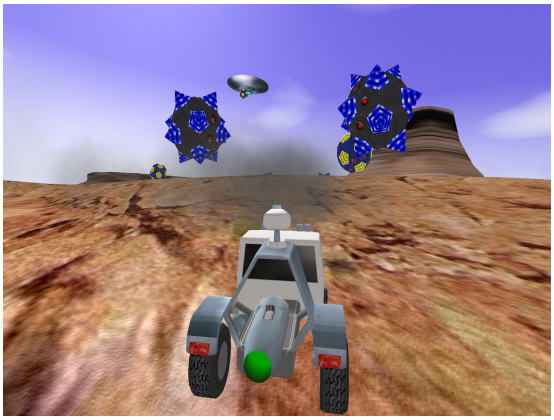
Building Networked Games With Delta3D Simulation Core

Alion Science and Technology
Naval Postgraduate School

Curtiss Murphy
cmmurphy@alionscience.com
Chris Rodgers
crodgers@alionscience.com
David Guthrie
dguthrie@alionscience.com
Brad Anderegg
banderegg@alionscience.com
Perry McDowell
mcdowell@nps.edu



I/ITSEC Tutorial (Nov 30, 2009)

Booth 835 - Alion
Booth 2663 - NPS

Tutorial Contents

- Introduction
  - The Problem & The Solution
  - Game Engines
- Tutorial Parts
  - Part 1 – Overview of Delta3D
  - Part 2 – Networking 101
  - Part 3 – Net Demo
  - Part 4 – Conclusion

TUTORIAL
PART 0

Introduction

Intro - Background

- Assumptions
  - Gaming is a valuable part of our training toolbox
  - Delta3D interests you because it is Open Source
- 5th Annual I/ITSEC tutorial
  - See previous tutorials/references (last slide)
  - Time limitations – topics covered briefly
    - Slides posted www.delta3d.org (Tutorials section)
- Audience - Technical
  - Software developer or manager
- Goal
  - Introduce networking essentials used to build a Delta3D Simulation Core game

Intro – The Problem To Solve

- The obvious stuff…
  - Your customer needs a networked 3D sim
  - Your project is willing to invest in development
- Where do you start?
  - Dozens of engines (proprietary and open source)
  - What if they don't meet your needs?
  - Can you customize?
  - Do you start from scratch?
- The solution …

Intro – The Solution

- Delta3D Simulation Core
  - Large library provides common M&S functionality
  - Ready to go – don't start from scratch
  - Open source
  - Widely used
  - **<u>Networking – HLA & Client/Server</u>**
- Let's back up a minute…

Intro - What is a Serious Game?

- Serious Game
  - Use of game technology for non-entertainment purposes (ex. Training)
- Why use Serious Games at all?
  - Experiential fidelity!
    - Dynamic Interaction
    - Engaging Immersion
    - Simple Interface
    - Interesting Decisions

Intro – What is a Game Engine?

- Visualization
  - Move around and 'see' the simulation – typically 3D
  - 3D Models (trucks, planes, tanks, soldiers)
  - 2D Textures (brick walls, satellite imagery, UI Icons)
  - Terrain (large or small, indoor or outdoor)
  - Shaders (detail mapping, specular highlights, bump maps)
- Behaviors
  - Moving and rotating in 3D space
  - Networking & Physics (collisions, gravity)
  - Character Animation (walking, running)
  - Weather (clouds, fog, rain, sun rise)
  - Particle Effects (smoke, explosions)
- Misc
  - User Interface (Heads Up Display)
  - Input (joystick, keyboard, mouse)
  - Sound (voices, explosions, music, ambient)
  - Tools (editors, export/import, level design)

Demo Time!

TUTORIAL
PART 1

Overview of Delta3D

Part 1 - What is Delta3D?

- Delta3D
  - Open Source Gaming Engine == FREE
  - Government maintained - Naval Postgraduate School (NPS)
  - Active community involvement
    - www.delta3d.org
    - ~ 2219 registered users, 15818+ Posts, 55+ Tutorials
    - Dozens of companies and organizations involved
- Specifically geared to the M&S community!
  - HLA, After Action Review (AAR), Large Terrains (Terra Page, OpenFlight), Learning Management System (LMS), 3D Simulations, Munitions, Entities

Part 1 - Legal Mumbo Jumbo

- Improved since last year!
- Most demos use the MIT license
  - MIT == do whatever you want
  - '… deal in the software without restriction'
- Delta3D & SimCore licensed under LGPL
  - Lesser GNU Public License
    - http://www.gnu.org/copyleft/lesser.html
  - Non-viral in nature. Applications built on top of Delta3D may retain a proprietary license.
  - Direct modifications to Delta3D & SimCore should be resubmitted

