A COM-based Toolkit for Real Time Visualization

Stefan Maas
Westfälische Hochschule Gelsenkirchen
Medical Engineering Laboratory
Neidenburger Straße 43
45877 Gelsenkirchen, Germany
stefan.maas@w-hs.de

Heinrich Martin Overhoff
Westfälische Hochschule Gelsenkirchen
Medical Engineering Laboratory
Neidenburger Straße 43
45877 Gelsenkirchen, Germany
heinrich-martin.overhoff@w-hs.de

ABSTRACT

Collaborative software development in different languages is not unusual, but leads to minor resource utilization during collaboration as a result of porting or reprogramming needs. Additionally in cooperative projects, frequently legal and market economic issues prohibit an exchange of source code between the project partners. Combining modules from different languages is possible using the Component Object Model (COM). Additionally COM offers an efficient way to combine modules from several development teams.

To solve the common issues of collaborative software development and to fulfil the needs of a real time visualization toolkit, "RTVCOM" was designed and realized. To demonstrate the capability of this approach an example client was developed that combines COM components written in OpenCL C, OpenGLSL, C++ and C#. It that processes 3D+t ultrasound data at 45.2 MB/s reconstructs the associated volume data and visualizes them in real time. The visualization is fully interactive, and different pre- and post-processing filters can be applied.

Keywords

Collaborative software development, COM, GPU, real time rendering

1. INTRODUCTION

Collaborative software development in different languages is not unusual. But this leads to minor resource utilization as a result of porting or reprogramming needs. Additionally in cooperative projects frequently legal and market economy issues prohibit an exchange of source code between the project partners. The Component Object Model (COM) [Mic15a] was designed to support development of software using different languages and offers an efficient way to combine modules from several development teams. COM enables inter-process communication and applications with loose coupling and high cohesion.

Using graphics processing units (GPUs) in real time visualization applications is state of the art. While GPU host programs are implemented in languages like OpenGLSL or OpenCL C, clients on central processing units (CPUs) are written in languages like C++ or C#.

To facilitate collaborative development of real time visualization software in cooperative projects the toolkit "RTVCOM" ("Real Time Visualization using the Component Object Model") was designed and realized. RTVCOM allows the implementation of clients in different CPU- and GPU-languages to gain a great variety of application possibilities.

To evaluate the capabilities of RTVCOM an example client was created that processes 3D+t ultrasound data at 45.2 MB/s in real time. Volume rendering including pre and post-processing filtering contribute to a complex functionality of this client.

2. RECENT SOLUTIONS

The open source "Visualization Toolkit" (VTK) [Kit15a] and the "Insight Segmentation and Registration Toolkit" (ITK) [Kit15b] are widely spread in the medical visualization community. Other toolkits like the "Medical Interaction Toolkit" (MITK) [Ger15a] and frameworks like "MeVisLab" [Mev15a] are built upon VTK and ITK or integrate them. The advantage of these toolkits and frameworks is the immense range of functions and the acceptance in the community. But these toolkits are not fully realized for GPU execution and hence the algorithms are too slow for 3D+t volume visualization in real time.

[Gob08a] created a pure GPU ray casting framework for massive volumetric datasets. The algorithms of that framework would be fast enough in principle. But it is only laid out for handling static volumes.

Some GPU frameworks as presented in [Sch11a], [Mem11a] or [Chu10a] are dealing with medical image segmentation or reconstruction, but not with continuous volume data.

Therefore, all of these toolkits and frameworks are incapable of visualizing volume data at rates mentioned above.

Simulator X [Lat12] is an example for software that is based on the actor model. It is fast enough to allow real time visualization with low coupling and high cohesion. But as the toolkits/frameworks above the actor model is not laid out to fulfil the needs of collaborative software development in different languages. Also the mentioned legal and market issues in cooperative projects are not regarded.

