Body-Centred Configuration in Collaborative Virtual Environments

Dave Snowdon
University of Nottingham, UK
d.snowdon@cs.nott.ac.uk
http://www.crg.cs.nott.ac.uk/~dns/dave.html

Kai-Mikael Jää-Aro
Royal Institute of Technology, Sweden
kai@nada.kth.se
http://www.nada.kth.se/~kai/

Abstract

With the increase in number of VR applications and increasing complexity of VR systems the user is provided with a multiplicity of different options for configuring their embodiment in the virtual world, the style of interface (immersive or desktop), application specific parameters and even whether they wish to use the system in a subjective or objective manner. However, there is no single, flexible way for the user to configure these parameters and even a single VR system may provide several different configuration files that must be tailored to a user's needs. To worsen the situation the user (from our own experience) may need to vary these settings at each invocation of the VR system or application. We therefore propose a new metaphor for configuring Virtual Environments and the applications within them, which we term Body Centred Configuration (BCC). Body Centred Configuration considers a user's preferences to be an integral part of her representation in a virtual environment. In this paper we describe Body Centred Configuration and an implementation using the SICS DIVE VR system.

Keywords: configuration, DIVE, user embodiment, subjectivity

1. Introduction

The inspiration for this work arose when the authors were building a prototype subjective virtual environment (Snowdon et al, 1995) using a subjective version of VR-VIBE (Benford et al, 1995) as an example application. VR-VIBE is an application designed to provide a 3D visual interface to a document store. It is a relatively mature application (as VR applications go) and currently provides the user with a relatively large number of configuration options and presentation methods which are normally specified at the command line.

When VR-VIBE is being used objectively, specifying the parameters when the application is invoked is perhaps cumbersome but sufficient since all users will be seeing the same visualisation. However, when VR-VIBE is being used subjectively each user will have their own view of the visualisation which may vary from the views of other users. In this case a new subjective view is created for each user as they enter the visualisation. Since the VR-VIBE interface allows some parameters to be varied while the application is running, one solution for the subjective case would be to give each user a standard view and allow them to adjust it. However, given the number of possible configurations and the delays necessary to re-compute the visualisation when some parameters are changed this was not felt to be ideal. At this point the authors had the idea that the user might bring the various parameters with them as they entered the world, and the concept of Body Centred Configuration (BCC) emerged from there.

Our approach to configuration considers all properties relating to a user in a virtual environment to be an extension of that user's embodiment in the VE. We shall begin by outlining the various type of configuration information that the user may want to supply to the VR system. The next section will briefly describe existing features of user embodiments. We shall then describe our approach to configuration and a prototype implementation of these ideas in DIVE 3.

1 This work was performed at the Swedish Institute of Computer Science
1.1 Configuration options

In this section we shall outline some of the types of configuration information that the user may have to supply in order to be able to most usefully work in a collaborative virtual environment.

- **User embodiment** As described below, the representation of the user in the virtual environment needs to be customised both so that users can be identified (and told apart) and to match the interaction devices available to the user and thus give as clear a view as possible of what that user is doing, in order to allow co-workers to be aware of each others' actions. Indeed we may even need to support multiple user representations if the user is present in multiple worlds simultaneously, or we have several users with subjective views of the environment.

- **Application configuration** The user may need to use one or more applications whilst present in the virtual environment. It may not be possible for the user to use command line arguments to configure applications since the application(s) may already be running (perhaps because it runs continuously or was started by another user). Text files in the user's filestore may not be a solution since, for security reasons, the application may not have access to them (it might be running at another site).

- **User interface** Due to different interaction devices or different needs, users may profit from different user interfaces. Obviously an immersed user and a desktop user will want to use different types of interfaces. For example an immersed user may be able to navigate easily in 3D using 3D input devices but may find text hard to read. Conversely, a desktop user might find 3D navigation more cumbersome but easily be able to handle text display and use of 2D menus.