Part 1 - Delta3D Features (partial) (1)

- Cross Platform
- C++ API
- Game Manager
- Actors, Messages, & Components
- 3D Audio
- Graphics (30+ formats)
- OpenGL Shaders
- HLA Networking
- Client/Server Networking
- DIS Networking
- STAGE Level Editor
- Learning Management System (LMS)
- Character Animation
- Large Terrain support
- Artificial Intelligence
- AI – Planning & Pathing
- Tool - Character Animator
- Tool - Particle Editor
- Tool - Model Viewer

Part 1 - Delta3D Features (partial) (2)

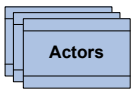
- Simulation Core
- NVidia PhysX™
- Bullet Physics
- Munition system
- Entity system
- Weather Integration
- Coordinate Conversion
- Dead reckoning & ground clamping
- Dynamic lighting
- Ocean Rendering
- After Action Review - Record/Playback
- Tasks/Objectives
- Binoculars, Compass, Night Vision Goggles
- Maps - Project Assets
- Python Bindings (partial)
- Unit tests (45,000+ lines)

Part 1 - Limitations of Delta3D

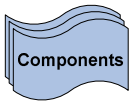
- Out-of-box experience – geared to engineers
  - How do you get started?
  - The number of libraries can be confusing
  - Limited pool of art assets (out of box)
    - Organizations don't always share
- Complex build procedure
  - Availability of source plus reuse of other open source libraries makes it complicated to compile
  - Docs & tutorials available
  - Multiple repositories & build steps
- Limited lighting model & shadows
  - Can be solved per project

Part 2 – A Quick Review (1)

- Quick review of Delta3D architecture
  - See Delta3D tutorials or prior I/ITSEC presentations
- Actors
  - Objects in the world that we care about
    - Ex: tanks, trees, missiles, tasks
  - Can be created in code or with a map (xml)
  - Receive some messages, by request

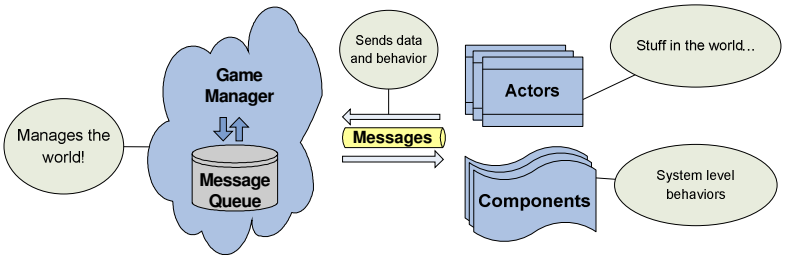


- Components
  - High level behaviors
  - System level stuff - networking, logging, HUD, input
  - Receives all messages from Game Manager

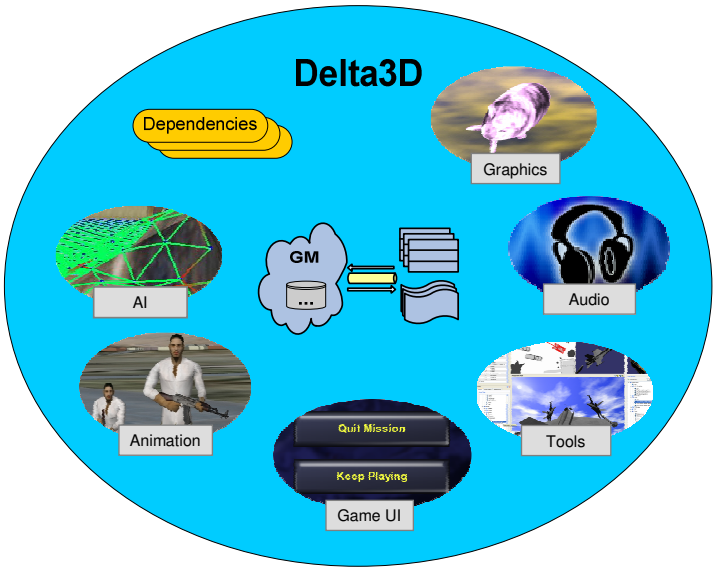


Part 2 – A Quick Review (2)

- Game Manager
  - Manages all Actors and Components
  - Primary job is to route messages
  - Controls time and ticks (via messages)
- What it looks like:



