character. At this point the animations, along with any objects (buttons, sliders, etc) used to control the specific activity are loaded. As the character moves from one environment to another, this information is updated and the user is presented with a new set of options.

These techniques are useful for actor interactions as well. That beer bottle may also have encoded in it the effects of drinking beer on the actions of the animated actor. After a few beers, the actor may display difficulty walking or standing, or whatever other effects the virtual world designer may have decided should accompany the drinking of

alcohol. The human participant, upon seeing this, realizes it's time to leave, but as he attempts to direct his avatar toward the door, he accidentally stumbles into another actor, who immediately grows angry and swings a fist at our helpless participant. In the same way that encoding the appropriate behavior into the bottle enables the actor to perform the act of drinking believably, encoding appropriate responses to behavior into the actor initiating that behavior allows us to maintain consistent and believable behavior even between actors with completely different sets of abilities. Our human participant may not have the appropriate fighting animation built into his avatar, but his attacker does. These animations contain, in addition to the set of attack animations, a set of possible responses, including those used by actors who do not know how to fight (like getting knocked across the room).

Not that this is limited only to violent situations. If instead of getting into a fight, our actor was asked to dance he should at least be able to make a clumsy attempt at it, even if he has never danced before. Along with the dance animations used by his expert partner, are those employed by our awkward, but earnest, hero, allowing the two of them to make a believable (though potentially comic) pair out on the dance floor.

8 “Lights, Cameras Action” – Behind the Scenes on the Virtual Soundstage

In the production of any movie, from the biggest Hollywood blockbuster to the smallest independent film, there are a number of people whose contributions make the movie possible but never appear in front of the camera. The director, the cinematographer, the editor, the props, sets and special FX crews and numerous others, all lend their expertise to the creation of the final product and the quality of their work is largely responsible for the success of the film. What if we could bring this talent and expertise to the creation of virtual worlds?

Unlike the movies, the interactive nature of virtual worlds means that the kinds of decisions that are normally made ahead of time during the making of a film, the choice of camera angles, the editing of shots, etc. all need to be made on-the-fly in response to the ever-changing (and unpredictable) actions of the human participants. How can we apply the same autonomous decision-making techniques used for our “actors” to the creation of these off-screen members of our production team?

8.1 Interactive Cinematography

In the “first person point of view” interface used in games like “Doom”, the user has direct control over their movement and actions, but is generally limited to one kind of view, greatly diminishing the number of cinematic possibilities. If we instead make the camera an independent actor in the environment, we open up the possibilities enormously, while freeing up the constraints on the camera. If we treat the camera as another character in the scene, we can give it its own set of behaviors. The camera can react to the action in the scene and adjust the shot accordingly, giving us the kind of real-time cinematography not possible in POV-type games.

8.2 Lighting, Sound & Special FX

Other elements of the environment can be “embodied” as well. Lighting, music, sound and special FX agents can be designed to be constantly responding to the changes in mood and tension, in addition to following cues triggered by the accomplishment of goals or the occurrence of key events in the story. The goal in creating these “backstage agents” is not to replace the creative individuals who specialize in these various cinematic art forms, but rather to allow these individuals to construct aesthetic rules that these agents will follow during the course of the interactive experience.

I want to point out that the use of these agents is not limited to gaming and interactive fiction environments. In MUDs (Multi-User Domains) and other freeform virtual spaces these agents can be used to maintain a certain visual and aural quality to the experience. These agents might also be custom-tailored to the individual. For example, each avatar may have its own soundtrack, audible only that participant, that accompanies them and reflects the changing circumstances surrounding the individual. Each person might view the world through an automated camera that tracks the avatar’s every move, always presenting the most dramatic or interesting camera angle. Lighting and weather agents can provide interesting and unpredictable atmospheres in reaction to events in the world and their own internal rules. All these can contribute to the creation of compelling experiences even within purely social virtual environments.

9 “To Be or Not to Be...” – The Possibilities for Interactive Drama on the Virtual Stage

Now that we have the actors and production crew ready, how do we bring it together into the creation of an engaging and powerful experience? You may be familiar with “branching narratives” in which, at key moments in the story, the player must choose from one of a few options and his/her decision determines how the story proceeds. In these systems, each alternative or “branch” is plotted ahead of time by the author. While we could apply these techniques to systems involving autonomous agents, the restriction of having to know everything in advance negates the advantage we gain from having actors who can continually improvise as they go along under rapidly changing circumstances. At the same time if we simply allow the characters to go about their business in an unstructured environment, there’s no way to guarantee that, as the human-avatar interacts with them in unpredictable ways, the tension will build and the stakes will continue to rise. A participant might fail to provoke the right people or wander around aimlessly for hours, or worse get herself killed right off the bat. In many gaming scenarios, especially those in which combat is the central theme, getting killed is ok, you just start over again and keep going until you get it right, and wandering around aimlessly is often part of finding your way through some maze. In games in which social scenarios play a much greater role, these complications can hinder rather than support the flow of the game. In these cases, we need to introduce one more, “invisible” agent, this time in the role of director. This agent is given the task of maintaining the drama, and is encoded with the rules governing how much time various parts of the story should take and what events need to occur in order to effect this. The point here is not to railroad the participant into following a specific course of action, but rather to ensure that the rest of the world responds to the user’s actions to keep the story progressing. The drama is

continuous, flows seamlessly around the user and, if done correctly, remains engaging and provocative, under almost any circumstance. The ability to create these experiences, once the province of science fiction (Star Trek's Holodeck, et al) is now a matter of creative talent and skill and no longer a limitation of the technology.

10 Background and History

The IMPROV Project began in the Spring of 1994, when Ken Perlin began applying procedural texture synthesis techniques developed in the early eighties to human-like animated characters. The result was "Danse Interactif" in which an animated dancer performs, following the directions of a choreographer (using an on-screen button panel) in real-time. Following the success of these initial experiments, we began applying these techniques to higher-level behaviors and multi-character interactions. To, in effect, allow the characters to make their own decisions, based on a set of pre-scripted rules and carefully controlled statistics governing how and when each character would perform a specific scene or script. Early experiments along these lines resulted in "Interacting with Virtual Actors" Continuing along these lines, we've begun implementing more sophisticated devices which will allow the characters to engage in even more complex behavior and interaction. In addition, we are exploring various multi-modal interaction paradigms and Wide Area Network environments.

11 Group Profile

The Media Research Lab at NYU was formed in the fall of 1993 with Prof. Ken Perlin as its director. The research staff consists of 5 full-time staff members, several graduate students and a number of artist collaborators. Currently the lab is involved in a number of research initiatives. In addition to IMPROV, we are also exploring multi-scale zoomable interfaces and various Web and Internet-related technologies.

12 Acknowledgements

The IMPROV system and all the research described in this paper is the result of a collaborative effort on the part of a number of dedicated artists, animators, programmers and scientists. Principal among these is Professor Ken Perlin, director of the Media Research Lab and developer of the core technology upon which IMPROV is based. Others who've played major roles in the IMPROV's development include Jon Meyer, who wrote most of IMPROV's rendering system, Mehmet Karaul and Troy Downing, who are continuing to develop IMPROV's network-distribution system, Eric Singer, Clilly Castiglia, Sabrina Liao, Andruid Kerne Seth Piezos, Eduardo Santos, Ruggero Ruschioni, Daniel Wey, Marcelo Zuffo, Kuo Chen Lin, Leo Cadavel, Tim Cheung and everyone else who helped make IMPROV possible.

13 Special Thanks

I'd like to thank Steph, just for being Steph.