- **Subjective or Objective display** Depending on the task the user is to perform she may wish to experience exactly the same environment as other users (objective display) or may wish to be able to tailor her view of the environment to her needs (subjective display).

A reasonable amount of information might be required to specify values for all the options the user requires. For example VR-VIBE alone currently has 15 different parameters that can be set to specify the form the visualisation will take. There is a need to make the configuration process easier and also to better match it with the way in which virtual environments are used.

2. User Embodiment in Collaborative Virtual Environments

A user's embodiment currently has two main purposes: to represent the user to herself and to represent the user to others. Seeing a representation of yourself in a virtual environment is important since it provides visual feedback of your actions (analogous to the feedback provided by a mouse cursor in a 2D desktop interface) and can also increase your feeling of presence in the environment (Slater & Usoh, 1994).

In a multi-user virtual environment, a user's embodiment signals the presence of that user to any other users who are currently in the environment. Benford et al. (Benford et al, 1995b) note that there are several pieces of information that an embodiment should convey in order to fulfil this role, including:

- **Presence** An embodiment should signal its owner's presence in a VE.

- **Identity** There are several levels of support for identity that an embodiment can provide. Firstly an embodiment should indicate that it represents a user (as opposed to simply being a piece of scenery). Secondly, it should be possible to differentiate that user from other users. Thirdly, ideally it should be possible to recognise who an embodiment represents (perhaps by using a texture map of the user's face on the embodiment or simply a name tag).

- **Viewpoint** An embodiment should convey a user's current viewpoint. This is important since this helps convey what a user is currently interested in. In a subjective view we by necessity compromise this requirement, since the same objects will not be visible in all views.

- **Actionpoint** An embodiment should indicate where a user is manipulating the VE and give other users some indication of the action being performed.

- **Capabilities** In environments that support varying degrees of sophistication of user-client software a user's embodiment should reflect the capabilities of the user's software. For example in an environment such as MASSIVE (Greenhalgh, 1995) where some users will have the ability to communicate using real-time audio and some will only have text, users will want to recognise what media another user supports in order to use the appropriate method to communicate with them.
It can be seen that the current uses for user embodiment are to provide information to yourself and other users. In the next section we argue that this can be extended so that the embodiment provides information to applications as well.

3. A new metaphor for configuration

Slater and Usoh coined the term Body Centred Interaction (Slater & Usoh, 1994) to describe a metaphor for interacting with a virtual environment based on the use of the virtual body. In a similar fashion we refer to our technique as Body Centred Configuration because the various configuration parameters (including those used to configure users’ bodies) are carried with users as part of their virtual bodies.

The X Window System (Scheifler & Gettys, 1986) provides a mechanism whereby configuration information (resources) can be gathered from various sources (the command line, user's files, application defaults files and application hardwired defaults) and merged into a single database which the application can query without needing to know which source this information came from. It has been argued that as VR systems and applications become more complex similar mechanisms will be required for VR (Snowdon & West, 1994). In addressing these issues we have two basic goals:

1. **On-demand embodiments.** It was noted early on in use of DIVE that while it really was not that hard to define a unique embodiment for oneself, other priorities were usually higher and most people would simply use an existing standard embodiment when entering the environment (fig 1A). This often made it difficult to tell who was who; the addition of a randomised colour on the body made it easier to tell participants apart, but still did not convey identity, rather less so, since a person would enter the world clad in a different colour each time. As a response to this problem Sundblad defined the Newt (Sundblad, 1996, chapter 4), a modular embodiment that uses the macro facilities and thus very rapidly can be personalised by changing the definitions of a number of constants that define its size, colour and accessories (fig 1B).

2. **Application configuration.** We would like to provide a more convenient and unified system for configuring applications. The X Window System uses resource files (Converse et al, 1989) to set up session-wide options for applications. The resources can be defined for classes, so that a property which should be the same for several applications can be defined as such at the same time as more specific settings for particular applications can be defined. Once users have created these resource files they do not need to know in advance which applications they will use in a given session - X will provide these resources to applications when they are invoked. In a similar fashion we would like the user not to have to be concerned with which applications they might find in a particular world - as long as the user has specified a configuration for a particular application then...
this should be made available to the application automatically without any further action on the part of the user.

