DELIVERABLE D7.2

Virtual Reality presentation demo: Human activities in VR

October 16, 2003

Authors: Vladimir Stepan, Jiri Zara, Vaclav Hlavac

Czech Technical University, Faculty of Electrical Engineering

Department of Cybernetics, Center for Machine Perception
121 35 Prague 2, Karlovo námesti 13, Czech Republic

{stepanv,zara,hlavac}@fel.cvut.cz, http://cmp.felk.cvut.cz

Project acronym: ACTIPRET
Project full title: Interpreting and Understanding Activities of Expert Operators for Teaching and Education

Action Line IV.2.1: Real Time Distributed Systems (Cognitive Vision)
Contract Number: IST-2001-32184

CENTER FOR
MACHINE PERCEPTION

CZECH TECHNICAL
UNIVERSITY

RESEARCH REPORT

ISSN 1213-2365

Virtual Reality presentation demo: Human activities in VR

Deliverable 7.2
of the project IST-2001-32184 ActIPret

Vladimír Štěpán, Jiří Žára, Václav Hlaváč

{stepanv,zara,hlavac}@fel.cvut.cz

CTU–CMP–2003–21

October 16, 2003

Available at
ftp://cmp.felk.cvut.cz/pub/cmp/articles/stepan/Stepan-TR-2003-21.pdf

Authors were supported by the project IST-2001-32184 ActIPret, by the project GAČR 102/03/0440, and by the Czech Ministry of Education under the project MSM 212300013.

Research Reports of CMP, Czech Technical University in Prague, No. 21, 2003

Published by

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

Abstract

The virtual reality tool is presented which allows to display the activity of a human operator performing a routine which involves manipulation with object. The tool has to transform the ActIPret activity plan into virtual reality entities in VRML. The main challenge has been in incorporating semantic information to the activity plan, converting it into animation, integrating time information, and describing parametrized avatar-object interaction. The use of standard 3D computer vision reconstruction tools was inquired for creating a 3D model of a real 3D scene however, the most of the models has been hand-crafted so far.

Contents

1 Introduction and the ActIPret project context **2**

2 VR presentation module **2**

3 Task Specification **4**

4 Related work **4**
- 4.1 Scene reconstruction 5
- 4.2 Human animation 5
- 4.3 Avatar-object interaction 5

5 Our Approach **6**
- 5.1 3D geometry and appearance issues 6
- 5.2 Animation 7
- 5.3 Avatar-object interaction 8

6 Implementation Issues **8**
- 6.1 Animation in VRML 9
- 6.2 The activity in VRML 10
- 6.3 Interaction events 11
- 6.4 Experimental application 13

7 Conclusions and future work **14**

1 Introduction and the ActIPret project context

This report points out several problems and solutions encountered in the process of development of the ActIPret virtual reality (VR) software module for presentation of a learned expert activity which is described symbolically by the activity plan. A demo implementation of this software module is described as well.

Learning and perception is tested in ActIPret scenarios which encapsulate the activity. The *'insert CD scenario'* is the simplest one used in the project. For the completeness, let us note that this scenario includes a human operator, the table with a CD player and CDs. The human opens the player by pushing the appropriate button, selects the CD on the table top, picks the CD up, inserts it into the player and closes the player. Even though this might look as a little too simple problem, from the VR point of view the scenario comprises all what a testing example should have. The animated avatar interacts with other objects in a scene - the CD is moved between two locations and CD player is opened (and closed) as a result of avatar's action. All experiments described later have been performed with CD scenario as a testing example.

To stress the position of the VR presentation module in the ActIPret framework, let us show the framework's block diagram (see Fig. 1).

Structure of the report

The report is structured as follows: Section 2 informally introduces the VR presentation module. Three relatively independent submodules are identified (3D scene reconstruction, animation of a human, and interaction between avatar and objects). The task specification is given in Section 3, which focuses in closer detail to the particular goals, which have been reached at present stage of the project. Section 4 summarizes the state-of-the-art in three mentioned areas. The solution we propose to the specified task is described in Section 5. Implementation issues are in Section 6 in which we mainly focus on the VRML prototypes and functions involved. Final Section 7 concludes the report and outlines intended future work.

