Design & Implementation of Virtual Simulations

Douglas Hodson, doug@OpenEaagles.org
David Gehl, dave@OpenEaagles.org

Agenda

- Big Picture
- Visual System
- Interoperability
- Dynamics Model
- Design Patterns
- Simulation
  - Graphics Hierarchy
  - Examples (100% Open Source)
- Summary

Big Picture

Anatomy of a Flight Simulator

Visual System
(Out-The-Window)

Pilot Vehicle Interface
(Control Inputs &
Heads Down
Displays)

Human Operator
(Pilot)

Anatomy of a Distributed Virtual Simulation

Network

Each "Simulation"
Shares Its State Data
Across a Network

Observations

- Because of the Human…
  - Visual System and Pilot Vehicle Interface (PVI) Must be Realistic Enough to be "Believable"
  - Simulation Must Respond to Pilot Inputs (e.g., Control Inputs) in a Timely Manner
  - Simulation Must Advance Time in Sync with the "Wall clock"
  - Must Execute Physics-based Models, such as an Aerodynamics Model of the Aircraft

Virtual Simulation

- Definition
  - Real People and/or Real System Hardware Interacting with a Simulated System
  - This is Not the Case with Most Simulations
- Result
  - By Including People in the Simulation System, the Software Design of the System is More Complicated

Virtual Simulation Requirements

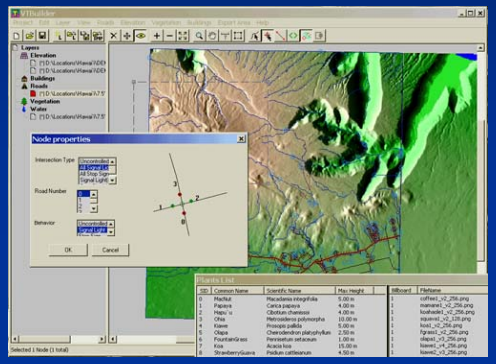
- Introducing Real-World Elements (People/Hardware) Imposes Timing Constraints on the Software System
- Systems with Timing Constraints are Called "Real-time" Systems
- Real-time Systems have Nothing to do with how "Fast" a Computer Runs, it has Everything to do with Reliably Meeting Timing Deadlines

Real-Time Concepts

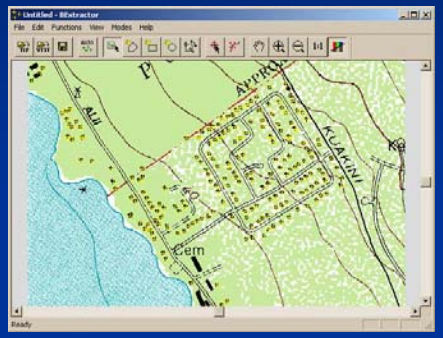
- Software Systems with Timing Constraints
  - Executes in Sync with Wall-clock
  - Interaction Response Characteristics
    - Time to Generate Outputs from Inputs
- Real-time Paradigm: Partitioning of Code
  - Foreground
    - Jobs that have a Time Deadline.
      - Example: Model Mathematics, Redrawing Interface Displays, etc
    - Executed on a Periodic Basis.
      - Example: 50 Hz for Models, 20 Hz for Interface Displays
  - Background
    - Jobs without Timing Constraints.
      - Example: Logging Data to a Hard Drive
    - Execute Whenever Possible. (But Must Finish at Some Point.)

The Visual System

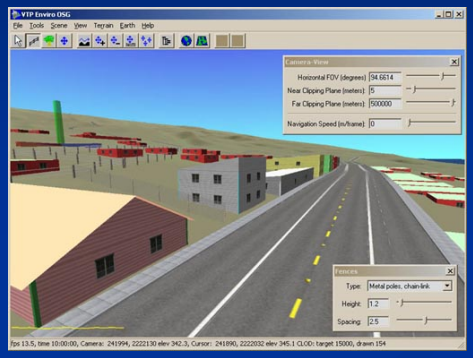
Virtual Terrain Project



Virtual Terrain Builder



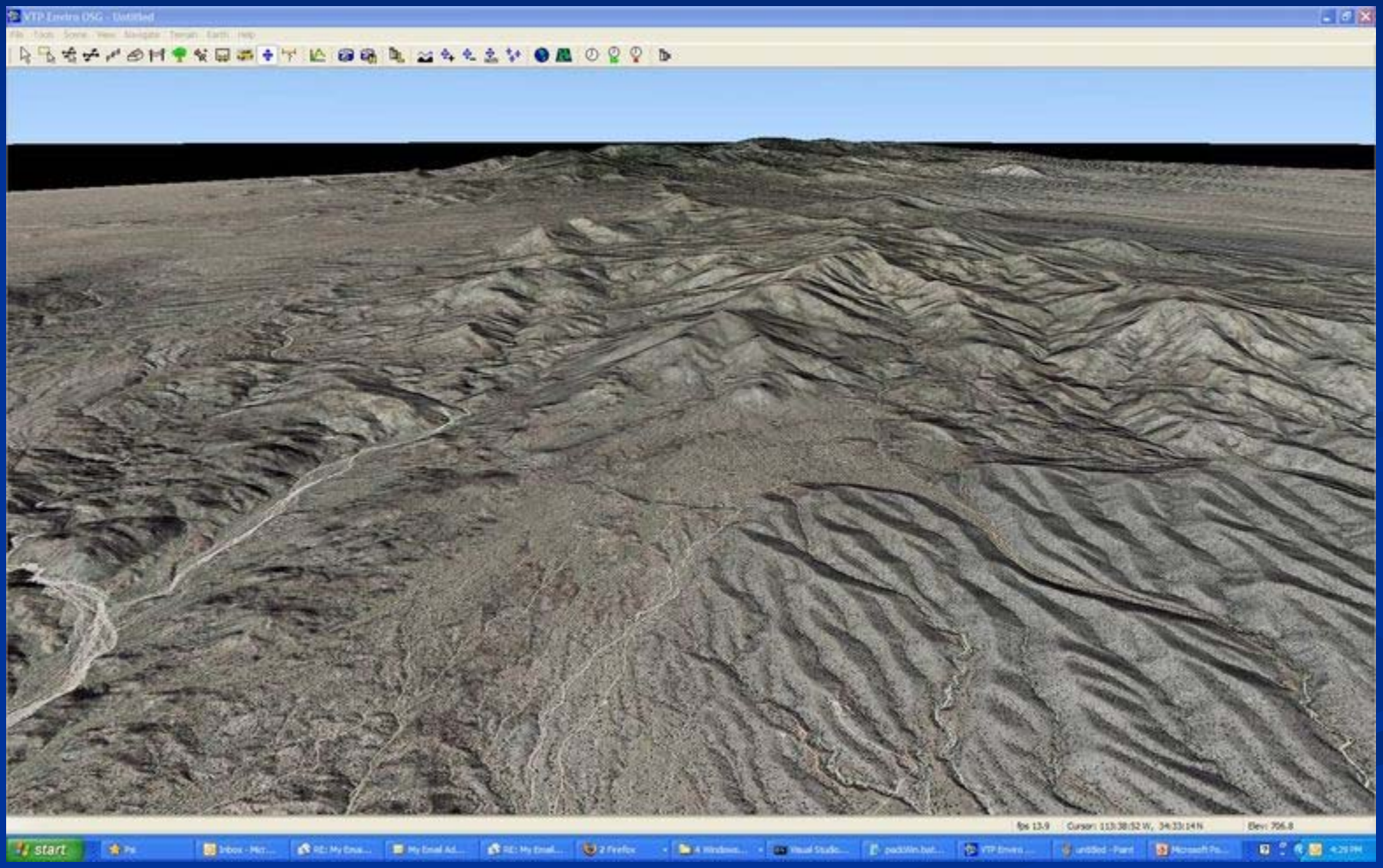
Building Extractor



3d Runtime Environment

- Open-source Tool to Build Visual Databases
- Well Documented with Online Tutorials
- Website Provides Good References for Source Data

