"It / I" : A Theater Play Featuring an Autonomous Computer Character

Claudio S. Pinhanez

IBM Research, T.J. Watson
Emerging Interactive Spaces Group
P.O. Box 218
Yorktown Heights, NY 10598 - USA
pinhanez@us.ibm.com

Aaron F. Bobick

Georgia Institute of Technology
GVU Center
801 Atlantic Dr.
Atlanta, Georgia 30332 - USA
bobick@cc.gatech.edu

ABSTRACT

"It / I" is a two-character theater play where the human character ***I*** (played by a real actor) is taunted and played with by an autonomous computer character ***It*** on a computer controlled, camera-monitored stage. The play was performed before live audiences in November of 1997 and, for the first time ever, brought an automatic computer character to a theatrical stage. This paper reports the experience and examines important technical developments needed for the successful production of "It / I". In particular we describe the *interval script* paradigm used to program the computer character and the ACTSCRIPT language for communication of actions and goals. Although our experiments have been restricted to physical interactive spaces, we believe that *interval scripts* and ACTSCRIPT can successfully address the control and management of any virtual environment with a complex temporal structure or a strong underlying narrative.

Keywords:

Interactive spaces; computer vision; programming virtual and augmented reality, scripting interaction; synthetic characters, narrative control; story control; computer theater.

1 INTRODUCTION

While the movie and music industries have been extensively using computers for decades, theaters have rarely employed any electronic equipment except for light and sound control, and ticket reservation. Although computers have been experimentally used to generate "electronic puppets" (where a human puppeteer controls a computer graphics character displayed on a stage screen, as pioneered by George Coates and Mark Reaney [1]), we are interested on performances where the computer automatically controls the character constituting what we call a *computer-actor*. Notice that with autonomous computer-actors it is possible to stage plays where a member of the audience can take the place of the human actors and have a first-hand, immersive, personal experience of the universe depicted on the play. This is possible because the performances can be easily repeated as many times as needed, without the participation of human actors. This creates an interesting perspective for theatrical audiences, where the story of a known play can be "lived" from inside — similar to the "rides" in theme parks that are associated with movies.

Although we are using a theatrical performance as our application scenario, it is important to point out that the technical difficulties faced when creating autonomous actors on stage present a significant challenge for the design and construction of robust sensing and control systems. The difficulties are, in fact, very similar to traditional issues on building interactive spaces and system. Moreover, computerized theater forces the investigation and improvement of tools and techniques that have not been addressed by the tele-presence and pervasive computing research communities. For example, the need of constant reworking of the material of a play during its rehearsal requires powerful and easy-to-use systems for defining the behavior of the characters and the space. Also, live performances before an audience requires robustness and reliability quite beyond the level typically present in laboratory-level demonstrations and experiments.

The computer theater play "It / I" was written, produced and performed with the aim of developing and testing tools for interactive immersive systems populated by characters and driven by a story. Unlike previous work involving automated characters (see [2-4]), our work addresses the situation where the actor or user's body is the body of one of characters, and the story is controlled by the computer. This requires the recognition of human actions as they are performed, not as written or clicked-in an interface by a displaced user. Finally, unlike any other previous work with computer actors, "It / I" implemented the autonomous system and run it before live audiences.

This paper starts by describing the concept of computer theater and how the 40-minute play "It / I" transforms the theatrical stage into an interactive environment. We follow with an analysis of the computer system controlling the computer character ***It***, and a brief description of technical aspects of the play. The two main technical developments featured in "It / I" are the *interval script* paradigm used for character and story scripting (based on Pinhanez et al. [5]) and the language for communication between characters, story, and physical devices called ACTSCRIPT. The description of both paradigms stresses the burden placed upon interactive, immersive systems by the need to recognize human action in a physical environment when controlling a complex interactive narrative. We end by describing the performances and further developments in the play and technology. In particular, we argue that interval scripts and ACTSCRIPT are potential solutions for similar problems encountered in the design and implementation of virtual and augmented reality environments with complex sequencing of actions or with an underlying narrative structure.

2 COMPUTER THEATER

Computer theater is a term referring to live theatrical performances involving the active use of computers in the artistic process, (see Pinhanez [6, 7] for a detailed exposition about computer theater, the origins of the term, and related works). The research described in this paper has concentrated on building automatic, semi-autonomous *computer-actors* able to interact with human actors on camera-monitored stages. However, automatic control does not mean pre-timed response. Rather, computer-actors should be built as *reactive* autonomous systems that sense the world, compare the current situation to a script stored in memory, and choose the appropriate line of action. Otherwise, the magic of performance, the energy of live theater, is lost as the human and computer actors are not responsive to each other or to the audience.

One of the most interesting aspects of having autonomous computer actors on stage is that we can have audience interaction in truly novel ways. It is possible, for instance, to control the character's behavior according to the response of the audience. After the conclusion of the actor/computer performance in "It / I", we transformed the stage into an interactive space where members of the audience could re-enact the story of the play, taking on the role of the main character. In this scenario, the play is expanded from the ritualistic happening of the performance into a universe to be explored and experienced by a user[1].

The performance context brings two simplifying factors to the construction of interactive, immersive systems (see also [7]). First, the human actor knows how to interact with the computer character within the limitations of the sensory apparatus. Second, after watching the performance,

[1] We employ the term "user" to differentiate members of the audience from the actors.

the members of the audience transformed in actors have probably learned the basic structure and interaction modes of the play. In this context, system design problems like the learning of the interaction and navigation become less prevalent. Also, the development of the story of the play can be shaped to facilitate the computer recognition of the actor/user activities.

3 *IT/I*: A COMPUTER THEATER PLAY

For testing these ideas one of the authors of this paper — Claudio Pinhanez — wrote the computer theater play "It / I"[2]. The play is a pantomime where one of the characters, ***It***, has a non-human body composed of computer graphic (CG) objects projected onto video screens. The objects are used to play with the human character, ***I***, performed by a real actor on a stage. The computer character ***It*** can also "speak" through images and videos projected on the screens, through sound played onstage speakers, and through the stage lights.

