Inviwo - An Extensible, Multi-Purpose Visualization Framework

Erik Sundén*, Peter Steneteg†, Sathish Kottravel‡, Daniel Jönsson§, Rickard Englund¶, Martin Falk‖, Timo Ropinski**

ABSTRACT

To enable visualization research impacting other scientific domains, the availability of easy-to-use visualization frameworks is essential. Nevertheless, an easy-to-use system also has to be adapted to the capabilities of modern hardware architectures, as only this allows for realizing interactive visualizations. With this trade-off in mind, we have designed and realized the cross-platform Inviwo *(Interactive Visualization Workshop)* visualization framework, that supports both interactive visualization research as well as efficient visualization application development and deployment. In this poster we give an overview of the architecture behind Inviwo, and show how its design enables us and other researchers to realize their visualization ideas efficiently. Inviwo consists of a modern and lightweight, graphics independent core, which is extended by optional modules that encapsulate visualization algorithms, well-known utility libraries and commonly used parallel-processing APIs (such as OpenGL and OpenCL). The core enables a simplistic structure for creating bridges between the different modules regarding data transfer across architecture and devices with an easy-to-use screen graph and minimalistic programming. Making the base structures in a modern way while providing intuitive methods of extending the functionality and creating modules based on other modules, we hope that Inviwo can help the visualization community to perform research through a rapid-prototyping design and GUI, while at the same time allowing users to take advantage of the results implemented in the system in any way they desire later on. Inviwo is publicly available at *www.inviwo.org*, and can be used freely by anyone under a permissive free software license (Simplified BSD).

Keywords: Inviwo, Interactive Visualization Workshop, Rapid Prototyping, Open Source Software

1 INTRODUCTION

Visualization algorithms and system are often based on the visualization pipeline [7], i.e. preparing, filtering, mapping and rendering data. Most advanced visualization algorithms must therefore be involved in the discovery process from the start. From acquiring the data to displaying it on the screen [5], which requires that the algorithms are tightly integrated into the domain scientists work flow [10]. However, from the perspective of a domain scientist the state-of-the-art in visualization algorithms has mostly been limited to showing their data through 2D graphs of various kinds, or preconfigured 3D representations with limited interaction and filtering possibilities. Even though many tools support more advanced 3D representations, the user is usually not able to configure and modify these representations freely. The fact that the user is not able to explore the visualization design space freely severely hinders the potential use of visualization and thereby scientific discovery. Furthermore, it is usually not enough to visualize the data using a single representation or algorithm. To fully understand the data several visualization algorithms need to be integrated and linked such that changes are reflected in the different views.

*Linköping University, e-mail:erik.sunden@liu.se
†Linköping University, e-mail:peter.steneteg@liu.se
‡Linköping University, e-mail:sathish.kottravel@liu.se
§Linköping University, e-mail:daniel.jonsson@liu.se
¶Linköping University, e-mail:rickard.englund@liu.se
‖Linköping University, e-mail:martin.falk@liu.se
**Ulm University, e-mail:timo.ropinski@uni-ulm.de

2 GENERAL VISUALIZATION FRAMEWORKS

A relatively large selection of general purpose visualization frameworks is available, whereby all have their individual pros and cons. One of the first general purpose frameworks for scientific visualization is VTK (The Visualization Toolkit) [9], which is a visualization framework written in C++, freely available under a BSD license. VTK itself cannot be used without knowledge of C++ and requires applications, such as ParaView, to be used by non-programming experts. Voreen (Volume rendering engine) [6] is a rapid application development framework for visualization for volumetric data sets. Voreen is written as a C++ library and is cross platform (Windows/Linux, with minimal Mac support). Similar as many major visualization frameworks, Voreen's user interface consist of a data flow network. VisTrails [1], which uses a similar data flow network as its interface, is still distinctive from other frameworks, as it maintains a detailed history of all the changes made by the user. Unfortunately, the connections in the dataflow network need to be performed in a relatively fine grained manner, which makes the framework more difficult and tedious to use. Another data flow application targeted for medical visualization is MevisLab [8], available under a commercial license. In theoretical chemistry VMD (Visual Molecular Dynamics) [3] is an application for displaying large biomolecular systems using 3D visualization. It accepts a large selection of biology related file formats, and supports many ways or rendering. The application Visit [2] was build on top of the VTK toolkit, designed for parallel visualization and graphics analysis, with the main aim of large scale visualization, as it is built around a client and server solution.