Virtual Terrain Project



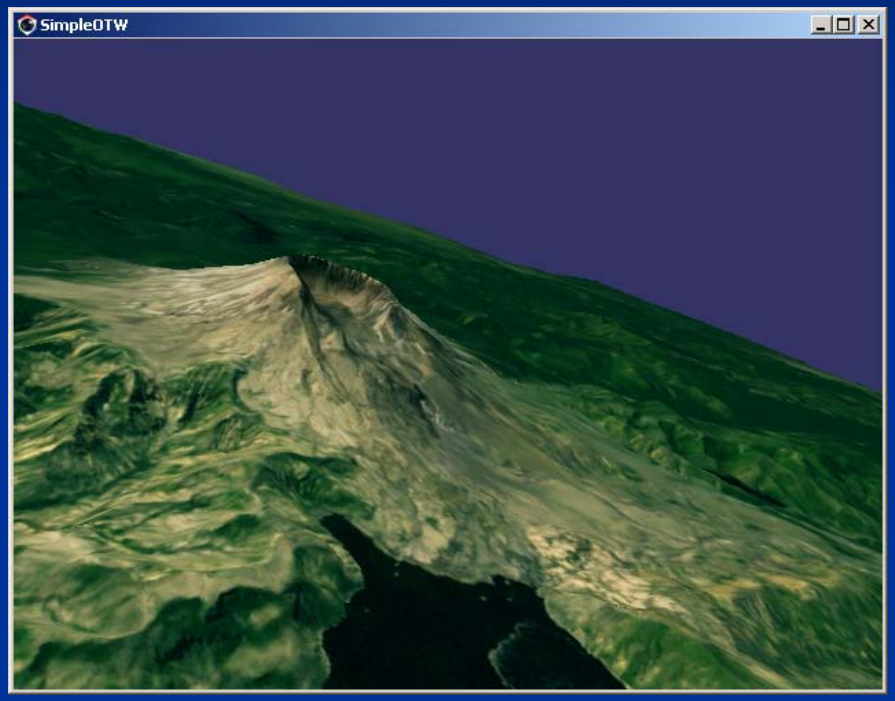
SubrScene IGS
(Image Generation Solution)

- Open-source Simulation Visualization Toolkit
  - Standalone Visual System
    - Can Drive Single Monitor or Multi-channel Dome System
  - SDK for Integration into Other Applications
  - Built with OpenSceneGraph

CIGI

- Common Image Generator Interface
- Open-source Interface Designed to Promote a Standard Way for a Host Device to Communication with an Image Generator (IG)

Out-The-Window Display

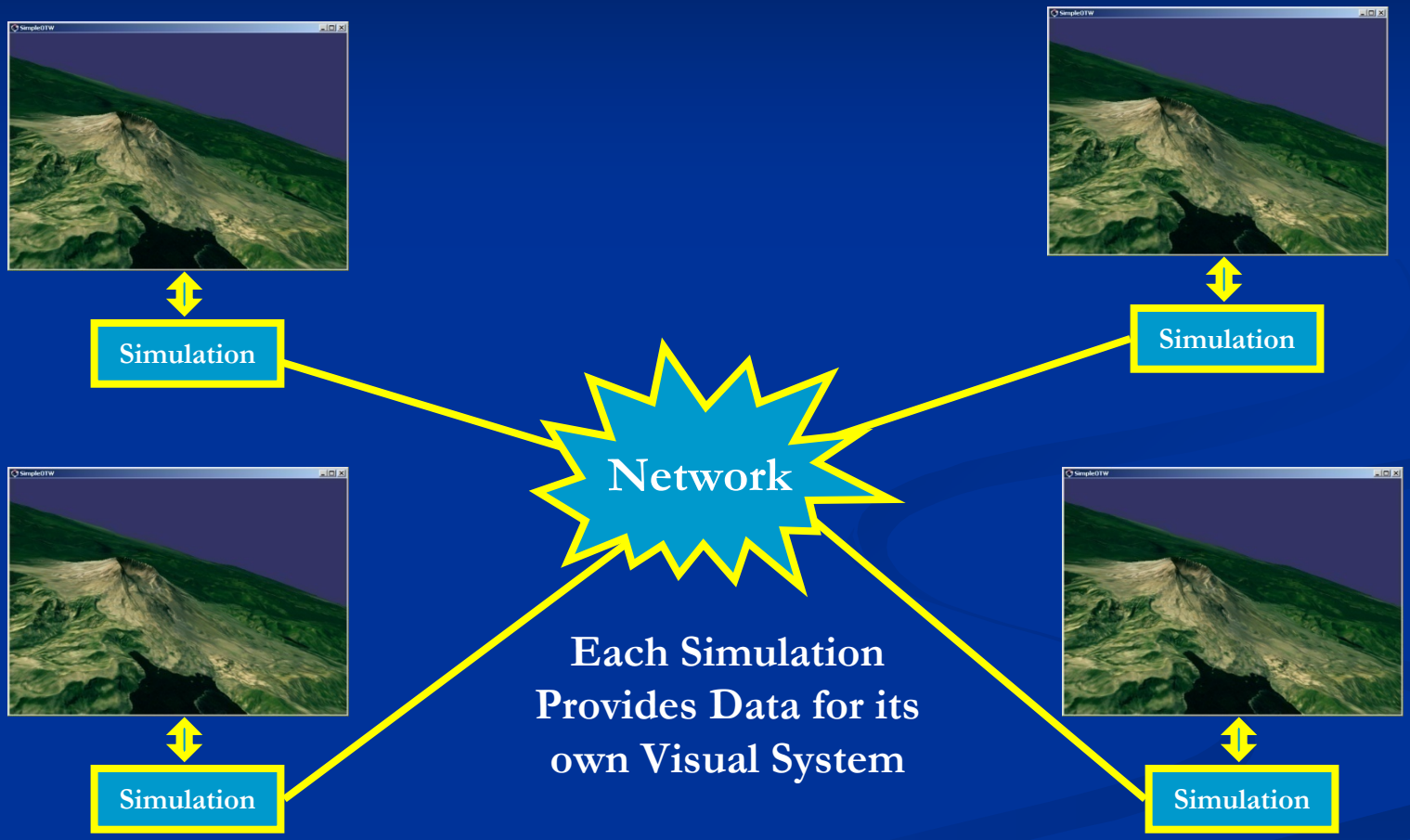


CIGI

Simulation

- Typically a Separate Application that Interfaces with the Main Simulation
- SceneGraph-based Graphics
- OpenSceneGraph is a Mature Open-source Framework to Build these Applications
- Common Image Generator Interface (CIGI)

Distributed Virtual Simulation



Interoperability

(Connecting Simulators)

Distributed Virtual Simulation

Distributed Interactive Simulation (DIS)

- Open Standard for Conducting Real-time Platform-level Wargaming Across Multiple Host Computers
- Defined by IEEE
- Encodes Basic Simulation State Information into Protocol Data Units (PDUs) and Exchanges them with Standard Network Protocols, such as UDP
- Widely-used, Well-defined, and it Works!

