Supporting Reusability of VR and AR Interface Elements and Interaction Techniques

Wolfgang Broll and Jan Herling

Collaborative Virtual and Augmented Environments
Fraunhofer FIT
53754 Sankt Augustin, Germany
{Wolfgang.Broll,Jan Herling}@fit.fraunhofer.de

Abstract. In contrast to 2D environments which apply well established user interface elements and generally accepted interaction techniques, VR and AR applications typically provide rather individual and specific realizations. This often leads to inconsistent user interfaces and a long and cumbersome development process. In this paper we show how we extended our approach on modeling VR and AR interface elements and interaction techniques represented by interaction and behavior objects by some simple yet powerful mechanisms: modules, templates, and inheritance. We will also show how specific examples could benefit from that approach.

Keywords: Virtual Reality, Augmented Reality, Mixed Reality, 3D User Interfaces, Multi-modal User Interfaces, Interaction Techniques.

1 Introduction

The majority of the overall effort and time required to develop Virtual Reality (VR) and Augmented Reality (AR) applications is spent for the development of specific user interface elements and underlying interaction techniques [2]. The first issue is also true for many traditional 2D desktop applications, although it is generally easier there due to the availability of appropriate user interface design tools and well established design guidelines (both not available for VR and AR). However, when it comes to interaction techniques, the difference is even more obvious. While in 2D desktop environments applications usually rely on the well known and well established WIMP metaphor, no standard interaction techniques for VR and AR exist [4]. In contrary, they often depend on the specific (3D) input devices available, the user's preferences as well as the specific requirements of the application. Further, most users have no or little experience using VR and AR technology and related devices and interaction techniques. This additionally complicates the user interface design and often results in either rather poor user interfaces or several iterative user trial and update cycles requiring a high implementation effort.

Existing approaches to overcome this problem include user interface description languages such as UIML [1] and InTML [7], authoring tools such as DART [13] or iaTAR [11], scene graph related approaches such as Behavior3D [6] and YABLE [5], and component based approaches such as BodyElectric [10] or Unit [12].

Our original approach [4] combines synchronous control and data flows with asynchronous event and network distribution. It is based on a predefined yet extendable set of components representing integral parts of VR/AR interaction techniques and related device handling. Several of these components are combined into a so called interaction or behavior object (aka interactive bit). These objects react to event input from input devices or other objects, or depend on the elapsed time or are synchronized with per frame scene updates.

While the overall approach has already proved to be quite powerful for fast and easy realization of application prototypes, reusability of already realized interaction techniques going beyond simple copy and paste has been a major request by the users.

In this paper we will present our recent extensions allowing for easy and powerful reusability of user interface elements and interaction techniques. In section 2 we will briefly explain the major concepts and components of our original approach. In section 3 we will then present our extensions allowing for an efficient reusability of interaction and behavior objects. In section 4 we will provide some example scenarios to demonstrate how specific user interfaces and interaction techniques can benefit from the approach, before concluding and looking at some future work in section 5.

2 Our Original Approach

In our original approach [4] we used a component based approach allowing for modeling rather than programming 3D user interfaces and interaction techniques by assembling interaction and behavior objects from a set of pre-defined components. These components represent integral parts of autonomous object behavior, user interface elements, and interaction techniques. Generally we distinguish between seven component categories:

- Base components, for receiving and sending events and to query information or register for updates or changes at other system components. This for instance allows for registering for an input device and to manipulate a scene graph object accordingly.
- Execution components are used to influence the control flow and to perform calculations or even more complex behavior by scripting.
- Time-dependent components are invoked independent of user input at specific timestamps or after certain periods, etc.
- Key-value mapping components are in particular used for realizing animations, but also for mapping between different data sets and for autonomous behavior based on state machines.
- Device input calculation components allow for easy calculation as necessary for using 2D devices in 3D interaction techniques and for modifying 3D input data. This includes components for transforming and filtering data.
- Data storage components, allow for temporal (memory) and permanent (file) storage and retrieval of data.
- Networked components, finally provide enhanced versions of other components for better supporting their usage in networked environments allowing for more advanced distributed interaction mechanisms than by replicated scene graphs only.