3. PROPOSED SOLUTIONS

RTVCOM was designed and realized to solve the issues mentioned in the previous section. While RTVCOM was primary implemented for visualizing medical image data it was laid out to include all kind of real time data in principle. RTVCOM was developed as a modular system consisting of COM components as “In-process-Servers” with defined interfaces. Clients are not part of the toolkit and can be implemented arbitrary.

3.1 Component Categories

RTVCOM includes components from six categories (Table 1), which were implemented by the authors and three different project partners in two collaborative projects:

- Reader

The category “Reader” contains interfaces to (raw) data providers like ultrasound devices or other devices with continuous 2D+t or 3D+t raw or image data streaming.

For prior recorded data support this category also contains readers for different file formats like DICOM or Insight Meta Image (.mha, .mhd). Components in this category are usually written in CPU languages.

- Preprocessor

COM-components for raw data processing or for processing image data before visualization belong to the category “Preprocessor”. This category consists of two subcategories:

  - “GPU”

  Real time processing filters can be found in subcategory “GPU”. These filters are written in OpenCL C or OpenGLSL and are used before the visualization process. For example the conversion of raw data to image data can be found in this category.

  - “CPU”

  During development time it can be reasonable to write filters using CPU languages first. For example to examine the quality of a serial filter in C++ before parallelizing it using OpenCL C. These filters belong to subcategory “CPU”.

- Viewer

“Viewer” contains all graphical COM-components that were built to visualize data. This category is divided in two subcategories: “2D(+t)” and “3D(+t)”. Currently all viewer-components use OpenGLSL vertex and fragment shaders.

- Postprocessor

It is often useful to enhance visualizations with post processing algorithms (“filters”). This category is divided in the same subcategories that are used in “Preprocessors” for the same reasons.

- Helper

This category contains all components that support the visualization indirectly like e.g. a GUI element for manipulating the opacity transfer function of volume visualizations. (The opacity transfer function itself is part of “Viewer” components.) Components in this category are usually written in CPU languages.

- Import/Export

Import/Export components are used to include functionality from third party products like MATLAB. The “MLApp”-COM server allows external applications to use MATLAB-functions. Since MATLAB is to slow for real time processing, this component is only used during the first phase of filter development. In the second phase promising filters are usually ported to OpenCL C or OpenGLSL.

3.2 Interfaces

The interfaces of the components are standardized. This simplifies the data exchange between the components on the client side. It is possible to convert data from OpenCL C to OpenGL shaders and vice versa without leaving the host. To minimize memory consumption and to enable fast data transfer, generally memory addresses or pointers are exchanged over the interfaces.

Category	Reader	Preprocessor	Viewer	Postprocessor	Helper	Import/Export
CPU	X	X		X	X	X
GPU		X	X	X		

Table 1. Overview of RTVCOM categories

3.3 Sharing Components

Sharing RTVCOM components is currently realized by a Microsoft Team Foundation Server 2010. This server is part of a Microsoft SharePoint Server 2010. These servers are the base of the quality management system used for developing RTVCOM.

Depending on the settings of an integrated rights management system the developed components can be shared, for example, with project partners without revealing internal knowledge. Also jointly created clients are possible, where each partner can use the COM components of the other partners. Source code access is only possible for developers with rights in accordance to the rights management system.

3.4 Client Creation

Clients can be built by assembling components from the given categories. While the usage of "Reader"- and "Viewer" components is mandatory, components from other categories can be used optionally in type and numbers.