High Level Architecture (HLA)

- General Purpose Architecture for Distributed Computer Simulation.
- Rather than a Network Standard like DIS, HLA Defines an Architecture with a Set of API Standards
- User(s) Define the Data to be Shared

…the poRTIco project…

- Fully Supported, Open-source, Cross-platform HLA RTI Implementation
- www.porticoproject.org

Dynamics Model

Dynamics Model

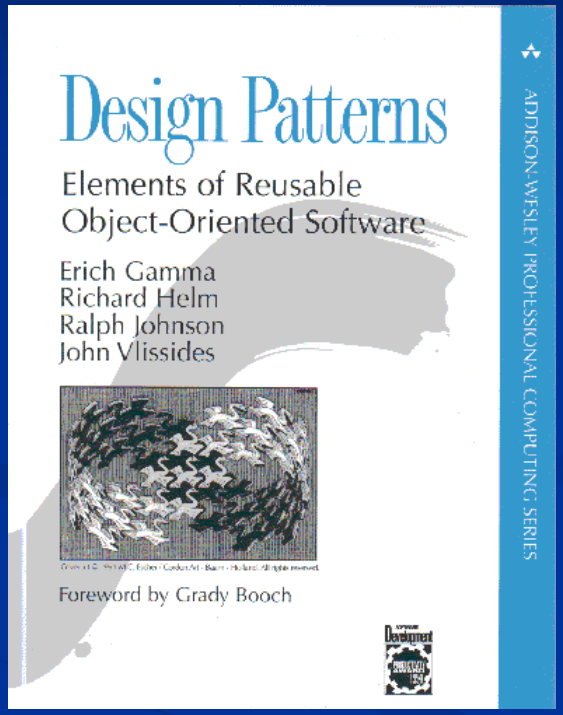
- JSBSim is an Open-source Cross-platform Flight Dynamics Model (FDM)
- Fully Configurable Flight Control System, Aerodynamics, Propulsion, Landing Gear Arrangement, etc.
- Interfaced and Utilized by OpenEaagles

Design Patterns

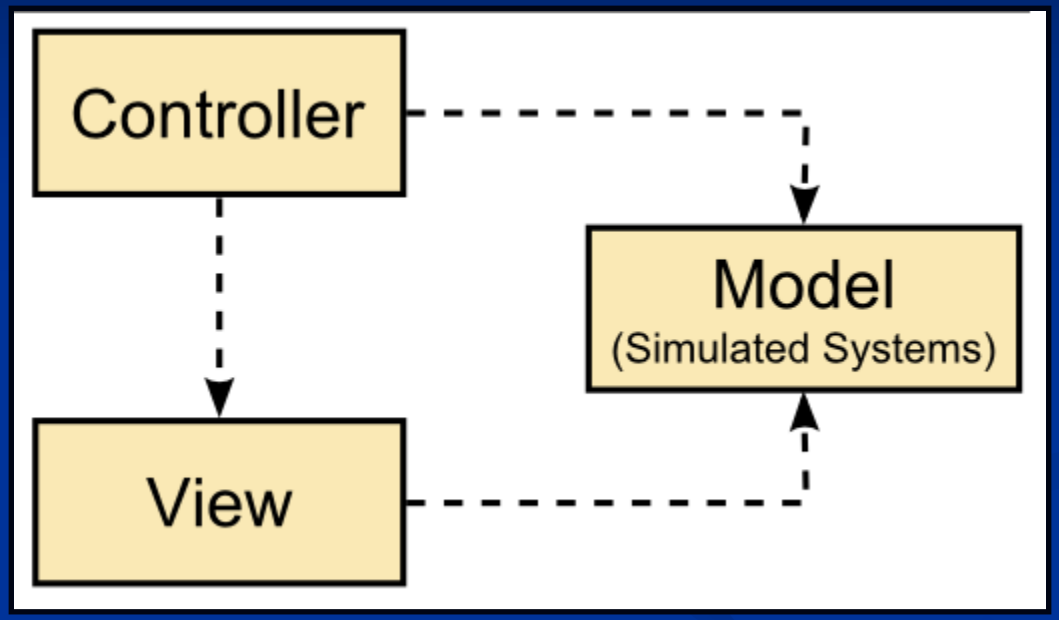
(Computer Science Perspective)

What is a Design Pattern?

- Is a General Reusable Solution to a Commonly Occurring Problem in Software Design.
- It is not a "Finished" Design that can be Transformed Directly into Code.
- It is a Description or Template for How to Solve a Problem that can be Used in Many Different Situations.
- Gained Popularity after Gamma's Book was Published in 1994.