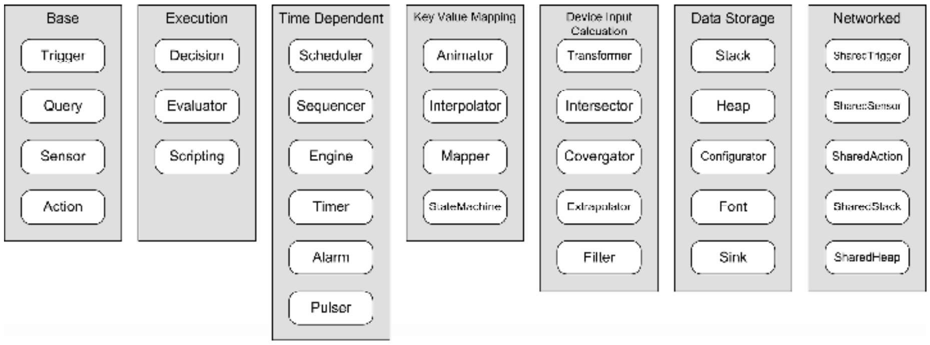

Fig. 1. Component hierarchy for assembling interaction and behavior objects

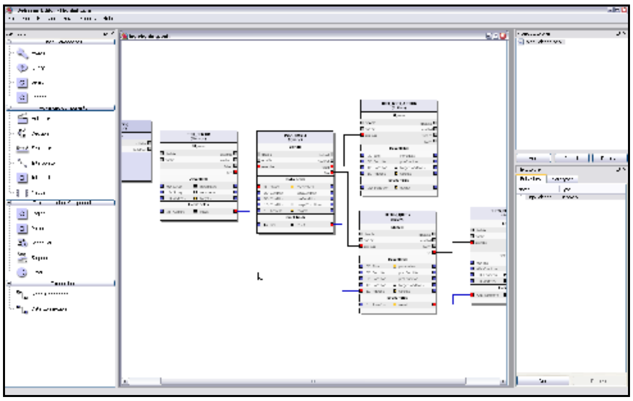

Fig. 2. Visual programming environment for defining interaction and behavior objects

Figure 1 provides an overview of the currently supported components.

Each interaction and behavior object may use an arbitrary number of components of each type, which are connected by a signal/slot mechanism. This mechanism allows us to transfer data values (such as integers, floats, strings, vectors, matrices, etc.) between the individual components. In order to specify the control flow for the execution of the individual components, event signals are used (also specified by the signal/slot mechanism). However, these mechanisms apply for the internal communication between the individual components of a single object only. For external communication, i.e. access to input and output devices, scene graph objects, or services such as object picking or collision detection, an asynchronous event based mechanism is used. Thus, it allows for easy application of the approach to networked/distributed environments, where events will need some time to be transmitted. This event based mechanism is also used for communication between different interaction and behavior objects.

While the interaction and behavior object description uses a text based description, we additionally realized a visual programming environment (see Figure 2) for easier arrangement and editing of the individual components, and their interconnection by data connections and event signals [3].

3 Our Approach to Support Reusability

In order to reduce the effort for realizing similar interaction techniques we extended our original approach by four mechanisms for supporting the reusability of already defined interaction and behavior objects. These are:

- Instantiation
- Templates
- Modules
- Inheritance

By instantiation we provide a mechanism which allows for easy attachment of interaction techniques to several user interface elements at once. Therefore a master object is defined, which is actually not immediately used. As part of the definition an attachment rule is defined. Whenever the attachment rule is fulfilled, a instance (copy) of the object defined is created and attached to the objects specified. As the specification of the target attachment is quite flexible (e.g. specifying specific scene graph objects by wild cards) and the actual attachment may be either explicit or implicit (e.g. whenever a new object is created), this mechanism allows for easy reuse of elements already defined. The short coding fragment below shows how an object is specified to be dynamically attached to objects in a certain branch of an X3D scene graph having a name with the prefix "CHAIR". The attachment is re-evaluated each time a file is loaded or a node is added to the scenegraph.

```
Behavior {
  targets XSG:X3D::MY_ROOM/FURNITURE/CHAIR*
  attachment [LOAD_FILE, ADD_NODE]
...
// definition of individual components follows here
Sensor GRAB {
  ...
}

Action MOVE {
  targets .  // i.e. the local entity the
             // behavior is attached to
  ...}
```

Further, we developed two template mechanisms: one for entire interaction and behavior objects and one for individual components. The first one allows for specifying a full object with additional parameters applied during initialization. Upon instantiation the parameters are then used to create the actual object. The individual components of such a template are invisible and cannot be accessed directly within the instantiated object. A simple example would be a behavior simulating the movement caused by a parabolic flight. As a parameter you may specify the velocity (direction and amount).

