Battlestar

Documentation

Team 11
David Gay
Scott Gunther
Yigit Katkici
Nathan Osborn

May 21, 2012
Revision 1.0

Table of Contents

Change Log

Preferrably, changes are referenced through the git repo log on GitHub. However, this project also requires that we maintain this change log, especially since most team members don't know git. This log lists significant project changes.

Commit history: **https://github.com/oddshocks/battlestar/commits/master/**

Member	Date	Message	Version
David Gay	April 11, 2012	Created repo	Design
David Gay	April 17, 2012	Created design doc	Design
Nathan Osborn	April 19, 2012	Added to design doc	Design
Yigit Katkici	April 22, 2012	Added to design doc	Design
David Gay	April 22, 2012	Edited and added to design doc	Design
David Gay	April 24, 2012	Heavy GUI work, continued program setup	Interface
David Gay	April 26, 2012	Added GUI panels and client networking, reformatted readme and todo	Interface
David Gay	April 28, 2012	Working client/server communication, work on command protocol functionality	Networked
David Gay	April 30, 2012	Working chat communication to server	Communication
David Gay	April 30, 2012	Corrected hard-coding of constants, fixed communication problem	Communication
David Gay	May 1, 2012	Working client/server messaging, all commands and communication working, stylized ViewZones, restructured and updated MPP file, minor changes to BattleConstants	Messaging, Commands
David Gay	May 3, 2012	Made it so clients can ready up, started creating ships	Readying
Scott Gunther	May 5, 2012	GUI changes, images added for icons	New GUI
David Gay	May 8, 2012	Created rest of ship classes, made	Ships Ahoy

		client add specific ship classes to grid, worked on ship icons	
David Gay	May 15, 2012	Added StatPanel accessor	Ships Ahoy
Scott Gunther	May 15, 2012	Worked on game functionality, allowing for race selection, and getting ships moving	Early Game
Yigit Katkici	May 17, 2012	Added sounds directory with a few sounds, changed color of ChatPanel, added race-specific colors to constants file for later implementation	Early Game
David Gay	May 17, 2012	Intentation and whitespace fixes, updated MPP file, fixed a few bugs, reformatted and updated design document	Early Game
Nathan Osborn, David Gay	May 17, 2012	Made StatPanel fully functional,	Early Game
Nathan Osborn	May 17, 2012	Added newly-updated UML document, aesthetic improvements to StatPanel	Early Game
Nathan Osborn	May 18, 2012	Few more changes to StatPanel, simplified by David	Production
David Gay	May 20, 2012	Added Yigit's sound-playing (later removed), implemented a few more of Nate's changes, added JavaDoc comments done by Yigit, corrected some bugs, added a more few stats to StatPanel, updated design doc, added help dialog, initial attempt at .sh and .bat files	Production
Scott Gunther	May 20, 2012	Prepared code for final release (refactoring, updates, presentation, etc)	Production
Yigit Katkici	May 21, 2012	Added Method Documentation and screenshot.	Production
Yigit Katkici, David Gay	May 21, 2012	Prepared JAR files and final project packaging stuff	Submission

Executive Overview

This is a two-player strategy game based on the popular television show *Battlestar Galactica*. One player represents the human race and the other player takes control of the Cylons, human-created beings bent on destroying their old masters. The Cylons have launched an attack on the humans and have destroyed the 12 human colonies. The Humans' goal is to reach the lost 13th colony, Earth, and the Cylons aren't going to make it easy.

This tactical game features two fields of play. The first is the Global View, which displays the locations of players' fleets on an interconnected, web-like map. When enemy ships meet, the interface switches to the Battle View. This field of play pits players' ships against each other in a chess-like format. The winner of the battle gets control of the contested territory, and the loser's fleet is destroyed.

VIPs (Very Important Persons) are placed by each team on ships (or fleets, perhaps) of their choosing. If all of a side's VIPs are destroyed, that side loses the game. Ergo, there are two victory conditions for each team.

Special events can occur outside of either player's control. These random events may give one player an extra edge, making the game a bit unpredictable.

