논문-98-3-1-04

버츄얼 스테이지: 시나리오에 의거한 가상현실 시스템

이 기 창*, 설 창 환*, 원 광 연*

Virtual Stage: A Scenario-Based VR System

Kee-Chang Lee*, Chang-Whan Sul*, and Kwang-Yun Wohn*

요 약

머지않아 가상현실 기법은 TV 프로그램 제작에 큰 영향을 미칠 것이다. 한 예로서, 이미 수년 전부터 가상 스튜디오는 제한적으로나마 프로그램 제작에 도입되기 시작하였다. 본 논문에서는 가상현실 기법을 이용한 또 다른 응용시스템을 소개한다. 가상 스테이지라고 이름 붙인 본 시스템은 가상 스튜디오와 개념적으로 흡사하나 가상세계 참여자의 상호작용성을 더욱 강조하고 있다. 본 연구에서 가상 스테이지는 일종의 카라오케 형태로 구현되었으나 실제로는 TV쇼 프로그램 등, 다양한 프로그램 제작에 사용가능하다.

Abstract

The virtual reality technology will affect the nature of TV programming in the near future one way or another. For instance, the so-called virtual studios have been introduced in numerous TV programs in a somewhat limited scope. In this paper, we shall describe another system, which is based on the virtual reality technology. The system, called Virtual Stage, is similar to Virtual Studio in its basic concept, but is given more weights to the interactivity of participants. In our work, Virtual Stage has been implemented in the form of Karaoke. But its usage is not limited to as such; the system has an enormous potential to be exploited as a general-purpose program development tool.

I. Introduction

Some of the noticeable trends of the entertainment business, especially of the electronic one, are that they are getting more and more interactive and three-dimensionalized. There are lots of examples such as video games and interactive TV shows. Another interesting observation is that the entertainment that involves group activities is gaining in popularity. Survival games and the so-called location-based entertainment (LBE) fall into this category.

We have been developing a karaoke system, named *Virtual Stage*, that reflects the current trends as mentioned above 3D, interactivity, and group activity. In this paper, we will begin with a brief introduction of our extended karaoke system. Our primary intention of this paper is to explain how we handle the so-called story telling, namely, the representation of virtual worlds, the stories inside, and the way of representing stories. Details on the system itself may be found elsewhere[14].

The motivation behind our developmental effort is three folds:

* 한국과학기술원 전산학과
Dept. of Computer Science, KAIST

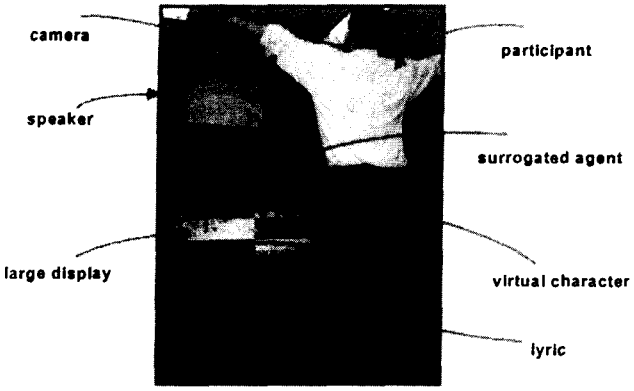


Fig. 1. The Setting of Virtual Stage

- *Virtual Stage* exemplifies the importance of representational issue commonly encountered in AI and virtual reality. Here, the representation of story is the primary concern, as a typical song delineates a well-defined story line. The story line is defined by an author; and is represented as pre-defined format of scenario. This scenario contains the characters that appear in the story, the behavior of each character at a point of story line, and the interaction between one character and others. We propose a scenario-based representation for *Virtual Stage*. The details will be discussed hereafter.
- *Virtual Stage* is a complex multimedia system, which demands advanced technologies of various fields: multimedia, virtual reality, and artificial intelligence. Traditional karaoke itself is, in fact, a multimedia system with limited functionalities. Thus our objective is to extend the karaoke system to embrace 3-D real-time graphics and animation technology, 3-D interaction technology, intelligent scoring of human performance and virtual agent technology for intelligent behavior in virtual environment. The implementation details can be found in [14].
- *Virtual Stage* has been developed as a common test-bed for our various research activities. They are motion capture / tracking of virtual agent, shared virtual environment in the networked environment, multi-modal interaction within the virtual environment integrating gesture, posture and speech, method for geometric and behavioral representation of the virtual environment, and real-time management of the virtual environment (rendering, simulation and scene management).

In *Virtual Stage*, the video image of the participant grabbed by a video camera is compounded with a rendered image of 3-D virtual environment. As a result, the participants will see the realistic video image of themselves. The participants will have a false feeling of immersion that they are existing in the VE. The resulting scene is displayed on a large screen. And the screen is located in front of the participant, as depicted in Fig. 1. By mirroring the video images of the participant before the composition, the large display functions as a digital mirror.

The participant first selects a song to sing-along from a local library. Each song has an associated scenario, which describes all the information needed to run the virtual environment. This information includes the geometric shape of virtual objects and their layout, the behavior of virtual characters, the mood, and the story line.

