everVoid

(temporary logo)

Software Development Project

Valentin Bonnet Pier-Luc Gagnon Etienne Perot

COMP 361 - In-house project
McGill University
Montreal, Canada

Monday November 1st, 2010

Table of contents

Project everVoid

- Team:
  - **Bonnet**, **Valentin** - valentin.bonnet@mail.mcgill.ca
  - **Gagnon**, **Pier-Luc** - pier-luc.gagnon@mail.mcgill.ca
  - **Perot**, **Etienne** - etienne.perot@mail.mcgill.ca
- Contact:
  - **Trac**: http://evervoid.biringa.com/trac
  - **Subversion repository**: http://evervoid.biringa.com/svn
  - **IRC channel**: irc://irc.freenode.net/everVoid (#everVoid on freenode)
  - **Guest access**: guest / comp361evervoid - (where relevant)
- Stream: **In-house**, game project
- Clients:
  - **Joseph Vybihal** - jvybihal@cs.mcgill.ca
  - **Jörg Kienzle** - joerg.kienzle@mcgill.ca

---

everVoid is a video game set in space in which players vie for galaxy domination. The game will be implemented in Java, mixing both 3D and 2D graphics. Game play revolves around planets, which yield income and allow players to advance through research “ages”. When a player no longer controls a planet, he is eliminated. Players fight over the control of solar systems, each of which contains a “random” number of planets. Spaceships of various types can be built in order to harass enemy planets, protect friendly ships, or bolster your economy. The economy is based on money, metal and gas. Metal is abundant on asteroids and is used primarily for building ships and buildings. Gas is found on gas planets (much rarer) and is used in research. Terran planets provide a small supply of both metal and gas as well as increasing population, which is needed to build ships. A more detailed explanation of game rules is provided lower in the document.

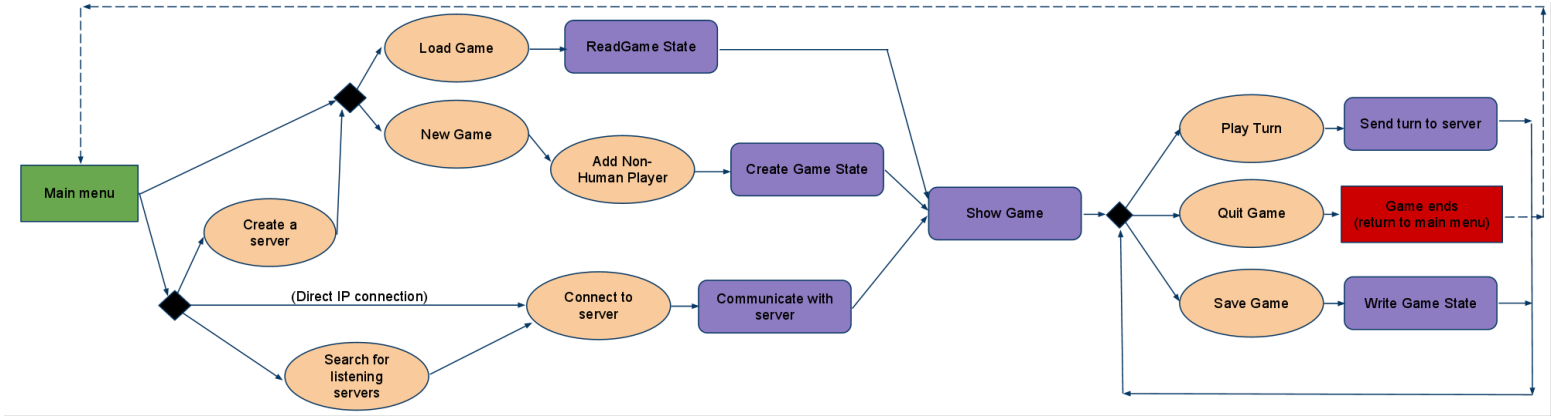
Game Overview

The game will have the following attributes:

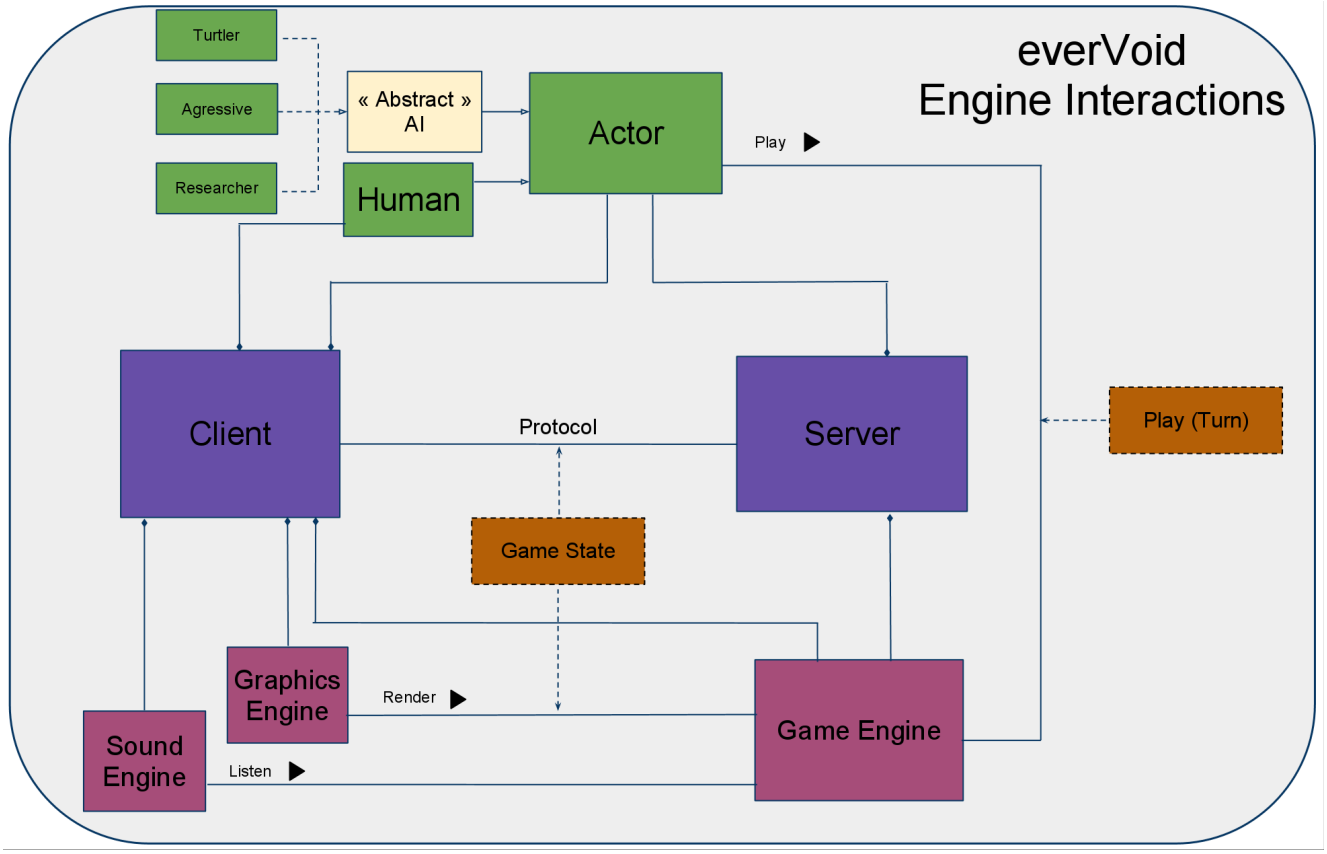
1. Basic turn-based play style
    - All turns handed in and then worked on
2. The objective of the game is to annihilate the enemy(ies)
    - Alternative win, through “wonder”-like structure
3. Three main views:
    - Galaxy (3D)
        - Contains the solar systems and connecting wormholes
    - Solar System (2D)
        - Contains planets and ships
        - Square grid
    - Planet (2D)
        - Static menu interfaces
        - Lists buildings and actions of the planet
4. Resources:
    - Money
        - Based on population (tax)
    - Metal
        - Used primarily for ship construction
        - Found on asteroids
    - Gas
        - Used primarily for research
        - Found on gas planets
5. Ships
    - Permanently “deployed” in space (cannot be docked)
    - Each ship specialised for its duty
6. Buildings
    - On planets, provide resources, ship-building capabilities and research levels
7. Research
    - Four ages/”tiers” that players can go through
    - Each tier unlocks new research levels
    - researching the levels improves the appropriate category (Shields, lasers, speed....)
8. Jump between solar systems are through “wormholes”. Radiation must be expended, but has a (slow) regeneration rate, which can be improved by closing in to a sun or by research.