While the visualization systems above boasts many features, they can be difficult to use either from a programming or end-user perspective, and their basis often do not utilize the low-level modern architecture to achieve high performance processing with low overhead. Our design of Inviwo has been done with the goal to compensate these shortcomings.

3 THE INVIWO FRAMEWORK

Inviwo is a framework for rapid visualization prototyping, written in C++, it exploits modern graphics hardware, and is freely available under the permissive Simplified BSD license. The strategies behind the design of the Inviwo framework are illustrated by Figure 1. The goal is to create a framework that can be used without any prior visualization knowledge, but still is powerful enough to be useful for a visualization scientist. By having a framework that both novice and expert users can use, we can effectively shorten the time it takes for advanced visualization techniques to reach a broader user base outside of the visualization field.

On a more technical level, Inviwo is designed in such a way that all specific platform APIs are excluded from the core, enabling the system itself to be used on any system (with C++11 support), even without a display system. Thus, all core concepts have C++ back-ends, and implementation of these concepts for APIs such as OpenGL and OpenCL is provided in optional modules. This graphics independent design makes Inviwo more suitable for cloud-based visualization as well as future API changes. Thus, the inclusion of other recent low-level frameworks, such as Metal (Apple) or Mantle (AMD), can be performed in a straight-forward way, while providing the option to use none, one or multiple interfaces to these APIs.

The application interface of Inviwo enables the user to design data flow networks, similar to others (see Figure 2). In Inviwo we represent the nodes in the network as *processors* which has a set of input and output ports that can be connected. Data flows from top to bottom following the connections in the network. Additionally each processor has a set of properties that defines the state of the processor. When properties change, affected parts are re-evaluated automatically.

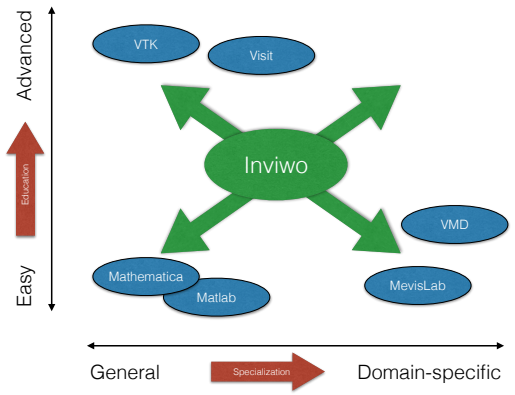

Figure 1: The goal of Inviwo is to be useful for a wide range of problems and users. It should be easy enough for a novice user to use it without instructions, yet still allowing an experienced visualization scientist full advantage of the system. It is designed with generality in mind, but can easily be extended with domain specific functionality.

3.1 Seamless data flow between architectures

To be able to bridge all available graphics hardware architectures, we have designed and implemented a data flow network which seamlessly update data on various architectures on-demand. Thus, the developer may utilize the incoming data for either CPU processing, explicit graphics pipeline (OpenGL) or a heterogeneous multipurpose platform (OpenCL), regardless of which architecture the valid data resides on. This is enabled through data converters which, if available, utilize already allocated memory to, for instance, shuffle data to and from the GPU on-demand, while keeping track on which architecture the latest valid data resides on. Thus, we are able to cache intermediate results across graphics architectures.

4 Application Cases

To demonstrate the flexibility of Inviwo, we briefly discuss two application cases realized through Inviwo in the past. Further application cases are available on (@ www.inviwo.org) within the sample workspaces deployed with the current version of the software.

4.1 Molecular Visualization

One of the key challenges in molecular visualization is enabling a better spatial comprehension in an crowded environment while preserving real time interaction. The Inviwo framework was used to develop a novel Depth of Field (DoF) technique that improves spatial comprehension [4]. This DoF technique exploits coverage based opacity estimation to achieve rendering quality comparable with multi-sampling DoF technique at interactive frame rate. With the aid of the rapid protoyping nature of Inviwo, we were not only able to develop new algorithms, but also to easily integrate existing techniques such as ambient occlusion, screen based DoF and order independent transparency. Furthermore, the scripting features in Inviwo enabled the evaluation of our algorithms. For example, python scripts can be used to drive the application (by extending/embedding Python API) and perform evaluation tasks such as measuring frame rate by changing camera view or comparing differences in images.