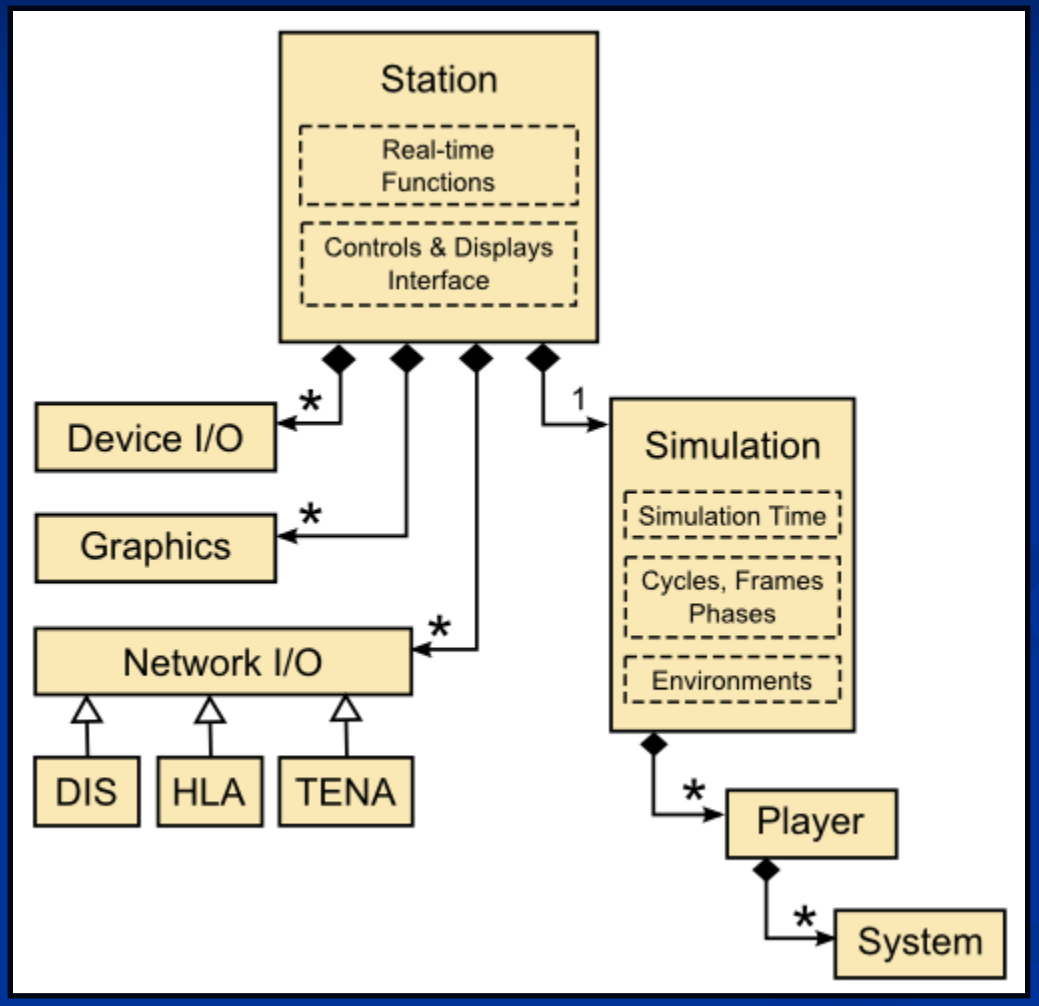


MVC Pattern



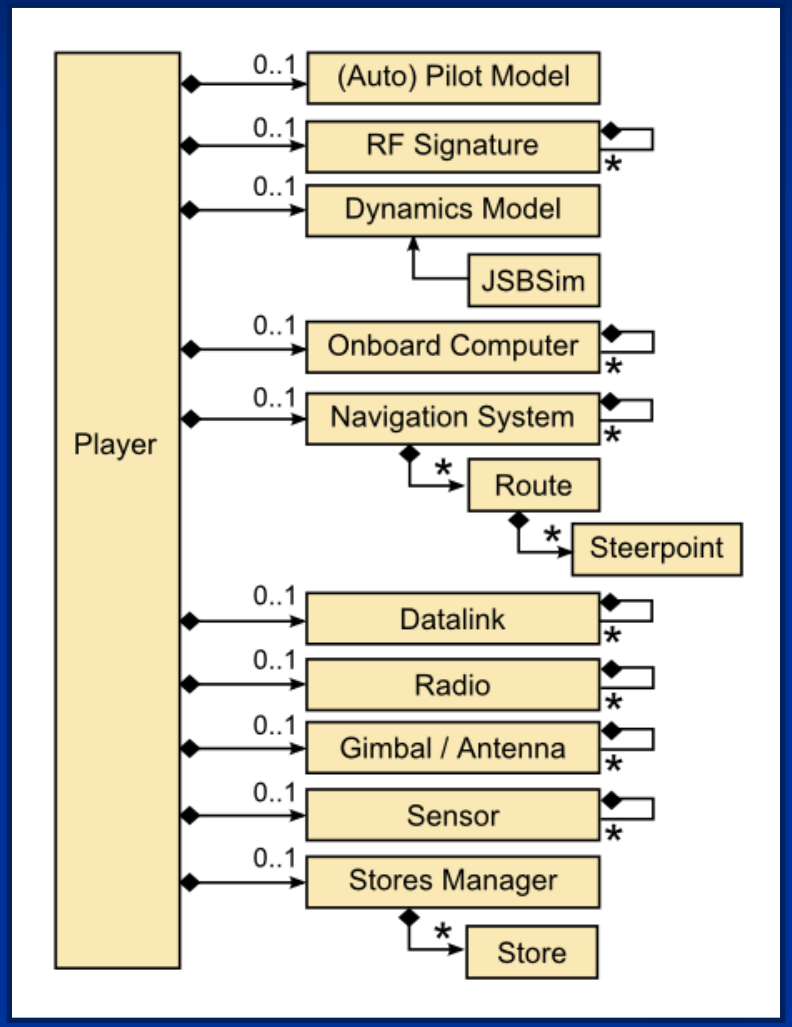
- Model is the Application's Domain Logic
- View is the Application's Graphical Displays
- Controller Connects Model to View(s)

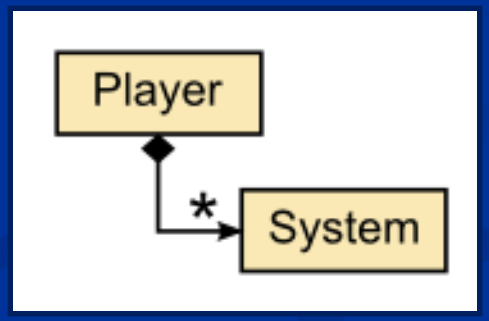
Simulation Pattern



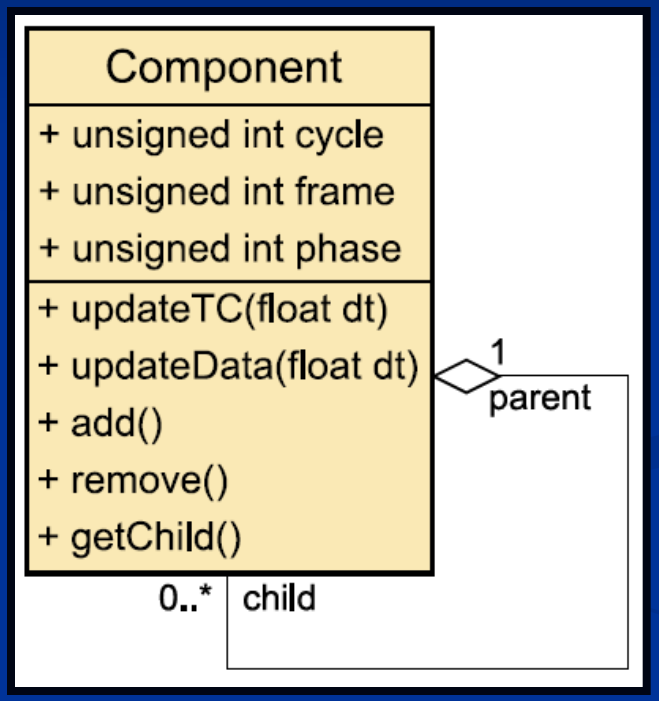
- Asynchronous Execution of Simulated System, Graphics and Network I/O
- Architecture Maps to Real-time Design Paradigms
  - Good "Fit" with Virtual Simulation Requirements
- Leverages Multi-cpu & Multi-core Systems