In the next section we describe our approach to configuration which provides an integrated solution to both these requirements.

3.1 Body Centred Configuration

We argue that in distributed virtual environments configuration should not, and in many cases cannot, depend on command lines to an application, or configuration files in a user's home directory. As the applications are distributed, the actual start-up of the application may occur at a distant site, and we as a user only access it through our user client. Indeed, several "applications" may be present in the same virtual environment. Since several users with potentially conflicting desires will access the application no single configuration file is appropriate for initialising the application, nor is the application likely to have access rights to a configuration file which may reside on a remote node. This indicates the necessity for different users to personalise their view, both in the sense of directing options to the application but also in the sense of having a personal, subjective view of the data they access. Since the tool we use to access data also equips us with an embodiment in the environment (necessary to indicate our presence and activity in the virtual environment) we use that to project our options to the application. This means that our embodiment will be matched to our choice of interface and data representation.

4. Implementation

We have implemented our Body Centred Configuration mechanism in DIVE, a distributed virtual environment platform developed by the Swedish Institute of Computer Science (Carlsson & Hagsand, 1993, Hagsand, 1996). DIVE is based on a notional shared database of objects, these are in fact replicated to all members of the process group that defines a world. Updates to an object at one node is distributed to all other nodes using IP multicast. In the latest versions DIVE also supports partial replication such that not all objects are automatically distributed to all nodes but are kept in a process subgroup which a process must explicitly join in order to received information about the subgroup's contents.

4.1 DIVE 3

The aspects of DIVE 3.0.13 that are relevant for this work are the following:

- All entities that can affect objects in the environment, i.e. application programs and interfaces for the human participants, are known as actors.
- Human participants have an embodiment in the environment. This body can be shaped in any manner, but the DIVE visualiser supports the notion of body parts, such as eyes, feet, head, etc. and if parts of the body are labelled accordingly they will be used for viewpoints, ground reference etc.
- Any object, including bodies, may contain Tcl code (Ousterhout, 1994) that gives it behaviour, so that they can react to user interaction, events in the environment or simply perform application computation. In this manner computation can be distributed over the objects (even though the actual computation normally will take place at a single node and then be distributed to all other member nodes).
- Any object may in addition use properties to hold information about itself which is publicly accessible to other objects and applications. Properties essentially consist of tuples specifying the name, type and value of each property. Applications can augment objects with additional properties as a means of storing application specific information. The property mechanism provides a very powerful way for applications and objects to exchange arbitrary information via the DIVE database.
- DIVE object files are passed through the C pre-processor (SUN Microsystems, 1988) before parsing, which allows very rapid and efficient construction of objects through the use of macros.
- DIVE visualisers use configuration files to set a number of parameters for the user, but our suggestion is that this be superseded by options set in the body files instead.
4.2 Using DIVE 3 for Body Centred Configuration

We use the functionality provided by DIVE to achieve our goals as follows: A body is constructed by calling a Tcl script, `makebody`. It will accept various keyword parameters to define an object file to use as body, definitions that can be used to customise that body (when using the Newt body for example), and definitions of property values or the inclusion of property files. Example:

```
makebody -define LENGTH 1.76 -define HEAD normal \  
-body newt.vr -prop VR-VIBE.spikes 3 \  
-out VR-VIBEbody.vr
```

This will create a bodyfile `VR-VIBEbody.vr` with a 1.76m tall Newt body with a normal head and with the `spike2` property for VR-VIBE set to 3, the results of which can be seen in Figure 2.