Delta3D Overview



TUTORIAL
PART 2

Networking 101

Theory & Solutions

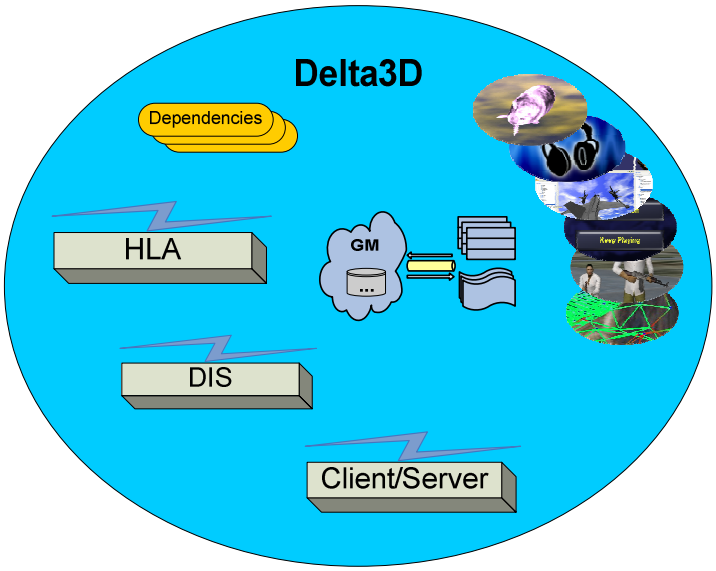
Part 2 – Network Types (1)

- Delta3D supports 3 types of networking
  - HLA, DIS, and Client/Server
- HLA - High Level Architecture (dtHLA)
  - Tested with RTI-S & RTI-NG Pro & CERTI
    - Run Time Infrastructure (RTI) not provided
  - “Connectionless” – no server
  - Works perfectly with Simulation Core (later…)
  - RPR 1 & RPR 2 ready
  - Configurable settings via XML
  - Robust - tested at many installations
    - dtHLA is the most proven Delta3D network layer

Part 2 – Network Types (2)

- Client/Server (dtNet & dtNetGM)
  - Traditional game networking
    - Server has control. Routes messages to clients
    - Client connects to server. Must have a server
  - GM Messaging works well with Client/Server
  - Moderately proven - used at multiple organizations
- DIS – Distributed Interactive Simulation (dtDIS)
  - DIS is the precursor to HLA. “Connectionless” also
  - Partial implementation
  - Least used Delta3D network layer

Delta3D + Networking



Part 2 – Entity Ownership

- Local
  - Entities/actors that you own (i.e. ownship)
  - Real time & 'live'
    - You know exactly where 'you' are
- Remote
  - Someone else's entities
  - In M&S, most entities are remote
  - In playback, **all** entities are remote
- Same actor class
  - Remove & Local often use the same actor
  - Game Manager was built with this in mind

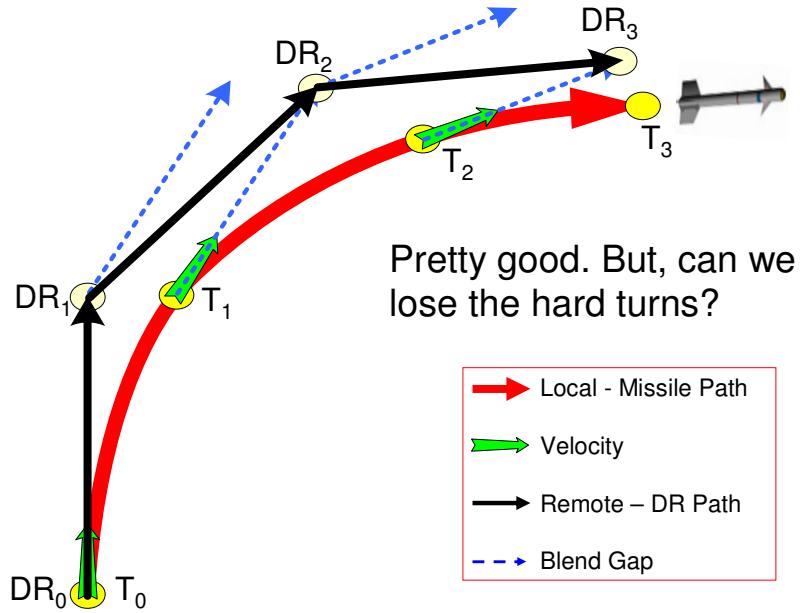
Part 2 – Publishing - Entities

- Client control (aka 'distributed' network)
  - First rule – no one has 100% ground truth
  - Clients feel 'lag' free
- Who publishes?
  - Local entities
- What to publish?
  - Sometimes ALL actor data (i.e. 'heartbeat')
    - Actor type, name, id, damage state, …
    - Helps with late joiners and dropped packets
  - More often, partial update
    - Position, velocity, rotation
- When? How Often?
  - Hint – It's NOT every frame
    - CPU speed is much higher than network