2 VR presentation module

The VR module is not directly coupled to other modules of the ActIPret framework, see Fig. 1. The VR module can be considered a stand-alone

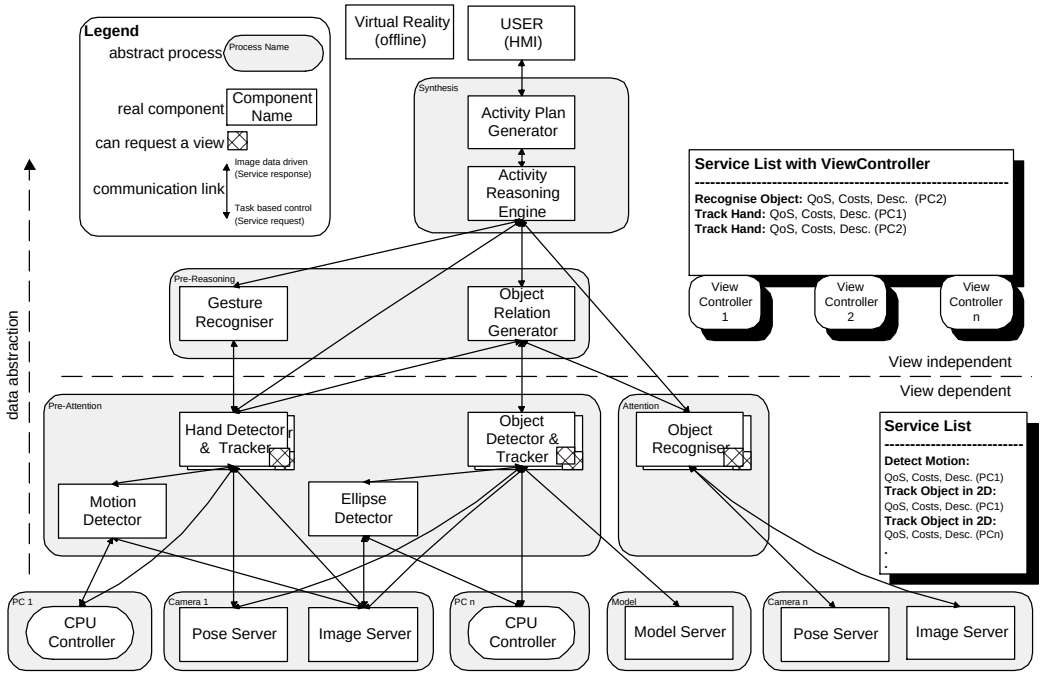

Figure 1: ActIPret framework and its modules.

program for presentation the project results and generally for human-oriented activities. It was decided to build the VR module on VRML standard.

The presentation of the human activity in VR can be divided into three parts: *3D geometry, animation* and *interactions*.

The 3D model of the scene can be captured off-line using 3D reconstruction techniques from computer vision. The model includes 3D geometry and appearance of the scene itself and involved objects. In the 'insert CD scenario', the model consists of CDs, CD player, the table and the walls of the room in the scenario. The avatar is one of the objects in the scene in terms of geometry.

All the motion in the scene can be included in the animation part. In the 'insert CD scenario', the operator reaches the button, pushes it, picks up the CD, the CD player opens and the human moves the CD into the player. Not all the motion is of the same kind. The motions can be categorized by their importance, basically into two categories – indigenous and dependent (master-slave). The motion of the human is indigenous and all other motion related to the activity depends on human motion, as it is triggered by it. Thus it is necessary to model only the human motion.

Pushing the button followed by CD player opening, picking up and later

releasing the CD are interactions between a human and the objects in the scene. These interactions belong to the third and the most difficult part of the activity description.

3 Task Specification

The task of the ActIPret VR presentation module is to define, create and present the three parts of the human activity VR model. We use the output of other ActIPret modules in a form of a conceptual language and an activity plan. Since this description is not exactly what we need for a VR model of the activity, we have to transform it to a description usable for the selected tools (the VRML standard).

Even though the activity plan provides the information on the interaction between the objects, it does not describe the motion of the human on the level we need it for VR presentation. Thus the VR presentation module must use the output of the hand tracker to obtain the information describing the human motion that can be converted to our chosen (VRML) animation format.

A large variety of human activities requires general tools for their description. Simultaneously these tools must be able to generate the animation data and interaction description for particular activities. For the experimental reasons we first created the hard-coded 'insert CD scenario' presentation as an example of a human expert activity model. We proceed in task of selecting the set of data describing the activity by abstracting the 'insert CD scenario' VR model.