An important aspect of "It / I" is that user/actor interaction keeps the basic structure of the narrative while providing the actor and the audience with a truly responsive computer character. The play follows the "less choice, more responsiveness" paradigm for interactive stories proposed in Pinhanez et al. [8], which downplays the importance of user choice among story alternatives over the use of local responsiveness of characters and the environment. Like in a traditional theater play, the overall development of the story is predetermined and known by all actors, but the details of the actual performance (intensity, gesturing, pauses, audience interplay) are modified each performance based on how the other actors are performing their roles, how the audience is reacting, etc.

---

[2] "It / I" was inspired by three plays by Samuel Beckett, *Act Without Words I, Ghost Trio*, and *Waiting for Godot*.

Figure 1. Scene from "It / I". The computer graphics object on the screen is autonomously controlled by the computer character *It*.

The play was also written considering the sensory limitations of computer vision. That is, actions of ***I*** that had to trigger a response from *It* were restricted to those that the computer could recognize automatically through image processing. In many ways, ***It***'s understanding of the world reflects the state-of-art of real-time automatic vision: the character's reaction is mostly based on tracking ***I***'s movements and position and on the recognition of some specific gestures (using [9]). Figure 1 shows a picture of a typical scene: the actor is on the stage interacting with the TV-like object projected in the screen behind him.

3.1 Physical Setup

Figure 2 depicts a diagram of the different components of the physical setup of "It / I". The sensor system is composed by three cameras rigged in front of the stage. The computer controls different output devices: two large back-projected screens; speakers connected to a MIDI-synthesizer; and stages lights controlled by a MIDI light board.

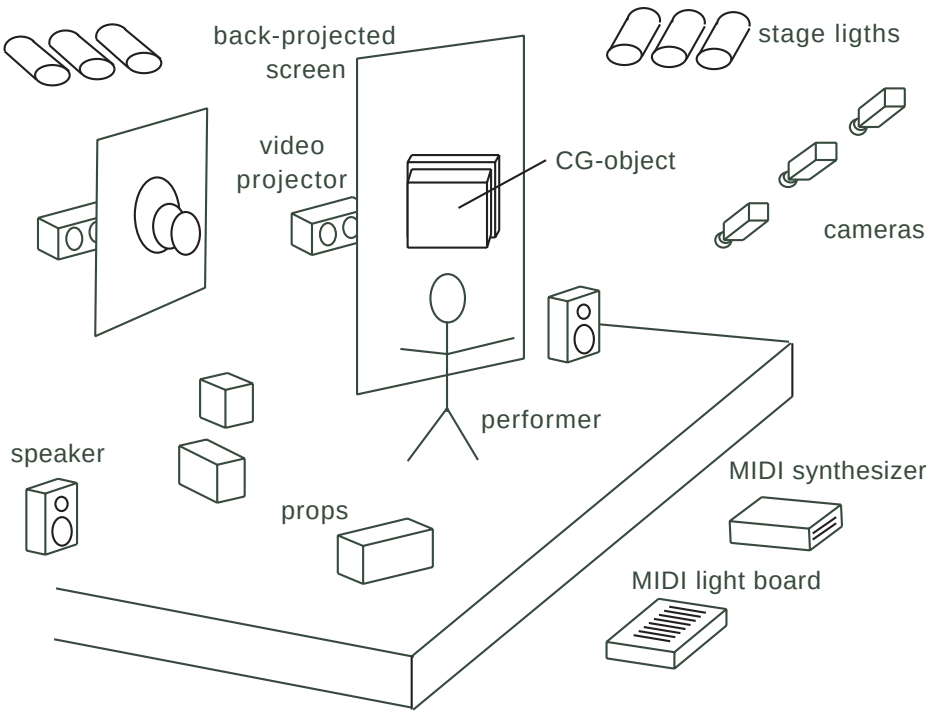

Figure 2. Diagram of the physical structure employed in "It / I".

3.2 The Play

Let us examine the basic plot of the "It / I" in order to access the sensing and controlling challenges faced by the autonomous system. When scene 1 of the play starts, ***I*** is sitting on the center of the stage, distracted by the music played off-stage by a pianist. ***It*** attracts ***I***'s attention by displaying a beautiful image of the sun on the left stage screen. When ***I*** stands up, the image moves away, and a CG-clock appears, running a countdown. ***I*** tries to hide from the imminent explosion, while ***It*** projects a movie showing that the clock can be stopped by a gesture. When ***I*** executes the gesture the clock disappears, immediately being replaced by a new clock. Clocks continue to appear, at a faster rate than ***I*** can stop them. After some time ***I*** gives up and protects himself from the coming explosions (which, in fact, end up not happening). The clocks disappear and the piano music returns.

The basic sensory capacity in scene 1 is the recognition of body shapes. The information that ***I*** has stood up triggers the disappearance of the sun image and the bringing of the clocks. The stopping gestures control the turning off of the clocks and their absence causes the explosion. For maximum responsiveness between the computer and human actors, the number of clocks, the time until they “explode”, and the gesture that stops them are not pre-determined by the script. Instead, whenever *I* succeeds stopping a clock, a new one is created with a shorter time to explosion and/or a faster tick speed. The result is that the human actor has actually to work as fast as he can to stop the clocks, but the gradual shortening of the time to the explosion will eventually make *I* fail to stop a clock. Independently of how many clocks were stopped, scene 1 proceeds as determined in the script of the play. This structure clearly exemplifies our commitment to the “less choice, more responsiveness” paradigm as discussed before.

In scene 2, ***I*** is again instigated to play by an image — a picture of a family. This time, a CG-object similar to a photographic camera appears on the right screen and follows him around. When ***I*** makes a pose, the camera shutter opens with a burst of light and the corresponding camera shutter sound. On the other screen a CG-television appears and, when ***I*** gets close, the television starts to display a slide show composed of silhouette images “taken" by the camera. After some pictures are shown, the camera “calls” ***I*** to take another picture. This cycle is repeated until ***I*** refuses to take yet another picture and stays in front of the television, provoking an irate reaction from ***It***, which throws CG-blocks at ***I*** while flickering the lights and playing really loud noise.

The actions of ***It*** in scene 2 are solely based on the position of ***I***. ***I***'s interest in ***It*** is assumed when he gets close to either the camera or the television. For instance, the refusal to continue taking pictures is detected when the camera “calls" three times and ***I*** does not move from the

front of the TV screen. Although simple, the sensing allows quite flexible interaction in this scene. The number of take-picture-watch-tv cycles is not pre-determined: the human actor decides to refuse to play when he — as a performer — believes the character (and maybe the audience) has reached the emotional potential to do that. Also, there is room for improvisation in terms of when and how the pictures are taken since the camera clicks only when the actor has been still for a few seconds.