As the song begins, the virtual stage system plays the music by MIDI instruments, and displays the lyrics like a traditional karaoke. In addition, virtual characters such as virtual band, virtual dancers, and one with its own mission according to the story of underlying song appear in the virtual environment. While singing-along the song, the participant interacts with those virtual characters in many ways. In other words, the participant becomes the main character in a real-time, interactive music video.

The goal of our work is to develop an authoring tool and a runtime environment for *Virtual Stage* applications grounded on scenario-based representation. Once the authoring is completed, the resulting scenario is executed in the runtime environment through the timeline, allowing the participant not only to sing-along to a song of his/her choice as in the traditional karaoke, but also to interact with the virtual characters.

II. Related works

In this section, we describe some major developments relevant to our work.

1. Story representation for the virtual world

Maes[6] generated a virtual environment, called ALIVE, which a user interacts with a virtual dog using simple gestural commands. ALIVE is kind of reflexive virtual reality system; and a typical simple reactive

system. Cooper[12] tried to construct a dramatic virtual world using frames composed of actions and properties of virtual objects. Pinhanez[8] used interval scripts and PNF calculus to handle interactions evolving through time. 3D Movie Maker[7] is a commercial authoring tool for kids to make his/her own movie but it doesn't provide fully 3-D environment. Participants just watch the movie, which an author made using the authoring tool, and cannot interact with virtual characters.

The general virtual reality systems can be regarded as kind of reactive system because the virtual objects in virtual environment respond to the act of other objects (sometimes participants) according to the interaction rules. They do not consider the development of the concrete stories that an author wants to unfold in virtual environment through time. In fact, the general virtual environments do not need scenarios like a movie script. So there are not many studies about the development of a story in virtual environment.

We already introduced the basic concept of *Virtual Stage* in [14]. Now we want to unfold a concrete predefined story line in this special purposed virtual environment. This is quite different from general virtual environments. Participants sing along to the background music just as the traditional karaoke and interact with virtual characters in many ways through time. In this paper, we suggest a scenario-based representation for virtual environment and improve *Virtual Stage* using the suggested representation.

2. Reflexive virtual reality

In Videoplace[5], the first attempt to realize the idea of reflexive VR, Krueger demonstrated the potentials of reflexive VR as a new communication media. The participants were self-represented by their silhouettes, and the underlying virtual environment (VE) was a flat, two-dimensional space inhabited by simple flat objects. Mandala[11,12] from Vivid was an extension of Videoplace in that it represented participants by their video image, while still having flat objects. Major restriction of these two predecessors of today's reflexive VR was that they only supported two dimensional virtual space and objects. Therefore the interaction between participants and virtual worlds was inherently bound to their 2D-nature. IVE(Interactive Video Environment) from MIT[3,6] is known as the first attempt to extend the virtual space, objects and interactions to 3-D ones. Video image of the participants captured by multiple video cameras is composed to an image of 3-D virtual environment inhabited by 3-D objects and agents. Thus allowing participants to interact within the virtual environment using their full body motions and gestures. We have built a similar system and reported its result in a journal independently.[9,10] In our system, we use only one camera to capture the participant. The concept of *Virtual Stage* is designed on the base of our own reflexive virutal reality system.[14]

Recently, other attempts to compose real video image onto rendered image of virtual environment are being made in the broadcasting and film industries. The technology called 'Virtual Set' or 'Virtual Studio' is expected to complement slow and expensive traditional studio sets by digital sets, which are attractive for both creative and economic aspects of production.[1]

Two areas of reflexive virtual reality and virtual studio have common research issues such as precise synchronization of real and virtual camera parameters, Z-keying, etc. But at this point the researchers in reflexive VR area focuses more on issues such as the interaction method, models for intelligent agent, and the precise tracking of human body. On the other hand, virtual studio focuses on visual fidelity and the camera tracking method. However, the ultimate goal of those two area is identical - to seamlessly merge the image of the participant and VE.

III. Scenario-based representation

A story, in general, is composed of three components: stage, scenario, and character. In the virtual environment, a stage is obviously located in 3-D virtual space. Scenario is deeply related to the autonomy of virtual character. If an author wants the content of a story to be fixed, the autonomy of virtual characters is reduced, and the development of a story is fixed by a scenario which is provided by an author. If the autonomy is more focused, a fixed story cannot be unfolded, that is, the content of a story in the future is not predictable.

We are interested in the concrete and deterministic story in the virtual environment. This is somewhat similar with storyline in movie, play or music video. An author builds a total plan, and describes a scenario. As a result, the scenario is developed through time by characters in some stages. In this section, we propose a scenario-based representation using the concept of

timeline model[14] to handle the development of interactions through time. Proposed representation is adopted to unfold the concrete and deterministic story, which an author plans and intends, in *Virtual Stage*.