Project Specifications

- Process
  - Incremental
- Protocol
  - All communications done through TCP/IP
  - Clients and Server are synchronized with a compressed JSON string containing the Game State (serializable)
  - Game is managed by a server program, which may run on a player's machine or as a dedicated listening server
  - Gameplay logic checks are done at the server side and at every client's side
  - Possible game types:
    - LAN game
    - Online game (a central server would host a list of online servers)
    - Direct IP connection
- Saving
  - JSON string saved to a file, compressed
  - Save files and communication should be encrypted. (or codes).
- Gameplay
  - Turn based: All turns handed in, engine runs when all players have submitted their turn
  - Research added
- Map layout
  - **Galaxy**: Random distribution of solar systems in a 3D graph
  - **Solar system**: Square grid
  - **Planet menu**: Menu, possibly with image in the middle (standard image?)
- Graphics
  - Views:
    - **Galaxy**: Overview of all solar systems (3D)
    - **Solar system**: View of a single solar system, and all its planets. Has fog-of-war (2D)
    - **Planet menu**: View of a single planet, and all its buildings (2D)
- AI Players
  - Multiple AI profiles (behavior) and difficulty
- External Engines
  - jMonkeyEngine - 3D graphics, 2D graphics, audio, input


Main menu
Load Game
ReadGame State
New Game
Add Non-Human Player
Create Game State
Create a server
(Direct IP connection)
Connect to server
Communicate with server
Search for listening servers
Show Game
Play Turn
Send turn to server
Quit Game
Game ends (return to main menu)
Save Game
Write Game State