After selecting this set of data we need to create a formal notation of an activity prototype in VRML. Completing this task will also serve to clear out the goals and next steps for our part of the ActIPret project.

Due to the unintegrated off-line position of the ActIPret VR presentation module does not need to model the activity in real time. The VR description is created once and then can be reviewed according to user's needs.

4 Related work

The task of the activity modelling consists of several separate subtasks - the 3D scene reconstruction, human animation and avatar-object interaction. All these topics have been extensively studied in different research areas - computer vision and computer graphics.

4.1 Scene reconstruction

Computer vision provides several techniques for 3D scene reconstruction [6]. These can basically be classified as active and passive. Among the active methods we can count time-of-flight range finders (LIDAR), structured light range finders the laser scanning and other such methods, typically quite demanding in terms of special hardware. Passive methods are based on reconstruction from two or more views provided that the correspondence problem is solved. The 3D geometry can be reconstructed from images captured by calibrated or uncalibrated camera [7].

There is also a tedious manual way to reconstruct the scene is the manual modelling using some 3D modelling software, e.g. 3D Studio Max.

The result of the scene reconstruction is the information about the 3D geometry of scenes and involved objects and their appearance (texture). For the needs of the activity modelling, the model should usually have certain structure and even semantics (the CD player opens and closes upon an event of pushing the button). Thus the scene reconstruction for needs of an activity modelling still needs large amount of user input.

The smart object [3, 4] approach that resembles the object oriented programming paradigm provides the most complex description of objects for the activity simulations. As far as the simple geometry is concerned, there is nothing special about smart objects.

4.2 Human animation

Many techniques of animating the virtual humanoid have been described. The forward and inverse kinematics techniques have been widely used. Also the approach of capturing the real actors motion has developed in a variety of technologies that can be basically divided into on-line and off-line methods. Especially the latter, represented by the optical technology is developing rapidly [1, 2]. Lately some rather interesting methods of animating the virtual humanoid have appeared, such as the virtual puppetry [5].

Usually a combination of the animation methods is used. For example, the limited information obtained by optical tracking part of human body can be used to animate the virtual humanoid when the constrains given by the human model (distances between joints, etc) are used.

4.3 Avatar-object interaction

Avatar-object interactions are commonly solved by programming them specifically for each case. This does not solve the problem for the wide range of

cases. There are attempts to solve involved manipulation problems automatically. Due to computational complexity, heuristics borrowed from artificial intelligence are often used. Advanced techniques as time dependent planning, reasoning and learning determine many manipulation variables during an avatar-object interaction. Seeing from this point of view, even the simplest activities in 'insert CD scenario' (opening the door, picking up the CD) are quite complex in terms of the interactions between human and the environment. This issue is quite a challenge.

The idea of the *smart object* [3, 4] was introduced to address this problem from opposite direction. The information about the way how objects can be used is incorporated in the objects in the scene. This information concerns the actions of the object as well as the expected behavior of the avatar (or, in case of cited research, of an autonomous agent). This approach is suitable for autonomous agents or for highly interactive systems. The smart object technique seems to be too sophisticated for our purposes.

5 Our Approach

We introduced the VR description of the human expert activity which consists of a set of actions leading to a certain goal. The actions of an expert are represented by the animation (the motion data) and by the avatar-object interactions. Our activity description is the VR form of the scenario with the specification of the scene settings and a description of the particular act.

The VR activity description includes the information on the geometry of the scene, its appearance, the duration of the activity, the animation data describing the motion of the avatar, and description of the avatar-object interactions.

The activity description is further enhanced with optional textual description, that can be displayed to the user when viewing the recorded activity. These hints are synchronized by the avatar-object interaction events.

5.1 3D geometry and appearance issues

Let us first show one frame from the input video sequence showing the scene of the simplest 'insert CD scenario', see Fig. 2.

As there is no general method to automatically reconstruct the scene (and no ambition to develop one in the project), we have decided to model each object in a scene either manually or with the help of some software for image based modelling. We chose to use commercial tool for 3D reconstruction

Figure 2: The example of a frame from the input video sequence of the 'insert CD scenario'.

from images Canoma [10]. The point correspondences should be clicked by hand and polygons representing texture as well.

A set of 3D models for each scenario analyzed with the ActIPret system has to be created. This process will always require a significant amount of user input. Some models will have to implement their own functionality. The CD player can be an example of it, as it opens and closes at the event of pushing the appropriate button. Here the geometrical model has to be combined with interactions and animations.

