Communication in Componentized 3D Graphics Multimedia Systems

Vlastimil Miléř
Postgraduate student
Czech Technical University
Karlovo náměstí 13
121 35, Praha 2, Czech Republic
xmiler@fel.cvut.cz

Bohuslav Hudec
Associate Professor
Czech Technical University
Karlovo náměstí 13
121 35, Praha 2, Czech Republic
hudec@cslab.felk.cvut.cz

ABSTRACT

This paper describes two aspects of architecture of componentized systems. First, properties of the basic building blocks (acquisition and presentation components) are discussed and their critical points are identified. Second, the communication between components in the system is analyzed. Different ways of communication and different communication scenarios are evaluated. The obtained results are useful when designing a new system, enlarging current systems or choosing a system for particular task.

The area of 3D graphics and multimedia was the main source for examples in the paper and the whole paper is oriented on applying the described principles in this area.

Keywords

System, Component, Cooperation, 3D graphics, Multimedia.

1. INTRODUCTION

As the requirements on software grow more complex, so does the size of the software systems. The obvious way to compensate for the complexity is to divide the system into components. The structure of the system and the quality of interfaces between components determine certain properties of the system – its extensibility, robustness, customizability and others. Sometimes, these properties are more important than the actual functions of the available components. The structure and inter-component communications have earned more interest in the recent time. This paper pursues this trend by analyzing the basic structure and ways of communication in componentized systems with 3D graphics and multimedia content.

2. COMPONENTIZED SYSTEM

Each system is a set of several components. The classification to systems and applications is hierarchical. From one point of view a set of applications may appear as a system. From another point of view this system is a component of another larger system.

The physical mapping of components into applications is discussed where applicable. A component can manifest itself as one application visible to the user. Alternatively an application can consist of more components or more applications can form one component.

Sometimes the term 'global system' is used. This global system consists of all applications available. Almost all applications are trying to become part of the global system by implementing the required interfaces (usually import/export functions).

WSCG POSTERS proceedings
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.

3. COMPONENT CLASSIFICATION IN A SYSTEM

Most components (and applications) can be classified according to their main purpose into one of two categories.

- Acquisition component.
- Presentation component.

This is very rough but sufficient classification. Presenting interactive multimedia and 3D graphics data to the user is the main purpose of majority of

real components – the presentation components. On the other hand, the data need to be obtained somehow. Therefore mechanisms for acquisition must exist – the acquisition components.

Acquisition component

A typical representative of this kind of component is an editor, where data are created manually, or capturing software, where automatic acquisition is performed.

Presentation component

A presentation component uses some form of already existing constant data and presents them to the user. For example it could present a 3D graphic on internet, playback audio or video.

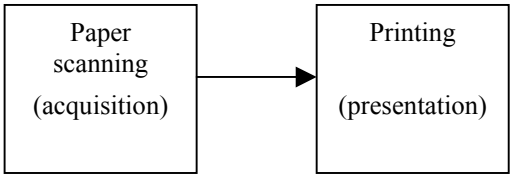

Figure 1. A minimum system with one acquisition and one presentation component

Examples of component diagrams

Each system must have at least one presentation and one acquisition component and an interface between them. The complexity of systems may vary.

Figure 1 shows components of a very simple system that is able to copy paper documents. The system is not able to do any image processing or preview before printing.

Figure 2 shows a simplified system where interactive 3D scenes with multimedia content are produced and presented to the user.

Componentized systems summary

It is typical that the number of distinct acquisition components is higher than the number of presentation components. The acquired data are usually used multiple times and there are more copies of the same presentation components distributed and used by users.

A question arises, which component type is more important for the whole system? There is another factor in the equation – the quality of the interface. Examining current systems, we can see that insufficient quality of any of these parts can make the whole system unusable. Bad acquisition component would produce low amount of data or low quality data. Bad presentation component would lower the end user experience and bad interface would prevent effective communication and future extensions. Therefore it is best not to focus on one part of the system only.

4. ACQUISITION COMPONENT ARCHITECTURE

Figure 3 shows typical architecture of an acquisition component in current systems. An acquisition component obtains data from user, performs operations on them and finally sends them to another component using defined protocol.