Engine interactions



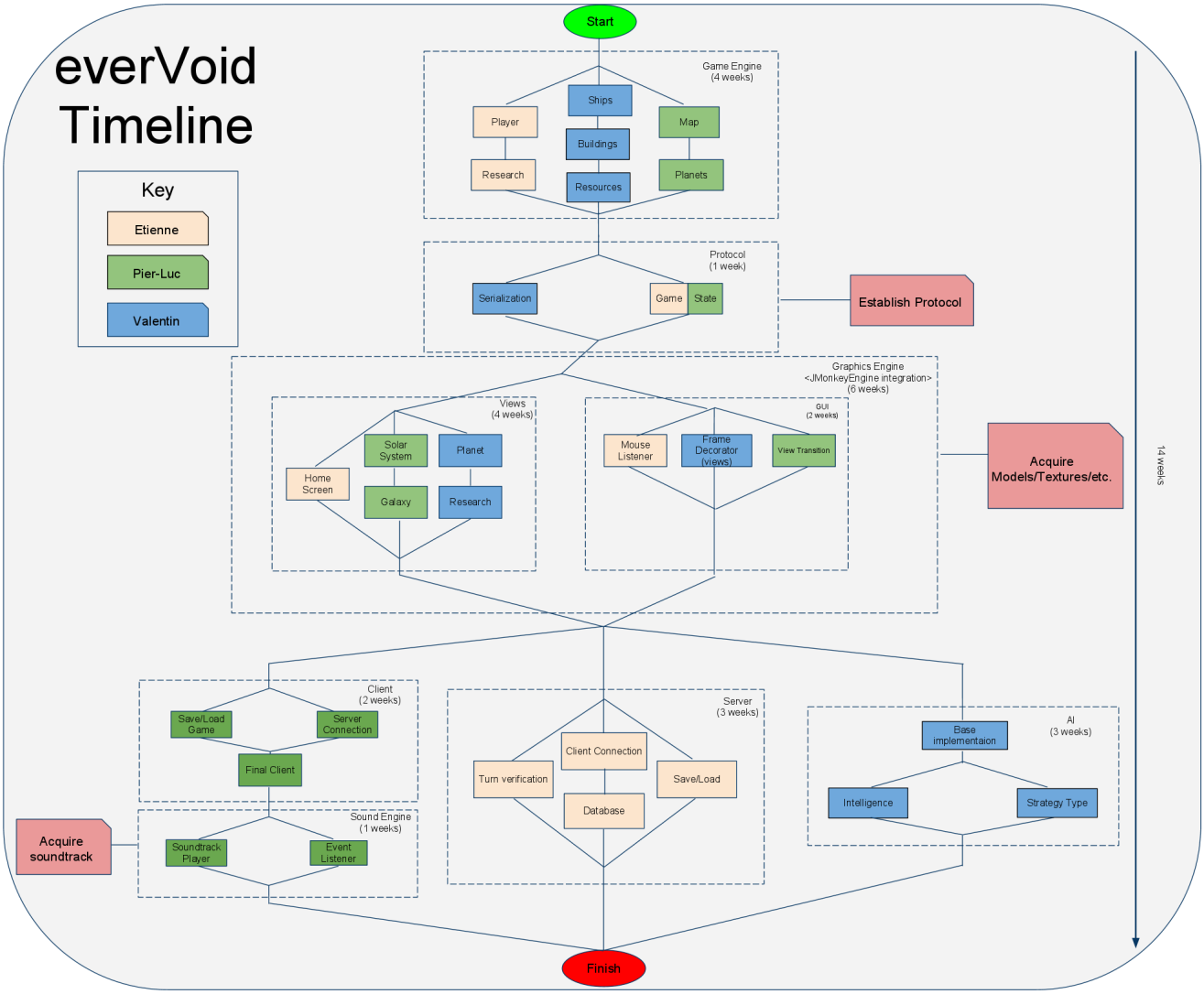
Engineering Methods

Project everVoid will be built using the Incremental development. We chose this development method for the following reasons. It makes full use of team member strengths by assigning modules to team members with prior experience in those domains. This is particularly useful due to the small size of the team, and the scope of structure challenges we will face. The development model also reduces the chance of propagating bugs from module to module by clearly delineating module borders. Therefore a module can, once tested, be used an integrated within the project with minimal effort (assuming test suits and correct implementation). Finally, the model reduces the gravity of design oversights done early in the project. Should our module design be flawed in any way (inexperience or incomplete implementation of the requirements) the model allows us to easily change the module with minimal impact on the rest of the project. Given that the entire team is new to game development, the wiggle room granted by this choice is a huge advantage of this model.

On a more personal note, this development model allows the team to acquire a deeper understanding of their specific modules. For educational and integration reasons, we would rather have well functioning modules that we understand well than a globally functional but messy implementation.

Timeline

Enlarged version available at:
http://is.gd/gymHw



Formal Phase Breakdown

Activity	Time Estimate (days)
Milestone 1: Game Engine	**12 weeks**
Step 1.1: Player *Activity 1.1.1: Game correctness* *Activity 1.1.2: Associations* *Activity 1.1.3: Point attribution*	6
Step 1.2: Ships *Activity 1.2.1: Movement* *Activity 1.2.2: Health* *Activity 1.2.3: Cost combat*	12
Step 1.3: Research *Activity 1.3.1: Research tree* *Activity 1.3.2: Technology tier* *Activity 1.3.3: Cost*	6
Step 1.4: Buildings	6
Step 1.5: Resources	3
Step 1.6: Planets	6
Step 1.7: Map *Step 1.7.1: Default maps*	9
Milestone 2: Protocol implementation	**3 weeks**
Step 2.1: Game State	9
Step 2.2: Serialization	3
Milestone 3: Graphics Engine	**18 weeks**
Step 3.1: Graphical views *Activity 3.1.1:* HUD *Activity 3.1.2:* Solar system view *Activity 3.1.3:* Galaxy view *Activity 3.1.4:* Planet view *Activity 3.1.5:* Research view	 6 12 12 9 6
Step 3.2: GUI *Activity 3.2.1: Main menu* *Activity 3.2.2: Lobby/matchmaking UI* *Activity 3.2.3: Client configuration menu*	 6 12 6