Although the functionality will be added to some models, this is not the *smart object* approach, because the object will not contain the information for the avatar. It is just the enhancement of the simple geometrical model.

The idea of further enhancing the models to become *smart objects* has not been completely aborted. It can prove useful for the process of creating particular models of activity.

In the activity description, all the objects in the scene are listed along with their locations and orientations in the scene and other attributes of the models.

5.2 Animation

All the dynamics in the scene related to the human activity depends on the actions of the human. The only animation data necessary to store for the activity is the data describing the motion of the avatar.

In the simplest 'insert CD scenario', the pose of the expert performing the activity is assumed to be known and unchanged during the motion. Only

the active hand (the 'insert CD scenario' is 'one-handed') is being tracked. Thanks to these constraints, it is not a problem to animate the avatar using the inverse kinematics algorithms provided the trajectory of the hand is known.

The inverse kinematics algorithm that seems to be most promising for our purposes is Cyclic Coordinate Descent (CCD). This algorithm was chosen for implementation.

However, the knowledge of the hand trajectory is not enough to animate the hand with all details. The information about the finger movements and position is not available.

This problem can be solved by the use of a database of gestures associated with certain actions (pushing the button), that will be indicated by the activity plan. Another solution might be the use of the *smart object* approach with the information on the positions of avatar's fingers incorporated into the manipulated object model.

5.3 Avatar-object interaction

The human changes the state of the scene by changing the state of particular objects (opening the CD player) or by moving objects without changing their state (picking up the CD). In case of describing the activity, we are interested in purposeful avatar-object interactions. That excludes motions of objects after the on-off stimuli.

The actual events of interaction that remain are activating the functions of objects and moving objects. The list of the events belongs to the activity description mentioned in the beginning of this section. This list is ordered by the relative time of the event occurrence which is also stored in order to synchronize the events with the animation.

6 Implementation Issues

The demo implementation of the VR presentation module has been written in Java language and uses the EAI (External Authoring Interface) to control the VR scene that is defined using the VRML standard. Beacuse of the EAI implementation, that has been used, the Parallel Graphics [12] Cortona VRML browser is necessary for running the demo. The humanoid avatar we use is compliant to the H-Anim 1.1 standard and was borrowed from the VRLab web site [13], where it is available for non-commercial use (by C. Babski [9]).

The program (see 6.4) works with the VRML description of the activity (see 6.2) that encapsulates the VRML animation (see 6.1) and the model of the scene.

6.1 Animation in VRML

We have designed an animation format that can be easily connected to any H-Anim humanoid [11]. The following VRML PROTO provides the information that describes the animation and its connection to the H-Anim humanoid. One of the possible uses of this prototype is in the description of the activity presented in the next section.

```
PROTO Animation
[
  exposedField MFString jointToIntMap []
  exposedField SFString name ""
  exposedField SFNode timer NULL
  exposedField MFNode interpolators []
]
```

Every animation is identified by its *name*.

The *timer* field contains the timer (TimeSensor) node of the animation and the *interpolators* field the list of interpolators. Last two of the interpolators belong to the *hanim_HumanoidRoot* joint node of the humanoid. The first of them is the PositionInterpolator and the second one is the OrientationInterpolator.

The *jointToIntMap* field serves the purpose of determining which joint belongs to which interpolator. When loading the animation, the application should connect the corresponding joint (from the list of joints) to appropriate interpolator (list of interpolators = the *children* field of the animation) for each of these strings different from *'INACTIVE'*. It is clear, that the sequence of joints is crucial. The joints in an appropriate field of the H-Anim Humanoid node are sorted by depth-first traversing the humanoid hierarchy and the list of interpolators has to follow this order.

This idea was borrowed and simplified from the animation gallery described on the VRLab web pages [9].

Let us demonstrate the nature of the VR presentation on the example from the 'insert CD scenario' in Fig. 3. The example demonstrates how the real example given in Fig. 2 by a frame from the input videosequence was simplified by its generalization in a functional sense.

Figure 3: The VR presentation of the 'insert CD scenario'.

