COMPONENTS FOR IMMERSION

Alexandre R.J. François

Integrated Media Systems Center
University of Southern California, Los Angeles, CA

ABSTRACT

A person is immersed when they feel part of an environment they experience and influence. While virtual immersion systems are usually designed on a case by case basis, and are not easily reusable or scalable, our goal is to specify and develop a framework for the design and integration of immersive systems. We address the issues raised in the design and implementation of the middleware components necessary for immersion, in a generic, extensible, modular architecture for Integrated Media Systems we have developed for efficient generic concurrent processing of data streams. Design principles are illustrated on specific visual immersion components. Utilization of these components is demonstrated with a real-time immersive interactive application.

1. INTRODUCTION

A person is immersed when they feel part of an environment they experience and influence (for example, we are all immersed in the physical world). Computer applications and simulations can provide various degrees of immersion, depending on the nature and behavior of the environment in which the user is immersed, and on the modalities through which the user experiences and influences this environment.

Immersive systems, examples of which include first person video games and virtual reality simulations, require real-time on-line processing and mixing of heterogeneous data, some persistent (but dynamic), such as the model(s) of the environment, some volatile, such as the user output (experience) and input (influence) data streams. Such integrated media systems are usually designed on a case by case basis, and are not easily reusable or scalable. Our goal is to specify and develop a framework for the design and integration of immersive systems.

Interactive 3-D frameworks such as VRML [6] and more recently X3D [7], do not allow consistent handling nor generic processing of data streams. Similarly, data stream frameworks such as MPEG21 [5] focus on multimedia delivery and consumption, and thus are not natural frameworks for immersion. A large number of independent libraries and toolkits implementing necessary functionalities are available from research and industry. However, they emphasize specific aspects of media processing, and do not address their integration. An example of such an heterogeneous set of libraries is Microsoft's DirectX [1].

In order to address these issues, we have introduced the generic, extensible, modular architecture for Integrated Media Systems presented in figure 1 [2]. The middleware

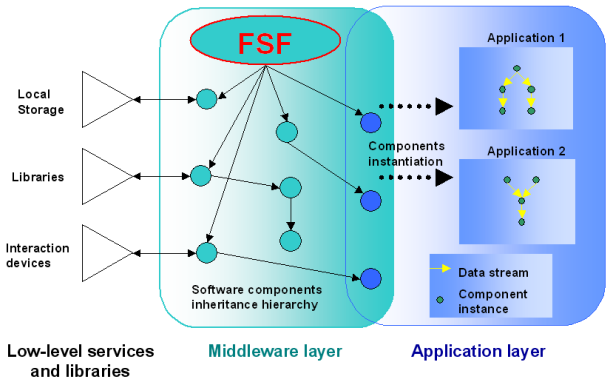

Figure 1. A modular software architecture.

layer provides an abstraction level between the low-level services and the applications, in the form of software components. The cornerstone of this layer is the Flow Scheduling Framework (FSF), an extensible set of classes that provide basic synchronization functionality and composition mechanisms to develop data-stream processing components. The application layer allows to compose and execute multimedia applications using the software components in a dataflow-based, immersive programming environment.

This architecture, whose implementation is released as an open source project [4], addresses and promotes critical aspects of integrated media systems component development and integration such as efficiency, scalability, extensibility, reusability and inter operability. Its suitability for real-time data stream processing has been demonstrated in the context of video analysis [3].

In this paper, we address the design of the middleware

components necessary for immersion. After an overview of the Flow Scheduling Framework in section 2, we analyze the design issues raised in the implementation of simulated dynamic environments and user interaction in section 3. Utilization of such components is demonstrated with a real-time immersive interactive application in section 4. We propose concluding comments and perspectives in section 5.

