Developing games using Java

Presented at CS196
Brown University
March 8, 2006

Brad Myers, President
Galactic Village Games, LLC
bmyers@galactic-village.com
Who is Brad Myers?

- Brad’s “Brag Sheet”
  - Founder and president of Galactic Village Games, LLC
  - Hard-core gamer since age 14.
  - Has been programming games on-and-off since 1985.
    - Galactic Village (in-progress)
    - AD&D Pool of Radiance
    - Sub Battle Simulator
    - Orbiter
  - Has been designing Java-based systems since 1996.
Who is Galactic Village Games?

• The Company
  • Founded in 1998
  • Core team of 3 people
  • Some contractors for artwork and sound effects.
  • Combined total of 25 years experience in games industry
  • Combined total of 200? years in computing.
  • Self-funded
    • Everybody has a day job
    • Work mostly nights and weekends
Our Game: Galactic Village

- Galactic Village is a 3D online strategy game, where thousands of players participate in running the galaxy by balancing religion and science, warfare and diplomacy, exploration and commerce as they strive to expand their spheres of influence, keep their people happy, and conquer space.

- When you play Galactic Village you are an immortal demigod with the ability to interact with the “little people” of the galaxy. But you don’t just move pieces around -- you get to participate in a vast interconnected society and to help it grow and prosper over time.
Our Game Design Goals

• Strategy first
  • Pacing -- give players time to think
  • Lots of different strategies, ways to compete and cooperate
  • Different ways to succeed

• Large online world
  • Persistent world with evolving content
  • Huge scope: an entire galaxy, down to individual people!

• Complex game, simple view
  • Needs to be challenging enough to last several years
  • Needs to be simple enough to start playing immediately
  • Game play should be 50% visual, 50% mental
Our Software Design Goals

- Modular systems
  - Extensible architecture that can adapt to growth
  - Model-View-Controller approach to software
- Lots of Data
  - Design for 500K subscribers with 100 - 200 units (1 billion units!)
- Lots of Parallelism
  - Concurrent AI processing
  - Simultaneous order processing from thousands of players
- Complex simulation and data visualization
  - Economic, religious, geopolitical, and climate systems
  - Spherical planets
Our tech vision – 2, 5, 15 yrs

• 2 years
  • Speech input with much better recognition
  • Better AI for planning and autonomous behavior
  • Procedural dynamic terrain generation

• 5 years
  • Totally cross-platform (including game consoles)
  • 3D stereo graphics and VR displays in the mainstream
  • Personal AI tutors

• 15 years
  • Fully immersive virtual reality
  • Omnipresent wearable computer agents
  • AI that passes Turing test
Where are games headed?

(In my humble opinion)

- Fully immersive interactive movies
  - Hollywood effects and stories
- Casual play-anywhere toys
  - Integrated cellphone, PDA, and GPS
- Experiential virtual communities
  - Hybrid real-life and virtual reality cross-over
- Personal storytelling kits
  - Intelligent life-like autonomous agents who will act out your directions.
Game technology will be used in:

- Serious training games
  - Military
  - First responders
  - Socio-political simulations
- 3D visualization of complex data
  - Financial systems
  - Medical systems
  - Climate systems
- Education
Languages for games

- C++
- Java
- C
- Assembler
- Perl
- Python
- Pascal
- Basic
- <Insert your favorite language here...>
Why not use Java?

- Perceived as “too slow for real games”
- No existing commercial game engines yet
- Memory-hungry
- Doesn’t run on game consoles, yet
- Write-once, test everywhere
Why use Java?

- Rich platform
- Single source code
- Lots of open-source utilities
- High productivity (8-10x C++)
- Cross-platform
- Standardized API
- Network-friendly
- Cross-platform 2D & 3D toolkits
What Java APIs are we using?

- Server
  - JDBC
  - JMS
- Client
  - Java Swing
  - Java 2D
  - Java 3D
  - Java Sound
  - JDBC
  - JMS
What other code is being used?