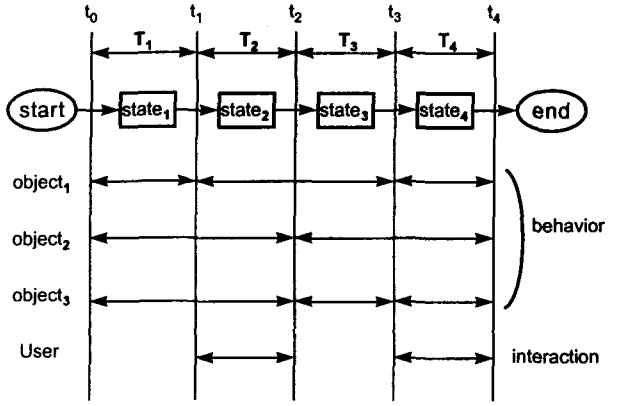


Fig. 2. Basic timeline model

1. Basic timeline model

Fig. 2 shows the basic structure of timeline model for the representation of a scenario. One story is represented as an ordered set of states just like ($state_1$, $state_2$, $state_3$, ... , $state_i$). And $state_i$ has the time interval T_i, which starts at time t_{i-1} and ends at time t_i. The behavior of each object and the interaction between an object and a participant are defined at each state.

The meaning of time in timeline model can be interpreted either as absolute or as relative. In case that time is absolute, t_i is mapped to a correct elapsed time. Because t_i is previously determined by author, if the time elapsed arrives at t_i in runtime, the state transition from $state_i$ to $state_{i+1}$ occurs. In case that time is relative, t_i is not determined before a story is unfolded, and the state transition takes place for other reasons such as the interaction between a object and a participant.

2. State transition

In timeline model, the development of a story through time is realized by a series of the transitions between one state and the following state. The causes of the state transition are as follows.

- **Accomplishment of a virtual character's goal** : The behavior of a virtual character may have a goal. The completion of a goal makes the virtual character itself and other virtual characters change one behavior into another behavior. It is regarded as the point of changing the flow of story, so state transition can take place.
- **Interaction between a virtual character and a participant** : When a participant touches a ball on the ground, for an example, the movement of the ball is changed and each object around it can selects an various alternative behavior. It is a proper position to decompose a story. Author partitions a story at this point and defines the behaviors and the interactions to the effected objects in the new state.
- **Time-constraint** : If the current time elapsed in $state_i$ get to the time-constrain t_i in runtime, the state transition occurs. The time-constraint causes *Virtual Stage* engine to develop a story naturally and continually under the time limit, which means that total running time is fixed. Our applications in *Virtual Stage* are grounded on songs. Naturally a song has the time limit explained above. So the time-constraint plays a decisive role to the transition between one state and the next state.

The cause of the state transition may be defined in company with other causes. That is, one state may have the causes of the state transition more than two. This circumstance is managed using logical operators like *and* and *or*. For example, if an author wants to generate the state transition by both cause *A* and cause *B* or by only cause C, he can express that situation as ((A and B) or C).

3. Object behavior

A virtual character acts a particular behavior in a determined state according to the scenario described by an author. The behavior can be classified considering the property of the behavior as follows.

- **Simple behavior** : In case that a virtual character takes a behavior that does not have any particular purpose, this behavior is classified as simple behavior. A virtual character taking this behavior does not affect the state transition. In Fig. 3, the behavior of simple agent and reactive agent has this property.

- **Goal-oriented behavior** : A virtual character with goal-oriented behavior has a particular goal. The result of the accomplishment of this goal can cause the state transition. State-based agent in Fig. 3 has this type of behavior. The term 'state' in agent model means the state of a finite state automata which indicates the status of a character. That is different from the meaning of the term 'state' in timeline model.

4. Participant action

An author may designate some actions to a participant at some particular state in the timeline. This action becomes a reason to generate an interaction with a virtual character and the virtual environment. As a result of it, the state transition may happen.

Participant actions, that are allowed in *Virtual Stage* and have to do with interactions, include three different types. One is the collision with a virtual character in the virtual environment. Another is the posture taken by a participant. The other is the gesture considered as a series of postures. An author has to predefine postures and gestures, which a participant can take in runtime.

In order to capture the interaction between a virtual character and a participant, various sensing and recognition technologies are required. Current Virtual Stage uses one camera for sensing a participant and employs some recognition methods like collision detection, posture recognition, and gesture recognition for the detection of the interaction (see Section V).

