

Virtual Reality - Quiz #4

April 3, 2008

(50 minutes)

1. Based on what you know, what are common features of VR toolkits? How can you classify them? Give examples. (2 point)

Some of the common features of VR toolkits are: they support a number of I/O devices, they allow importing CAD models and a variety of formats and have support for multi-user interaction.

They can be classified by:

- Whether they are text-based (e.g., Java3D) or graphical-programming (e.g., Vizard or PeopleShop).
- The type of language used and library size. For example: WorldToolkit uses C++, Java 3D uses Java and Vizard uses Python scripting language
- The type of I/O devices supported. For example, Java3D only supports “natively” mouse and keyboard interaction, while Vizard supports a wide range of devices, including the 5DT glove, Polhemus, Ascension and Intersense trackers, among others.
- The type of rendering supported.
- Whether general purpose (e.g., Java 3D) or application specific (e.g., PeopleShop, for military applications)
- Whether proprietary (e.g., Vizard Toolkit) or public domain (e.g., Java3D)

2. What is a scene graph, and what types do you know? Explain how and why a scene graph changes from frame to frame, and what happens when hierarchy changes in the scene (such as when an object is grasped or when it is destroyed)? Do you think a graphic scene graph traversal is synchronous (at the same time) with a haptic scene graph traversal for the same simulation? Why Yes/No (2 points)

A scene graph is a hierarchical organization of objects (visible or not) in the virtual world (or “universe”) together with the view to that world. A scene graph is represented as a tree structure, with nodes connected by branches. There are two main types of scene graphs: graphic scene graphs and haptic scene graphs.

In a scene graph, an object is said to be a child of a node if it is defined in terms of the coordinate system defined by that node. For example, a hand model and a ball may be children of a general coordinate system for the entire world. A scene graph then may change from frame to frame as objects may “move” to another coordinate system. For example, when the user grasps a ball, the ball node is now a child of the hand node, so that whenever the hand is moved, the ball moves with it. When the user releases the ball, its node is detached from the hand node and added back to the “root” node of the world. Similarly, when an object is destroyed, it should be removed from the scene graph, so that it is not rendered in subsequent frames.

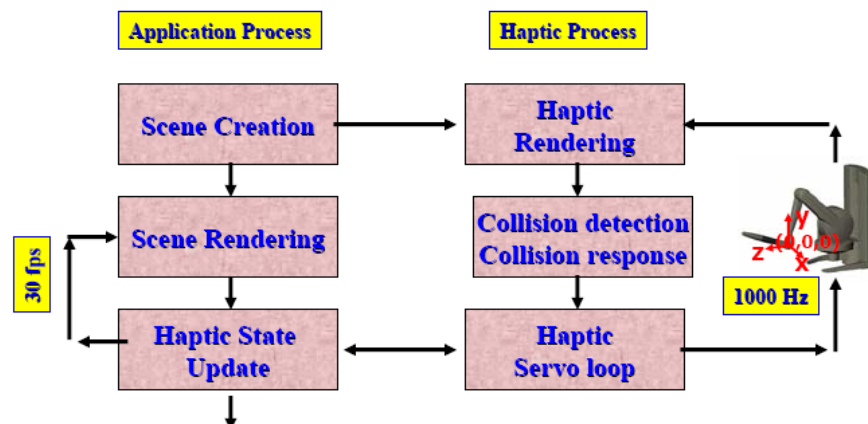
The graphic scene graph and haptic scene graph are not synchronous since not all graphic elements have haptic response, and because both pipelines have different update rates. Instead, the interaction between the two scene graphs can be done via call-back functions.

3. Why do you think it is easier to model avatars using People Shop Di-Guy in military simulations, than using VRML? Why you don't need a sensing suit to interact with the Di-Guy simulation in real time? In what way is People Shop advantageous in networked simulations? (2 points)

People Shop Di-Guy makes easy the modeling of avatars, because it provides a set of pre-defined joint motions which are interpolated in real-time. These joint motions correspond to high level movements, such as walking, running, etc. Since the joint motions are pre-stored, there is no need for tracking individual joints with a sensing suit. In networked simulations, there is need for updating the movement from each joint, which can be costly. Instead, People Shop only transmits updates at the task level (action, position, velocity), which reduces considerably the amount of information sent through the network and provides a smoother shared simulation.

4. What is the communication mechanism between the graphics and haptics scene graphs in case of the GHOST toolkit? How are haptic effects implemented in GHOST and how many Phantom interfaces can be used to interact with a non-distributed virtual environment (single user and no network involvement)? How do we insure the proper mapping between the virtual scene (camera workspace) and the Phantom interface work envelope? Does this have any consequences on the physical model of the virtual world? (3 points)

The graphics and haptic scene graphs have different refresh rates (30 fps vs. 1000 Hz for the Phantom). Therefore communication is done through message passing, namely call-backs. The user defines certain nodes in the haptics scene which have call-back functions associated with them. The application calls *updateGraphics* to have graphics information updated. Nodes that have a graphics call-back defined, and have a new state since the last call to *updateGraphics* will copy their current state to a defined data structure. Call-backs pass new state information of the haptic scene nodes from GHOST haptics process to the application process. For example, the user can create a callback for the graphics representation of the position of the *gstPHANToM* node. This should change to callback of *gstPHANToM_SCP* after collision, so the user can see the location of the contact point on the object.



Haptic effects are implemented with the *gstDynamic* nodes (with the four derived classes *gsdDial*, *gstButton*, *gstSlider*, and *gstRigidBody*). Up to 4 Phantom interfaces can be used by a single user, and they are connected in a daisy chain. The 4 Phantoms are mapped to

4 *gstPANToM* nodes in the haptics scene. The camera workspace needs to be oriented such that its z axis is aligned with the axis of the Phantom workspace. Furthermore the width of the camera workspace has to accommodate the whole left-right movement of the real Phantom interface. This generates a scale factor $S_{frustum} = D_{xmax} / D_{phantomxmax}$

The geometrical scaling has to also map to the scaling of the spring and damping constants, such that objects feel the same regardless of geometrical scaling.

$$SurfaceKspring_{new} = SurfaceKspring_{current} / S_{frustum}$$

5. In what ways does 3DGame Studio help us author games? Comment also in relation with proprietary game engines (1 point)

3DGame studio allows character creation and animation with a Model Editor, objects can be created with a world editor, then multi-textured with bump and light maps. Physics behavior and collision detection can be done in a script, and 3D sound sources can be added. Compared to proprietary game engines, 3DGame Studio offers the advantages of openness and reduced cost.