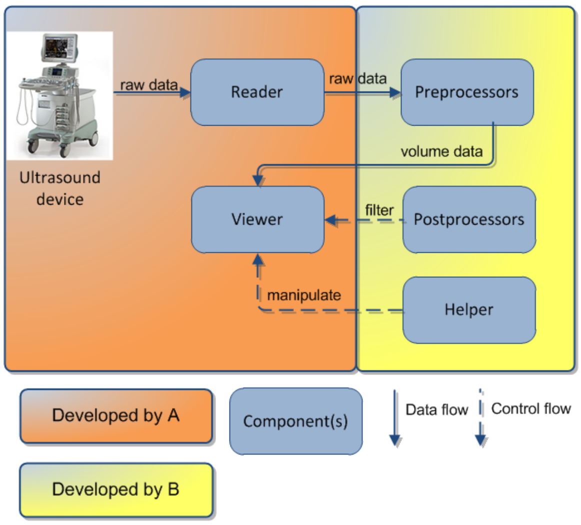

Figure 1. Example for a jointly created client regarding developers

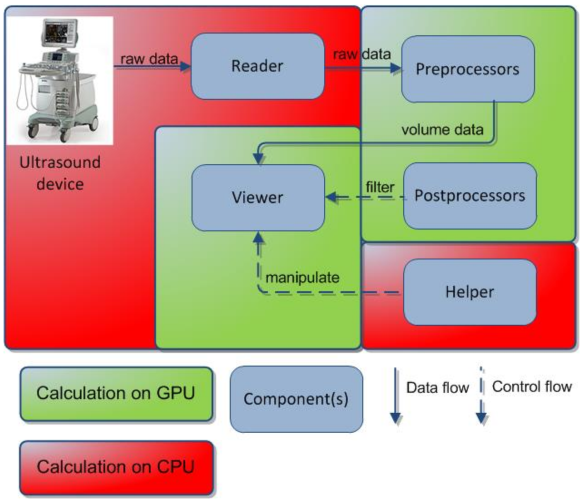

Figure 2. Example for a jointly created client regarding processing location

A client, that was created by the authors and one industrial partner during a jointly project, is shown in Figure 1. In this client the industry partner ("A") provides an ultrasound device including software components of category "Reader" and "Viewer". This software allows only a conventional view on ultrasound images. The authors "B" integrated components from category "Preprocessor", "Postprocessor" and "Helper". During development "B" is able to use the components from "A" but cannot access their source code due to missing access rights. After the project ends "B" can provide "A" with the developed components without revealing any knowledge.

A different view on this example client (see Figure 2) illustrates that the use of RTVCOM enables both partners to create GPU and CPU components and merge them in a joint client.

4. EXPERIMENTAL RESULTS

To demonstrate the capability of RTVCOM an example client was developed that combines COM components written in OpenCL C, OpenGLSL, C++ and C#. It acquires raw data streams at 45.2 MB/s and 50 Hz from an ultrasound device, reconstructs the associated volume data and visualizes the data in real time.

The used hardware consists of an Intel Core2Duo, 2.6 MHz CPU, PC with an NVIDIA Geforce 760 GTX. On this system the visualization is fully interactive, and different post processing filters can be applied without losing the real time capability.

4.1 Client Details

The data acquisition, data flow and visualization process of this C#-based client (see also Figure 1) in detail:

1. The first component ("Reader"; written in C++) starts the data acquisition from the ultrasound device. Raw data (short values, 8 bit) are transferred at 50 Hz as 2D+t slices from the ultrasound device via a network card to the visualization computer.
2. Within the second component ("Preprocessor"; written in C++ and OpenCL C) the short values are uploaded to the GPU. Due to a better float value support of GPUs the short values are converted to float values in the first OpenCL kernel. In the next step the volume data are recalculated from the raw data using four different OpenCL kernels.

