VR Applications

Class 5

Dr. Nabil Rami

http://www.simulationfirst.com/ein5255/

Last week

- Output Subsystem
  - Visual
  - Audio
  - Haptic/Tactile
- Input Subsystem
  - Locomotion
  - Tracking
  - Gesture
  - Voice

Simulation Software

- Is the core simulator component that enables the other components to perform their functions within the simulation system
- Collects data from the user via the input devices
- Processes the data and updates the simulation state

Simulation Software

- Presents the simulation state to the user via the output devices
- Shares the local state with other simulators via the communications pipeline

Simulation Software

- This class focus is on what makes a good simulation software
- Rather than talking about specific software

Simulation Software

- Simulation hardware, software, and end user requirements are constantly changing
- The software must be able to pull together available technology and integrate it into a seamless package that solves a particular simulation problem

Functional Requirements

- Start by defining a set functional requirements:
  - Ease of Use: To promote efficiency and productivity in the development process
  - High Performance: To support real-time, large scale simulation
  - Maintainability: To minimize life-cycle cost

Functional Requirements

- Scalability: To support new requirements and ever increasing scope
- Cross-Platform Support: To support multiple operating systems to enable maximum selection of available hardware
- Open Standards Support: To gain maximum leverage from existing and future simulation products

Programming Paradigm

- A programming paradigm is a specific set of rules and models that the programmer, who subscribe to that paradigm, has to follow
- The challenge is to use a programming paradigm that allows the software to easily fulfill the functional requirements

Programming Paradigm

- The closer the paradigm matches the problem-space, the easier it will be to solve the problem
- For example, a command line interface to a graphics program will require a great effort from the user than a graphical user interface

Programming Paradigm

- Dynamically reconfigurable and extendable object-oriented paradigm is an example of a programming paradigm that works well with simulation software

Programming Paradigm

- Some of the fundamental principles of this paradigm are:
  - Support Modularity: Enables functional disassembly of the simulation problem space
  - Minimize Dependencies Between Modules: Simplifies code changes enhancing ease of use

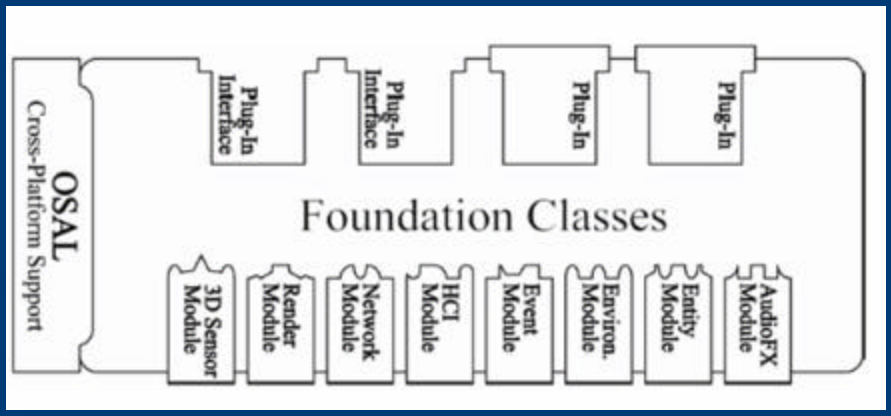
Programming Paradigm

- Support Dynamic Reconfiguration: Allows systems to be reconfigured at runtime to improve usability
- Support Extension of Functionality: Provides a mechanism to extend the functionality, which enhance maintainability and scalability

Architecture

- Functional boundaries and methodologies defined in the requirements and programming paradigm are crafted into a blueprint that would support the implementation
- The same as building a house

Architecture



Architecture

- The architecture is based on four key components types:
  - Operating System Abstraction Layer (OSAL)
  - Foundation Classes
  - Modules
  - Plug-Ins

OSAL

- It is a set of software utilities that provides a common set of operating system (OS) level services while encapsulating the underlying OS implementation
- Services include timing, synchronization, dynamic loading of runtime libraries…