In scene 3 the saga between ***It*** and ***I*** continues, with ***I*** being haunted by an agitated, restless electric switch-like CG-object which always keep itself far from the reach of ***I***'s hands. ***I*** is then shown (by video clips) that he can control the switch's position by making special gestures while standing on the top of a block. When he finally gets close enough and seems to be able to turn the switch off, ***I*** discovers that the objects are only projections on a screen. In a Beckettian way, ***I*** tries to hang himself — without success — and provokes another angry reaction from ***It***.

In the beginning of scene 4, ***I*** decides to ignore ***It***, which then brings all the objects to the screen, trying to attract ***I***'s attention. Finally, given the lack of response, ***It*** brings the CG-switch to the screen and turns it off, leaving ***I*** in an empty, silent, dark world.

4 SYSTEM ARCHITECTURE

Figure 3 displays the control architecture used in the performances of “It / I”. It is a 2-layer architecture in which the upper layer contains information specific to the computer character and the bottom layer is comprised of modules directly interacting with the actual input and output devices. This control model is a simplification of the 3-layered architecture, called *story-character-device* architecture, or *SCD*, proposed by Pinhanez [10].

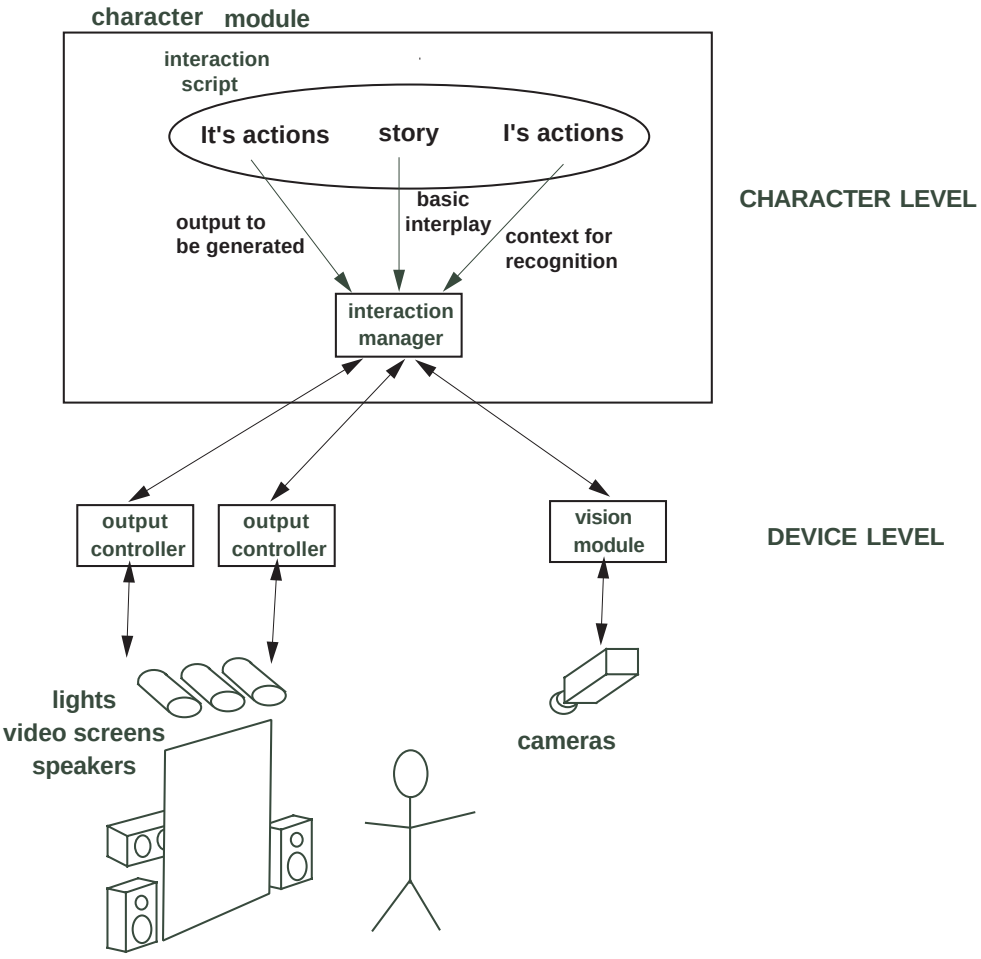

Figure 3. System architecture of "It / I".

As shown in fig. 3, the computer character control system is composed of one active element – the *interaction manager* – that processes the interaction script. The interaction script is separated into three parts: the description of the *story* in terms of actions to be performed by the computer and the human characters; the specification of ***It***'s actions, that is, what the computer character actually does when trying to perform the actions needed by the story; and the description of how the human actor's movements are recognized according to the current moment in the story, or ***I****'s actions*.

