vjVTK

Kristopher J. Blom
University of Hamburg, Germany
interactive media/interactive environments[2]

blom@informatik.uni-hamburg.de

vjVTK

Figure 1. vjVTK

vjVTK is, simply put, the merger of VTK[3] and VRJuggler[4]. More specifically it brings VTK's native use of OpenGL into the context VRJuggler's VR hardware control. VTK's visualization power directly rendered with VRJuggler's flexibility for devices and displays.

Table of Contents

Project History

The vjVTK project has both a long and a short history. While the code that makes up vjVTK was developed by me in the last months (mid 2005), as a member of the interactive media/virtual environments group[5], the idea of this project reaches back a number of years. The idea came from working with vtkCAVE by the Future Labs at Argonne National Laboratories.[6] Going from there to ISU and VRAC[7], where VR-Juggler was under development, I thought about doing this project based on the vtkCAVE code. That was the summer of 1999. Well, it never did happen and I lost the original code from ANL. One of the reasons for that was most people could get by with vtkActorToPF[8], a great project by Paul Rajlich. Finally I found a few afternoons to sit down and make this work in 2005. vjVTK is born, with a first release in September, 2005.

A short paper over vjVTK's design was published at EGVE '06. The paper is availabe on the im/ve publications website[9].

ChangeLog

Changes from Version 0.7 to 0.8

- moved to VTK 5.0 stable release. VTK versions prior to 5.0 are no longer supported.
- Windows Support added,
- Initial development release of Interactor/Widget code for vjVTK. This should be considered experimental
- Various code clean-ups

Changes from Version 0.6 to 0.7

- Ray picking in VTK
- Integration of VTK with OSG and OpenSG including examples
- Compatibility with Interim VTK release 4.4, and 5.0 series
- Fix of a number of limiting bugs with display

Contributions

Contributions - 0.8 relase

Francois Rioux

Windows build system contributions

Downloads

A tarball of the source code for the current version of vjVTK can be downloaded here.[10] The current version of vjVTK is *0.8.0*.

This text you are reading, for what it is worth, can also be downloaded here in pdf[11] form.

Requirements and Installation

Requirements

vjVTK is currently availabel for Windows and Linux systems. Availability on Mac platform is not yet provided, as I do not have access any development machines currently. Examples provided may place further requirements, such as the vjVTK_logo example requiring the Hybrid module in VTK.

- VRJuggler 2.0
- VTK 5.0 or newer

Optional requirements

- OpenSG
- OpenSceneGraph

Build and Installation

The vjVTK build and installation systems are relatively primitive; however, quite functional. The build processes are derived from the VR Juggler build system and as such should be familar VR Juggler application writeres. Under linux this entails a traditional make based system, and under Windows there are VC project files for 2003 available. vjVTK installation is, at the moment, left largely up to the user. The build system is responsible only for the creation of the library and examples. After building the system, installation may be desired. For this there are a number of options. The two most common are discussed here, installation into Juggler and as a separate installation.

Building vjVTK - Linux

vjVTK's build system is a bit primitive at the moment. Steps to building vjVTK

1. untar the tarball to an appropriate directory
2. Make certain your VJ_BASE_DIR environment variable is set
3. Either set the VTK_DIR env variable to the base of the VTK installation or use the default of /usr/local/ if appropriate
4. Run the *make* command

The examples and test programs can be compiled by executing the *make test* and *make example* commands respectively. The examples can also be individually compiled if desired.

> **Note:** if copying the examples to another directory there will be a path problem with the make as there is an assumed relative path

Building vjVTK - Windows

vjVTK's build system is a bit primitive at the moment. Steps to building vjVTK

1. unzip the tarball to an appropriate directory
2. Environment variables for Juggler dependencies must be set, as explained in the VRJuggler guides. VJ_BASE_DIR must be set and the libraries should be found in the user's path for run-time loading. If you are using OpenSceneGraph or OpenSG the appropriate PATH and environment variables.
3. Load the vcproj for vjVTK into Visual Studio.
4. Set the location of the VTK and VRJuggler libraries in the project configuration.
5. Build the project. This will build a Debug build of vjVTK in the Debug directory directly under the main vjVTK directory.

The examples each have thier own project file and can be configured after the vjVTK library is compiled. As with the main library, the correct settings in the configuration of the project for VTK, VR Juggler, and vjVTK must be given. Additionally, vjVTK.dll has to be found in the user's path, just as any used library must.

Building Optional Components

Optional components of vjVTK include the ability to combine VTK and a SceneGraph together. Actually you don't need to perform any builds to use this feature, as it is defined through one of the App files, VTK_OSGApp.h or VTK_OpenSGApp.h.

The examples which come with require to be manually compiled, that is they are not automatically built. In most cases, where Juggler is set up with OSG/OpenSG, simply running make should be enough. The build is based on the Juggler example build process for examples of OSG/OpenSG. If this does not work, please check that the proper environment variables are set correctly: OSGHOME, OPENSGHOME, PATH, and LD_LIBRARY_PATH.

