

Including Haptic Display in the Explorable Virtual Human (EVH)

Elizabeth Prince
University of Colorado, Health Sciences Center
Colorado School of Mines

Alyn Rockwood
Colorado School of Mines

Karl Reinig
University of Colorado, Health Sciences Center

To enhance the experience of using the EVH, we have added haptic display of the Interactive Anatomic Animations (IAAs). The haptic display is currently rendered by PHANToM desktop and premium force reflective devices. The software and operating system requirements to install the PHANToM device drivers limit the portability available to any haptics application. Installation of the drivers require either Silicon Graphics Incorporate (SGI) workstations running Irix 6.5 or Intel or AMD based processors running an NT based operating system. New updates also allow Red Hat Linux 7.2. In addition, a running PHANToM device requires the processor to complete the servo loop at a rate of 1 kHz. For these two reasons, most haptics applications are developed in C++, a language that performs much faster than Java and needs at most a recompiling of the original source code to switch among the systems indicated.

The design for the EVH intentionally allows for a wider audience. By choosing Java as the development language and through testing in various environments, the EVH is available to many system setups excluded by haptics requirements.

So the question becomes: How can we enhance the EVH experience through the sensation of touch without losing a large portion of our audience? One solution comes through extending the functionality of the existing application through a service available only to those users with the PHANToM drivers installed. This solution provides the additional feature of touch to those owning a PHANToM, while keeping those who don't own one from even knowing anything is missing.

A simple Java graphics/ C++ haptics environment requires three basic components: shared models, a common cursor position, and common transform matrices. Our applications utilize gl4java for graphics as well as the GHOST SDK for various haptics methods.

One of the most straightforward methods of sharing the data between Java and C++ is via the JNI interface provided by Java. This provides a relatively fast loading method for the models and works fine for single models without transformations. However, even when operating in separate threads, applying transformations to both the haptics and the graphics causes CPU management issues. As previously mentioned, the haptics requires a 1kHz rate through the servo loop, or the PHANToM will cause a timeout error. The graphics, on the other hand, should be updated at a rate of 30Hz with small dips below this rate having little effect on the experience. While changing the priority of the threads can help ensure these rates for small models, as the number of polygons increases, this requirement becomes increasingly difficult to guarantee.

The problem with using Java threads lies in the difficulty of truly managing the resources. One can increase the priority of the graphics thread to make the graphics update appear continuous, which generates a haptics timeout error. Alternatively, one can increase the priority of the haptics thread to prevent timeout errors, causing significant lags in the graphics update. In the middle range, one will unpredictably experience either a graphics lag or a haptics timeout, suggesting there is no good middle ground. Yet on a dual processor machine, while 100% of one processor is used, the other remains largely idle.

With this observation, we moved from the JNI/Java thread based model to a model in which the C++ haptics runs as a completely separate process from the Java graphics, allowing the operating system rather than the Java Virtual Machine (JVM) to determine resource allocation, in particular CPU scheduling. The processes communicate any updates via sockets. This model allows CPU scheduling more conducive to the needs of each process. In the dual processor machine, up to 20 models have been loaded and successfully manipulated.

The addition of the haptic interaction makes the EVH a far more powerful tool. It provides a tactile aspect to the learning environment. Interaction with the mouse is limited to 2D, or at best manipulations can be added by allowing the user to choose a manipulation from a menu or by clicking a button. Manipulation with the PHANToM stylus allows 3D manipulations of the models directly, giving greater control to the user. Providing the users with a more complete experience, it allows for the possibility of adding tangible qualities, such as mass and texture, to the models.

Currently we are developing several enhancements to this system. For one, while it is possible to feel any model chosen from a file, the models all feel the same. Whether bone or muscle or any other tissue, all feel equally firm. We are adding stiffness properties to the models for the haptics side and appropriate deformations to the graphics for a more realistic interaction. In addition, we are moving away from a single point of contact with the models to touching the models with tools of finite size, such as needles and spheres.

In conclusion, for most applications requiring haptic functionality, one would not choose to compromise the speed of C++ for the more universal nature of Java. Requirements of the PHANToM driver itself greatly limit the systems on which it will run. However, to enhance an existing Java application with tactile ability, running the haptics as an independent application out-performs JNI for manipulation of models or multiple polygonal model situations.