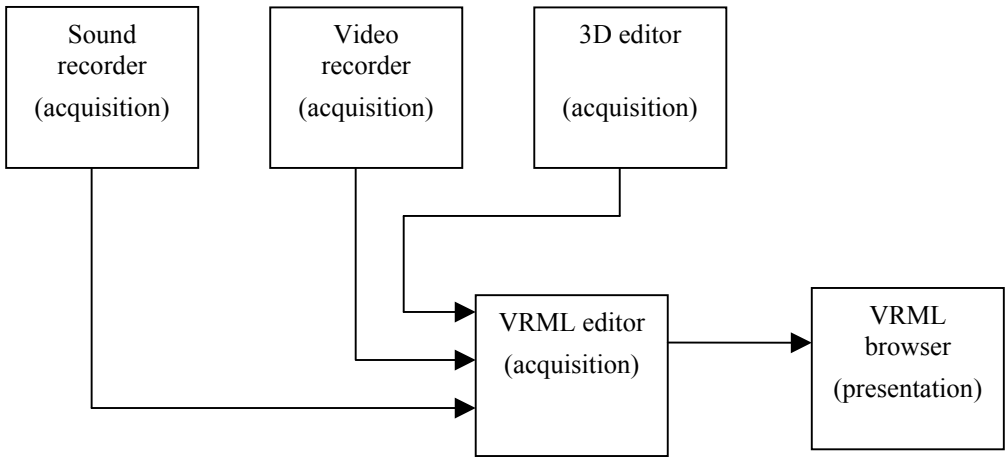

Figure 2. In this nonlinear system, the data from the three elementary acquisition components are assembled in the VRML editor. The elementary components are independent.

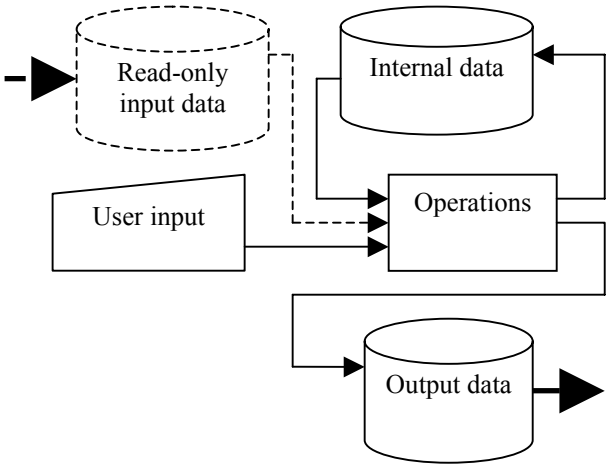

Figure 3. An acquisition component produces defined output data. It may use data produced by another acquisition component and enhance them.

The 'Internal data' block is used because it is important when building complex systems with bidirectional communications.

The thick arrows symbolize communication channel used between components. An acquisition component has one output channel and an optional input channel.

- The 'User input' block is an abstraction of a physical user who enters data manually or automatically using scanner or similar device.
- The 'Output data' block is a simple pipe that transforms internal data into a specified format that is recognized by other components in the system.
- The 'Internal data' block acts as a cache if the capturing operation cannot be performed in a fully-pipelined mode. Some components support saving the captured data in an internal format and continuing the operation later.
- The 'Operations' block accepts the data from the 'User input' block and performs various component-specific operations. If it is unable to produce the output data directly, the 'Internal data' block is used as a cache. The output data are produced only once.
- The optional 'Read-only input data' block allows cooperation between multiple acquisition components. An acquisition component can import initial data using this block.

5. PRESENTATION COMPONENT ARCHITECTURE

Figure 4 shows the basic architecture of a presentation component. The diagram contains basic blocks found in every presentation application. This diagram is partially similar to the Model-View-Controller schema – 'User input' is equivalent to Controller, 'User output' to View and the other blocks are the Model part.

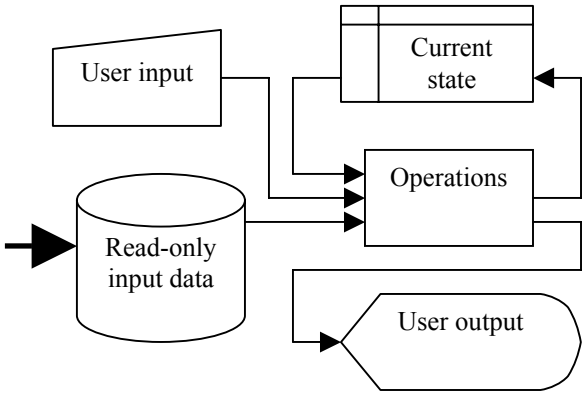

Figure 4. Presentation component delivers data to an external entity. The presented data are read using defined inter-component interface.

