N95-14032

INTERACTING WITH VIRTUAL WORLDS

David Zeltzer
Massachusetts Institute of Technology
Cambridge, Massachusetts

What tools should we provide the user for defining and interacting with objects and agents in virtual environments at varying levels of complexity? Understanding the appropriate simplifications to make is critical for modeling non-trivial environments and agents with varying levels of autonomy. I describe a set of abstraction mechanisms appropriate for constructing and interacting with virtual worlds, and I discuss how programming, and direct manipulation or *guiding* techniques, can be used to afford users interactive access to graphical simulations at appropriate levels of abstraction.

Structural abstractions provide a mechanism for describing the kinematic structure of objects, such as transformation hierarchy for a jointed figure, and for defining physical attributes of objects, e.g., mass, stiffness, etc.

Procedural abstractions are a mechanism for defining processes that control rigid and non-rigid object motion, such as elastic deformation, collision detection, forward dynamics or inverse kinematics, independent of the structure of the agen or object.

Functional abstractions are the means with which we associate procedures with objects and object sub-assemblies, for the purpose of defining meaningful behaviors. This allows complex, largely unmanageable systems with many DOFs - like the human figure, with over 200 DOFs—to be decomposed into a set of small, manageable subsystems, each with few DOFs.

An *agent* consists of a structural definition, a set of functional units which define the behavior repertoire of that entity, and some means for selecting and sequencing (possibly concurrent) behaviors. The mechanism for selecting and sequencing these motor skills must link *action* with *perception*.

The lowest layer of abstraction is represented by objects and structures with no procedural attachment, and is called the *machine level*. At the other extreme, the *task level*, guiding and programming tools interact with agents through their repertoire of behaviors. Task level interaction *emerges* when guiding and programming techniques are applied at higher layers of abstraction.

Guiding and programming tools have a dual nature: they can be used to construct and modify instances of any of the abstraction types; or they can be used at runtime as control instruments. I describe how programming and guiding can be used with each of the four classes of abstraction.