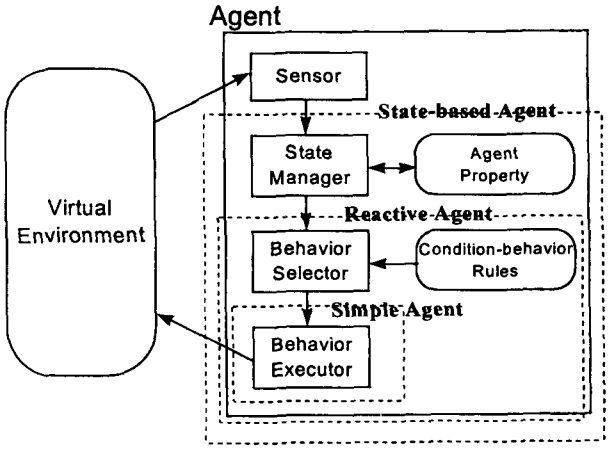


Fig. 3. Virtual character represented as agent model

5. Timeline model in Virtual Stage system

In *Virtual Stage*, a scenario is closely associated with each song. The scenario may be regarded as the blueprint for interactive music video. A scenario consists of three parts: music data, lyric data, and interaction data. All parts are determined by an author using the authoring tool for *Virtual Stage*, and saved as a form of our own script file format defined already. Music data is about the underlying song to be played by *Virtual Stage*. It contains MIDI file name to play and the total running time. Lyric data is composed of pairs of lyric itself and starting time to display that lyric in screen. Interaction data defines the behavior of each virtual character and the action of the participant itself in the virtual environment. The behavior of each virtual character is defined by both of the action itself and the event condition. The event condition determines the initiation and the termination of the action. In most cases, the event condition is more or less related to the time events in the timeline. That is, the behavior of each character is decided by time.

An author arranges the fixed scenario based on the continuity in the timeline using the authoring tool for *Virtual Stage*. The behavior of each virtual character and the interaction between a virtual character and a participant are defined along one timeline. So the authoring tool have to get the ability which expresses a scenario in the timeline. When the scenario is executed in *Virtual Stage* engine, the flow of time controls and changes the behavior and the interaction of virtual characters.

IV. Authoring environment for Virtual Stage

Producing exciting and entertaining sessions demands a high degree of complexity in scenario. Writing a complex scenario using only low-level tools such as conventional programming language is not a simple task, and can be a tedious, time-consuming and labor-intensive process.

Hence, it is desirable to have an authoring tool and environment that facilitates an authoring of scenarios for *Virtual Stage* applications. Authoring for *Virtual Stage* differs from the traditional multimedia authoring in that, firstly, it requires easy expressiveness of interactions in the VE. Secondly, the underlying virtual world is 3-D. But the most of traditional multimedia

authoring tools deal with the 2-D iconic ones. For these reasons, a specialized authoring environment is needed for an effective authoring of *Virtual Stage* scenarios.

1. Architecture of authoring tool for Virtual Stage

VSATE(*Virtual Stage* Authoring Tool and Environment) is an integrated set of authoring tools for an easy authoring of the scenarios of *Virtual Stage* applications. The components of VSATE are depicted in Fig. 4.

Authoring of scenarios starts by selecting a song written in standard MIDI file format. The scene manager parses the MIDI file, and extracts timing information (it shows the events along the timeline). Using the lyric editor inside the scene manager, authors add the lyrics to the song. Object manager allows authors to control the background scene along the timeline and the initial position of virtual characters in the scene. Author determines the overall background of the scenarios by choosing the scene among many choices. These include, for example, a rock concert stage, a roof of a one hundred-story building, the surface of the moon, etc. Changing the scene at some point of the song, such as at the start of each verse, functions like a cut in a movie or music video. By 'scene', we refer to static objects like the buildings, desks, and stages. They are modeled in 3-D CAD program and represented as sets of polygons.

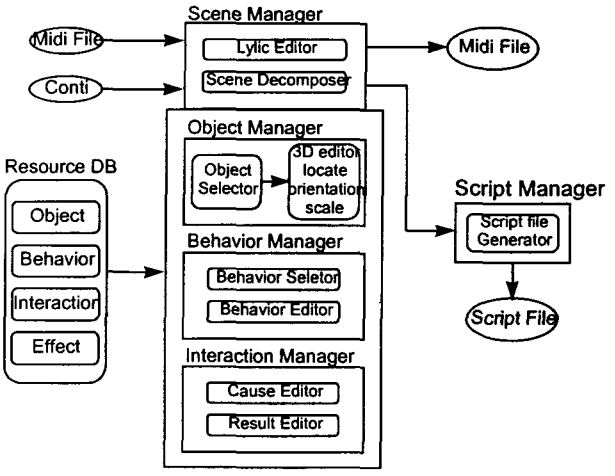


Fig. 4. Components of VASTE

The interactive aspects of the *Virtual Stage* are specified using the behavior manager and the interaction manager. Behavior is a set of rules like "if it is the climax of the song, move the arms faster." Using behavior manager, behavior is defined and saved as a script. In describing the behaviors, authors use pre-built basic behaviors such as "Follow the singer", "Waving gesture is activated", and "Fade-out".

Interaction manager binds behavior script and virtual characters. Authors specify, for each virtual character, what behavior to take, and when to begin the behavior. The causes that activate behavior can be the time in the timeline or a participant's action. Cause editor inside interaction manager specifies the causes of a behavior at any point of the timeline. Effect editor allows authors to determine what behavior is taken according to a cause. All data authored in VSATE are saved as a form of predefined script file. And this file is used by *Virtual Stage* engine as a scenario.

