Interactive Software Prototyping for VR Applications

Gerwin de Haan Michal Koutek Frits H. Post

Computer Graphics Group, Delft University of Technology
{g.dehaan, m.koutek, f.h.post}@tudelft.nl

Virtual Reality Application Development

The development of domain-specific Virtual Reality applications is often a slow and laborious process. The integration of domain-specific functionality in an interactive Virtual Environment requires close collaboration between domain expert and VR developer, as well as of domain-specific data and software integration in a VR application. The software environment needs to support the entire software life cycle, from the early stages of iterative, rapid prototyping to a final end-user application. In our paper, we propose the use of flexible abstraction layers that, combined with an interactive development environment, support the VR application development process.

Development Approach

We propose flexible abstraction layers through a single abstraction language. In our approach, the abstraction layers are code fragments that abstract, combine and wrap lower level code. We want to facilitate:

(1) continuous, iterative software development, including features such as rapid prototyping, profiling and debugging
(2) flexible integration and configuration of heterogeneous VR and external software
(3) seamless evolution from early software prototypes to flexible end-user applications and
(4) ease-of-use, lowering the learning curve and empowering end-users.

Software Architecture

A schematic software overview of our Interactive Virtual Reality (iVR) system is shown in Figure 1. The set of components and domain-specific applications, combined with the Python scripting language layer, now form the basis for creating a VR application. The prototype is mainly intended for experienced VR developers, but some high level features geared towards end-users are also demonstrated. Furthermore, new application-specific high level facilities can quickly be developed based on the new abstraction layers.

Development Environment

The iVR functionality is directly available in the running Python interpreter after importing its wrapping modules. We use IPython, an enhanced interactive interpreter to enhance the usability and interactivity of this process. IPython provides many extra development features, including object and code inspection, command history, integrated debugging and saving of interactive prototyping sessions, see Figure 2 (left). We also use an experimental environment for the construction of Python-based VR code using the Notebook metaphor, see Figure 2 (right).

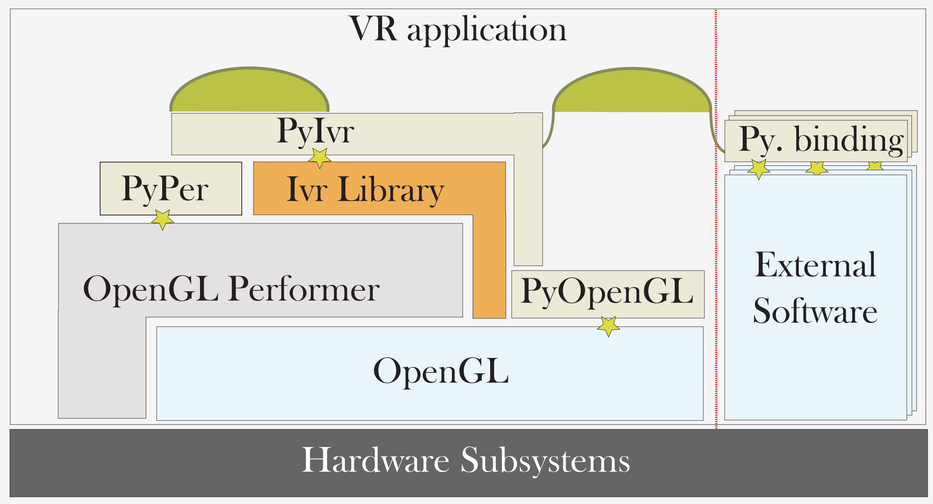

Figure 1: iVR software layers. The VR application has native access to various C++ components and external software or through Python. Stars indicate Python bindings on the underlying libraries. Arches indicate custom Python layers which provide higher level abstractions.

Building iVR applications

One can directly construct his application by using and extending a set of standard widgets, graphical objects and interaction techniques. The close integration of Python and the original C++ code allows us to, gradually, transform existing code toward Python oriented programming methods. We currently extend internal functionality for rapid VR development and integrate several external software libraries, see Figure 3.