Player Pattern



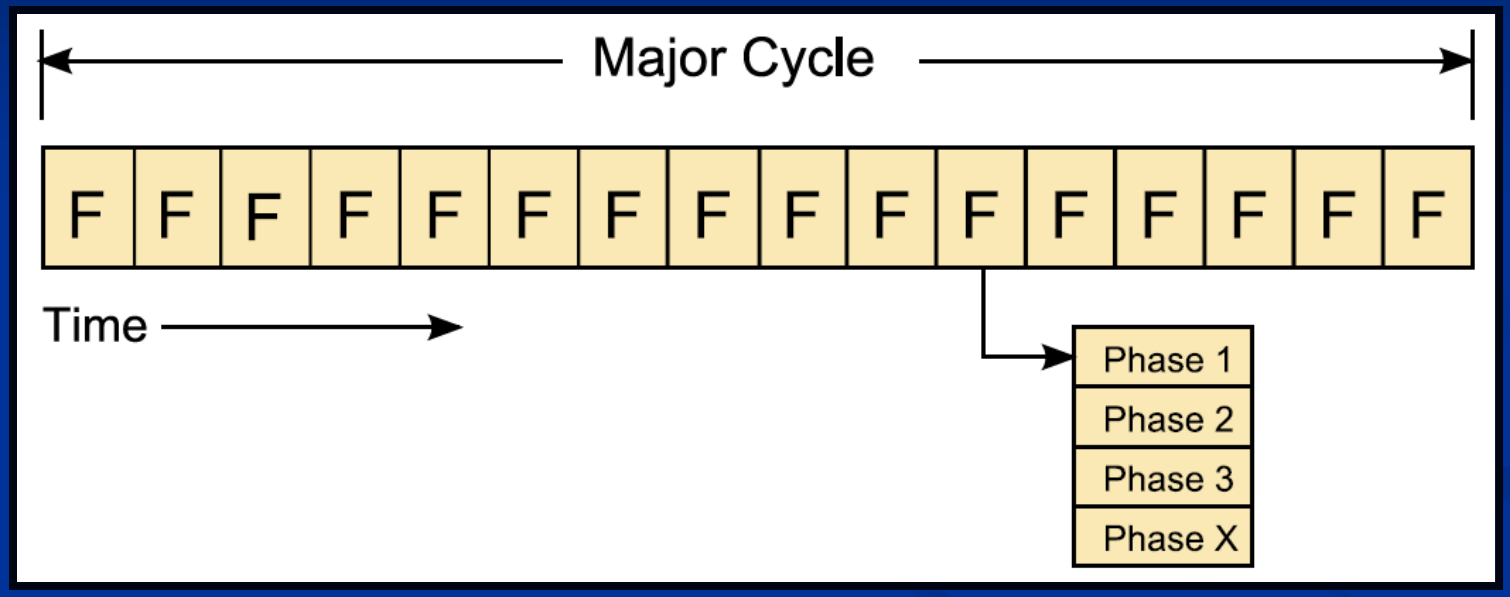


Real-Time Component For Hierarchical Modeling



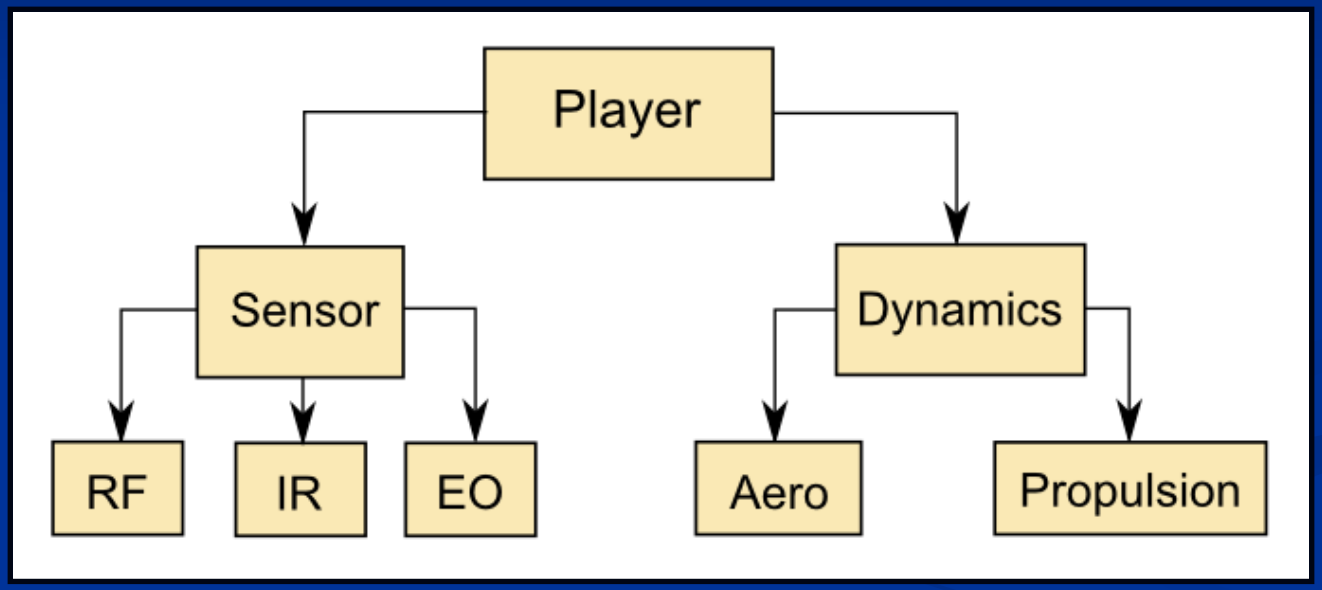
- updateTC – Placeholder for Time Critical Jobs
- updateData – Background Processing

Scheduling Model Code
(Cyclic Scheduler)



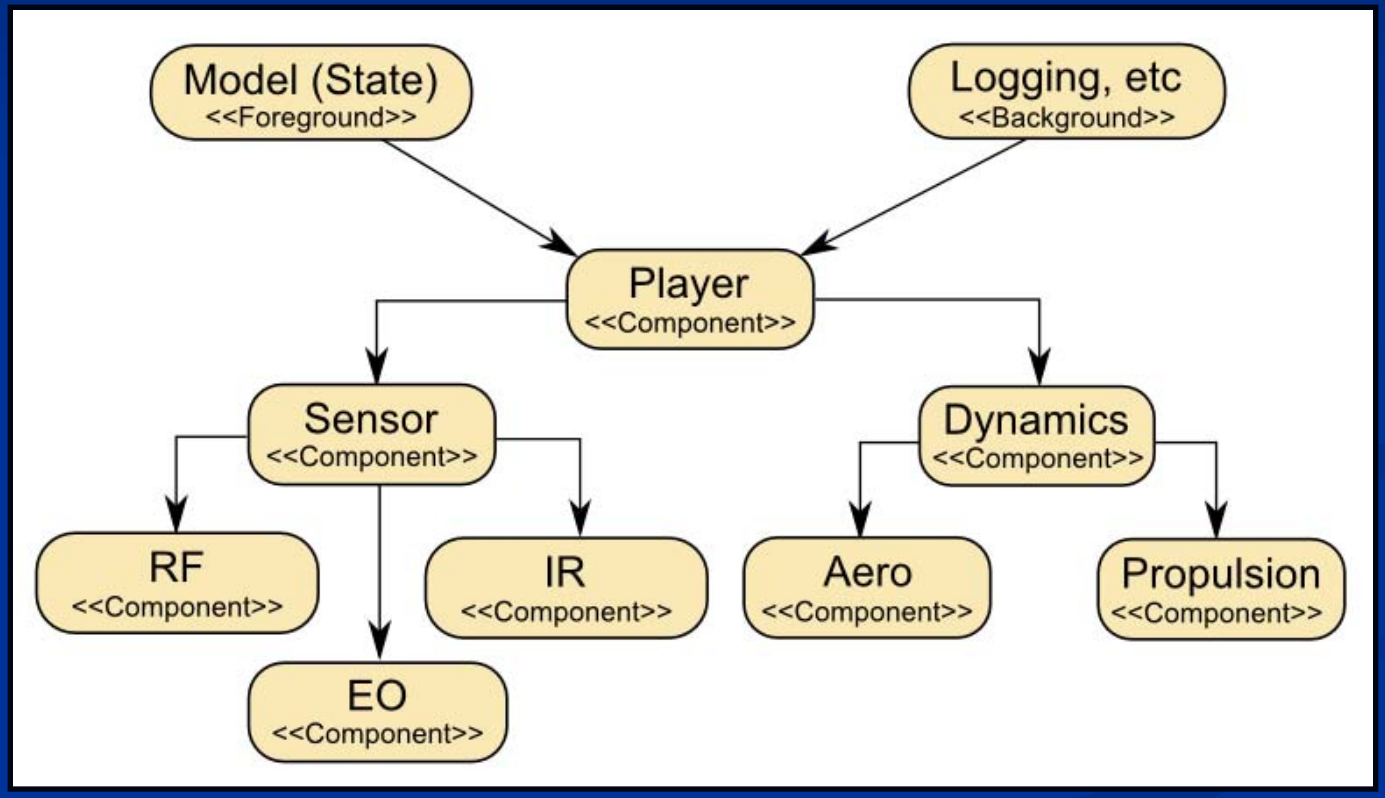
- Provides More Modeling Flexibility
  - Code can be Scheduled to Execute in Different Frames
  - Phases Provide Order
    - Example: Player Dynamics Computed in First Phase of Each Frame
    - Example: RF Sensor Calculation Performed in Second Phase