The second template mechanism applies to individual components. A component template is always based on one or several already existing components (or component templates). In addition to parameters a component template has to define the data and control flow interfaces of the new template component. Those are mapped to the corresponding data and signal slots of the underlying sub-components. This allows us to combine several components into a new component template, which then may be used similar to built-in components.

Further, our approach supports a mechanism for realizing interface and interaction technique modules. It may be used to include previously defined interaction and behavior objects or templates. It allows for the provision of frequently used interaction and interface modules (e.g. for rigid 3D manipulation such as positioning and rotating, support for tangible user interface elements, or individual navigation styles for specific types of input devices).

Finally we provide an inheritance mechanism. It allows us for the reuse of already defined behaviors object, enabling the rapid construction of new objects with rather small modifications (either replacing parts of their behavior or by applying additional features). For example, a standard object behavior which highlights a scene graph objects upon selection can be easily modified and extend to use a color frame and a sound feedback instead, while the underlying selection mechanism is kept). Inheritance applies during instantiation of an object and applies to any part of its definition.

A big advantage of the four approaches presented is their ability to be combined and even nested arbitrarily.

4 Example Scenarios

In this section we would like to present and discuss some example scenarios. In particular we will show how those examples benefited from the mechanisms introduced in this paper.

4.1 Autonomous Behavior

In this little demo example we realized an arena containing a configurable yet arbitrary number of autonomous robots. Each robots goal is to explore the given arena and to kill as many other robots as possible. The entire demonstration scenario is based on two 3D files (one containing the geometry of the arena and one for the geometry of a robot), three behavior objects (in a single file) and a couple of sound files representing the individual activities of the robots. The three behavior objects are:

- One defined as template, where the actual instance used allows for specifying the number of robots and the size and location of the area for their potential initial starting position. This one will include the robot file as many times as specified, giving each of them an individual name (ROBOT1 … ROBOTn) and define an arbitrary starting position for each robot.

- One for registering the virtual 3D scene with the tracked marker on the table. This is a standard marker tracking behavior which is loaded from an appropriate module using the include mechanism.
- One for the actual behavior of the individual robot. Its main component is a *State-Machine* (scan, fire, cruise, explode, idle) activating and deactivating individual tests (i.e. picking and/or collision detection) and animations. The behavior object is defined as master allowing for an automatic instantiation and attachment of an individual copy to each robot's geometry upon creation.

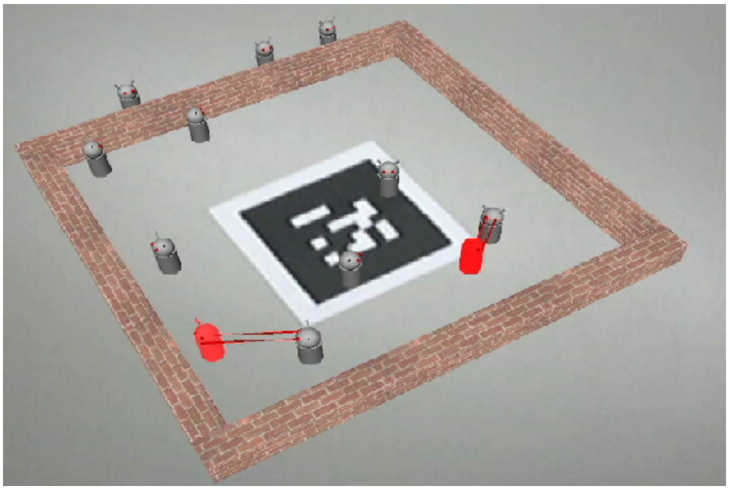

Fig. 3. Demo application showing autonomous robots trying to destroy each other

4.2 Application Specific Interaction Styles

TimeWarp [8] – an application showcase of the integrated project IPCity [9] - is a location-based Mixed Reality game where the players have to solve challenges in different locations distributed throughout the old part of Cologne. In addition to enhancing the real environment by the virtual challenges, the game takes the players to different epochs in the history of the city by augmenting the real environment visually and acoustically. Players use either head-mounted optical see-through displays or handheld ultra mobile PCs applying video see-through AR. While the individual challenges are quite different regarding their content, the interaction techniques for selecting and manipulating content are shared by all challenges. As each challenge typically is realized independently (even by different developers or game designers), it was important to ensure that the interaction techniques used are independent of the individual challenge. For that reason an interaction library module was created and included in each challenge implementation. This also ensured that changes to the interaction techniques apply to all challenges automatically. Specifically the reusability mechanisms also allowed us to support different flavors of the interaction techniques depending on the devices used. While for head-mounted displays we relied