OSAL

- Exclusively utilizing the OSAL services, the software remain OS and platform independent
- Results in a software easier to maintain

Foundation Classes

- The foundation classes are the unifying software infrastructure that provides the common set of services needed by all other components of the architecture
- These services include:
  - data communications between the components
  - conflict resolution
  - scheduling
  - data capture

Modules

- Modules are the primary mechanism that the architecture uses to provide:
  - Ease of use
  - Maintainability
  - Key simulation functionality
- A module is a reusable, modular software library that provides specific functionality to the simulation system trough a well defined and published interface

Modules

- The guiding principle is to design modules that contain essential simulation functionality in a package that depends on a minimal amount of information from the remainder of the simulation

Modules

- Modules of the same type are interchangeable because they are written to satisfy the same standardized interface
- Knowing the interface, other developers can implement their own modules

Modules

- Common modules that can be seen in a simulation software:
  - AudioFX: Responsible for providing spatialized audio information
  - Entity: Responsible for gathering, maintaining, and transmitting entity (object) information
  - Environment: Responsible of managing the state of the virtual world and providing state information

Modules

- Event: Responsible for gathering, maintaining, and transmitting events such as collision detection, detonation…
- Human Computer Interface (HCI): Provides access to input and Output (I/O) devices
- Network: Responsible for capturing and transmitting all simulation network information
- Render: Responsible for providing 3D visual information and 2D overlays
- 3D Sensor: Provides access to 3D tracking devices

Plug-Ins

- A plug-In is a reusable, modular software library which provides specific functionality, and operates as a subsystem of a specific application
- Plug-ins differ from modules in that plug-ins are implemented using the functionality provided by the foundation classes, modules, OSAL

Plug-Ins

- Plug-ins are OS and platform independent, but software architecture dependant
- All plug-ins have the same interface that consists of simple member functions that allow the developer to initialize, synchronize, and shut down… modules

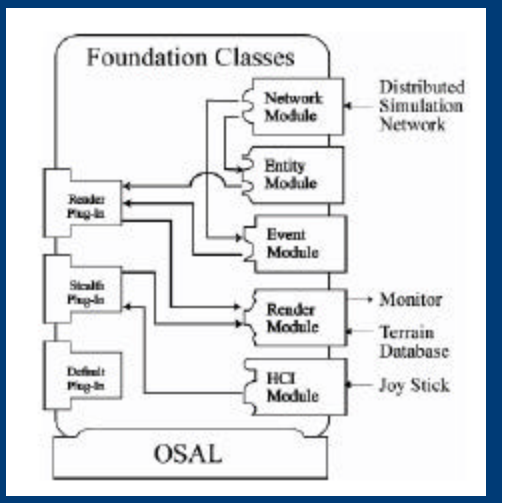
Plug-Ins

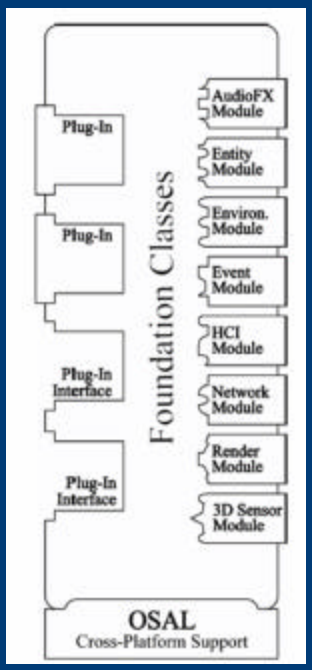
- A collection of plug-ins that interact to solve a simulation problem is called an application

Sample Application

- A Stealth application allows the user to see the virtual world without being detected by other simulations on the network
- A basic Stealth allows the user to fly around the virtual world using a joystick
- The viewpoint is completely controlled by the joystick

Stealth





Next Week

- Chapter 7
- Discussion
- Exam Review