2. THE FLOW SCHEDULING FRAMEWORK

The FSF [2], cornerstone of the middleware layer, is an extensible set of classes that provide basic synchronization functionality and composition mechanisms to develop data-stream processing components. The FSF specifies and implements a common generic data and processing model designed to support stream synchronization in a concurrent processing framework. Applications are built in a data-flow programming model, as the specification of data streams flowing through processing centers, where they can undergo various manipulations. This extensible model allows to encapsulate existing data formats and standards as well as low-level service protocols and libraries, and make them available in a system where they can inter-operate.

Information is modeled as data *streams*. A stream represents one or several synchronized objects, i.e. expressed in the same time referential. An application is specified by a number of streams of various origins and the manipulations they undergo as they pass through processing nodes called *cells* (see figure 2). Streams carry time samples, which can represent instantaneous or integrated data. *Active streams* carry volatile information, while *Passive streams* hold persistent data in the application. Passive streams are maintained in the system by objects called *sources*.

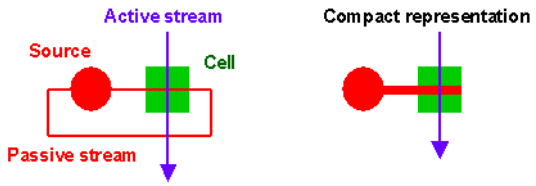

Figure 2. The FSF generic processing unit.

2.1. Generic Data Model

We define a structure that we call *pulse*, as a carrier for all the data corresponding to a given time stamp in a stream. In an application, streams can be considered as pipes in which pulses can flow (in one direction only). The time stamp characterizes the pulse in the stream's time referential, and it cannot be altered inside the stream. Time referential transforms require inter-stream operations.

As a stream can represent multiple synchronized objects, a pulse can carry data describing time samples of multiple individual objects as well as their relationships. In order to facilitate access to the pulse's data, it is organized as a mono-rooted composition hierarchy, referred to as the pulse structure. Each node in the structure is characterized by its type and identified by its name. The framework only defines the base node type that supports all operations on nodes needed in the processing model, and a few derived types for internal use. Specific node types can be derived from the base type to suit specific needs, such as encapsulating standard data models.

2.2. Generic Processing Model

The FSF processing model is designed to manage generic computations on data streams. A generic cell type, called Xcell, supports basic stream control for generic processing. Custom cell types implementing specific processes are derived from the Xcell type to extend the software component base in the system.

In an Xcell, the information carried by an active stream (active pulses) and a passive stream (passive pulses) is used in a process that may result in the augmentation of the active stream and/or update of the passive stream. Each Xcell thus has two independent directional stream channels with each exactly one input and one output. Part of the definition of the Xcell type and its derived types is the process that defines how the streams are affected in the cell.

An application is specified by a graph of cells with two independent sets of links: active and passive. Active and passive connections are different in nature, and cannot be interchanged.

3. COMPONENTS FOR IMMERSION

Virtual immersion is the interaction between a simulated dynamic environment, and a user that experiences and influences this environment. We analyze the design issues raised in the implementation of both aspects in our framework, and illustrate with specific design of visual immersion components.

3.1. Virtual environments

Simulating a dynamic environment requires the use of models that describe its state at a given time, and the way it evolves in time. These models are clearly persistent (and dynamic) data in the system.

Implementing the necessary components in the FSF

model requires to carefully discriminate between purely descriptive data, processes, and process parameters, and analyse how these model elements interact in the simulation.

Our design for 3-D modeling is directly inspired from VRML [6] and the more recent X3D [7]. The main descriptive structure of the model is a scene graph, which constitutes a natural specialization of the FSF data model. Descriptive nodes, such as geometry nodes and the Shape, Appearance, Material and Texture nodes, are straightforward adaptations of their VRML counterparts (see figure 3).