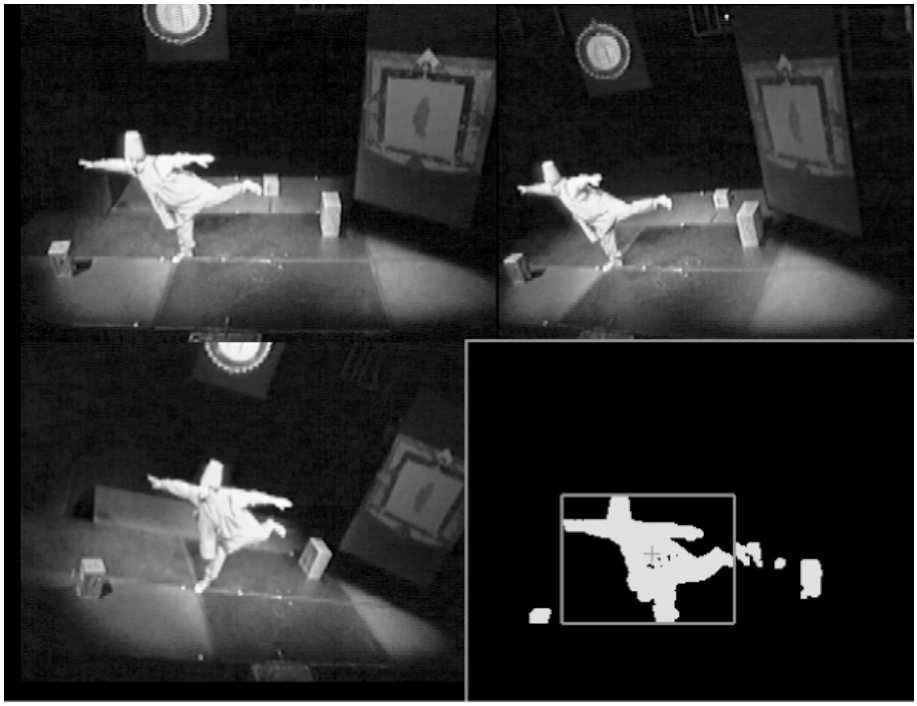

Figure 4. The 3-camera input to the vision module and the computed silhouette of the human actor (enclosed in the rectangle).

In the remaining of this section we describe in more detail two of these device modules since they were implemented based on some innovative ideas for the construction of autonomous interactive computer graphics characters and interactive spaces.

4.1 The Vision Module

The vision module answers queries about the position (x, y, z coordinates) and size of the actor and four large blocks; about number of persons on stage (none, one, more than one); and queries about the occurrence of the pre-trained gestures.

In the performance setup we employed a frontal 3-camera stereo system that is able to segment the actor and the blocks, computing a silhouette image that is used to track and recognize gestures. The stereo system, based on Ivanov et al. [11] work, constructs off-line a depth map of the background — stage, backdrops, and screens. Based on the depth map, it is

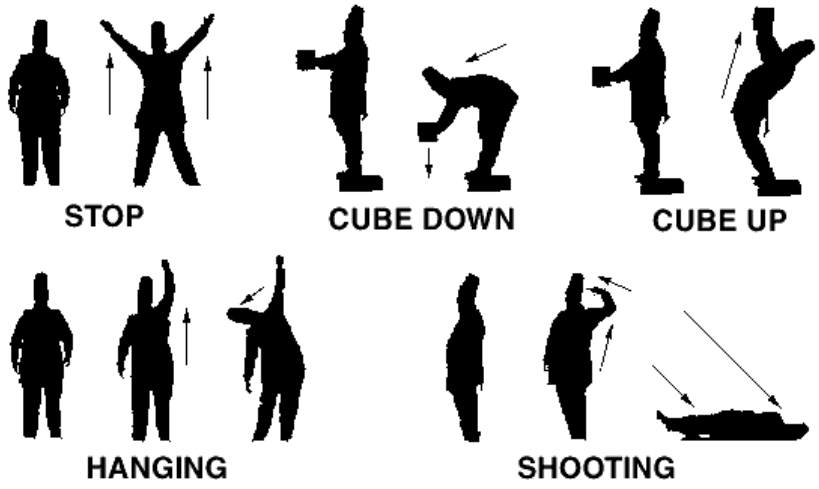

Figure 5. Gestures recognized by the vision module.

possible to determine in real-time whether a pixel in the central camera image belongs to the background or to the foreground, in spite of changes in the background imagery. This is a considerable improvement over background-subtraction vision systems traditionally used in interactive environments [12] since background-segmentation works even when lighting changes and when video is projected on screens or other surfaces. In particular, it enables color lights, light fades, and similar elements of theatrical lighting.

Figure 4 shows a typical visual input to the system and the silhouette found. Using the silhouette, a tracking system analyses the different blobs in the image to find the actor. The analysis is performed under assumptions about the continuity and stability of movement, position, and shape of the actor. Smaller blobs are labeled as blocks.

The vision system is also trained to recognize 5 different static gestures. For this, we employed a simplification of the technique described by Davis and Bobick in [9]. Figure 5 depicts the 5 gestures recognized by the system.

4.2 The Computer Graphics Modules

The computer graphics modules control the generation and movement of the different objects and flat images which appear on the two stage screens. Each CG module basically processes requests (in the ACTSCRIPT format described later) translating them into *Inventor* commands.

In our system, whenever an object or image moves, sound is produced. The importance of sound to enhance image is well known in the movie industry, but, surprisingly, only a few interactive CG characters and environments seem to have really explored this facet of animation. Here we are not talking about the inclusion of sound effects in the post-production phase of CG videos, or about synchronizing speech to talking characters (as in Morishima et al. [13]). Rather, we are interested in the automatic generation of pre-defined sound effects as the objects move around the world.

We implement sound production as short MIDI-files associated to each object's translation and rotational movements through an *Inventor Engine* object. The MIDI-file is sent to the synthesizer — through a request command to the sound module — according to a threshold function describing when the file must be played. Typical examples of such functions used in "It / I" are: a function that plays the sound periodically while the movement lasts; a function that looks for peaks in the velocity; a function that triggers whenever there is a peak in the derivative of the curvature of the path. The last function is used quite successfully to automatically control swiveling sounds when the CG-camera is performing fast rotational movements.

The coupling of sound to animation seems to have played a major role in making the character *It* "alive", compelling, and expressive. Due to limitations in the computers used during the performances, complex animations are not possible. However, we easily controlled the

"mood" of the character (playful, irate, suspicious, etc.) simply by associating sound files that have that particular mood.

5 SCRIPT REPRESENTATION

The major technical novelty in "It / I" is the scripting language used to describe the story, the interaction with the human actor, and the behavior of ***It***. The 40-minute interaction between ***It*** and ***I*** is written as an *interval script*, a language for interaction based on the concept of time intervals and temporal relationships (see [5, 14]

Previous work on scripting languages can be divided into two types. The first deals with languages for scripting movements and reactions of characters, like the work of Perlin [15], Kalita [16], and Thalman [17]. In these cases, the emphasis is on realistic-looking ways of describing human motion with little provision for scripting interaction and sensory input. As mentioned above, our character It is constructed exhibits quite simple movements, so our system did not require advanced futures in this area.

The other type of scripting languages corresponds to languages for description of interaction as for example *Director* [18], or *MAX* [19], and the works of Buchanan and Zelllweger [20], Hamakawa & Rekimoto [21], and Andre and Rist [22]. Those languages, however, lack appropriate ways to represent the duration and complexity of human action in immersive environments: hidden in the structure is an assumption that actions are pin-pointed events in time (coming from the typical point-and-click interfaces those languages are designed for) or a simple sequence of basic commands.