Audience

This document is for anyone who wants to learn about this program and how it was designed and implemented.

Perhaps you're an interested programmer. We hope that be reading this document, for whatever reason, you are able to get a good feel of how our project came to be.

Our game's audience is of course actual *Battlestar Galactica* fans. We confess that they compose only 25% of our team. Thankfully, this means that Battlestar is easily accessible (and fun) for anyone who is able to read the manual!

Assumptions

It is likely that we need to grasp all aspects of the game before going ahead and programming. For this purpose, we sketched out on paper how we thought the game would look. That gave us the ability to make right assumption and ensured that all of us understand how the game works.

Since the game is going to be multiplayer over the Internet, we need to add network functionality. For this project, we will be implementing both a server and client. The client will send information during the game, and the server will do the number-crunching.

Another assumption we can make is that a new player will have no idea where to begin or how to play. We will include a game manual, as well as instructions accessible through our game menu that will explain the rules and perks of the game.

Gantt Chart

The project file is located in the same directory as this file (project.mpp).

Program Overview

BattleClient.java

Contains GUI code, connects to server, holds 4 panes, a menu bar, and a status bar.

BattleConstants.java

Holds game's constants. Implemented by many classes.

BattleServer.java

Handles clients and their interaction. Outputs useful status messages during startup and operation.

ChatPanel.java

Holds chat interface components.

ControlPanel.java

Holds action buttons for gameplay.

Ship.java

There is a base Ship class on which all ShipX classes are based.

ShipX.java

There is a Ship<ship name>.java class for each type of ship. This class sets the ship's attributes and allows for its control.

StatPanel.java

Holds statistics interface to provide game feedback to the user.

ViewObject.java

The class that the Ship class extends. A ViewObject is anything that can be placed in a ViewZone.

ViewPanel.java

Holds game interface, showing location of things on the grid. Also allows user to select ships and perform actions using the ViewZones as buttons.

ViewZone.java

A class extending Jbutton on which a Ship or other ViewObject can be placed. Can be clicked to select things and perform actions.

Method Documentation

BattleClient.java

BattleClient Main method instantiates panelView, panelStat, panelChat, panelControl, move and attack buttons.

getStatPanel: Accessor returns established panelStat item. Returns panelStat item.
command: Initiates and writes them as println items, if the command is empty it writes the argument.
quit : Sets message to the statpanel as the software exits, closes the window.
sendAttackCommand : Verifies whether it is users turn, notifies the user to select a ship, verifies the ship and allows the user to attack.
SendMoveCommand : Verifies whether it is users turn, notifies the user to select a ship, verifies the ship and allows the user to move.
statusBar : Instantiates status bar item adds passed message values to the statusBar.
setMessage Takes in message string and sets it as message variable.
getPw : Returns the command item.
serverRead : Reads the commands and validates them. Calls the set ship method depending on the type of the ship to set the location of the ship. Creates threads for each set of commands.
chooseRace Creates a thread to choose race to make sure the program does not stall during the initiation. Sets the thread to sleep while waiting for other player to connect.

BattleServer.java

BattleServer Main method, instantiates Cylon ship objects, human ship objects, clients as vectors and game start boolean.

Handler takes in client socket, verifies the number of clients connected.
run instantiates clientMessage variable for message transfer. Instantiates inputStreamWriter, BufferedReader, InputStreamReader, PrintWriter, OutputStreamWriter for server side to be able to read and write messages. Calls getOutputStream to be able to read messages from the client side. Sends connection status messages. Catches socket exceptions. Verifies client messages.
validateAttack takes string array of positions to validate attacks. Checks ship positions, attack positions, checks distance to validate ship attack range. If attack is successful method changes the turn to opponent.
validateMove takes string array of positions to validate moves. Checks ship positions, move positions, checks distance to validate destination of the ship. If move is successful method changes the turn to opponent.
calcDistance takes in actionPosition and shipPosition, calculates the distance and

verifies the move. Returns the distance as double.

