# MIKE: a Multimodal Cinematographic Editor for Virtual Worlds

Bruno de Araújo, André Campos, Joaquim A. Jorge

Department of Information Systems and Computer Science
INESC-ID/IST/Technical University of Lisbon
R. Alves Redol, 9, 1000-029 Lisboa, Portugal
{brar, acampos}@rnl.ist.utl.pt,jorjej@acm.org

**Abstract.** We present an interface that allows creating camera scripts and storyboards for virtual scenes through a multimodal combination of speech and gestures. Users can specify shot positioning and cinematographic movement of several virtual cameras and create a rendered sequence of the scenario in real time. The interface provides a storyboard as a sequence of frames and scripts for each camera. In contrast to existing storyboard editors ours allows scripts to be specified concurrently with animation previews providing for live cuts. Preliminary evaluations show good promise and constitute an encouragement to explore synergies with smarter camera planning.

**Keywords** Graphical user interfaces, Multimodal interaction, Animation, Cinematographic camera movement

## 1 Introduction – Related Work

Computer Graphics owes a lot of its recent expansion to its applications to gaming industry and movies. However, programming computer animations within virtual worlds is a difficult, technical task not easily accessible to traditional film editors, requiring specialized knowledge in Computer Animation and often programming skills. Thus, existing applications for cinematographic editing are not suited to the traditional language of cinematography. This work presents an application that allows creating camera scripts and storyboards on virtual scenes through a multimodal interface which combines vocal and gestural interaction. The application allows users to control different virtual cameras, specify shoot positioning and camera movement such as traveling and panning in real time, through commands, as well as generating scripts of camera commands or storyboards. From a task analysis we focus on the synergistic combination of speech and gesture to enable multimodal interaction. Finally we present a preliminary evaluation analysis of combining voice and gestures to achieve complex scripting of scenes using a simple command set.

Existing works on camera control do not take advantage of multimodal interfaces. Drucker's [4] seminal work laid out the foundations for camera manipulation. Bares [1] developed a camera control in real time for virtual worlds based on a constraint system where the user can specify directing at a functional level (e.g. informative,

mixed or dramatic takes) and cinematic controls. McDermott [6] proposes an interface to compose sequences of shots, which is evaluated according to cinematic guidelines. While this provides means to produce a storyboard, it is not possible to render continuous sequences. Greenhalghm [5] work makes it possible to replay virtual scenarios by controlling the camera and using scene objects for camera positioning. However it is not possible to move the camera towards objects. The Craven et al's Avatar Farm [3] uses a table with a top view of the world. Thus it is possible to use this overview to place and move cameras. Other approaches use constraints to manipulating cameras, such as Christianson [2] who implemented a declarative language to specify camera position and defining shoots. While this approach is suitable for automatic camera control it is difficult to use by a traditional editor.

## 3 Functional Overview – Actor and Shot Information

MIKE is an application that allows users to create rendered animations of three-dimensional scenes with avatars. MIKE reads scenarios with model definitions including actors, scenes and scripts. Using MIKE users can control up to four virtual cameras in the virtual world, while the animation is running. There are two modes of operations. During prepare mode the animation is paused the user can specify camera positioning of all four cameras. During action mode, one camera is active and films the scene. Users can switch between cameras and specify cinematographic movements. In this mode, we can prepare non-active cameras to take a predefined shot. All commands are specified vocally while arguments are provided either verbally or using the pointing device. In action mode, a storyboard composed by key images is automatically created.

At any given time it is possible to save storyboard and the sequence of all recognized commands seen so far to a file. The collection of camera movements and positions is saved in a specific format annotated with time information to allow off-line replaying of scenes.

Default camera positioning uses avatar information such as position, orientation and dimension of actors extracted from scene information. Positioning and space occupation use the Oriented Bounding Box for each avatar. We also keep track of Head Position and Orientation and look vector. Finally we store mid-body articulation position from each avatar's geometric description. Each avatar gets assigned a number or nickname for reference using vocal commands.