- The 'User input' block presents the input part of the UI of an interactive system. In almost each current system, a real GUI and user is involved.
- The 'Read-only input data' block contains data that are read-only from the viewpoint of the whole system. Usually, the amount of the data is large and some system-specific filter is used to quickly access the relevant subset of data.
- The 'Current state' block keeps the accumulated state of the system. Serialization of state data is performed in this block if it is needed.
- The 'User output' block presents the output part of the UI. It must adjust the data according to the output device.
- The 'Operations' block takes the actual input data, persistent input data and current state and produces new output data and new state data.

6. COMPONENT COMMUNICATION

Communication means can be classified according to two main criteria:

- Level of integration – at which level are the components communicating.
- Communication scenario – how are the data exchanged between components.

Communication levels

Naturally, some kind of interface is needed for the components to communicate.

6.1.1 Native source code interface

This kind of communication is the least available and least general. Two components must be implemented using same programming language and they must be running on the same platform. On the other hand, the data transfer can be efficient because there are no additional rules.

6.1.2 Platform-dependent binary interface

This communication level is more general than the previous one. The components must still run on the same platform, but they can be implemented in different programming languages. An example of platform-dependent binary interface is a custom interface based on COM or other binary standard.

6.1.3 Platform-independent interface

In this case the components may run on different platforms but they still may be integrated into one application. For example a platform independent protocol (HTTP) can be used to connect a video capturing component to a player.

6.1.4 Standardized file format

This kind of communication is the most spread one. This approach is very general and it is available in majority of current components (applications).

6.1.5 Manual data transfer

This is the most general approach that can be applied on every component pair. User must manually transfer every piece of data. The efficiency is low and the integration of components virtually impossible.

Communication scenarios

Let's analyze the cost of change in the data. If there are multiple acquisition components connected and if data are changed in the first component, it must go through all the other components and all the actions performed by each component must be re-done. One of the options, how to enable iterative acquisition of data, is to use shared communication medium. Figure 5 shows schema of such system.

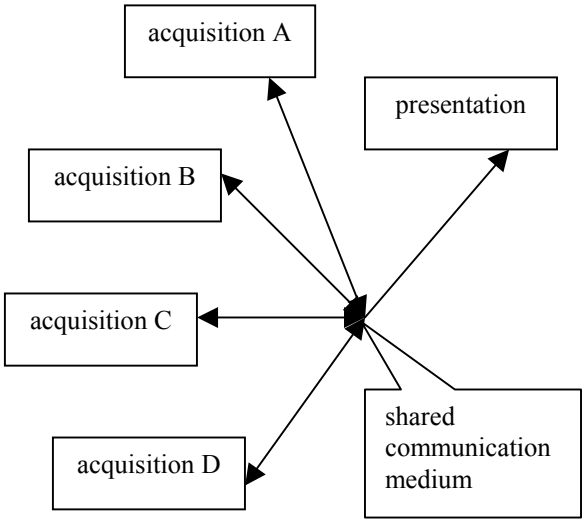

Figure 5. The acquisition components are able to cooperate using the shared communication medium. The components must not destroy each other's data.

The weakness of this system is obvious. Designing the generic interface is very hard but several systems build around the described architecture can be found (XML). A standardized file format is used as the communication medium. These file formats are based on two principles: 'ignore and preserve what you do not understand' and 'hierarchical layout'. Using these principles, multiple acquisition components can work on shared file without destroying each others data. Each component just adds its specific data bits to the final file. Although the result may not be always correct, it is usually satisfactory. This method can be enhanced by defining additional rules how to work with unknown data to keep them consistent.

6. SUMMARY

The shared communication medium scenario appears to have big potential, and will be the subject of future work. Once the set of cooperation rules is defined, it will become possible to merge and maintain data from multiple independently developed components with unrelated data models. This will allow parallel development of the components and as a consequence it will lower the development cost.

7. REFERENCES

[Ack96a] Ackerman, P. Developing Object-Oriented Multimedia Software - Based on the MET++ Application Framework. 1996

[Wag97a] Wagner., B. A Visual Programming Environment for Composing Interactive Performance Systems. ICMC'97, 1997.