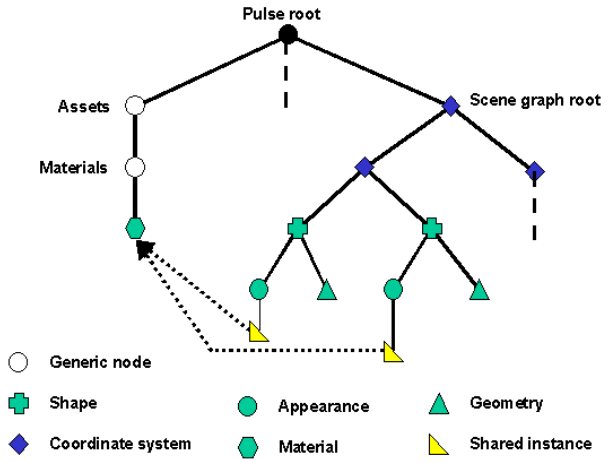

Figure 3. Scene graph in the FSF data model.

The VRML Transform node, however, is an example of semantic collapse, combining a partial state description and its alteration. A transform should actually be considered as an action, applied to a coordinate system. It can occur according to various modalities, each with specific parameter types. Consequently, in our model, geometric grouping is handled with Coordinate System nodes, which are purely descriptive. In order to provide a complete state description in a dynamic model, the Coordinate System node attributes include not only origin position and basis vectors, but also their first and second derivatives. Various transforms are implemented as cells, with their specific parameter nodes, that are not part of the scene graph (although they are stored in the same source).

Scene graph elements can be independently organized into semantic asset graphs, by using the Shared Instance node, instances of which can replace individual node instances in the scene graph, as illustrated for a material node in figure 3.

Animation, i.e. scene evolution in time, is handled by cells implementing specific processes, whose parameter nodes are not part of the scene graph. This ensures that both persistent data, such as the scene graph, and volatile data, such as instantaneous description of graph components evolution, are handled consistently. This aspect is critical when simultaneous independent processes are in play, as it is often the case in complex simulations. Note that process parameters can also evolve in time, either as a result of direct feed-back or through independent processes.

The analytical approach followed for modeling, by separating descriptive data, processes, and process parameters, supported by a unified and consistent framework for their description and interaction, allows to integrate seamlessly 3-D graphics, video streams, audio streams, and other media types.

3.2. Interaction

Interaction is a particular data stream loop feeding the user with a perceptual representation of the internal model (experience), collecting the user's reaction through various sensory devices and modifying the state of the internal model accordingly (influence). From the system's point of view, these data streams are volatile, and the processes involved in producing and processing them are of the same nature as those carrying procedural internal evolution tasks. Interaction components should therefore be modeled consistently in the FSF.

A minimal interaction subsystem, introducing the different categories of cells involved, is presented in figure 4.

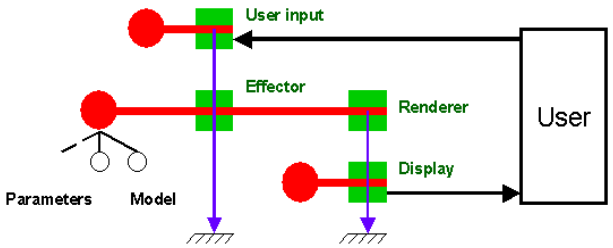

Figure 4. Minimal interaction sub-system graph.

Input cells collect user input and generate the corresponding pulses on active streams. They encapsulate such input devices as mouse, keyboard, 3-D tracker, etc. Effector cells use the input data to modify the state of the internal model (including possibly process parameters). In some cases, the data produced by the input device requires some processing in order to convert it into data that can be used by the effector. This is the case for example with visual interfaces, in which the input device is a camera, which produces a video stream that must be analyzed to extract the actual user input information. Such processes can be implemented efficiently in our architecture (see [3]), and be integrated consistently in an interactive application graph.

Renderers are particularly complex processes in terms

of simultaneous volatile and persistent data manipulation. They provide a perceptual instantaneous snapshot of the environment captured by a (virtual) device such as microphone or camera, which is itself part of the environment. The FSF intrinsic parallelism and synchronization mechanism are particularly well suited for well defined, consistent and efficient handling of these type of tasks.