2. Production of a continuity

Before an author makes out a scenario using authoring tool, he/she must draw up a continuity manually. This continuity contains the behavior of each virtual character and the aspect of the interaction between virtual characters and a participant. It has a shape like timeline model presented in previous section. According to the position of a timeline, the behavior of a virtual character and the interaction between a participant and the virtual characters may have different meanings. Fig. 5 is the example of a continuity that the author draws up. This continuity plays an important role just as a blueprint in authoring a scenario. Author makes a scenario based on both a MIDI file and a continuity using VSATE.

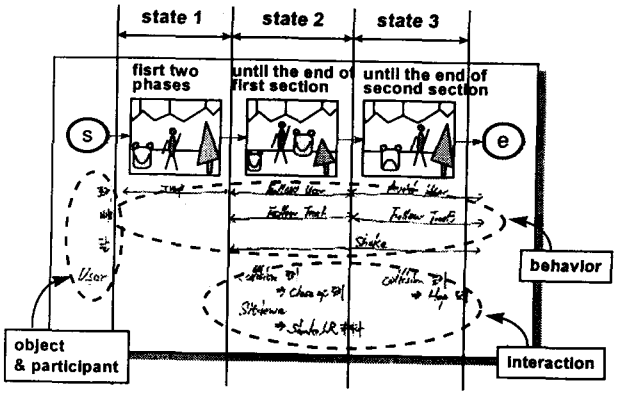


Fig. 5. Example of a continuity

3. Modeling the interaction

Our model of *Virtual Stage* operation is based on events. There are several sources of events: the time event, participant-related events generated by position and posture, and events produced as output or side-effect of action for each interaction entity (virtual character and surrogated agent). These events are combined into the 'if' part of behavior scripts.

Using the 'then' part of the behavior scripts, authors cause changes in the status of VE. The change in status of VE means many different things. Changes a behavior can and cannot cause are strongly related to the amount of interactivity in a scenario. In case of high interactivity, behaviors of virtual characters and participants are able to change virtually everything that describe the underlying VE. Since it is often the case that the participant is the most important interaction entity, enforcing a fixed story line is quite hard; participants are free to do anything and we cannot predict the status of VE at the branching point of the scenario. Interaction in this case functions as a key factor that determines the flow of the scenario and has a long-term effect. In case of low interactivity, behaviors are not allowed to update some of crucial status variables so that the predefined story line cannot be altered. Interaction in this case has a minor, short-term effect. In VSATE, basically we consider the low interactivity to reflect the fixed story line intended by an author. But Authors can specify certain status variables as protected to lower the interactivity, and remove the protection to raise the interactivity.

V. Virtual Stage engine

Virtual Stage engine takes the scenario as its input and executes it. (See Fig. 6) The sensor module processes the video image to extract the higher level information of the participant. The information extracted from the sensor module is fed into the *Virtual Stage* event manager (VSEM) to produce the participant related events. The VSEM notifies the events that each virtual character interested in. The *Virtual Stage* event manager, the core of the *Virtual Stage* engine, simulates the behavior of each character by executing the associated scripts once for each frame. VSEM manages the events produced in the simulation process. As a result, the runtime representation of the VE is updated, and the scene is rendered on a frame-basis. The VSEM includes lyrics agent, which superimposes lyrics on top of the rendered image.

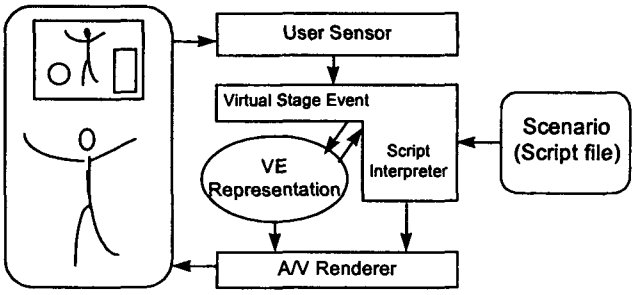


Fig. 6. Flow diagram of Virtual Stage engine

1. 2½-D Virtual space & interaction model

In *Virtual Stage*, like in any other interactive system, knowing the intention of the participant is crucial to enable the participant(s) to interact with *Virtual Stage* in a meaningful and consistent way. We use live video images from a video camera to get basic data such as the position of the participant, and to extract the intention of the participant during the playing of the scenario. In our current design, only one camera is used, this has its limitations.

The necessary information for interactions between a surrogated agent and the VE comes from two different sources. The information for the surrogated agent is obtained by analyzing the sequence of the 2-D video images. The information for VE is supplied by VE representation, which changes continuously in runtime. While there is a high level or non-geometric information in VE, this information, at its lowest level, is about 3-D geometries. The two types of information, one from 2-D pixels and the other from 3-D digital data, are not compatible to each other at geometric level. Therefore we transform the video image into 3-D compatible data by our simple 2½-D model.

In the 2½-D model, we assume the following conditions:

- We assume that the camera is fixed.
- We assume that the participants maintain the pose facing the camera at nearly a right angle (90°) most of the time.
- We assume that at least one body part of the participants touches the floor all the time.