Player Example

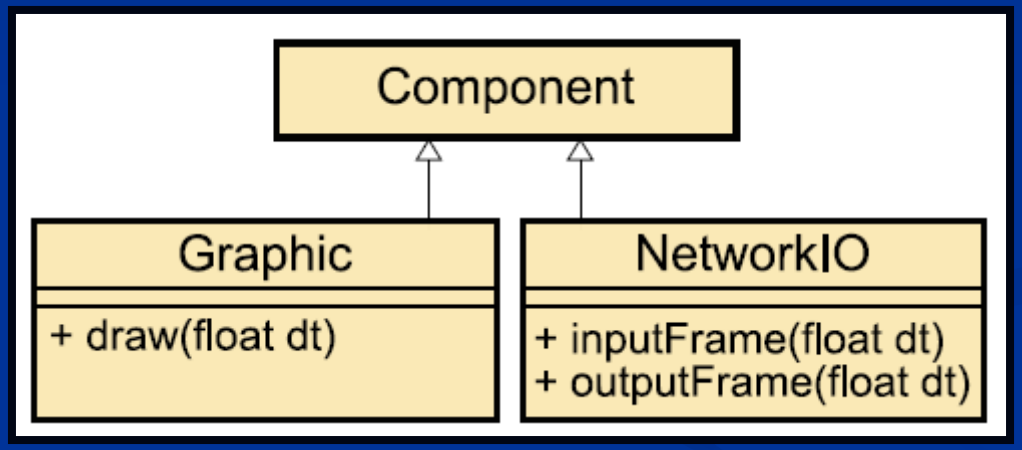


- Modeled as a Hierarchal System
- Based on Component to Execute in Real-time

Player Implementation



Extending Component
(Graphics and I/O)



The Simulation

(Introducing the OpenEaagles Simulation Framework)

Features

- Implements MVC and Component Design Patterns
- RF & IR Modeling Environment, Sensors, etc
- Vehicles, Missiles, Bombs, Navigation, etc
- Support for Reading Dafif & Terrain File Formats
- State Machine to Build AI Agents
- Extensive Graphics Library to Build Simple or Complex Interactive Displays
- Support for CIGI-oriented Visual Systems
- DIS, HLA & TENA Interoperability Interfaces
- Input File Structure & Parser

Design Concept

- Constructive Features
  - Flexibility to Define New Simulations and Scenarios from Databases of Reusable Components
  - Systems and Missions
  - Change Behavior or Properties of Components and Systems via Input Files
- Virtual Features
  - Techniques and Rules to Ensure Models can Meet Time Critical Requirements
    - Pilot-in-the-Loop
    - Hardware-in-the-Loop

Features

- Software Toolkit
  - Consists of Configurable and Extendable Simulation Components
  - Allows Users to Configure Their Simulation to Meet Their Own Unique Requirements
- Performance
  - Designed for Real-Time Performance
  - All Components Contain an Interface for a Frame-based, Time-Critical Thread
  - Standard Real-Time Simulation Rules Govern how Time-Critical Elements of the Component are Modeled

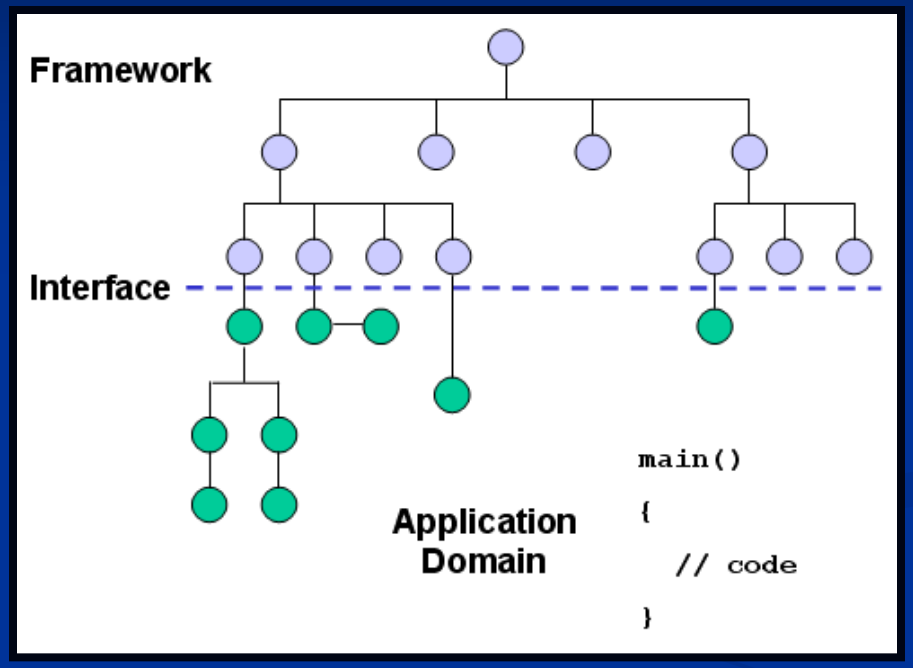
Features

- Object-Oriented Components
  - Provide a Basic Object System from which All Component are Built
  - Common Framework to Build Constructive and Virtual Simulation Components
  - Define Interfaces and Enforce Coding Standards
- Flexibility & Scalability
  - Common Simulation Components and Their Interfaces are Defined as Part of the Simulation Foundation Classes
  - Classes can be Created and Reconfigured from Input Files
    - Attributes and Behaviors can be Extended by Deriving New Classes
  - Users can Build and Add New Higher Fidelity Components as Needed, and Intermix these Components with Other Lower Fidelity Models