Part 2 – Dead Reckoning (DR)

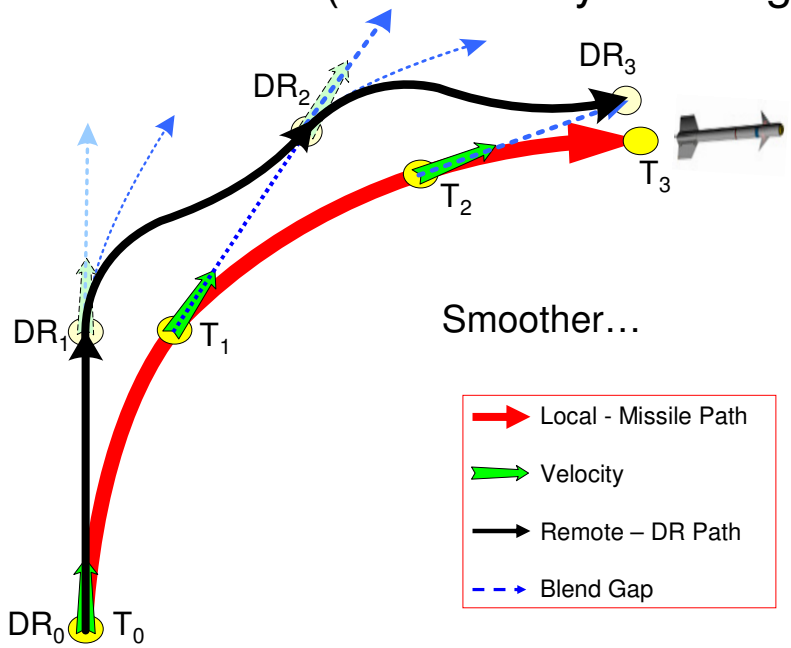
- Dead Reckoning
  - Between updates - estimate position based on last known pos and velocity
    - Use linear velocity and angle
    - Optional – acceleration and angular velocity
    - Blend to 'corrected' location
- How?
  - Remote Entity – DR actor each frame. Then, blend values on updates to fix 'gaps'
  - Local Entity
    - 1) Calculate DR of self
    - 2) Compare DR vs current pos
    - 3) Publish when threshold exceeded
      - Example - 0.5m or 3 degrees
      - Max publish rate (ex 3-5 times/sec for virtual sims)
  - Thresholds & rates usually agreed in advance

Part 2 – Dead Reckoning In Action



Pretty good. But, can we lose the hard turns?

Part 2 – DR (with velocity blending)



Part 2 – Dead Reckoning (cont)

- What about Rotation?
  - Same exact problem all over again!
  - Similar solution, but more forgiving ☺
  - Cannot guess (vehicles go back/sideways)
- Acceleration and Angular Rotation
  - Can makes a big difference (affects curve)
  - But - difficult to know – future knowledge
- Difference between M&S and 3D vis
  - M&S is often less visual – ‘close’ maybe OK
  - 3D games and sims are very visual
    - Humans perceive discontinuities easily

Dead Reckoning – DEMO!

Part 2 – Ground Clamping

- Ground Clamping
  - Drop remote entities to the ground
- Why do this?
  - Solves terrain correlation issues
    - Minor variations explode in 3D
    - Unusual for 3D to perfectly match 2D
  - Solves DR issues (i.e. floating cars)
- How?
  - Variable level of detail (better up close)
  - Use multi-point, single point, or intermittent
  - Note - can be expensive with many entities

Part 2 – Damage Model

- "I killed you" … "No you didn't!"
  - Who decides?
  - No 'right' way - as long as everyone agrees
  - Here's how we do it – HLA & DIS method
- Gentleman's agreement
  - Direct Fire (50 Cal, 9mm)
    - If you shoot, you decide if you hit, using the DR location. Publish a 'shot hit' message
    - Ownship – owner of a local entity decides the amount of damage and the effect
  - Indirect Fire (Grenades, bombs)
    - Area of effect damage. Simply publish Fire message with explosion location
- Security
  - Add server side validation
    - Many games use this hybrid approach
  - Training games don't care – who has time to cheat?