6.2 The activity in VRML

We have defined the VRML PROTO that combines all the information necessary to describe human expert activity in order to create its model in virtual reality.

```
PROTO Activity
[
   exposedField SFString name ""
   exposedField MFNode scene []
   exposedField MFString sceneUrl []
   exposedField MFVec3f scenePositions []
   exposedField MFRotation sceneRotations []
   exposedField MFVec3f sceneScales []
   exposedField MFString sceneParameters []
   exposedField SFNode animation NULL
   exposedField MFFloat eventKeys []
   exposedField MFNode events []
   exposedField MFString textOverlay []
   exposedField MFNode viewpoints []
   exposedField MFNode children []
   eventIn MFNode addChildren
   eventIn MFNode removeChildren
]
```

This prototype encapsulates all the data necessary to create the model of the scene and view the activity within the given scenario. Its instance identifies itself to the application with (preferably unique) *name*.

The *scene* and *sceneUrl* fields are alternative ways to provide the scene description. In the first case, the VRML code of all the objects is present in the activity description file. Since it is better to model the scene objects as reusable prototypes stored separately, we should rather load them from their location. This is what *sceneUrl* is for.

The *scenePositions*, *sceneRotations*, *sceneScales* and *sceneParameters* fields provide additional information about the objects and should be of the same length as the fields describing the scene objects. These fields are used for proper positioning of reusable prototypes included into a scene via *sceneUrl* field.

The *sceneParameters* field yields information on possible parameters of objects. For instance, the prototype of the CD uses the Switch node to determine whether the color or texture will be used, other parameters are then color or texture URL. For such model the entry in this field can look like this:

which SFInt32 1 url MFString ../textures/cd4-1.gif

In general, it can be either empty string or the string:

*(<target field><VRML type><value of that type>)**

The VRML data type must be present so the application can create the proper VRML event to send the the data to.

The *animation* field contains the node with the animation data. It is an instance of the prototype described in previous section. This animation data describes the avatars move throughout the described activity as a whole.

The last three fields – *eventKeys*, *events* and *textOverlay* – refer to the avatar-scene interaction events. The field *eventKeys* lists the key-frames indicating the relative time when the event occurs. The field *events* is an array of the special nodes describing the events themselves. We mention it later in more detail (see Section 6.3). These two fields have to be of the equal length. The last field (*textOverlay*) is the array of strings that could be displayed to the user as an explanation. The number of these strings is equal to the number of events plus 1. This field is optional.

6.3 Interaction events

The avatar-object interaction event is represented by the following VRML PROTO.

```
PROTO ActorEvent
[
   exposedField SFString eventValue ""
   exposedField SFString targetNode ""
   exposedField SFString eventType ""
   exposedField SFString eventName ""
   exposedField SFNode nextEvent NULL
]
```

In terms of VRML, any avatar-object interaction must be expressed as an VRML event sent to the appropriate node. In the simplest ActIPret 'insert CD scenario', there is the event of touching the button with the response of opening (closing) the CD rack and the event of moving object (the CD).

This *ActorEvent* prototype has been designed to provide the information the application needs to create the necessary VRML events.

For our needs, the VRML event is determined by the its type (*eventType*) and value (*eventValue*), the target node (*targetNode*) and its field (*eventName*). All these fields of the *ActorEvent* prototype are strings. The mechanism of creating appropriate VRML events from these strings must be on the application side.

Moving objects by avatar in VRML scene is best implemented by changing the location of the object in the graph of the scene. For example, when the CD is picked up, it is removed from its original location and becomes a subgraph of the animated avatar's hand, thus accepting all transforms of the hand.

The avatar-object interaction event of picking up the CD consists of two VRML events - removing the object and adding it somewhere else. The *nextEvent* field was added to the *ActorEvent* prototype for such multiple events.

The *ActorEvent* for picking up the CD would be as follows:

```
ActorEvent {
    targetNode "root"
    eventType "MFNode"
    eventName "removeChildren"
    eventValue "node 4"
    nextEvent ActorEvent {
        targetNode "human sites 0"
        eventType "MFNode"
        eventName "addChildren"
        eventValue "node 4"
    }
}
```

A simple language was defined to describe the several possibilities of target VRML node. The possible targets are root of the scene, Site node [11] of the avatar or any scene object identified by its index in the array of objects (*scene* or *sceneUrl* fields).

In our example, the fifth object (index 4) in the scene description (*sceneUrl*), which is our CD is removed from the root of the scene where it was originally placed and added to the first (index 0) of the Site nodes listed in the H-Anim [11] humanoid definition, which happens to be located in avatars right hand.