Features

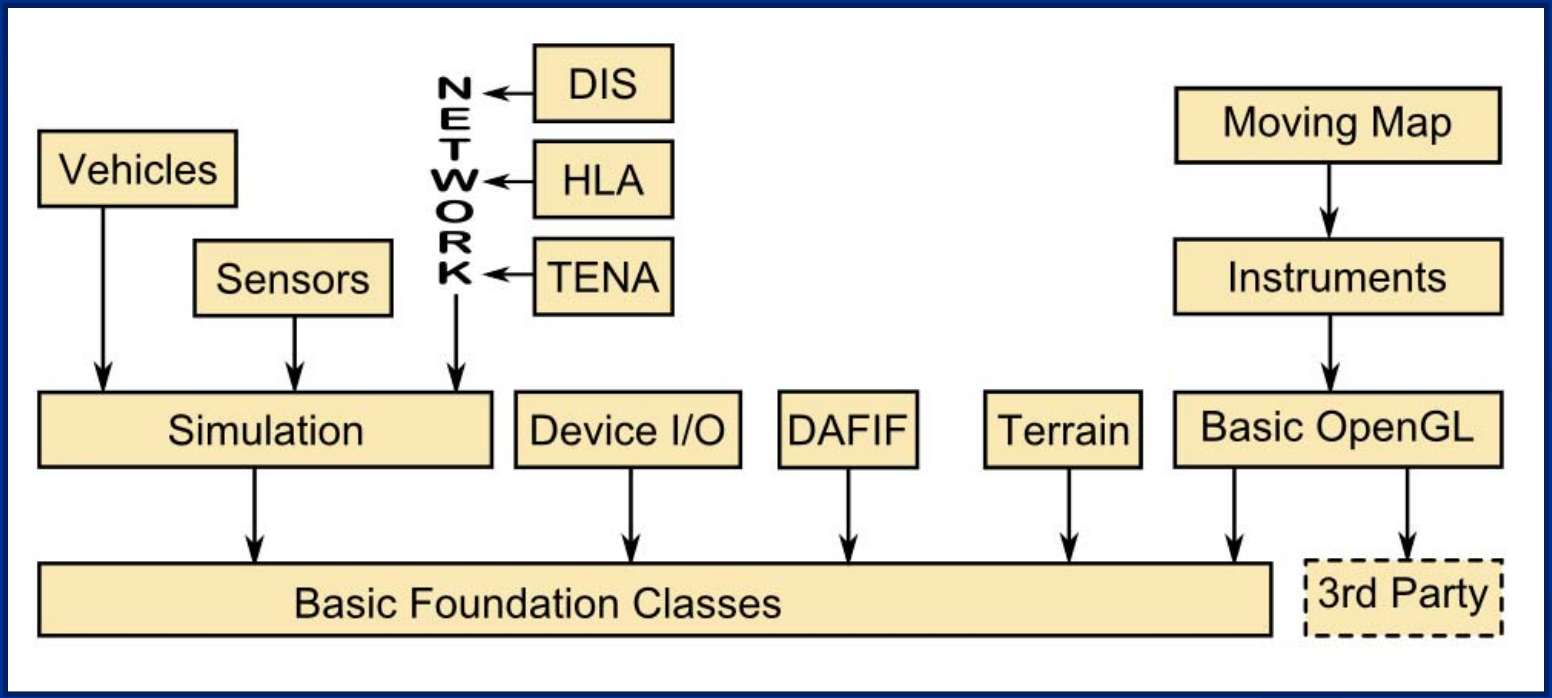
- Distributed, Interactive Simulations
  - Can be Run as a Single Constructive or Virtual Program, it is Designed to Allow Users to Distribute their Simulation Environment Across Numerous Computers
- Open System
  - Windows, Linux, etc
- Graphics Toolkit
  - For Modeling Interactive Pilot Vehicle Interfaces (PVI) and Control Displays
    - Includes a Library of Reusable Aircraft Instruments

Simulation Application



- Application Developer Provides
  - Specifics, Data and/or Maybe Additional Models
  - Process/Threading Environment
    - Supports Single and Multi-core Architectures
  - main() function

Libraries/Packages

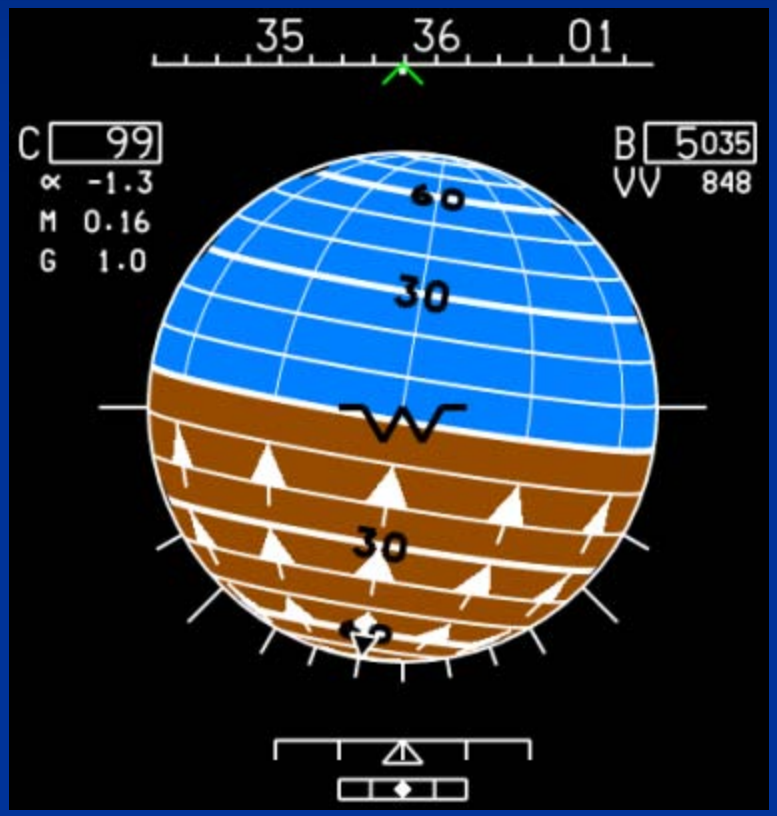


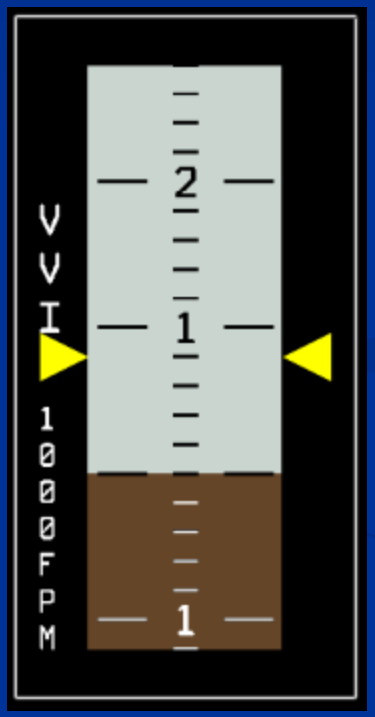
Mature Graphics Hierarchy for Building Operator-Vehicle Interface Displays