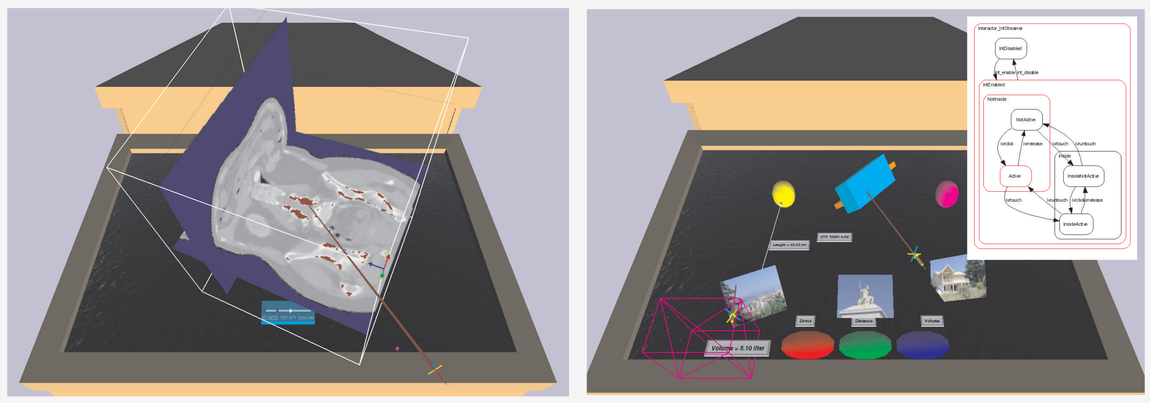

Figure 3: Demonstration of bi-directional integration of VTK in the VR environment (left) and interaction prototyping with graphing support (right). The Python glue facilitates expressive commands that combine VR and external library statements.

Conclusions and Future work

The current approach improves catalyzes the learning, re-use and re-design of VR software mechanisms and changes development philosophy, see also Figure 4. We envision an integrated development and run-time environment providing interactive control using higher level, visual programming and debugging tools. We expect the abstraction layering and integration of external tools to be key aspects in achieving this goal.

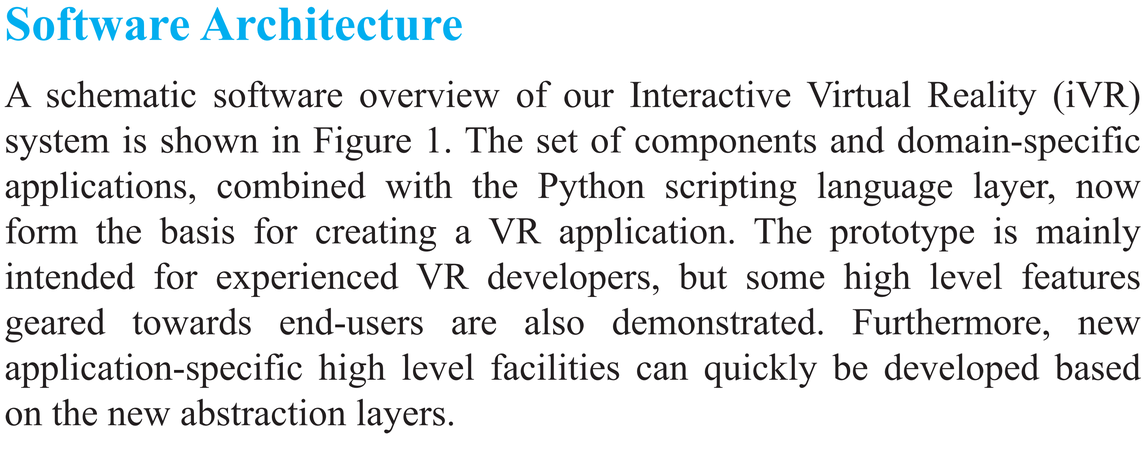

Figure 2: The interactive Python shell (left) and the notebook editor (right). The interactive shell can be used while the VR application is running. The notebook or worksheet (right) shows code editing, loading and saving operations, integrated graphics, and available documentation and command completed parameters.

Figure 4: Interactive application prototyping. During a VR workshop for PhD students, our prototyping approach was used for interactive demonstrations (left) and hands-on work on various VR systems (right).

Part of this research has been funded by the Dutch BSIK/BRICKS project.
For more information of our work visit http://visualization.tudelft.nl

BRICKS TU Delft
Delft University of Technology