Team 12

**Shadow Nova**

Software Design Document

Names: Katt Owens, Theron Myers, Kyle Carter

Date: (03/15/2018)

Software Design Document

**TABLE OF CONTENTS**

**1. INTRODUCTION 3**

1.1 Purpose 3

1.2 Scope 3

1.3 Overview 3

1.4 Reference Material 3

1.5 Definitions and Acronyms 3

**2. SYSTEM OVERVIEW 4**

**3. SYSTEM ARCHITECTURE 4**

3.1 Architectural Design 4

3.2 Decomposition Description 5

3.3 Design Rationale 6

**4. DATA DESIGN 6**

4.1 Data Description 6

4.2 Data Dictionary 7

**5. COMPONENT DESIGN 13**

**6. HUMAN INTERFACE DESIGN 18**

6.1 Overview of User Interface 18

6.2 Screen Images 18

6.3 Screen Objects and Actions 19

**7. REQUIREMENTS MATRIX 19**

**1. INTRODUCTION**

**1.1 Purpose**

This software design document describes the architecture and system design of Shadow Nova to the instructors and team members.

**1.2 Scope**

Shadow Nova is a browser based roleplaying game set in an ever expanding galaxy. Each server should be able to handle around 100 players. Our goal is to create a functional alpha version that is functional and able to be expanded upon later. With this, we could potentially pull more people into the development (volunteers), and build other components atop of the core functionality.

**1.3 Overview**

The player controls a noble House and the family/characters that make up that house. One could compare it to the Game of Thrones books, where the player would control the entirety of a House, such as Stark or Lannister, and the characters of that house.

The goal of the software is to provide a mechanical framework to build story writing on, thus is not a traditional style of game. For this project, we do not plan to implement the entirety of the games features, just the core features needed to show the design off.

**1.4 Reference Material**

* Game Shadow Nova is based on and aiming to replace: <http://www.imperiumnova.com:8080/Imperium/>
* Old Wiki containing old roleplay elements of IN: <https://imperiumnova.info/wiki/start>
* Game Design Doc, has additional features we are not implementing: <https://docs.google.com/document/d/1iYqGYNLc1ffTukhdn2A1Frj3CH2QWK10C8SSqB5OmyE/edit#bookmark=id.l5g5yxf286ku>

**1.5 Definitions and Acronyms**

* (Server) Heartbeat: Interchangeable with Server Tick.
* Triggered at regular time intervals, causes the game to update things that would advance with time, such as resource consumption/production, ship movement, battles, etc.
* House: What the player controls at a broad level. Analogous to a noble house of ancient eras, or more directly, a noble house in the book series Dune.
* Character: An individual person in the House’s family and/or employ. Can be assigned to different tasks as a leader.
* Industry: A field of commerce, such as Agriculture or Healthcare.
* Holding: The collective assets of a single industry that a House owns on a planet.
* Upgrade: A single level of assets that make up a holding, such as a production plant or a fortification.
* NPC: A computer controlled house that operates on a handful of worlds.
* Campaign: Extended battle on a planet, either in space or on planet. Represents a war and is made up of numerous, calculated battles.
* Player: Interchangeable with User
* Spacelane: Interchangeable with Hyperlane. Used as connections/roads between planets.

**2. SYSTEM OVERVIEW**

Shadow Nova is broken into two distinctive parts, the server, with its heartbeat loop, supported by specific classes, and the Browser UI. Both interact with a central database.

This architecture meets the barebone requirements for Shadow Nova to be functional, but has some extra bits for expanded functionality. Additions and trimming will occur as the project moves forward.

**3. SYSTEM ARCHITECTURE**

**3.1 Architectural Design**

The main components of our game system are the Functions within Classes, the Heartbeat Loop, Databases, Verification, and Browser UI.

Browser UI: Role to provide information to the player and take commands from the player. The Browser UI requests data from the Databases which returns that data after verification has determined the user is allowed to see that data. Browser UI can submit changes (spending resources to build something for example), which are verified as possible and allowed, before the changes are made to the database. Browser UI is what the user and admin use to interact with the game.

Verification: Role to check commands and requests from the Browser UI to make sure they are acceptable for the user or admin to use/view. Users can not access all of each others info. Admins can see everything potentially, but might have limits on commands.

Databases: Central repository for all saved data. Databases: Battles, Campaigns, Characters, Houses, Planets, Military Units, User Data, Game

Heartbeat Loop: Role is to progress the game forward. The Heartbeat Loop would run every set amount of time, probably every 30, 45, or 60 minutes, making the game move forward in time by updating various elements of the game. It will get data from the databases and store in local objects, call class functions to calculate changes and modify those objects, then when it is finished, override the old database files that have been changed. This method may run into issues with the speed requirement, requiring a pseudo-lockdown on Browser UI input.