changeTurn changes the turn for the player after move and attack.
setMyTurn takes in boolean to set myTurn variable.
printWriter returns printWriter object.
i**sReady** Returns a boolean for whether the ship is ready.
IsHuman Validates whether the player is human and returns a boolean.
IsCylon Validates whether the player is cylon and returns a boolean.
startGame Sends go command at the beginning of the method, sets ships and ship locations.
command Takes in command string, argument string and mode integer. Instantiates clients printWriter. Validates command, argument and mode.
printMessage Takes in message string and prints it out.
getPW Returns printWriter object.
checkWin Validates win conditions. Checks whether winner is cylon or human.

ChatPanel.java

print takes in string message, appends it into text area.

Ship.java

getMoveRange Returns moveRange as integer.
setMoveRange Takes in integer and sets moveRange value.
getAttackRange Returns attackRange as integer.
setAttackRange Takes in integer and sets the attackRange value.
getWeaponDamage Returns weaponDamage as integer.
setAttackRange Takes in integer value and sets the i weaponDamage value.
imageIcon Returns imageIcon item.
setImageIcon Takes in imageIcon item and sets it's value.
setName Takes in String value and sets name value.
setType Takes in string value and sets type value.
setHits Takes in integer value and sets hits value.
takeDamage Takes in integer value and sets takeDamage value.
type Returns type value as string.
name Returns name value as string.
hits Returns hits value as integer.
setPosition Takes in integer value and sets position value.
position Returns position value as integer.
toString Formats name, type, hits, position values as a readable printout.

ShipBasestar

ShipBasestar Main method. Sets image icon. Takes in string value to set ship name

and integer value to set ship position.

ShipGalactica

ShipGalactica Main method. Sets image icon. Takes in string value to set ship name and integer value to set ship position.

ShipRaider

ShipRaider Main method. Sets image icon. Takes in string value to set ship name and integer value to set ship position.

ShipHeavyRaider

ShipHeavyRaider Main method. Sets image icon. Takes in string value to set ship name and integer value to set ship position.

ShipRaptor

ShipRaptor Main method. Sets image icon. Takes in string value to set ship name and integer value to set ship position.

ShipViper

ShipViper Main method. Sets image icon. Takes in string value to set ship name and integer value to set ship position.

StatPanel.java

StatPanel Main method implements swing timer timer for time values. Instantiates time values, matchid and turn. Sets the panel configuration.

update Takes in ViewZone object, sets the zone for the selected ship. Prints out text to the panel depending on set stat values.

turn increments turn.

setShip Takes in ship value and sets it to selected ship.

setId Takes in integer and sets it as id.

ViewObject.java

getIcon Returns icon object.

setIcon Takes in icon imageObject and sets it as icon. .

ViewPanel.java

ViewPanel Main method, instantiates zones as vector, selected zones, action zones and client objects. Booleans for moving and attacking.

setShip Takes in position as integer and ship object. Takes the position and sets the ship position.

setAttacking Takes in attacking boolean. If it is true it sets the attacking variable. If it is null it sets the attacking value as null.

setMoving Takes in moving boolean. If it is true it sets the attacking variable. If it is null it sets the attacking value as null.

getZone Takes in integer and sets it as the zone.

getSelectedZone Returns selected zone object.

getActionZone Returns action zone object.

ViewZone.java

ViewZone Main Method takes in statPanel object and sets the value. Sets background, foreground, margin, compound and line. Sets the border as compound value.

unselected Sets the unselected button color as white.

selected Sets the selected button color as green.