6.4 Experimental application

Our demo application implements the basic functions for viewing any record of time dependent action - start, stop, pause and restart. The specifics of implementation of these functions for the VRML based animations were described in our paper published in 2002 [8].

The mechanism of viewing the animation was improved a little to suit better more articulated humanoid models and the need to resolve the avatar-object interaction events. One more link was added to time-fraction passing chain, see Fig 6.4.

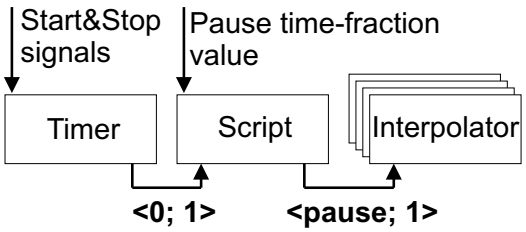

Figure 4: The time-fraction passing chain for animation control.

Whenever the timer that controls the animation is turned on it starts sending the time-fractions beginning with 0. This causes a problem when we want to restart the paused animation without starting from the beginning again.

The new link is the script node that remembers the position of the pausing event on the original animation time-line and stores it in its internal variable. This node is linked between the timer and the interpolators and transforms the time-fractions from the timer. When its output reaches value of 1, it resets its internal variable back to 0 - the animation has reached its end.

Our demo application consists of the VRML and Java part. Their roles and relationships are indicated by the Fig. 5.

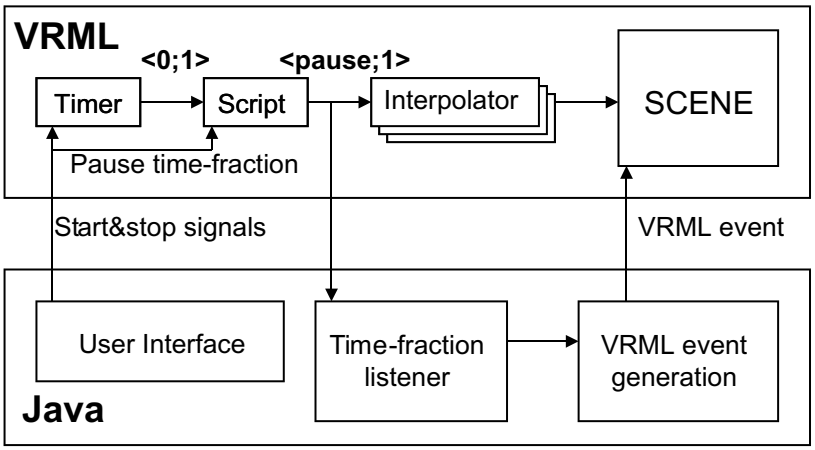

Figure 5: The roles of VRML and Java parts of our demo.

The activity viewing program allows to select the activity from the list and reads the selected file with the activity description. Then it builds the activity set-up, loads the animation and connects it to the avatar. When this is done, its main purpose is to view the animation with regard to the avatar-object interaction events.

The time-fraction listener checks the output of the script and compares it with the values listed as the relative times of avatar-object interaction events. It triggers the creation of an appropriate interaction event according to the result of this comparison. This event is then passed to the VRML part, where the appropriate node receives the appropriate data. This results in the desired action happening in the VR scene.

The example of the testing application for the simplest ActIPret project 'insert CD scenario' is shown in Fig. 6. Notice the control buttons in the bottom of the figure and the overlayed text in red color in the middle left of the figure.

Of course, viewing the activity from various viewpoints is supported. Two viewpoints with the highest information value are typically the original camera position and dynamically changed view through the virtual expert's eye.

7 Conclusions and future work

We have determined the type of data required for presentation of the human expert activity in VR. We proposed the formal description of the expert