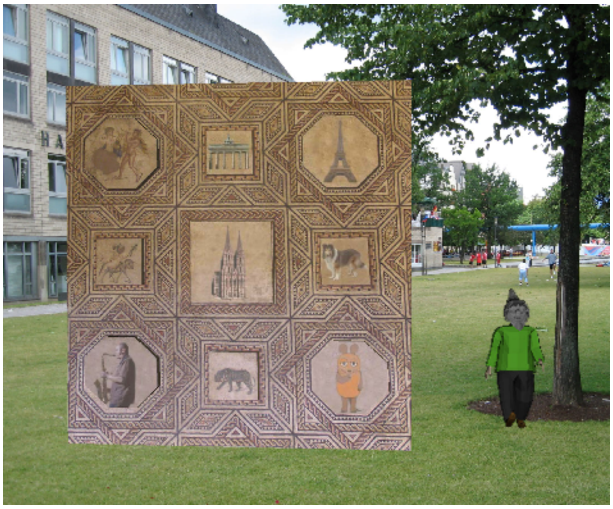

Fig. 4. A virtual version of the famous Roman Dionysus mosaic in Cologne was modified and had to be re-arranged to represent the original figures by the players

on ray-based picking along the viewing direction in the center of the current view, ray-based picking on the handheld devices providing a touch screen was more flexible as objects could be directly selected by the players using their fingers. Thus, we just inherited and extended the original ray-based mechanism, adding the inherited interaction technique to the module.

Additional reusability mechanisms were applied to realize individual challenges. The challenge to reconstruct the original tiles of the famous Roman Dionysus mosaic (see Figure 4) consists of nine interactive tiles. Each of them may be turned by 180° based on user interaction. While each tile seems to consist of two faces, there are actually three different texture applied to each tile in a sequence. Thus, the behavior object for turning a tile and replacing the textures from a set uses the template mechanism to specify the individual texture as parameters.

4.3 Guided Tour

Another application we realized using the mechanisms presented was a guided tour for a stereoscopic 3D presentation of a reconstruction of the Bamiyan Buddha statues in Afghanistan for a museum (see figure 5). The camera was animated smoothly flying from one point of interest to the next one. This was the standard mode in the museum. For guided tours however, there was a SpaceTraveler as navigation device. By that, a guide could interrupt the animated camera at any point and continue individual navigation. The automatic animation of the camera would resume after a certain time of inactivity, flying smoothly to the next POI in line.

Here, the SpaceTraveler navigation is loaded from a navigation module containing navigation support for various input devices as template. The default 6-DOF

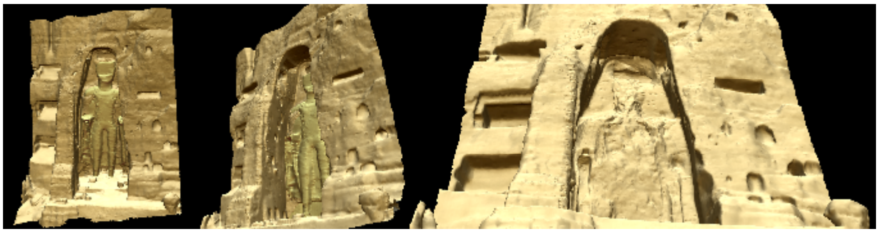

Fig. 5. Guided tour and interactive navigation for a presentation of the destroyed Bamiyan Buddha statues in the Gandhara region of Afghanistan

navigation provided by the device is reduced to 5-DOF by disabling the roll axis for easier use by occasional users, since user tests showed that they otherwise get easily lost in the scene. This was done by inheriting from the template a modified behavior template, thus combining the template and the inheritance mechanism.

5 Conclusion and Future Work

In this paper we introduced our approach on supporting the reusability of interaction techniques and user interface elements in VR and AR environments. We showed how the approach overcomes the limitations of existing approaches in respect to flexibility, scalability, and usability.

Beside the detailed introduction of the mechanisms applied, we presented several examples from ongoing projects, showing the actual feasibility of the individual mechanisms and the usability of the overall approach.

In our future work we will further extend the interface and interaction libraries of predefined behavior templates and base classes, reflecting all major 3D interaction techniques and interfaces. We further intend to combine this with a tutorial on best practice examples for easier use by new interface developers and in particular for using the approach for teaching VR and AR user interface classes.