In our 2½-D model, the participant is represented by a bitmap on a plane P_r moving in real world coordinate as the participant moves. The direction of P_r is fixed, to the direction of the participant moving back and forth about the camera, i.e., P_r always has the form z_r=depth(t). (See Fig. 7) Therefore the dimension of the virtual space inhabited by the participant may be regarded as lying at somewhere between 2 and 3. The surrogated agent is also represented by a bitmap on the plane P_v moving in VE. The movement of P_r and P_v are tightly related. The shapes of all objects and characters in VE bar the surrogated agent, are defined in 3-D space. The resulting VE is 3-D space occupied by 3-D objects and a 2½-D surrogated agent.

The restriction we imposed to the dimensions of space for the participant also affects the way the interaction is defined. Interactions in VE can be defined at various levels. Among many levels of interaction, only physical interaction level, which includes navigation and collision, is affected by the restrictions in our model. 2½-D navigation is obvious - as the participant navigates in real space, P_r and P_v moves accordingly in two directions as in Fig. 7. We defined collision between the 2½-D surrogated agent and 3-D object as the intersection between bitmap of the surrogated agent and the bounding box of the object as in Fig. 7.

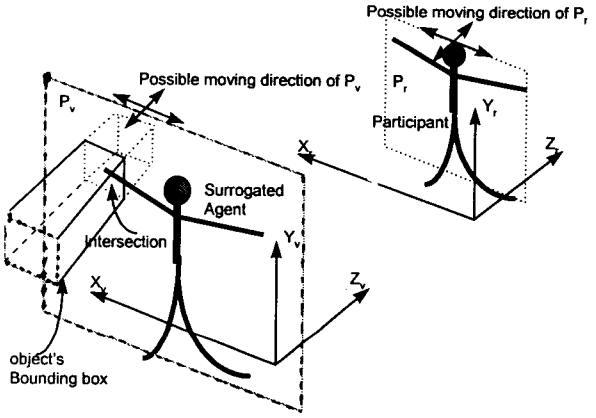


Fig. 7. $2^1/_2$-D Model of virtual space and collision

2 Participant sensor

The participant sensor module analyzes the video image of the participant, and updates the 2½-D participant model by sending events describing the changes found. As a first step, the region that belongs to the background is removed from the input image. Separation of the foreground from the background pixels can be performed using either special hardware or software. The first option, chromakeying using special hardware is used widely in the broadcasting and film industry. The other option does not assume a uniform or a static background. It varies from the simple method based on frame difference to the relatively complicated statistical approach.[13] Despite the cost of setting a homogeneous background and the need for carefully controlled illumination, the first option is preferred as it meets the real-time requirement and produces visually superior results.

From the 2-D bounding rectangle that encloses the foreground pixels, a bounding rectangle in 2½-D space in real world is obtained from simple trigonometric calculation and the assumptions we made. We find the Z-depth first by finding the intersection between the floor and the back-projection of the bottom line of the 2-D bounding rectangle. Then we make plane P_r from the intersection line. The intersection between P_r and the back-projection of the 2-D bounding rectangle becomes 2½-D bounding rectangle, as depicted in Fig. 8. The camera parameters needed for the back-projectioning are obtained through a camera calibration process[11], which can be done beforehand.

A posture of a participant is a snapshot of the foreground image. To recognize the posture of the participant, we use a simple template matching method. At each frame, the binary templates for each predefined posture are matched against the foreground image normalized and sub-sampled. The posture with the highest matching score over predefined threshold is considered to be the current posture. We use a simple gesture model: a gesture is defined as a sequence of postures. To recognize the current gesture, we use dynamic programming technique.

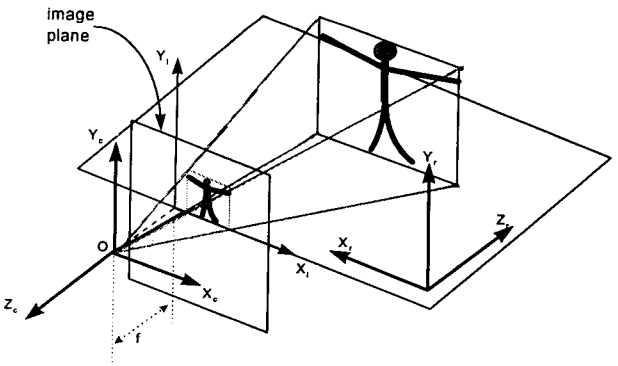


Fig. 8. Position determination

3. Audio and Video Rendering

As the scenario is played interactively, the status of the continuously changing VE is updated every frame and the resulting VE is rendered.

For visual feedback, the image of the surrogated agent, which is the bitmap of the participant, is transferred to texture memory as texture data. The background portion of the image is marked as transparent, and the resulting texture is mapped onto a rectangle that corresponds to the bounding box in P_v. The position and size of the bounding box to display the surrogated agent is controlled by the events produced by the participant sensor module. By adding one layer of mapping from the participant to the surrogated agent, many interesting effects become possible; such as distortion or exaggeration of both appearance and the movement of the participant. This alleviates restrictions in participant movement governed by physical laws in real world.

Because of the way the image of the participant is presented, no further process is needed. The graphics pipeline does not know if it is rendering the participant or the virtual characters and it need not know that. Z-keying is done by the ordinary Z-buffering which in most cases is processed by graphics hardware for the purpose of hidden surface removal in 3-D graphics.