Installation in Juggler

For installation in your VRJuggler installation the requirements are dependant on how you wish the users to access vjVTK. Two versions are highlighted here. The first version is the simple method, but likely the least appeasing to VRJuggler purists. The second version makes the vjVTK portion appear to be part of VRJuggler, some minor modifications must be made to the installed headers and all of the examples and documentation will be slightly flawed due to path discrepancies.

> **Note:** Below is written for Linux based systems; however, the only critical difference for Windows systems is that library has the ending .dll and the .lib is also required.

For the simple version: From the build directory of vjVTK the following commands should suffice for a user with permissions to write to that directory. Copy the lib-vjVTK.so into ${VJ_BASE_DIR}/lib/ The second portion is to copy the header file information to the appropriate place, cp *.h ${VJ_BASE_DIR}/include. Everything should be relatively straight forward now when using a makefile that is "VRJuggler aware."

For purists: Installation of vjVTK to make it look like it is part of VRJuggler is a bit more complicate, but only in the aspect of your users. vjVTK users require only two parts installed, the library and the header files. The library, `libvjVTK.so` should

be installed in `${VJ_BASE_DIR}/lib/` The header files should be placed in a newly created directory, `${VJ_BASE_DIR}/instlinks/include/vrj/Draw/VTK/`.

With this installation the examples and documentation will no longer be totally correct. The problem lies in the path to the includes, as the code, examples, and documentation assume the vjVTK is installed as a single library. Most easily changed is that the includes must be expanded to "#include <vrj/Draw/VTK/*.h>"

vjVTK Installation as a single library

The simplest version of this is, build the project and it is ready. Relocating the build is not a problem. For the "bare bones" installation, the users need only the `libvjVTK.so` library and header files. Everything else can be left out of the installation. Regardless of where it is installed, the users require the usual information of where the headers are for their includes on the compile line and also to set their Library path.

Usage

This section discusses the usage of vjVTK. The guide is set dependant on your background, as novice to VTK and Juggler, VTK user, or Juggler User.

For Novice Users

The area of novice to VTK and Juggler could included a number of possibilities. However, in the most cases I would suggest taking at look at VTK itself and then starting out from there. Since VTK has such a well developed set of interfaces and examples, it is an easier place to start. After learning out to set up your VTK pipeline, integrating it into Juggler using vjVTK will fairly easy. The examples included here are either directly showing a VTK example in using Juggler/vjVTK or an adaptation of one. The one thing to note is vjVTK and Juggler are always going to require C++ programming. VTK however, has a selection of language bindings. Luckily bindin gs for each language mean the syntax is different, but the basics of the commands are the same. If you are familiar with tcl or python learning VTK with that language may be easier for you. Converting to the C++ version is mostly syntax changes, as can be seen in the navgrab example, which is based on a tcl VTK example.

Please refer to the Section called *the example programs* for help on running the provided examples.

For VTK Users

For those users already familiar with VTK adaption to VRJuggler shouldn't be to difficult, given an assumption, that the juggler configuration files for your display system are functioning and ready for you. Documentation for that portion is in the VRJuggler documentation. vjVTK programming is always in C++. Luckily for those of you who have used one of VTK's other supported languages the interfaces are very similar. Even easier you don't have to worry about the RenderWindow and Renderer anymore, vjVTK and VR Juggler take care of it.

The best starting point is likely to simply start with one of the examples. In a function called initScene you will find the VTK code. Simply placing your code there instead will load your scene. However, here is the biggest caveat, you need to do one little thing different. vjVTK has it's own Renderer, vjVTKRenderer. Any vtkActors etc. must be added/removed from it.

VRJuggler works be defining an "application class" which is run. Applications you write will be derived classes from the VTKApp class. This class performs the pieces required for joining VTK and VRJuggler. In most cases your derived class will require only a initScene and a preFrame function. The initScene function is the place for performing your initial VTK pipeline setup, in my experience the largest chunk of code is here. The preFrame function is used in Juggler for dynamic things in the scene, i.e. anything that changes over time. Interaction with the visualization goes here.

Please refer to the Section called *the example programs* for help on running the provided examples.

For VRJuggler users

For Juggler users the Juggler side of things will be quite familiar. The class VTKApp is the parent class from your application class. Differences are slight in the setup of your scene and interaction. Navigation is best explained by taking a look at the provided example, shortly said simply give the normal Juggler transformation matrix over to vjVTK and it takes are of everything. VTK has it's own engine for the graphics, which is best understood by looking at the documentation provided by them. Further explanation of what you need to know about how vjVTK works with VTK follows.

VTK works using a pipeline. The built VTK pipeline creates the visualization ending in OpenGL geometry. The Geometry is encapsulated in an object called an vtkActor or a vtkProp VTK normally handles the display system itself, consisting of Renderer's and RenderWindow's. This aspect is handled by vjVTK. As a user of vjVTK you simply use the vjVTKRenderer which is a part of the VTKApp class using the accessor function 'getRenderer()'. Lighting objects, vtkLight, can be added to the renderer also to control the lighting.