All position information is updated for each frame and can be used for camera positioning relative to the avatar. Users can choose which shot they want from and indicate several types of camera positions (headshot, body shot, etc). Groups of avatar can be framed in a group shot. For each shot up to three different relative orientations and eight elevations can be specified such as front, left back, right, low angle etc.

Users can specify camera positions using speech commands. The position can be specified relative to avatars using shot definitions previously described or relative to the actual camera position (up, down, backward, forward, right and left, etc.). This is used for specific view angles.

So far we described static (position) commands. However, active cameras can be moved using movement relative to actual position and orientation or relative to actors in the scene. Relative movement can be up, down, left, right, backward and forward from current position. Other common effects are provided such as rotating left or right around the center of view and zoom in, and zoom out.
Movement relative to avatar includes traditional cinematographic panning and traveling, allowing the director to specify a transition to a specific actor either using its name or by means of a deictic gesture. Panning movement is a transition over the orientation of the camera and the traveling over the position of the camera. As the script scenario is running and avatars are moving users can track them, fixing the view of the active camera on a specific actor. For the three camera commands present above, we maintain focus on the "current" actor during transition to a different view. Relative camera movements reset shot definitions whereby we assign a default shot transition.

## 6 Storyboard, Script Control and User Interface

Whenever MIKE is in *action* mode, it creates a storyboard using frames taken from the active camera. Each time we execute a command the actual frame of the active camera is saved and presented to the user described in next section. Camera events such as switching active camera also cause key frames to be saved. A *save* command Actor and Shot Information causes all saved frames to be externally stored as Bitmaps to define a storyboard. Figure 2 shows such a storyboard.
The scenario script which define the filming of the scene can be controlled by switching between *action* and *prepare* modes. When *prepare* mode is active the scenario script is paused, allowing the user to ser different cameras at will. When in *action* mode, the scenario script is running. *Prepare* mode is entered by uttering TIMEOUT command and action is resumed with PLAY.
The interface presented to the user operates on two windows: The *interactive window*, shown in Figure 1, shows the overview of the system indicating individual and camera positions and view parameters as well as a view from each camera and the storyboard being created. The *action window* shows the film being taken by the active camera.

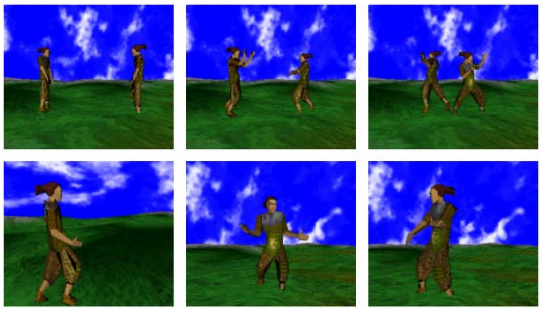

**Figure 1: A storyboard created using MIKE**

*The interactive window* is the main window, where all the control information is to be found. The window is divided in four areas, which serve different purposes. The left area displays the sequence of frames that comprise the storyboard. Every time a frame is collected to the storyboard, it appears on the list. Users can navigate through the storyboard access the action as if it were a cartoon. Center top shows an overview of the scenario, showing avatars and cameras. For each camera, frustum lines are displayed together with feedback about status. Camera colors denote active (red) or prepared state (blue). Users are able to select avatars and cameras in this window by pointing at them (*deixis*). Center bottom are provides feedback about recognized commands, every time an input is detected, whether vocal or gestural, the user receives feedback in this area, including the timestamp for each recognized event. The right area show the view from each of the four cameras and their status, (border highlighted in red for active, blue for prepared and green for inactive). These views are constantly updated. Users can select a camera by pointing at its corresponding view or icon, or by uttering their names. The background color of the interactive window informs whether MIKE is in prepare mode (red) or in action mode (yellow).
The *action window* shows the scene as viewed from the active camera, showing the results of edit commands. This Window can be presented on a different screen such as an off-screen beamer or projector, to allow for live shows and controlling presentations. When MIKE is in *prepare* mode, this window shows the view from the camera being prepared. This helps to define the camera position more precisely.