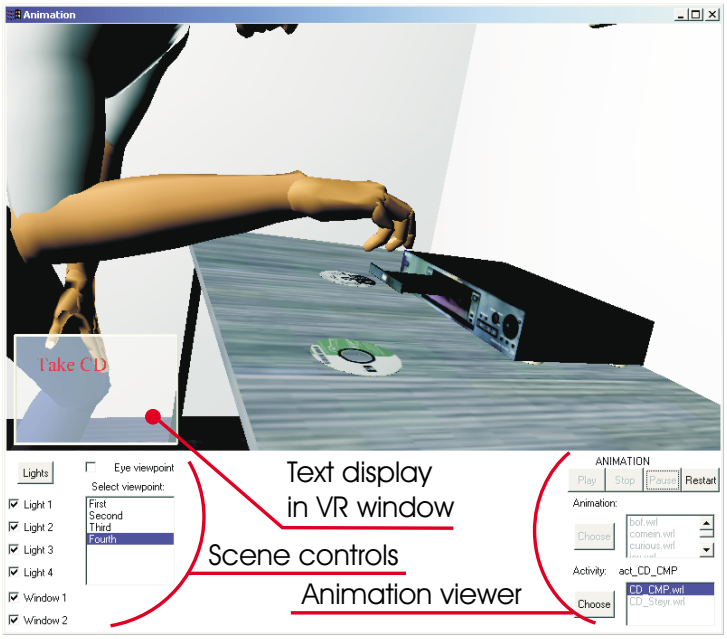

Figure 6: Our demo application and its user interface.

activity and created a program tool that uses this description to view the recorded activity. This was the first step in creating a general ActIPret demonstration module. The ActIPret simplest testing 'insert CD scenario' was used to test the methods of modelling the expert activity in VR. In its simplicity, it still contains examples of problems that can be encountered, namely the types of avatar-object interaction.

The proposed activity description is based on the VRML standard. The viewer program was implemented in Java language (on both stand-alone application and an applet platform) controlling the VRML scene via the EAI interface. For running this program it is necessary to have Parallel Graphics [12] Cortona VRML browser installed. This browser is available at the Parallel Graphics web-site free of charge.

The objects used in the demo were modelled manually as well as the animation data. Obtaining the human animation data from the hand trajectory using the inverse kinematics algorithm (CCD) is being worked on.

We aim to test the approach on more complex scenarios with richer expert activity and 3D scene in the second and third year of the ActIPret project.

The future work will be also devoted to the interface and link between the cognitive part of the project and VR module. Currently, the activity plan and conceptual language for its description is sketched only vaguely. Not all

information needed for VR presentation is included.

We have to seek the ways how to obtain data we have selected as crucial for modelling the activity. The process of creating arbitrary model of the activity to be viewed has to be designed. This must be done in cooperation with other partners of the ActIPret project.

References

[1] R. Boulic, P. Fua, L. Herda, M. Silaghi, J. Monzani, L. Nedel, and D. Thalmann. An anatomic human body for motion capture. In *Proceedings of EMMSEC '98, Bordeaux*, 1998.

[2] P. Fua, L. Herda, R. Plankers, and R. Boulic. Human shape and motion recovery using animation models. In *Proceedings of the 19th Congress, International Society for Photogrammetry and Remote Sensing, Amsterdam, Netherlands*, July 2000.

[3] L. M. Goncalves, M. Kallmann, and D. Thalmann. Programing behaviors with local perception and smart objects: An approach to solve autonomous agents tasks. In *Proceeding of SIGGRAPH 2001*, 2001.

[4] M. Kallmann and D. Thalmann. Modeling behaviors of interactive objects for real time virtual environments. *Journal of Visual Languages and Computing*, 13, 2002.

[5] S. Oore, D. Terzopoulos, and G. Hinton. Local physical models for interactive character animation. In *Computer Graphics Forum, The International Journal Of The Eurographics Association*, volume 21, pages 337 – 346, September 2002.

[6] M. Šonka, V. Hlaváč, and R.D. Boyle. *Image Processing, Analysis and Machine Vision*. PWS, Boston, USA, second edition, 1998.

[7] M. Urban, T. Pajdla, and V. Hlaváč. Projective reconstruction from n views having one view in common. In Bill Triggs, Richard Szeliski, and Andrew Zisserman, editors, *Vision Algorithms: Theory & Practice*, volume 1883 of *LNCS*, pages 116–131, Berlin, Germany, September 1999. Springer.

[8] V. Štěpán and J. Žára. Teaching tennis in virtual environment. In *Proceedings of SCCG 2002*, pages 38 – 43, 2002.

[9] Homepage of C.Babski, LIG - H-Anim humanoids and animations. http://ligwww.epfl.ch/~babski/.

[10] MetaCreations Canoma 3D reconstruction tool homepage. http://www.metacreations.com/products/canoma/.

[11] H-Anim (Human Animation Working Group). http://www.hanim.org.

[12] Parallel Graphics home-page (Cortona VRML browser). http://www.parallelgraphics.org.

[13] Virtual reality lab (former LIG), EPFL Lausanne. http://vrlab.epfl.ch.