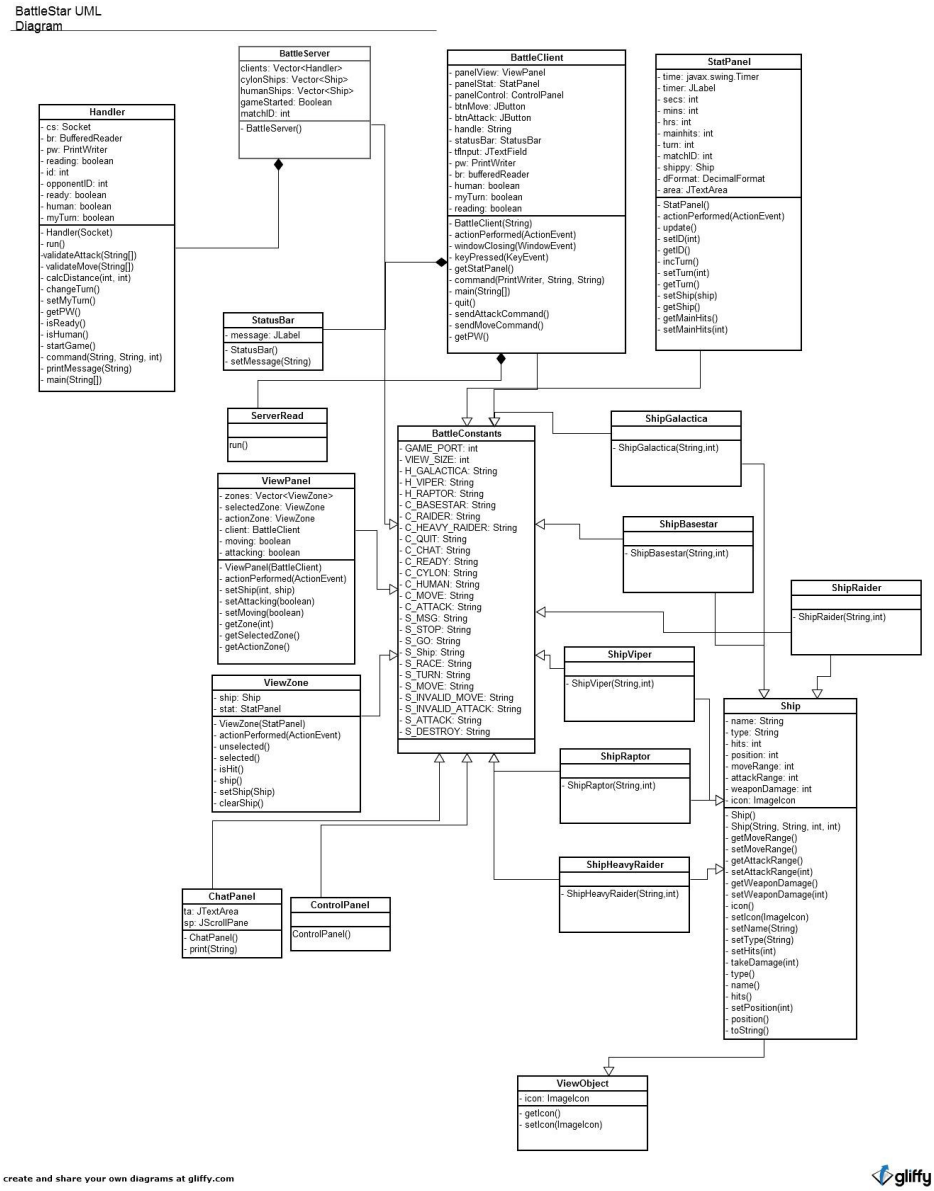
ship Returns ship item.

setShip Takes in ship item and sets the ship. Sets the image icon for the ship.

clearShip Sets the ship null and clears the icon.