Acknowledgments

The authors thank their colleagues at the Collaborative Virtual and Augmented Environments Department at Fraunhofer FIT for their comments and contributions. Special thanks to our colleagues Lisa Blum for her work on the visual programming environment, Richard Wetzel for realizing the TimeWarp Dionysus mosaic challenge and Robert Menzel who realized the Gandhara guided tour. They further wish to thank their project partners of the IPCity project for their ideas, cooperation, and their support. IPCity (FP6-2004-IST-4-27571) is partially funded by the European Commission as part of the 6th Framework.

References

1. Abrams, M., Phanouriou, C., Batongbacal, A.L., Williams, S.M., Shuster, J.E.: UIML: An Appliance-Independent XML User Interface Language. In: Proceedings of the Eighth International World Wide Web Conference, pp. 1695–1708. Elsevier, Toronto, Canada (1999)
2. Bowman, D.A., Kruijff, E., LaViola, J.J., Poupyrev, I.: 3D User Interfaces: Theory and Practice. Addison-Wesley, Boston (2004)
3. Broll, W., Herling, J., Blum, L.: Interactive Bits: Prototyping of Mixed Reality Applications and Interaction Techniques through Visual Programming. In: Proc. of the 3rd IEEE International Symposium on 3D User Interfaces 2008 (IEEE 3DUI 2008), pp. 109–115. IEEE Computer Society, Los Alamitos (2008)
4. Broll, W., Lindt, I., Ohlenburg, J., Herbst, I., Wittkämper, M., Novotny, T.: An Infrastructure for Realizing Custom-Tailored Augmented Reality User Interfaces. IEEE Transactions on Visualization and Computer Graphics 11(6), 722–733 (2006)
5. Burrows, T., England, E.: YABLE—yet another behaviour language. In: Proceedings of the Tenth international Conference on 3D Web Technology, Web3D 2005, Bangor, United Kingdom, March 29-April 01, 2005, pp. 65–73. ACM, New York (2005)
6. Dachselt, R., Rukzio, E.: Behavior3D: an XML-based framework for 3D graphics behavior. In: Proceeding of the Eighth international Conference on 3D Web Technology, Web3D 2003, Saint Malo, France, March 9-12, 2003, p. 101. ACM, New York (2003)
7. Figueroa, P., Green, M., Hoover, H.J.: InTml: a description language for VR applications. In: Proceeding of the Seventh international Conference on 3D Web Technology, Web3D 2002, Tempe, Arizona, USA, February 24-28, 2002, pp. 53–58. ACM, New York (2002)
8. Herbst, I., Braun, A., McCall, R., Broll, W.: TimeWarp: interactive time travel with a mobile mixed reality game. In: Proceedings of the 10th international Conference on Human Computer interaction with Mobile Devices and Services, MobileHCI 2008, Amsterdam, The Netherlands, September 2-5, 2008, pp. 235–244. ACM, New York (2008)
9. IPCity – Integrated Project on Interaction and Presence in Urban Environments, `http://www.ipcity.eu`
10. Lanier, J., Grimaud, J.-J., Harvill, Y., Lasko-Harvill, A., Blanchard, C., Oberman, M., Teitel, M.: Method and system for generating objects for a multi-person virtual world using data flow networks. United States Patent 5588139 (1993)
11. Lee, G.A., Nelles, C., Billinghurst, M., Kim, G.J.: Immersive Authoring of Tangible Augmented Reality Applications. In: Proceedings of the Third IEEE and ACM international Symposium on Mixed and Augmented Reality (ISMAR 2004), November 2-5, 2004, pp. 172–181. IEEE Computer Society, Los Alamitos (2004)
12. Olwal, A., Feiner, F.: Unit: modular development of distributed interaction techniques for highly interactive user interfaces. In: Spencer, S.N. (ed.) Proceedings of the 2nd international Conference on Computer Graphics and interactive Techniques in Australasia and South East Asia, GRAPHITE 2004, Singapore, June 15-18, 2004, pp. 131–138. ACM, New York (2004)
13. MacIntyre, B., Gandy, M., Dow, S., Bolter, J.D.: DART: a toolkit for rapid design exploration of augmented reality experiences. In: Proceedings of the 17th Annual ACM Symposium on User interface Software and Technology, UIST 2004, Santa Fe, NM, USA, October 24-27, 2004, pp. 197–206. ACM, New York (2004)