## 8 Implementation and Results

MIKE was tested in 3D virtual environment based in OpenGL API. The application was showing a fight between two animated characters. At each moment of the animation, MIKE had access to the location and spatial occupation of booth avatars.
The speech recognizer was built on Microsoft Speech SDK. The grammar uses 50 tokens and a vocabulary of 48 words.
To increase recognition rate, all sentences start with “Mike”, so in order to issue the command “CUT CAMERA ONE”, users need to utter “MIKE CUT CAMERA ONE”. This reduces error due to noise.
As describe above, users can select either the camera or the actor by pointing at the interactive window. This allows commands to be issued in different ways for example; the following three phrases are equivalent:

```
(1) <point at camera two> Say  “MIKE CUT”
(2) Say  “MIKE CUT CAMERA TWO”
(3) Say  “MIKE CUT” <point at camera two>
```

We used a sliding window to distinguish valid (deictic) gestures. For example if we say “MIKE CUT”, the user has five seconds to complete the sentence by pointing. Symmetrically, a selection remains valid for five seconds, if there is an ambiguous command the selection is used, if no command is issued within five seconds, the selection is discarded as happens after a complete command is issued.

We performed task analysis and did preliminary testing with ten users using specific tasks from a script. Preliminary assessment indicates that using multimodal commands provides for a simpler interface as compared to the work reviewed. We noted that users tended to use speech alone when following scripts, but tended to use deixis and speech in less constrained settings. Users seemed to like the system enough after a few interactions even though about 20% of the commands had to be repeated due to misrecognition. Users liked especially the possibility of preparing one or more cameras during an ongoing presentation which raises interesting possibilities for interactive storytelling and guided tour scenarios in Collaborative Virtual Environments.

## 10 Conclusions and Future Work

Feedback from users indicates that we should extend the interface to allow more camera commands (chasing an actor, moving the camera to a specific point using marks, and first person view). Some options had to be curtailed to cope with speech recognizer limitations: the actors could have more natural names. Also camera representations in the top worldview need to be less ambiguous, the difference between *prepare* and *active* status needs to be more visible. We will implement a "quick tour" to help the user and a new navigation modes need to be implemented for more complex scenarios. We intend to extend MIKE for use in interactive VR systems.

## References


1. Bares, William, H., Lester, James, C.: Intelligent multi-shot visualization interfaces for dynamic 3D worlds. In: Proceedings of the 4th international conference on Intelligent user interfaces, Los Angeles, California (1999) 119-126
2. Christianson, D. B., Anderson, S. E., He, L., Weld, D. S., Cohen, M. F., Salesin, D. H.: Declarative camera control for automatic cinematography. In: Proceedings of AAAI `96, Portland, OR (1996) 148-155
3. Craven, M., et. al..: Exploiting interactivity, influence, space and time to explore non-linear drama in virtual worlds. In: Proceedings of ACM CHI 2001, ACM Press (2001) 30-37
4. Drucker, S.M., Galyean, T.A., Zeltzer, D.: CINEMA: A System for Procedural Camera Movements. In: SIGGRAPH Symposium on 3D Interaction. Cambridge, MA. (1992)
5. Greenhalgh, C., Flintham, M., Purbrick, J., Benford, S.: Applications of Temporal Links: Recording and Replaying Virtual Environments. In: Proceedings of IEEE VR 2002(Virtual Reality), (November 2001) 1-8
6. McDermott, S., Li, J., Bares, W.: Storyboard frame editing for cinematic composition. In: IUI'02, San Francisco, CA, (January 2002) 13-16