Examples

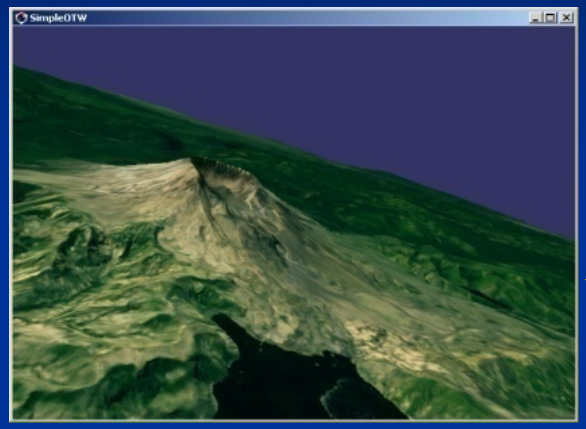
Primary Flight Displays

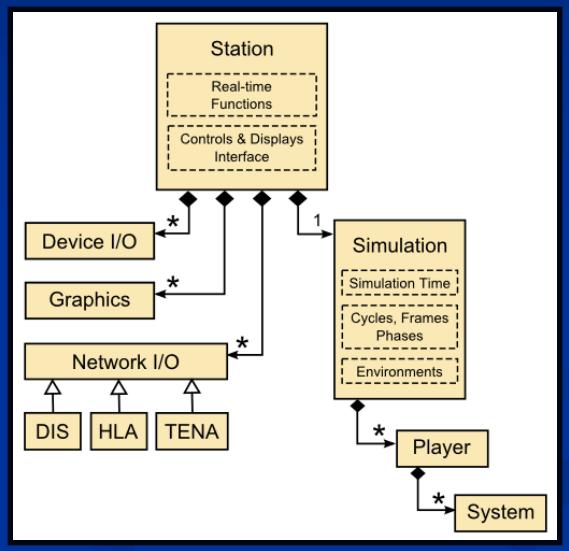
PFD / Instruments

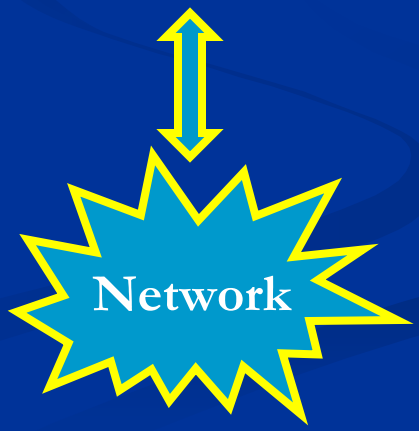




Putting it All Together

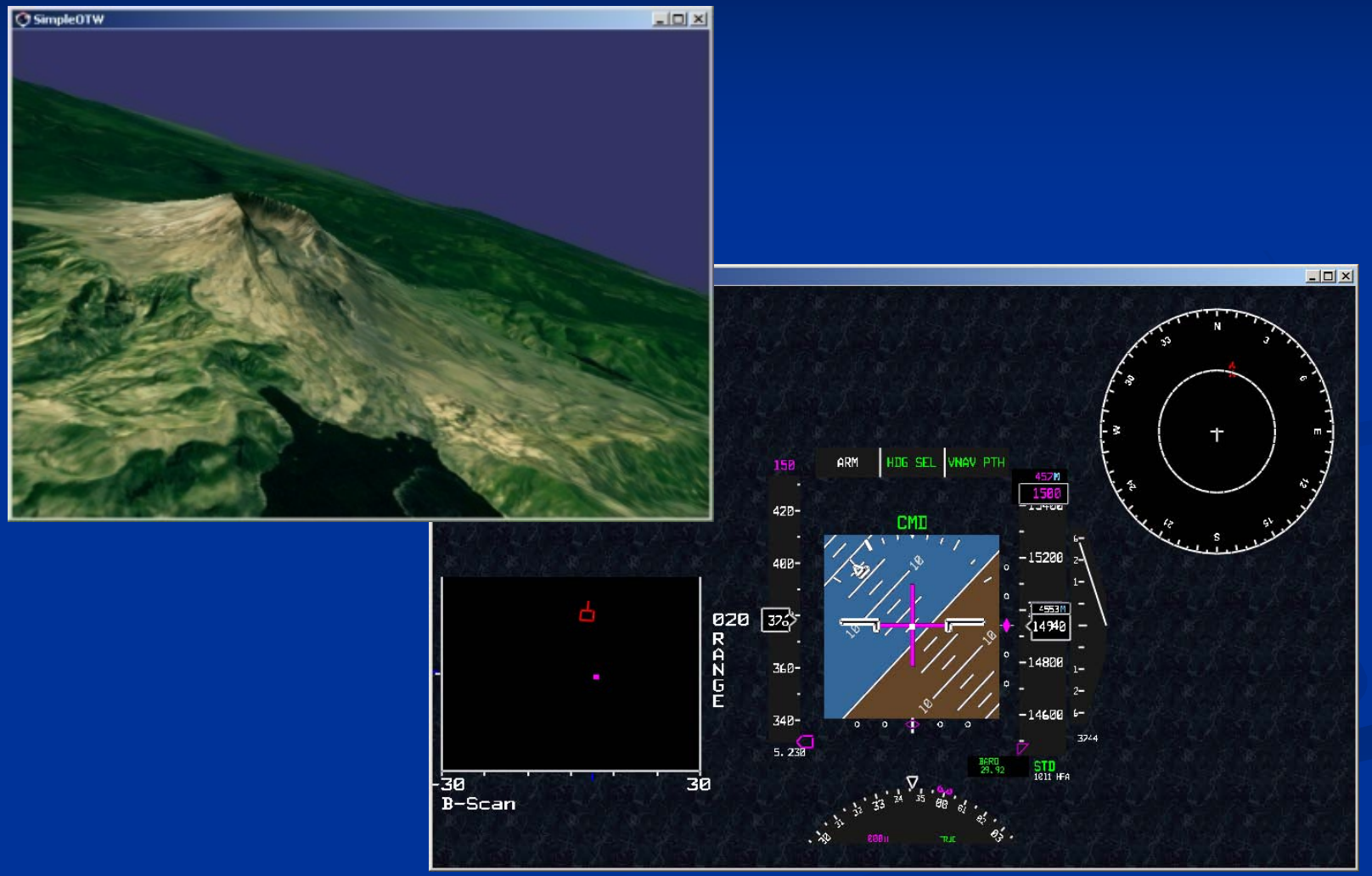






Radar Simulation

Example Simulation



Summary

- Virtual Simulation Characteristics
  - Real Time System
- Open Solutions
  - Visual Systems
    - Virtual Terrain Project
    - OpenSceneGraph
    - SubrScene
    - CIGI
  - Interoperability
    - DIS - IEEE Standard
    - HLA – “poRTIco project”
  - Dynamics Model
    - JSBSim
  - Simulation Framework
    - OpenEaagles

References

- "Design & Implementation of Virtual and Constructive Simulations Using OpenEaagles" by Rao, Hodson, Stieger, Johnson, Kidambi and Narayanan, 2007
- "Networked Virtual Environments: Design and Implementation" by Singhal, Zyda
- "Building Distributed Simulation Utilizing the EAAGLES Framework" by Hodson, Gehl and Baldwin, I/ITSEC 2006.
- "Real-Time Design Patterns in Virtual Simulations" by Hodson, Baldwin, Gehl, Weber, Narayanan

Backup Slides

Interoperability Pattern