Deliverable 1: Game with Graphics	
Milestone 4: Server	**3 weeks**
Step 4: Server *Step 4.1: Listening server* *Step 4.2: Game state operations* *Step 4.3: Game rules checking/sanitization*	 2 4 6
Deliverable 2: Standalone Server	
Milestone 5: Client	**3 weeks**
Step 5.1: Client *Step 5.1.1: Communicate with server* *Step 5.1.2: Integrate graphics engine* *Step 5.1.3: Integrate sound engine*	 3 6 3
Deliverable 3: Client - Server interaction	
Milestone 6: AI	**3 weeks**
Step 6: AI *Step 6.1: Basic AI (common behavior)* *Step 6.2: AI subprofiles* *Step 6.3: Adjustable hardness*	 6 3 3
Deliverable 4: Fully functional Game	
Total	**42 weeks**

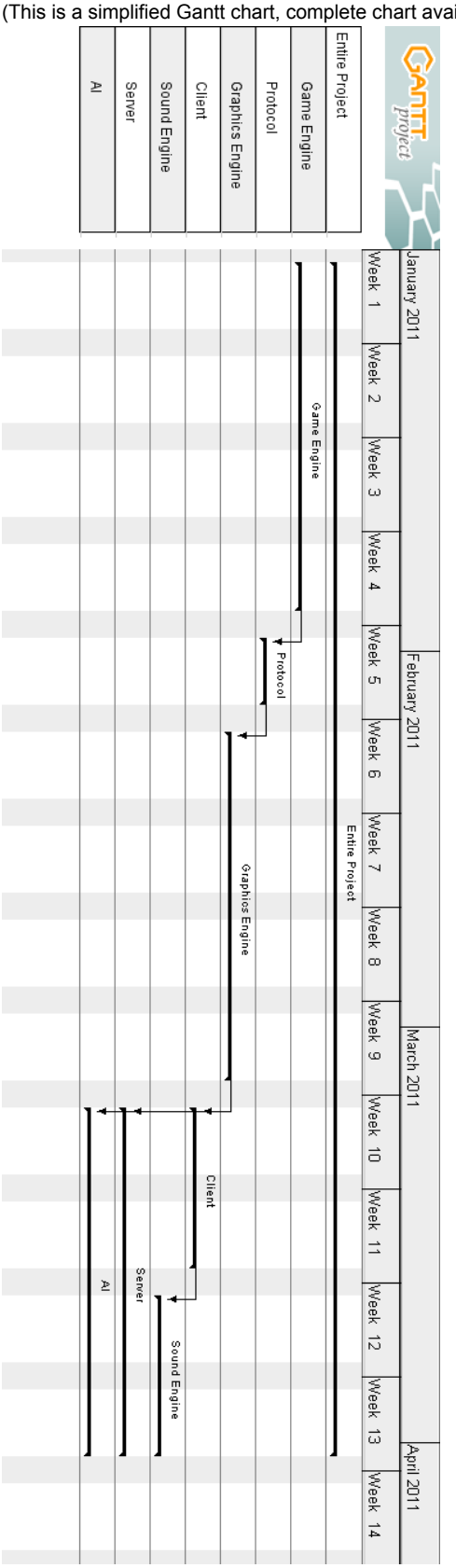
Notes:

1. Weeks are assumed to have 4 workdays
2. 42 weeks total. we have three programmers, so this is a 14 week project.

Milestone	Description	Dependency
Milestone 1: Game Engine	Implementation of the game rules, move validity, etc.	
Milestone 2: Protocol implementation	Establish how the protocol will work: Client-Server communication, master server list, what information should be passed, what signals should the server respond to, etc.	M1
Milestone 3: Graphics Engine	Graphics rendering and graphical user interface implementation.	M1, M2
Milestone 4: Server	Standalone, listening, no-graphics server.	M1, M2
Milestone 5: Client	Integrate the Game Engine within the Graphics Engine; then add a Sound Engine.	M1, M2, M3
Milestone 6: AI	Implement intelligent non-human players able to interact with the game engine and the server using the defined protocol.	M1, M2

Deliverable	Description	Dependency
Deliverable 1: Game with Graphics	A working game client with a graphics engine and a sound engine	M1, M2, M3
Deliverable 2: Standalone Server	A standalone listening server to which clients can connect	M1, M2, M4
Deliverable 3: Client - Server interaction	A full client+server package, with the client being able to run a server on the same machine as itself	M1, M2, M4, M5
Deliverable 4: Fully functional Game	Fully functional game, still needs testing for gameplay.	M1, M2, M3, M4, M4, M6

(This is a simplified Gantt chart, complete chart available at http://evervoid.biringa.com/images/gantt.png)



Project Modules

- Game Engine
    - Resource management
    - Research management
    - Unit management
    - Building management
- Graphics Engine
    - Main menu
        - Lobby interface
        - Matchmaking interface
        - Save/Load dialogs
    - Galaxy View
        - Solar Systems, Wormholes
    - Solar System View
        - Props (Ships, Planets, pretty art things, ...)
    - Planet view
        - Buildings
        - Current Action Progress
    - Research view
        - Progress
        - Research Tree
    - Models/sprites/art
        - Logo, UI elements
        - Ship, Building, Planet, Star sprites and models
- Protocol
    - Client (Uploading saved games to server, auto-reconnect)
    - Server (Resuming saved games, ping clients and notify players of drops
    - Master server
        - Server list database (Purge list regularly)
    - Game state synchronization:
        - State diff (sending only differences)
        - State hash (confirmation)
        - State resolver (movement conflicts)
- Sounds
    - Soundtrack
    - Sound effects (Explosions, lasers, etc)
    - Game milestones announcements (research finished, tier reached, planet destroyed, game won...)
- Artificial Intelligence
    - Type
    - Difficulty

Resource Requirements

- Graphics:
    - Logo, icon, splash screen
    - Ship sprites, buildings sprites, planets sprites
    - 3D planet models
    - Wormhole effects
    - UI design
    - Icons for buttons, research, etc
- Audio:
    - Soundtrack (main menu music, in-game background music)
    - Ship sound effects (firing, explosion, dropping, building, moving)
    - Planet sound effects (construction, explosion, resource gathering)
    - Announcer (game events, research complete, player dead)
    - UI sounds (button click, game paused, new message blip)
- Server:
    - Always-on master server (holds list of online servers)
        - Requires Apache and MySQL
    - “Official” dedicated servers
- Skills:
    - jMonkeyEngine expertise/learning curve
    - Knowledge of SQL
    - Knowledge of Java

Cost estimate

- We will be using Java with jMonkeyEngine and MySQL, all of which are free
- Audio/graphics royalties if necessary
- Estimated cost range: $0 - $100

Risk Estimation

Risk	Component	Severity	Fix Time
Time estimates are off / Workload too big - cannot finish project	Whole project	Critical	N/A
Learning curve is too important for external packages (jMonkeyEngine)	Game client	Major (if requires an engine change)	3 weeks
Server takes too long to process turns	Server	Major	1 week
Game play is unbalanced (too hard or too easy) / Game is not fun	Gameplay	Major	1 week
Game is too similar to other games; no competitive advantage	Gameplay	Minor	N/A
Cannot find free soundtrack / graphics	Audio	Minor	Do not fix
Unappealing Graphics/sprites	Graphics	Minor	Increases costs, hire an artist