5.1 Interval Scripts

An *interval script* associates a temporal interval to every action in the script. To each interval a label, `past`, `now`, or `future` is assigned during run-time, corresponding to the

situations where the action has occurred, is occurring, or have not yet occurred, characterizing what we call the PNF-state of the action. This is a significant departure of traditional event-reaction based systems which can not distinguish between events that already occurred (past) from the ones that are still to occur (future).

An interval script is a computer file containing the description of each action and statements about how the intervals are related temporally. A complete description of the syntax and semantics of interval scripts is beyond the scope of this paper (see [14] for a detailed description). We opted to present here an example that illustrates some of the main features of the paradigm.

Figure 6 shows the description of five intervals occurring in the beginning of the first scene of "It / I". They control a short segment of the scene where ***It*** brings an image of the sun to the screen and moves it away when ***I*** shows interest on it by standing up.

Interval script files are converted into C++ code through a special compiler that encapsulates the different intervals in function calls. The semantics of each interval is defined by three functions: STATE, a method to compute the current PNF-state of the action; START, a function to start the action or sensing corresponding to the interval; and STOP, a function to end it.

```
%%%%%%%%%% detection of seating %%%%%%%%%%%%
"I is seated"=
{STATE:
   [> if (vision.hasPersonAttribute (SEATED))
        return NOW;
      else return PAST-OR-FUTURE;
   <].
}.

%%%%% bringing the image to the screen %%%%
"bring sun image"=
{START:
   [>
    ScreenCoord centerSmall (0.0,0.0,-1.0);
    ScreenCoord startFrom (0.0,0.0,-40.0);
    leftScreen.move (sunimageId,centerSmall,
                     startFrom,NULL,0,40,
                  "decrescendo");
   <].

 STOP:
   [> leftScreen.stopMove (sunimageId);<].

 STATE:
   [> return leftScreen.moveState (); <].
}.

%%%%%%%% removing the image to the screen %%%
"remove sun image"=
{....  (similar to "bring sun image") ....}.

%%%timer to inforce permanence of image  %%%%
"minimum time of sun image"=TIMER(10.0).

%%%%%%%% the scene with the sun image %%%%%
"sun image scene"=
{
 WHEN "bring sun image" IS PAST
   TRYTO START "minimum time of sun image".

 WHEN "I is seated" IS PAST
    AND "minimum time of sun image" IS PAST
   TRYTO START "remove sun image".

 START: TRYTO START "bring sun image".

 STOP: TRYTO START "remove sun image".

 STATE:
   IF "bring sun image" IS NOW
    OR "remove sun image" IS NOW ASSERT NOW,
   IF "remove sun image" IS PAST ASSERT PAST.

 "bring sun image", "remove sun image",
    "minimum time of sun image"
   ONLY-DURING "sun image scene".
}.
```

Figure 6. Example of an interval script from scene I of "It / I".

These functions can either be written as C++ code (inside the special symbols `[>` and `<]`) or as a combination of previously defined intervals. For example, in fig. 6 the interval `"I is seated"` determines its `PNF`-state by calling a method of the C++-object `vision` that queries the vision module at the device level. *"bring sun image"* `and` *"remove sun image"* define `START` functions calling methods of the C++-object `leftScreen`, sending requests to the CG module of the left screen to move the sun image. The interval *"minimum time of sun image"* is defined by the macro `TIMER`, a C++-object that takes 10 seconds to make the interval go to the `past` state after the interval is started.

The intervals described so far simply encapsulate C++ code. One of the goals of the design of interval scripts was to have easy ways to build more complex intervals from basic ones. The interval `"sun image scene"` is such a case. The definition of this interval describes event relationships between the intervals defined before using the two `WHEN` statements and the interval's own `START`, `STOP`, and `STATE` functions. The first `WHEN` asserts that when the bringing of the sun image is over the timer *"minimum time of sun image"* is started; the second starts to remove the image when ***I*** is not seated anymore, provided that the minimum 10-second period of image exposure is completed.

The syntax of interval scripts allows the definition of start and stop functions in terms of previously defined intervals. Two examples are the `START` and `STOP` functions of the interval *"sun image scene"* which are defined by calls to the `START` function of *"bring sun image"* and *"remove sun image"*, respectively. When the *"sun image scene"* interval is set to start (by a `WHEN` statement not shown in the figure), the executed action is to call the `START` function of the *"bring sun image"* interval, executing its corresponding C++ code. Interval scripts allow complex combinations of start and stop functions of different intervals when describing higher

level functions. The `STATE` of *"sun image scene"* is also defined based on previously defined intervals. If either *"bring sun image"* or *"remove sun image"* are happening `now`, the `PNF`-state of the interval is defined to be `now`; if `"remove sun image"` is over, the state is `past`; otherwise, an undetermined `PNF`-state is assumed as default.

5.2 Temporal Relationships and Constraint Propagation

One of the major concerns of our work on scripting languages is to provide a structure which can handle complex temporal relationships. Human actions take variable periods of time; also, the order of the performance of actions to achieve a goal is often not strict. In other words, actions — and thus, interaction — can not be fully described neither by events (as *Director* [18] does), nor by simple tree-forking structures as proposed by Perlin's or Bates' works [2, 3], nor by straight encapsulation such as suggested by structured programming.

As in Andre and Rist [22], we adopted Allen's *interval algebra* [23] as the temporal model of interval scripts. According to this algebra the temporal relationship between any two intervals is defined as a disjunction of 13 basic primitives. Using the interval algebra, it is possible to represent temporal relations between character and system actions and states as constraints based on the 13 primitives, and to determine which actions should be happening by performing constraint propagation during every cycle of the system. However, unlike Andre and Rist [22], our system can perform very fast constraint propagation using the PNF propagation algorithm developed by Pinhanez and Bobick [24] (see also [14] for a more thorough description). It is beyond the scope of this paper to detail how the algebra works as a programming language for interactive spaces, or our approach for fast constraint propagation, or how the interval script is actually computed (see [14] for details). Our objective here is to exemplify how temporal

constraints can be incorporated into interval scripts allowing a significant increase in script expressiveness.