4.2 Data Presentation

It can be highly beneficial to utilize interactive visualization for educating the public about scientific data. Such applications should be engaging and intuitive for both novice and expert users, which often require highly interactive solutions, making the performance of the visualization pipeline critical. Inviwo has been utilized in numerous interactive application, in a public setting, for example show-casing brain activity based on data from the Human Connectome Project (see Figure 3).

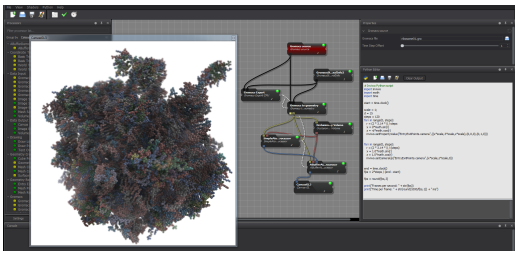

Figure 2: Molecular visualization functionality implemented inside an Inviwo module. Various operations are performed inside processors and the result from each processor flows between connected ports in an top-down topology structure.

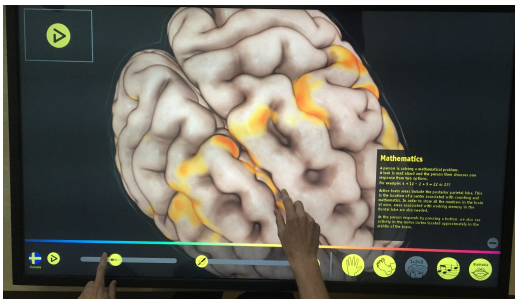

Figure 3: Application designed for public exhibition, utilizing Inviwo for as the visualization pipeline. The application allows users to explore the brain structure (MRI) and learn about brain activity (fMRI).

5 Conclusions

This poster just describes a subset of the possibilities of Inviwo. We hope that, in the case of acceptance, the discussions around the poster stand will enable us to better inform other researchers about Inviwo, in order to support them with their research.

References

[1] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. Vistrails: visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 745–747. ACM, 2006.

[2] H. Childs, E. Brugger, K. Bonnell, J. Meredith, M. Miller, B. Whitlock, and N. Max. A contract based system for large data visualization. In *Visualization, 2005. VIS 05. IEEE*, pages 191–198. IEEE, 2005.

[3] W. Humphrey, A. Dalke, and K. Schulten. Vmd: visual molecular dynamics. *Journal of molecular graphics*, 14(1):33–38, 1996.

[4] S. Kottravel, M. Falk, E. Sundén, and T. Ropinski. Coverage-Based Opacity Estimation for Interactive Depth of Field in Molecular Visualization. In *Proceedings of the 2015 IEEE Pacific Visualization Symposium*, pages 255–262, 2015. to appear.

[5] K.-L. Ma, I. Liao, J. Frazier, H. Hauser, and H.-N. Kostis. Scientific storytelling using visualization. *Computer Graphics and Applications, IEEE*, 32(1):12–19, 2012.

[6] J. Meyer-Spradow, T. Ropinski, J. Mensmann, and K. Hinrichs. Voreen: A rapid-prototyping environment for ray-casting-based volume visualizations. *Computer Graphics and Applications, IEEE*, 29(6):6–13, 2009.

[7] K. Moreland. A survey of visualization pipelines. *Visualization and Computer Graphics, IEEE Transactions on*, 19(3):367–378, 2013.

[8] J. Rexilius, J.-M. Kuhnigk, H. K. Hahn, and H.-O. Peitgen. An application framework for rapid prototyping of clinically applicable software assistants. *GI Jahrestagung (1)*, 93:522–528, 2006.

[9] W. J. Schroeder, K. M. Martin, and W. E. Lorensen. The design and implementation of an object-oriented toolkit for 3d graphics and visualization. In *Proceedings of the 7th conference on Visualization'96*, pages 93–ff. IEEE Computer Society Press, 1996.

[10] C. T. Silva and J. Freire. Software infrastructure for exploratory visualization and data analysis: past, present, and future. *Journal of Physics: Conference Series*, 125(1):012100, 2008.