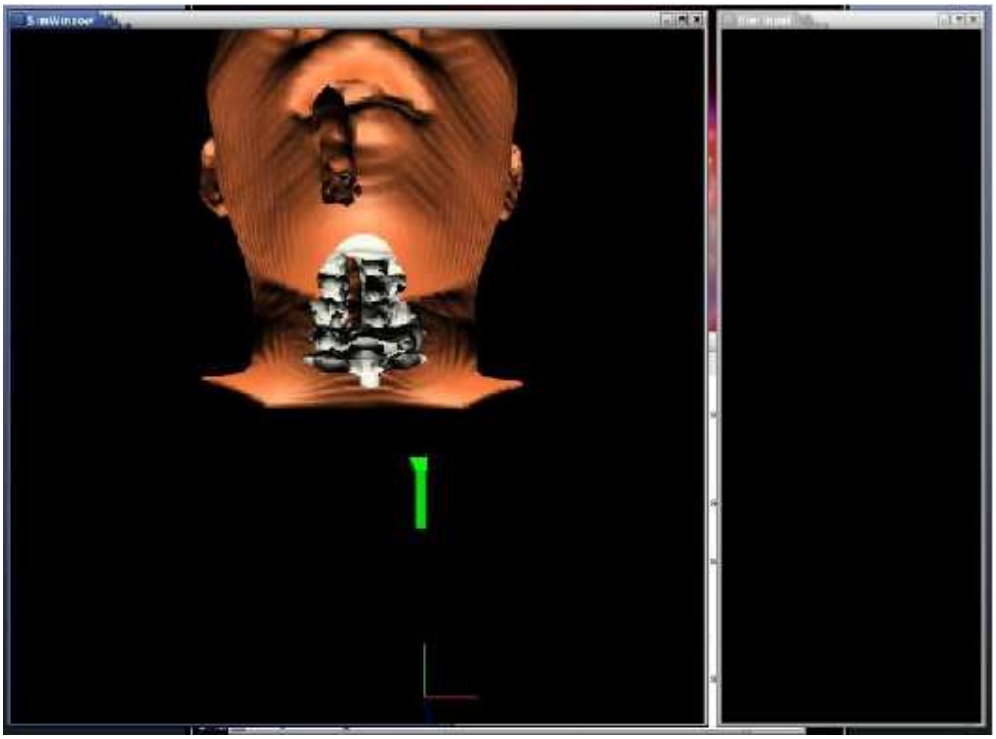
Please refer to the Section called *the example programs* for help on running the provided examples.

the example programs

The example programs are designed to be starting points for building vjVTK based apps. At least in the beginning, simply replacing the VTK portions of the code in the examples' initScene() will be enough. Later adding interactivity to things get more complicated. I would suggest starting with the Medical example, as the navgrab is a complex juggler example adapted to include VTK.

After compiling the applications, non-juggler users should be aware of a few things. VRJuggler uses configuration files to determine the display setup at run-time. This achieves great flexibility as your program can be run on the desktop and then later in an stereo IPT without recompiling. What that means as a beginner is that you must remember to include config files on the command line.

In addition, the Medical example itself requires a few things. The VTK_DATA_ROOT environment variable must be set the wherever the VTK example data is. An example line is: **VJMedical2 ${VTK_DATA_ROOT}/Data/headsq/quarter ${VJ_BASE_DIR}/share/vrjuggler/data/configFiles/simstandalone.jconf**



VJMedical2 example in simstandalone

Figure 2. VJMedical2 example in simstandalone

Two additional examples are available, OSG and OpenSG. These examples demonstrate how to combine VTK with a OpenGL based SG in juggler. The VTK and the SG are completely ignorant of each other, but can be used in the same environment. see the READMEs in the relevant example directory for further direct

Development Plan

vjVTK is a library in development. Further improvements are planned before hitting a 1.0 release. Unfortunately, since I have no active projects using vjVTK the time devoted to development is quite limited. This section outlines the development process as it is envisioned at this point. For all comments, requests, bugs, etc. please send them to me at blom (at) informatik (dot) uni-hamburg (dot) de.

Additional planned Features for 1.0 release

- Possible further improvements to the build/installation process
- Easy end-user Transformation integration (i.e. without having to deal with the context information as it stands now.)
- Extended and better Examples
- distributed juggler apps, testing, and implementation if needed

Notes

1. mailto:blom@informatik.uni-hamburg.de
2. http://imve.informatik.uni-hamburg.de
3. http://www.vtk.org

4. http://www.VRJuggler.org
5. http://imve.informatik.uni-hamburg.de
6. http://www-unix.mcs.anl.gov/fl
7. http://www.vrac.iastate.edu
8. http://brighton.ncsa.uiuc.edu/~prajlich/vtkActorToPF/
9. http://imve.informatik.uni-hamburg.de/publications.htm
10. http://imve.informatik.uni-hamburg.de/blom/vjVTK_current.tar.bz2
11. http://imve.informatik.uni-hamburg.de/blom/vjVTK.pdf