![Figure 2](image.png)

**Figure 2** A 1.76m tall Newt standing in front of a VR-VIBE display with spikes.

This bodyfile can then be used as the body argument when starting a DIVE visualiser. The body will be constructed in the intended manner and the given property inserted in that body. The VR-VIBE application, on noticing that a new actor has entered the environment can query it for any properties that are either applicable to VR-VIBE specifically or to any application in general. In this case it will find the `VR-VIBE.spikes` property and use the "spiked" display accordingly.

Indeed, the bodyfile does not have to be stored, as the script in absence of an outfile argument will store the description in a temporary file and return that filename as a result, consequently "backquote" syntax can be used to insert that body in the command line as in:

```
vishnu -world VR-VIBE -body `makebody \  
```

2 VR-VIBE can attach "spikes" to selected documents whose size and direction provide additional information about the information stored in the document
Figure 3 An embodiment generated on demand in front of a VR-VIBE display with the relevance threshold set to 10 and shadows drawn under the document icons. Note the name tag above the user's head.

Now VR-VIBE is usable both in a subjective and in an objective mode. If the presentation is subjective adapting the presentation to a new participant is naturally not a problem, but in the objective case it may or may not be allowable to change the representation for all users upon the entry (or command) of a new user. In such cases we put up a warning to the new user to let her retract the change request, but if she persists, warnings are posted to all others users to let them veto the change if it would be contrary to their intentions (Previous experience suggests the necessity to time-out these warnings so that participants who have gone off for coffee will not halt all further interaction.)

For persistence, each application which supports settings of options will have to rewrite changed settings to the body so that they are retained even if the application should crash (and be re-started). At any time the user can store all, or a subset of, current settings to a file, so that they can be reused.

5. Extensions to Body Centred Configuration

If we extend the Body Centred Configuration approach to allow configuration options to be embedded in any DIVE object (this is already supported by the current implementation) then there are some additional possibilities that could be explored. One possibility would be to use objects in the DIVE world to hold collections of options. An application might support a 3D "drag & drop" style of interface that allowed users to present the application with objects containing sets of options and thereby change the behaviour of the application. A user might have a collection of such objects which could be used to switch between commonly used configurations in a single operation. For example, in
the case of VR-VIBE a user might have an object that could be “given” to VR-VIBE to instruct it to visualise a particular data set with a particular visualisation style.

A second possibility might be to use these options as an equivalent to the resource fork of a file on the MacOS operating system. In this case an application would inspect these properties to find out some meta-information about an object and decide what to do with it. Several DIVE applications, such as VR-VIBE, already use specific properties in objects to allow them to “recognise” a DIVE object as one they created.

A third possibility would be to provide more dynamic information concerning the current state of users and objects. For example, a user's body might contain information about the current 'level-of-activity' of the user (e.g. how long since the user last moved). Other users and applications could then examine this property to determine if the user was currently present in the world or whether they'd temporarily abandoned their virtual body in order to attend to some task in the real world. Tcl scripts could be attached to the user's client process that would watch these properties and create a visual representation of them in the virtual world - for example, placing “Zzzzz” above the user's head, or rotating the user's virtual body so that it was horizontal (sleeping) to indicate that the user is not currently actively controlling their virtual body. Implementing dynamically updated properties of this kind would be a trivial extension to the current implementation.

6. Summary

We have noted that neither command line options nor configuration files are suitable for setting parameters for distributed virtual environment applications. Instead we propose that the embodiments used to interact with the environment and any applications therein should carry with them parameter settings that are then interpreted by applications in the environment.

We have implemented this method in the DIVE environment with scripts that support the on-the-fly generation of bodies, settings of parameters and the re-storing of changed parameter settings.

Acknowledgements

Many thanks to Lennart Fahlén and the Swedish Institute of Computer Science (SICS) for funding Dave Snowdon as a guest researcher at SICS for the period of this work. We also thank Olof Hagsand, Mårten Stenius and Emmanuel Frécon for help with DIVE.

References


Ousterhout, J.K. (1994), Tcl and the Tk Toolkit, Addison-Wesley


Sun Microsystems (1988), cpp --- the C language pre-processor.