The last line of *"sun image scene"* is a statement that declares that *"bring sun image*", *"remove sun image",* and *"minimum time of sun image"* can only happen during *"sun image scene".* This statement creates a temporal constraint linking the `PNF`-state of all these actions and preventing, for instance, the call of the start function of *"remove sun image"* if *"sun image scene"* is in the `past` state, even if all the conditions listed in the `WHEN` declaration apply. With statements like these, it is easy to create hard constraints among the actions and states that virtually inhibit the execution of an action even in the case of sensing error (see [14]).

Events, tree-structures, and encapsulated actions and other basic elements from other scripting languages are all subsumed by Allen's algebra temporal relationships. But the interval script paradigm allows also the description of more complex relations that occur in real interaction, like parallel and mutually exclusive actions, and even causality.

6 COMMUNICATING ACTIONS

In [25], Pinhanez & Bobick revitalized Roger Schank's *conceptualizations* [26] as a formalism to represent action that enables shallow reasoning. We are currently developing this work further into a language, ACTSCRIPT, able to represent actions, requests, queries, and goals in a physical environment. Unlike previous work in languages for CG characters (for example, [3, 16, 17]), ACTSCRIPT allows recursive decomposition of actions, specification of complex

```
(request
 (action "left-cg-module"
  (move (object "camera")
   (direction
    to (location (0.0 0.4 0.5))
    from (location (0.0 3.0 -1.0)))
   (path
    (location (0.0 -0.03 0.0)))
  (when NOW)
  (velocity (special "crescendo"))
  (interval (timed duration 5.0)))))
```

Figure 7. Example of a request for a movement of the clock in ACTSCRIPT.

temporal relationships, and translation to/from natural language[3].

One of the key features of ACTSCRIPT is its small number of primitives, notably for the description of action verbs. ACTSCRIPT employs 10 primitive actions: `produce`, `attend`, `propel`, `move`, `grasp`, `ptrans` (physical transportation of an object), `atrans` (attribute transference), `mtrans` (memory or concept transference), `mbuild` (memory construction), and the generic action `dosomething`. Complex descriptions use causal links described by the keywords `result`, `reason`, and `enable`. It is not feasible to describe the syntax and semantics of ACTSCRIPT in the scope of this paper (refer to [14]).

Figures 7 show examples of a movement and an action goal, respectively, as expressed in ACTSCRIPT. Figure 7 is an example of a request from the ***It*** module to the "left-cg-module" to move the computer graphics camera from a determined location to other going through a

[3] Although we have not ventured in building a natural language translator, we believe that NL translators similar to the ones built by Schank and his team are likely to successfully parse natural languages utterances from/to ACTSCRIPT due to the language's structural resemblance to Schank's conceptualizations

Table 1. The composition of the interval scripts for the different scenes of *"It / I"*.

	total number of intervals	"C++"-only intervals		intervals defined on previous intervals		timer intervals		when	constraints
scene I	120	51	43%	31	26%	38	32%	77	11
scene II	80	33	41%	17	21%	30	38%	52	18
scene III	92	46	50%	18	20%	28	30%	55	11
scene IV	115	41	36%	27	23%	47	41%	77	20

specified `"path"` position. The movement should take 5 seconds, start immediately, and follow the pre-defined velocity profile curve called "crescendo". Upon receiving this request, the CG module checks if the object is available for movement, and in the positive case, the movement is started. A message — also in ACTSCRIPT — is sent back to the character module to communicate the state of the request. When the movement is finished, another message is sent to the main module (see [14] for other examples).

7 THE COMPLEXITY OF THE INTERVAL SCRIPT OF *"IT / I"*

For performance safety, we implemented each scene of *"It / I"* in a separate script, generating four different executable files that were loaded in the beginning of the corresponding scene. Table 1 summarizes the composition of the interval scripts of each scene of *"It / I"*, displaying the total number of intervals, the number of those that were composed exclusively of C++ code, the number of intervals defined based on previous intervals, the number of timers, and the number of `WHEN` statements, and explicitly defined constraints.

As can be seen in the table, we have approximately 100 intervals per scene, of which about half are related to the interface to other elements of the system — the "C++"-only intervals. On average, 20% of the intervals were constructed based on previously defined intervals (as discussed above). In fact, in our experience we have found that the ease of abstracting interval

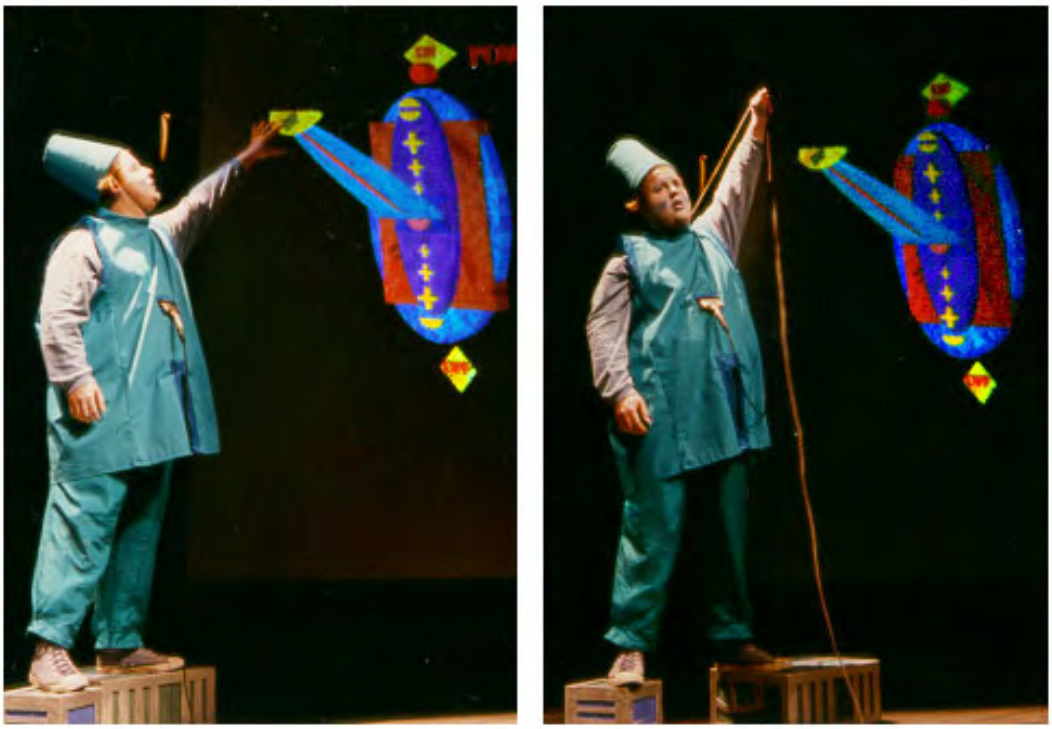

Figure 8. Scene 3 of "It / I". After discovering that the switch is just an image, *I* tries to hang himself.

scripts to be extremely helpful for the design and development process, becoming a major feature of the interval scripts system.