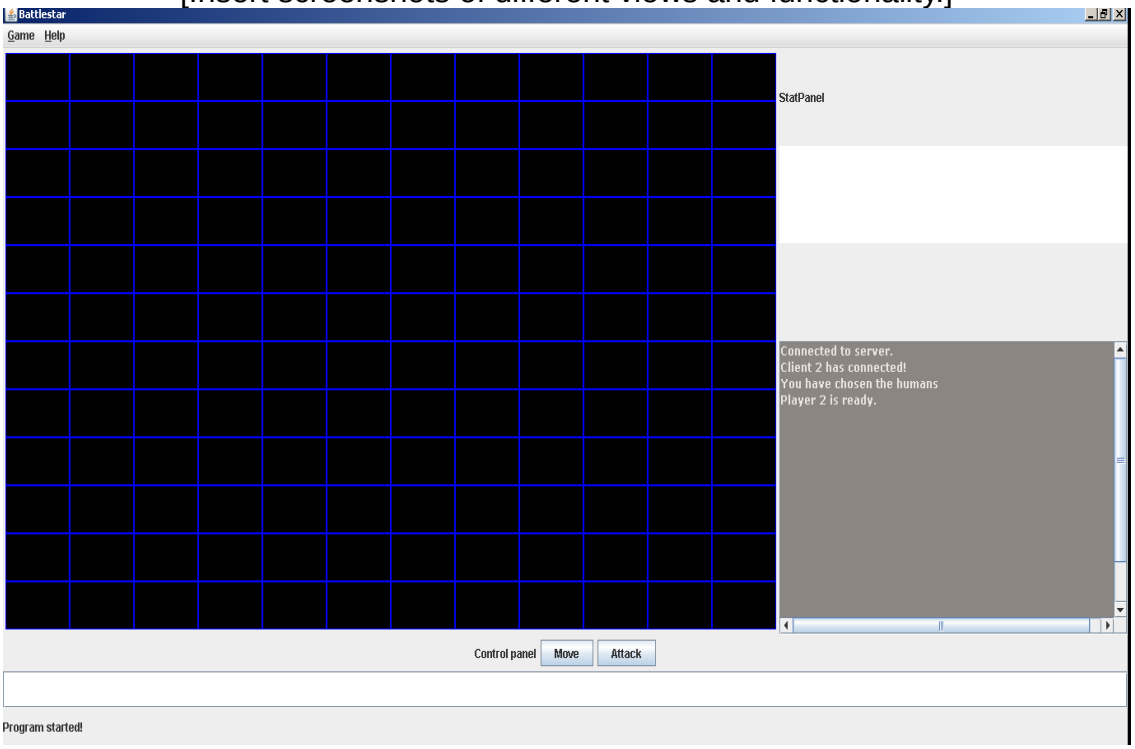
UML Diagram

UML diagram created with gliffy (gliffy.com).

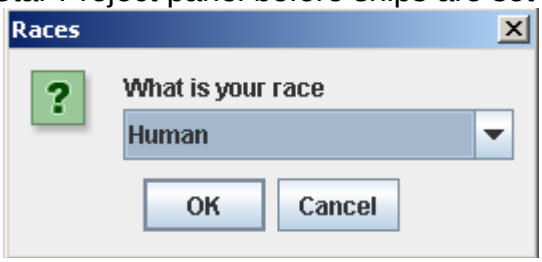


Screenshots

[Insert screenshots of different views and functionality.]



Empty Battle Star Project panel before ships are set on the board.



Battle Star Project race selection.

Protocols

The server is run on a host with an IP address which must be used as an argument when starting the client (ex. $ java BattleClient 10.100.100.131). All client/server communication is done via port 16789.

The client/server communication is done via a command protocol. Messages generated by the server and sent to a client or multiple clients is prefaced by a S_, and messages generated by a client and sent to the server are prefaced by a C_. All commands are found in the constants file, BattleConstants.java. The server and client(s) are all continuously listening for commands sent from the other.

Chat Interaction

If a user sends an input in the text field which is not prefaced with a slash command (ex. /ready), “/chat” is appended to the message and sent as a chat message. /chat can be manually used, but takes more time. The server broadcasts the message to all clients.

Data

Battlestar currently does not generate data files or use any connections outside of those defined in the Protocols section. In the future, we had planned to implement server logging and a data file storing match number and other records.

TODO

See TODO.rst in root directory for prioritized future plans and completed tasks.

Issues

See the issue tracker for bugs, enhancements, feature requests, and the like.

Issue tracker: https://github.com/oddshocks/battlestar/issues

License

Battlestar is released under the GNU GPL in the name of education and freedom. All Battlestar trademarked names and phrases are property of Battlestar Galactica's owners.