As the scenario proceeds in runtime, MIDI song is played by a sound server, which is a separate computer dedicated to MIDI sequencing and additional sound effect generation. The main computer controls *Virtual Stage* and the sound, and uses a MIDI connection to synchronize and communicate with the sound server. Once the song begins to play, the scenario is executed according to the time events in the scenario. The lyric agent keeps track of the time elapsed and displays lyrics on the screen accordingly, indicating the current position of the song as done in the traditional karaoke box.

VI. Implementation

We have implemented a platform for *Virtual Stage*. The system consists of two machines and periphery devices such as video camera, wireless microphone, large screen monitor, MIDI sound module and audio. One machine, a SGI Impact workstation equipped with the real-time video processing board is responsible for most of the processing. The *Virtual Stage* engine, the user sensing module, the VSEM, and graphics pipeline all run on this. The other machine, a PC running Linux(PC Unix code), works as the sound server. MIDI sequencing and sampled audio service are done by the sound server. The overall configuration is shown in Fig. 9.

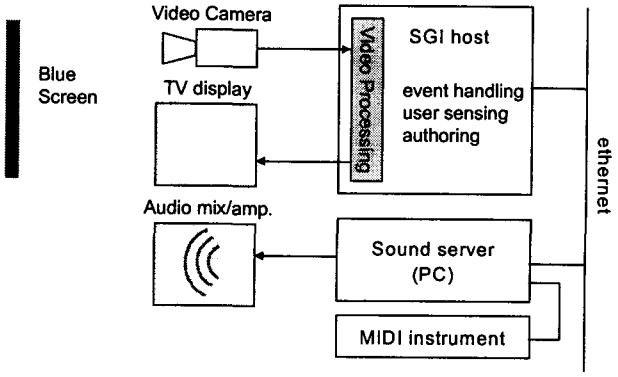


Fig. 9. Hardware configuration of Virtual Stage

The input video is shot against the blue background and is chroma-keyed. The keyed image consists of the image of participant in the foreground with non-zero values in alpha channel and the image of background scene replaced by a uniform color with a zero values in alpha channel. The sensor module analyzes this image, extracts the necessary information such as position and posture. It then generates the participant-related events. The VSEM manages the events generated by various sources including the sensor module. Since the VSEM runs as a single process, the event management must run in serial fashion. The incoming events are delivered to the central event queue, and then are checked when the action script for each interaction entity is executed.

Meanwhile, the sound server plays the underlying song by sending the MIDI messages to MIDI sound module. Since the sound server generates the time events as well as the channel messages, the synchronization between the song and the underlying scenario is guaranteed.

Examples of *Virtual Stage* play are shown in Fig. 10.

VII. Summary & Future works

In this research, we have designed and developed an authoring tool and a runtime environment for *Virtual Stage*, a virtual karaoke system, grounded on scenario-based representation. While singing-along with a song, the participant of *Virtual Stage* interacts with the virtual characters who possess the intelligent

behavior. The participant becomes the main character in a realtime, interactive 3-D music video based on the scenario produced by an author using a specialized authoring tool. The authoring tool allows the author to generate easily complex scenarios including many virtual characters with their own behavior. *Virtual Stage* analyzes video image of the participant to extract 2½-D position and posture data.

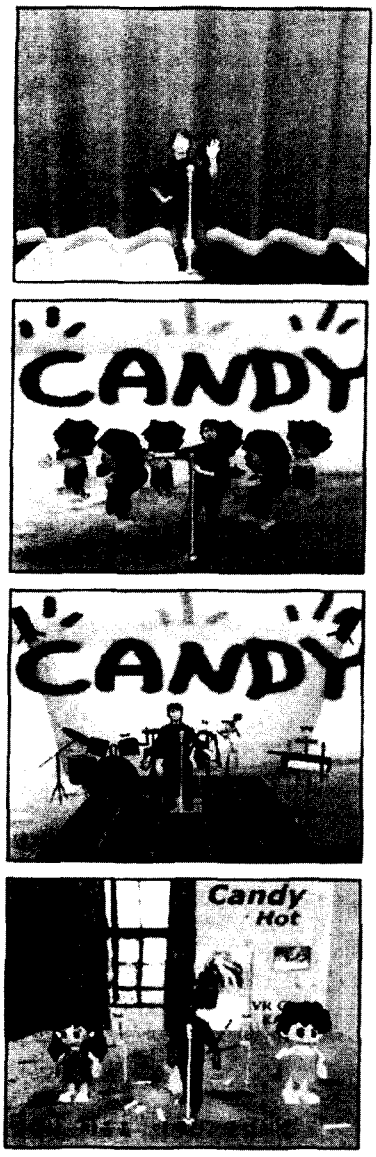


Fig. 10. Scene from Virtual Stage

There are still many issues to tackle to make *Virtual Stage* a more complete, robust, and exciting platform for interactive musical entertainment. Our current and future works include the following issues:

- We need to study on more powerful methods to describe a story. The proposed scenario-based representation for *Virtual Stage* applications is adequate for the low interactivity. It is basically caused by the properties of song. The model to represent general story line should be flexible and powerful enough that it could handle a various range of interactivity.
- Research on a more sophisticated method for participant sensing is needed. The richness and accuracy of interaction in VE mostly relies on the analysis of the video image of the participant. Current 2½-D model of interaction and posture detection by template matching is not sufficient for rich interaction. In addition, a method to utilize voice input to extract participant's intention or emotional status is needed.
- We also need to study on the intelligent scoring and the monitoring of participant's performance. In the current design, the song is played regardless of the participant's status, which is not desirable. Using technologies from signal processing and automatic accompaniment, the system should be able to adjust itself to the participant's lead.