Notice that about 40% of the intervals are simply timers. This does not come as a surprise to anyone who has actually built an interactive space, since it is extremely important to have pauses and "beats" to smooth and connect the different pieces. Table 1 also shows that the primary mechanism for controlling the experience in "It / I" ended up being the `WHEN` statements and not the temporal constraints. However, temporal constraints proved to be fundamental to avoid undesired situations and to deal with unexpected conjunctions of events and errors.

8 PERFORMANCES AND AUDIENCE PARTICIPATION

"It / I" was produced in the summer/fall of 1997 with direction of Claudio Pinhanez, art direction of Raquel Coelho, and actor Joshua Pritchard (see fig. 8). The play was performed six

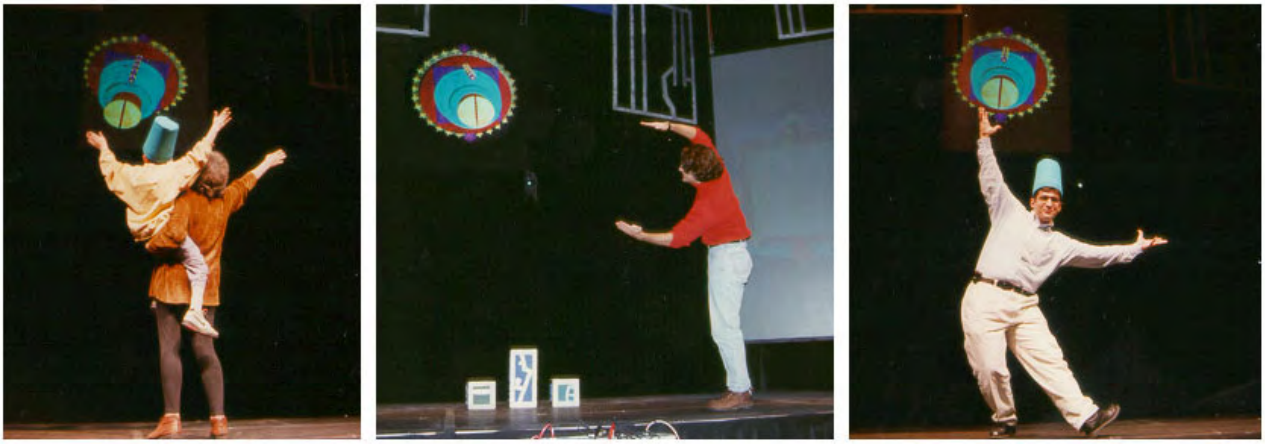

Figure 9. Audience playing with the computer character *It.*

times at the MIT Media Laboratory for a total audience of about 500 people. We clearly observed that the audience easily understood the computer character's actions and intentions. Particularly, we believe the play managed to keep the "suspension of disbelief" throughout its 40 minutes. The sound effects seem to have played a key role on creating the illusion that ***It*** was alive and to convey the mood and personality of the character. The play unfolded as a dialogue between the two actors, with the initiative shifting from one character to the other, as determined by the plot.

Each performance was followed by an explanation of the workings of the computer-actor. After that the audience was invited to go up on stage and play scene 2 — the scene where ***It*** plays with a camera and a television, first in front of the audience, and individually afterwards (see fig. 9).

However, we never observed the audience-transformed-into-actors to get deeply involved in the story. One of the reasons for this is that they performed in front of an audience and it is hard for non-actors to become emotionally engaged in front of a large group of people. Besides that, the control system was not really designed to handle normal users. In particular, it had poor ways to handle user confusion and no strategies to provide help or suggestions. To solve the first

problem we thought about creating pieces of scenario that would materialize the fourth wall (that is, cover the front of the stage). To address the confusion problem, we decided to rethink the play as an interactive space for users in what became later the "It" project (described in [14]).

In the performances held in November of 1997 the recognition of gestures was not robust enough and the occurrence of some gestures in two of the five scenes had to be manually communicated to the system. Otherwise, the 40-minute performance was completely autonomous, including during the audience participation time. Since the recognition of this kind of gesture has been demonstrated in computer vision (see [9]) and in real-time environments [12], we believe that there are no technical obstacles for a fully autonomous run of the play.

9 CONCLUSION

Both the interval script paradigm and the ACTSCRIPT communication language, developed and tested in "It / I", address problems that are certainly not restricted to computer theater or to physical spaces. The key characteristic of "It / I" is a strong narrative structure that needs to be represented in the computer and controlled under difficult sensing conditions by a system composed of multiple modules. Such conditions are encountered many times in scenes with multiple characters in computer graphics movies and in virtual environments populated with characters.

In particular, we do not see any fundamental difficulty in using interval scripts to control narratives in virtual reality-like environments where the user is one of the characters. In fact, in many practical situations in VR would probably benefit by the use of strong and robust development and control tools similar to ours. For example, in maintenance training systems the accomplishment of a task is normally associated with a correct sequence of operations and interactions with other members (possibly virtual) of a maintenance team. Clearly, the

representation of this scenario in a machine requires design/development systems that are able to capture the structured, narrative-like elements of the task.

To our knowledge, “It / I” is the first play ever produced involving a character automatically controlled by a computer that was truly interactive. However, “It / I” is just the first step in our work of understanding and developing technology for story-based, interactive, immersive environments. Also, the performances of “It / I” in November of 1997 were only the first phase of our intended work with the play itself. After that, we have developed an interactive art installation, called “It”, that is an user-specific version of the story of the play. This installation, premiered in the spring of 1999 at the MIT Media Laboratory and described in detail in [14], aims to create an interactive experience for an user without previous contact with the material of the play.