The snapshots produced by the renderer cells are presented to the user by display cells, which encapsulate output devices such as screen image display, stereo displays, sound output, etc. In some cases, e.g. for haptics, input and display functions are handled by the same device and the corresponding cells are combined into a single cell.

The very general model for interaction presented above, applies to any type of input and output modalities. It subsumes low-frequency interaction models such as the familiar desktop metaphor. In the context of parallel data stream processing of the FSF, it offers complete and consistent support for the high frequency interaction required for immersion.

4. EXAMPLE APPLICATION

To illustrate the design principles introduced above, we have implemented a real-time, dynamic environment mapping immersive application.

The user is placed in the center of the world, which is modeled by a cylindrical polygonal surface on which a video stream pre-recorded from a panoramic camera system is texture-mapped in real-time. The user only sees the part of the environment he is facing, and controls his viewing direction using a tracker, all in real-time. Although the user cannot modify the world he perceives, he can modify his perception of the world in real-time, which provides a sense of immersion that can be reinforced by the use of natural sensory input and output devices (such as a 3-D tracker and a head mounted stereo display).

The application graph is presented in figure 5. In this particular case, the input and output streams are handled synchronously, by the use of a single active stream. The pulse rate is dictated by a pulsar cell (Pcell).

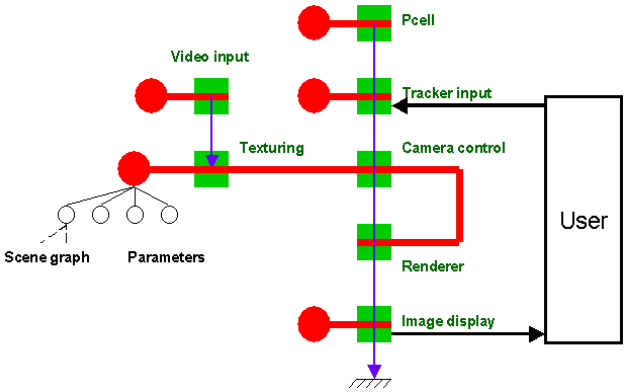

Figure 5. Example immersion application graph.

This application, although simple, illustrates the flexibility and efficiency of our architecture. In particular, thanks to the modularity of the framework, the same core application (model and rendering) can be effortlessly combined with various input/output modalities (e.g. mouse or tracker input, screen or head mounted display).

5. CONCLUSION

We have presented a framework for the design and integration of immersive systems. It is based on a generic, extensible and modular architecture designed for efficient and consistent data stream processing. The generality of the framework requires a similarly general approach in the design of components for immersion. We have outlined some generic design principles, and illustrated them with 3-D visual immersion components, used to integrate a real-time immersive interactive application.

Although the immersion level provided by the existing set of components is still limited, the proposed framework provides, by design, support for consistent integration of components for handling of all media types and interaction devices. We are currently adding to the component base (e.g. audio, haptics, networking) in order to reach our ultimate goal of total sensory immersion, or immersipresence.

6. ACKNOWLEDGEMENTS

This research has been funded by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152, with additional support from the Annenberg Center for Communication at the University of Southern California and the California Trade and Commerce Agency.

7. REFERENCES

[1] DirectX. http://www.microsoft.com/directx/

[2] A. François and G. Medioni, "A Modular Middleware Flow Scheduling Framework." Proc. ACM Multimedia 2000, pp. 371-374, Los Angeles, CA, November 2000.

[3] A. François and G. Medioni, "A Modular Software Architecture for Real-Time Video Processing." Proc. International Workshop on Computer Vision Systems, Vancouver, Canada, July 2001.

[4] A. François. Modular Flow Scheduling Middleware. http://mfsm.SourceForge.net

[5] MPEG21 “Multimedia Framework”. http://mpeg.telecomitalialab.com/

[6] The Virtual Reality Modeling Language. International Standard ISO/IEC 14772-1:1997. http://www.vrml.org

[7] X3D. International Standard draft. http://www.web3d.org