- Open source code / tools
  - J3D.org
    - Overlays
    - Textures
    - Geometries
  - Milkshape Loader
  - J3DTree
  - JOrbis
  - ExplicitLayout
- Databases
  - Postgres
  - HSQLDB
Commercial Tools

- SwiftMQ
- Genesis F/X particle system
- Milkshape
- 3D Studio Max
- Maya
- Poser
- Bryce 3D
- Adobe Photoshop
- Corel Draw
- Optimizelt
- SimBionic
What GUI approach do we use?

- Prototype #1: Java 2D & Swing
  - Quick development for proof-of-concept
- Prototype #2: 100% Java 3D
  - Decided 3D was needed to meet player expectations
  - Learning curve was short but steep
- Current: 50% Java3D & 50% Swing
  - Combination of both approaches looks good and allows quick development
- Future: 80% Java3D
  - 3D good for complex visualization, immersion
  - 2D good for lists and simple visualization
Galactic Village screen shot
What's our network layer?

- **Prototype #1: Standard packet stream**
  - Simple to develop, but doesn’t scale well.

- **Prototype #2: Pure JMS**
  - Easy pub/sub and peer-to-peer, communication

- **Current: Custom serialization w/JMS**
  - Better control over messaging
  - Compression helps bandwidth usage

- **Future:**
  - Encryption
  - java.nio
  - Possibly some movement info using UDP
How about the database?

- Data is everything
  - Almost everything in GV is data-driven (89 tables)
  - Standard SQL statements and data types
- Server database must be industrial strength
  - Fast and secure
  - Must scale to handle thousands of simultaneous users
  - Must be quickly recoverable
- Client database must be lightweight and cross-platform
  - Must be able to deploy and upgrade seamlessly
  - Must be fast for queries
Our SDLC: Prototype and Refine

• Every aspect of the game design and game code has gone through several iterations
  • Game Design
    • More than 5 iterations on planet representation!
    • 5 iterations on primary game rules and mechanics
  • Code
    • 4 iterations on GUI code
    • 3 iterations on shortest-path algorithms
    • 3 iterations on AI design
    • 2 iterations on order-processing

• Java is great for prototyping!

• This is a work-in-progress -- refactoring never ends!
Was it a good experience for us?

- What went well
  - Prototyping in Java is fast!
    - Initial proof-of-concept took 2 - 3 months.
  - Game design played to Java strengths
    - Not an accident!
  - Lots of OO design resulted in adaptable infrastructure

- What went less well
  - Getting from proof-of-concept to current design.
    - Several iterations of Java3D experimentation
    - Java3D is very powerful but has a steep learning curve
  - Sometimes design was too OO and didn't scale well
    - Example: Paths worked fine for 400 tiles, blew up with 16,000.
Where did we innovate in tech?

- Persistence management
  - All data objects know how to read/write themselves to database.
- Custom serialization
- 3D planets
  - Wrote our own 3D tile-based coordinate system
- Mouse and keyboard controls
- Swing components
  - Customized GameFrame hierarchy
  - Translucent 3D overlays
What would we do differently?

- Start smaller & grow
  - Start with small, playable game
  - Finish it and launch it.
  - Grow based on player feedback.
  - Target game #2 to be the "killer app"
- Get artwork and media specs decided early
- Choose only one area to innovate
Lessons learned

- Java3D and other APIs have a steep learning curve
- Design for the big picture but implement iteratively
- Frames-per-second is not a Holy Grail!
- Don't innovate everywhere
  - Choose where to break new ground
  - Avoid "million-dollar" items
  - Make innovations truly excellent
Some handy URLs

- Galactic Village Games
  - www.galactic-village.com
- Java resources
  - www.j3d.org
- Java Gaming resources
  - http://www.java-gaming.org
- Java developer community
  - http://www.java.net