As part of the development process of “It”, the interval script paradigm and its associated language has been reexamined and improved. The current version of the interval scripts system comprises a simpler and more expressive language and a stronger run-time control engine, as described in [14]. Similarly, ACTSCRIPT has been extended and a full set of run-time communication routines has been incorporated in the language (see [14] for the full syntax of the language).

Evaluating the success of a new language or paradigm is always difficult. We have not performed controlled tests to determine whether intervals scripts and ACTSCRIPT do simplify the development process of an interactive space. However, based on our experience in “It / I”, we can assert that both paradigms are quite expressive and convenient to use. In particular, we do not believe that it would be possible at all to create and perform “It / I” without the tools provided by interval scripts and ACTSCRIPT.

10 ACKNOWLEDGMENTS

The research presented in this paper was partially sponsored by DARPA contract DAAL01-97-K-0103. “It / I” was sponsored by the Digital Life Consortium of the MIT Media Laboratory. Claudio Pinhanez was partially supported by a scholarship from CNPq, process number 20.3117/89.1. We thank to all members of the crew, in particular Prof. Janet Sonenberg of the M.I.T. Music and Theater Arts Department, John Liu, Chris Bentzel, Raquel Coelho, Leslie Bondaryk, Freedom Baird, Richard Marcus, Monica Pinhanez, Nathalie van Bockstaele, and the actor Joshua Pritchard.

REFERENCES

1. Reaney, M., *Virtual Scenography: The Actor, Audience, Computer Interface.* Theatre Design and Technology, 1996. **32**(1): p. 36-43.

2. Bates, J., A.B. Loyall, and W.S. Reilly. *An Architecture for Action, Emotion, and Social Behavior*. in *Proceedings of the Fourth European Workshop on Modeling Autonomous Agents in a Multi-Agent World*. 1992. S. Martino al Cimino, Italy.

3. Perlin, K. and A. Goldberg. *Improv: A System for Scripting Interactive Actors in Virtual Worlds*. in *Proc. of SIGGRAPH'96*. 1996.

4. Tosa, N. and R. Nakatsu. *Interactive Poem System*. in *Proc. of ACM Multimedia'98*. 1998. Bristol, England.

5. Pinhanez, C.S., K. Mase, and A.F. Bobick. *Interval Scripts: A Design Paradigm for Story-Based Interactive Systems*. in *Proc. of CHI'97*. 1997. Atlanta, Georgia.

6. Pinhanez, C.S. *Computer Theater*. in *Proc. of the Eighth International Symposium on Electronic Arts (ISEA'97)*. 1997. Chicago, Illinois.

7. Pinhanez, C.S. and A.F. Bobick. *Computer Theater: Stage for Action Understanding*. in *Proc. of the AAAI'96 Workshop on Entertainment and AI/A-Life*. 1996. Portland, Oregon.

8. Pinhanez, C., et al., *Physically Interactive Story Environments.* IBM Systems Journal, 2000. **39**(3&4): p. 438-455.

9. Davis, J.W. and A. Bobick. *The Representation and Recognition of Human Movement Using Temporal Templates*. in *Proc. of CVPR'97*. 1997.

10. Pinhanez, C. *The SCD Architecture and its Use in the Design of Story-Driven Interactive Spaces*. in *Proc. of 1st Internation Workshop on Managing Interactions in Smart Environments (MANSE'99)*. 1999. Dublin, Ireland.

11. Ivanov, Y., A. Bobick, and J. Liu. *Fast Lighting Independent Background Subtraction*. in *Proc. of the IEEE Workshop on Visual Surveillance (VS'98)*. 1998. Bombay, India.

12. Bobick, A., et al., *The KidsRoom: A Perceptually-Based Interactive Immersive Story Environment.* PRESENCE: Teleoperators and Virtual Environments, 1999. **8**(4): p. 367-391.

13. Morishima, S., et al. *Hyper Mask - Projecting Virtual Face onto a Moving Real Object*. in *Proc. of Eurographics'01*. 2001. Manchester, England.

14. Pinhanez, C.S., *Representation and Recognition of Action in Interactive Spaces*, in *Media Arts and Sciences Program*. 1999, Massachusetts Institute of Technology.

15. Perlin, K., *Real Time Responsive Animation with Personality.* IEEE Transactions on Visualization and Computer Graphics, 1995. **1**(1): p. 5-15.

16. Kalita, J.K., *Natural Language Control of Animation of Task Performance in a Physical Domain*, in *Department of Computer and Information Science*. 1991, University of Pennsylvania: Philadelphia, Pennsylvania.

17. Thalmann, N.M. and D. Thalmann, *Synthetic Actors in Computer Generated 3D Films*. 1990, Berlin, Germany: Springer-Verlag.

18. *Director's User Manual*. 1990: MacroMind Inc.

19. *MAX Getting Started Manual*. 2002: Opcode.

20. Buchanan, M.C. and P.T. Zellweger. *Automatic Temporal Layout Mechanisms*. in *Proc. of ACM Multimedia'93*. 1993. Ahaheim, California.

21. Hamakawa, R. and J. Rekimoto. *Object Composition and Playback Models for Handling Multimedia Data*. in *Proc. of ACM Multimedia'93*. 1993. Ahaheim, California.

22. Andre, E. and T. Rist. *Coping with Temporal Constraints in Multimedia Presentation Planning*. in *Proc. of AAAI'96*. 1996. Portland, Oregon.

23. Allen, J.F., *Maintaining Knowledge about Temporal Intervals.* Communications of the ACM, 1983. **26**(11): p. 832-843.

24. Pinhanez, C.S. and A.F. Bobick. *PNF Propagation and the Detection of Actions Described by Temporal Intervals*. in *Proc. of the DARPA Image Understanding Workshop*. 1997. New Orleans, Louisiana.

25. Pinhanez, C.S. and A.F. Bobick. *Approximate World Models: Incorporating Qualitative and Linguistic Information into Vision Systems*. in *Proc. of AAAI'96*. 1996. Portland, Oregon.

26. Schank, R.C., *Conceptual Dependency Theory*, in *Conceptual Information Processing*. 1975, North-Holland. p. 22-82.