Part 2 – Physics & Networking

- The problem
  - You control your local entity with physics
  - You collide with a remote vehicle
  - What should happen?
- Huge Nastiness!!!
  - Options
    1) (BAD) Do nothing. Drive through other entities. Sometimes the only solution ☹ (ex people).
    2) (BAD) Perform all physics on the server. Doesn't scale well, not distributable, doesn't 'feel' responsive.
    3) (GOOD) Do physics on yourself only. Assume remote does also. Hope it works out.
    4) (BETTER) Simulate remote entity temporarily. Assume remote does also. Blend the two after initial collision.
    5) (BEST) Take ownership of remote object temporarily. Simulate it. Publish remote pos and then release control.

    * See GDC 2009 - Glenn Fiedler – www.gafferongames.com
- Net Demo uses #3 - Not an ideal solution

Part 2 – Other Considerations

- Initialization – what to publish
  - Map/terrain – can be pre-defined or published
    - Net Demo does both. Map defined, terrain published.
- Articulations add further complexity
  - Relative - DOF - Degrees of Freedom
  - Partial publication (i.e. only yaw or pitch)
- Control states
  - Getting inside another vehicle
  - Controlling a weapon turret
- The Butterfly Effect
  - Items that are not published – recreate locally
  - Ex. Remote Tracer visualization

TUTORIAL
PART 3

Net Demo
Example App

Part 3 – Net Demo (1)

- Ready to go - example networked game
  - Completely functional, physics based vehicle simulators
    - Open Source – all source available to guide developers
  - Demonstrates proper use of Simulation Core
  - Multiplayer networking - client/server
  - Physics - uses Nvidia PhysX™ or Bullet
    - Forces from munitions, weather, driving
    - Individually modeled bullets
- Two vehicle types
  - Four Wheel Vehicle – (ground)
  - Hover Vehicle – (flying)

Part 3 – Net Demo (2)

- Non-trivial examples using SimCore
  - Weapons, munitions, and damage models
  - Vehicle articulations
  - AI for steering, pathing, and planning
  - Custom Actor Library & Game Entry Point
  - GameLogicComponent
    - Game states and transitions, map loading, connectivity, actor creation.
  - Heads Up Display Component (UI)
    - Uses several controls - Crazy Eddie GUI (CEGUI)
  - Input Component
    - Key inputs and motion models
    - Controlling vehicles and weapons

Part 3 – Sim Core Basics

- Simulation Core
  - Large library of behaviors needed by most Sims
  - Sits on top of Delta3D
  - Part of the Delta3D-extras repository
- Essential M&S features
  - Entities – networking and publishing
  - Munitions System
  - Terrain & Weather Integration
  - Dynamic Lights
  - Art Assets (shaders, models, sounds, …)
  - Demo apps – Net Demo & Driver Demo
- Stealth Viewer…

Part 3 – Stealth Viewer

- Part of Simulation Core
  - View any sim core app!
- Commodity Stealth features
  - 3D View
  - Entity Searching
  - Binoculars, Help, NVG, Compass
- HLA & Client/Server Configuring
- AAR - Record and Playback
  - Built in, stand-alone AAR
  - Record your sim as it happens
  - Replay from any angle/location

Demo of StealthView

TUTORIAL
PART 4

Conclusion

Organizations Using SimCore

- A partial list of some of the known users…
- USMC - PMTRASYS & TECOM
  - Deployable Virtual Training Environment (DVTE)
- Navy - NAVAIR – Manned Flight Sim
  - MH-60R Tactical Operational Flight Trainer (TOFT) #3
- Navy – Naval Service Training Command
  - Damage Control & Flooding Trainer
- Army – PEO STRI
  - Common Sensor Model including NVG
- Army - West Point – Stealth Viewer
- Joint Forces Command – Stealth Viewer
- JIEDDO – Counter IED Trainer – Tactical Gaming
- Plus many companies…

References

- www.delta3d.org – Best Reference Material!
  - Tutorials, Knowledge Base, Forums
- I/ITSEC 2008 Tutorial
  - **"Building Training Games with the Delta3D Simulation Core"**
- Game Programming Gems 7
  - "**Support Your Local Artist – Adding Shaders To Your Engine**"
- I/ITSEC 2007 Tutorial
  - "**Creating Low-Cost Game-Based Trainers with Delta3D**"
- I/ITSEC 2007 Paper (Honorable Mention)
  - "**Are You Ready? The Open Technology Development Challenge"**
- I/ITSEC 2006 Tutorial
  - "**Building Game-Based Trainers with the Delta3D Game Manager**"
- Game Programming Gems 6
  - "**Exposing Actor Properties Using Non-Intrusive Proxies**"

Final Demo and Questions

THE END

Building Networked Games With Delta3D Simulation Core

I/ITSEC 2009

Thank you for attending!

Please visit our booths to learn more about Delta3D!