Acknowledgements

The work described in this paper was partially funded by CAIR (Center for Artificial Intelligence Research) and SAIT (Samsung Advanced Institute of Technology).

References

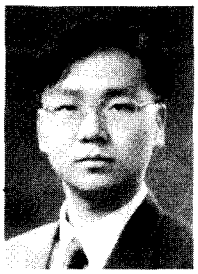
[1] L. Blonde, M. Buck, R. Galli, W. Niem, Y. Paker, W. Schmidt, and G. Thomas, "A virtual studio for live broadcasting: The Mona Lisa project," *IEEE Multimedia*, vol. 3, no. 2, pp. 18-29, 1996.

[2] M. Cooper and I. Benjamin, "Envisionments constructing dramatic virtual worlds," *Proc. VRST'94, Singapore*, pp. 251-265, 1994.

[3] T. Darrell, P. Maes, B. Blemberg, and A. P.

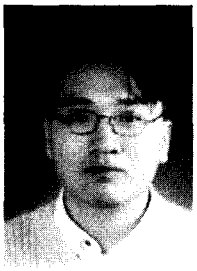
Pentland, "A novel environment for situated vision and behavior," Tech. Rep. Perceptual Computing TR-261, M.I.T. Media Laboratory, 1994.
[4] G. D. Drapeau and H. Greenfield, "MAEstro-A distrubuted multimedia authoring environments," *Proceedings of USENIX Summers 1991*, pp. 315-328, Jun. 1991.
[5] M. W. Krueger, *Artificial Reality II*, Addison-Wesley, 1990.
[6] P. Maes, "ALIVE: an artificial life interactive video environments," *Computer Graphics Visual Processing*, ACM Press, 1993.
[7] *3D Movie Maker*, Microsoft Corporation, 1995.
[8] C. S. Pinhanez, K. Mase, and A. Bobick, "Interval scripts: a design paradigm for story-based interactive systems," *Proceedings of CHI97*, pp. 287-294, Mar. 1997.
[9] C. Sul and K. Y. Wohn, "The VR system using mirror metaphor," *KISS spring conference*, 1995.
[10] C. Sul and K. Y. Wohn. "The design and implementation of virtual studio," *International Workshop on New Video Media Technology*, Seoul, Korea, 1996.
[11] R. Y. Tsai, "An efficient and accurate camera calibration technique for 3D machine vision," *CVPR*, 1986.
[12] S. Warme, "The mandala virtual world system, or, virtual reality-no strings attached," *Virtual Reality World*, vol. 2, pp. 65-71, Mar.-Apr. 1994.
[13] C. Wren, A. Azarbayejani, T. darrell, and A. Pentland. Pfinder: Real-Time Tracking of the Human Body. Tech. Rep. Perceptual Computing TR-353, M.I.T. Media Laboratory, 1994.
[14] C. Sul, K. Lee, and K. Wohn, "Virtual stages: a location-based karaoke system," *IEEE Multimedia (accepted for publication)*

저 자 소 개



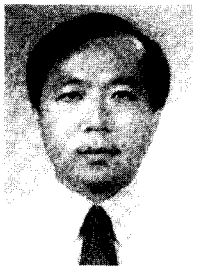
이 기 창

1995년 2월 연세대학교 전산과학과 졸업
1997년 2월 한국과학기술원 전산학과 석사학위 취득
1997년 3월 ~ 현재 한국과학기술원 전산학과 박사과정
주관심분야:가상현실감, 가상세계의 표현, HCI



설 창 환

1993년 2월 한국과학기술원 전산학과 졸업
1995년 2월 한국과학기술원 전산학과 석사학위 취득
1995년 3월 ~ 현재 한국과학기술원 전산학과 박사과정
주관심분야:가상현실감, 인체 애니메이션, 모션캡춰



원 광 연

1974년 2월 서울대학교 응용물리학과 학사
1974년 ~ 1979년 국방과학연구소 연구원
1981년 Univ. of Wisconsin 전산학과 석사
1984년 Univ. of Maryland 전산학과 박사
1984년 ~ 1986년 Havard Univ. 응용과학부 강사
1986년 ~ 1990년 Univ. of Pennsylvania 전산정보 과학과 조교수
1990년 ~ 현재 한국과학기술원 전산학과 부교수
주관심분야:가상현실감, HCI, culture computing