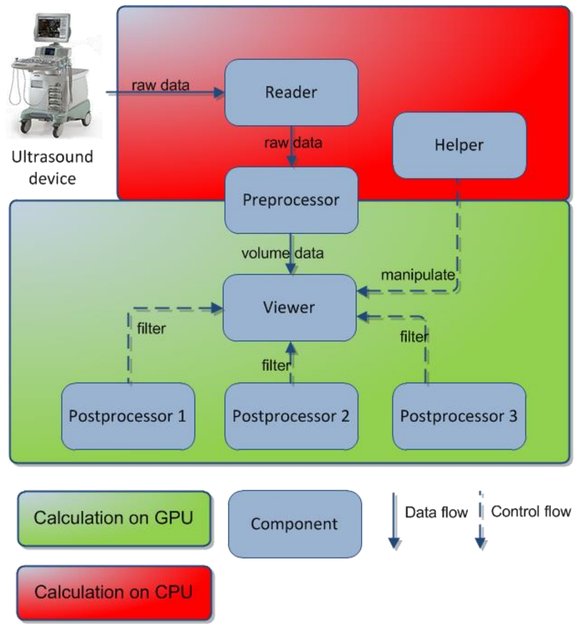

Figure 1. Schematic overview of example client

3. The next component ("Viewer"; written in C# and OpenGLSL vertex and fragment shaders) visualize the reconstructed volume using a raycasting technique. This 3D+t volume is updated every incoming slice resulting in an update rate of 50 Hz.
4. Three filter components ("Postprocessor"; written in C# and OpenGLSL compute shaders) were added for enhancing image quality. These filters can be activated/deactivated interactively.
5. Additionally the visualization of the volume can be optimized by manipulating the opacity and grayscale transfer functions using another component ("Helper"; written in C#).

4.2 Comparison

Before the example client was implemented, the visualization was realized by a single main program. This data and control flow was exactly like in the example client but without any COM components. Because COM is said to be slow regarding data transfer, a performance comparison between both implementations were implemented. It revealed no measurable difference. Both implementations were able to visualize the 3D+t data at 45.2 MB/s.

5. CONCLUSION

It has been demonstrated that RTVCOM and RTVCOM-based clients can solve the common issues of collaborative software development and to fulfil the needs of a real time visualization toolkit:

- Support of collaborative software development in different CPU and GPU languages
- Enabling development of jointly created clients in cooperative projects regarding legal and market economy issues
- Visualization in real time

The advantage as well as disadvantage is the fact that RTVCOM is not laid out to reveal source code. So this approach cannot directly be used for open source projects.

Future works will extend RTVCOM to support the Distributed Component Object Model (DCOM). This will simplify the exchange and replacement of COM objects and make the Team Foundation Server obsolete for this purpose. Additionally the number of filters will be enlarged to increase visualization quality for specific applications. Furthermore readers for other types of data like real time image data from magnetic resonance tomography (MRT) will be implemented.

6. REFERENCES

[Chu10a] Chunlan, X., and Anyuan Z., and Liu, D.C. Optimized GPU Framework for Ultrasound B-Mode Imaging. Bioinformatics and Biomedical Engineering (iCBBE) 2010, pp. 1-4, 2010.

[Ger15a] German Cancer Research Center. MITK. http://mitk.org/wiki/MITK.

[Gob08a] Gobbetti, E., and Marton F., and Iglesias Guitián, J.A. A single-pass GPU ray casting framework. The Visual Computer, pp. 797-806, 2008.

[Kit15a] Kitware, Inc. VTK. http://www.vtk.org.

[Kit15b] Kitware, Inc. ITK. http://www.itk.org.

[Lat12a] Latoschik, M.E. and Tramberend, H. A Scala-Based Actor-Entity Architecture for Intelligent Interactive Simulations. Software Engineering and Architectures for Realtime Inteactive System (SEARIS) 2012. 5th Workshop on, pp. 9-17, 2012.

[Mem11a] Membarth, R., and Hannig F., and Teich J., and Körner M., and Eckert W. Frameworks for GPU Accelerators: A Comprehensive Evaluation using 2D/3D Image Registration. 2011 IEEE 9th Symposium on Application Specific Processors (SASP), pp. 78-81, 2011.

[Mev15a] MeVis Medical Solutions AG. MeVisLab. http://www.mevislab.de.

[Mic15a] Microsoft Corporation. Microsoft COM. https://www.microsoft.com/com/default.msp.

[Sch11a] Schmid, J and Iglesias Guitián, J.A., and Gobbetti, E., and Magnenat-Thalmann, N. A GPU framework for parallel segmentation of volumetric images using discrete deformable models, The Visual Computer, pp 85-95, 2011.