Classes: Roles vary on class, but generally they are responsible for holding functions used by the Heartbeat Loop and each other. Classes: Battle, Campaign, Character, House, Planet, MilitaryUnit, UserData

General Overview of System

Class & Respective Functions

Heartbeat

Loop

Databases

Verification

Browser UI

User

Admin

**3.2 Decomposition Description**

Classes have many duplicated data members for search time reduction, but not diagraming out every time they need to update due to changes in another class. That is assumed to be true.

Heartbeat Loop

Starlanes

Move Units/Resources, Decay Health

Check if new form

Holdings

Consume, Build, Generate Income, etc

Lock Databases, Read in Data, store in Objects

Destroy Old Data

Character

Various tick updates for characters

Holdings

Check for overflow of resources

Campaigns

Consume Resources, Check for Battles

Planets

Consume, Population Checks, Derive Stats, etc

Wait for timer to restart loop

Save data to database, release database lock

Misc

Various updates, checks, and clean up

Campaign/Battle Diagram

Starlane

Campaign

Collision of hostile forces, triggers battle

Battle triggers

Points lost for military units. Exp potentially gained.

MilitaryUnit

Battle

Upgrade Destroyed

House

**3.3 Design Rationale**

We considered implementing Shadow Nova as a self-contained program, but Shadow Nova really benefits from being accessible from any computer. Additionally, implementing a browser based GUI is far simpler to program and design. It requires fewer new systems for our team to learn and lets us add new features relatively quickly. Since we needed a browser UI and a program that would update databases, a central repository for data storage that could be shared between the two was required.

**4. DATA DESIGN**

**4.1 Data Description**

Classes: Battles, Campaigns, Characters, Houses, Planets, Military Units, User Data

Databases: Battles, Campaigns, Characters, Houses, Planets, Military Units, User Data

Databases will use a style that can be easily read by C# and PHP or CHTML. MySQL is a strong contender.

Every time Heartbeat Loop triggers, it will flush its data and reload all the databases that are current, as some data will be kept only for historical reference like non-active battles and campaigns, and store the info in class objects. This will lock the databases from Browser UI input. Once Heartbeat loop is completed and data is saved back to the databases, it will release the locks.

Unsure how it will read in the data, either passing a string to a class constructor and letting it translate that into its variables, or having the loop parse the string and calling a constructor with all of the needed variables.

**4.2 Data Dictionary**

Providing both variables and functions, with small description. Some use things like PID, CID, etc as parameters. These are just of Int type, but specifying what that int represents (usually ID of some description). These could be replaced by pointers potentially.

Classes:

* Battle: Stores historical battle data.
  + Data:
    - BID (Int): ID for Battle
    - Loc (PID): Planet it occured on. 0 for starlane
    - Houses (2d Array of HID/Int/Int): Houses in the fighting/which alliance they belong to/what type of war they are waging
    - Date (Int): Date of battle
    - LossUnit (2d Array of MID/Int): Unit, number of points lost
    - LossUpgrade (2d Array of HID/Int’s): Static array for Ints, each for a different type of upgrade.
  + Functions:
    - Sets
    - Gets
    - Battle (PID): Constructor that generates the entire battle.
    - CalculateBattle (): Determines outcome of battle.
* Campaign:
  + Data:
    - CamID (Int): ID for the campaign
    - Houses (2d Array of HID/Int/Int): Houses in the fighting/which alliance they belong to/what type of war they are waging
    - Planet (PID): What planet it is on.
    - SDate (Int): Start of the campaign.
    - EDate (Int): When the campaign ended, 0 for still ongoing.
    - Battles (Array of BID’s): What battles happened during campaign.
    - Settings (Array of Int): Static array of options for the campaign
  + Functions:
    - Sets
    - Gets
    - Campaign (): Constructor
    - BattleChance (): Calculates odds of battle happening, creates battle if true.
    - Valid (): Checks if there are houses who are fighting still.
* Character:
  + Data:
    - CID (Int): ID of Character
    - Age (Int)
    - Birthday (Int): tick number they were created or backdated too.
    - Gender (Boolean): 1 for female, 0 for male.
    - Health (Boolean): 1 for live, 0 for dead.
    - Exp (Static Array of Int): Experience (Military, Specific Industries, etc)
    - Relations (2d Array of CID/Int): Relations to other characters (Brother, Father, Rival, Friend). Int for type.
    - Preg (CID): Pregnancy. List Father or 0 for not preg.
    - Descript (String): Description of Character
    - Loc (PID): Location
    - Owner (HID): House they belong to.
    - Pos (Int): Position they are in for Owner. Int for type.
  + Functions:
    - Sets
    - Gets
    - CreateNew (HID): Create new character for house creation. Backdate the birthdate based on generated age.
    - BirthNew (CID, CID): Using Mother and Father for genes.
    - BirthNew (CID): Mysterious father, use Mother for genes.
    - Experience (Int): Int is type of experience, typically used by Industry and Military to potentially raise characters experience.
    - Age (): Ages character by a tick, chance of death based on age.
    - Pregnancy (): Checks if character becomes pregnant.
* House:
  + Data:
    - HID (Int): ID of House
    - Relation (2d Array of HID/Int): Int for type.
    - Home (PID): Where their center of power is. Like what a capital city is for a nation.
    - Quote (String): A short sentence for the house. Like “Winter is Coming.”
    - Chars (Array of CID): Characters belonging to House
    - Units (Array of MID): What units they control
    - Planets (Array of PID): What planets they have holdings on.
    - Setting (Array of Int): Static array of settings for the house.
  + Functions:
    - Sets
    - Gets
* Planet:
  + Data:
    - PID (Int): ID of Planet
    - Starlane (Array of SID): What starlanes attach to the planet.
    - Loc (Int, Int): Location of planet on map
    - Size (Int): General size of the planet
    - Terrain (Int): What kind of planet? Desert, Ice, Marine, Temperate, Verdant, Airless
    - SecTerrain (String): A minor tag to put in front of the Terrain, like Cold Desert or Icy Marine. Flavour Text.
    - Descript (String): Sentence that can be blank or give some interesting tidbit about the world, like “3 to 1 gender ratio” or “intoxicating atmosphere” to borrow examples from IN.
    - ExpLabour (Int): Produced by Academic Industry
    - GenLabour (Int): Naturally produced
    - Minerals (Int): Mineral richness of the planet
    - PlanetStock (Array of Ints): Static array that holds the resources for the planet to use.
    - Holdings (2d Array of HID/Ints and CID’s): Ints and CID’s are statically placed, representing various stats of a holding, Eg: Upgrades, Resources, industry settings, and characters controlling
    - Units (2d Array of HID/MID’s): What units are on planet
    - Total Population (Int)
    - EduLevel (Int)
    - Wealth (Int)
    - PopGrowth (Int)
  + Functions:
    - Sets
    - Gets
    - Consumption (): Consumes resources.
    - BuyResources (): Buys resources from selling houses.
    - PopGrow (): Adds to GenLabour.
    - PopDie(): Lowers GenLabour and ExpLabour
    - TotPop(): Calculates total population by (GenLabour+ExpLabour)\*X%
    - EducatePop (): Converts GenLabour into ExpLabour based on education facilities on planet.
    - EducateLvl (): Calculate education level of planet.
    - MineralDecay (): Chance Mineral richness decreasing based on mining occurring on planet.
    - Calculate Wealth ():
* MilitaryUnit:
  + Data:
    - MID (Int): ID of Military Unit
    - Owner (HID):
    - Commander (CID): Character that commands unit, 0 for none.
    - DefMob (Boolean): Defensive or Mobile Unit.
    - Type (Boolean): 1 for Space, 0 for Land
    - Points (Int): How strong the unit is.
    - Exp (Int): How experienced the Unit is
    - Active (Boolean): Active or Reserve? Maybe change to scaling readiness level.
    - Camp (CamID): Part of which Campaign? 0 for none.
    - Moving (Array of SLID’s): Array of what SLID’s, in order, are to be taken to move locations.
  + Functions:
    - Sets
    - Gets
    - Experience (): If active, chance to gain exp. If reserve, chance to lose exp.
* Starlane:
  + Data:
    - SLID (Int): ID for Starlanes
    - FPlanet (PID): First planet it attaches too.
    - FPlanet (Array of Int/Int): Stores location of first planet for mapping
    - SPlanet (PID): Second planet it attaches too.
    - SPlanet (Array of Int/Int): Stores location of second planet for mapping
    - Dura (Int): How much health the starlane has, 0 means it dies.
    - Units (Array of MID/Int/Int): What units are currently on the starlane, which direction did they start (1 for First, 0 for second), and what percentage of progress are they.
  + Functions:
    - Sets
    - Gets
    - MoveUnit (): Moves unit
    - MoveResource(): Moves Resources
    - CheckCollision (): Checks if two units will collide and if so, if they are hostile. Trigger battle if Hostile
* UserData:
  + Data:
    - UID (Int): ID of User
    - Login (String): User’s login
    - Pass (String): Password
    - Display Name (String): What the user’s name will be.
    - House (HID): House they control.
  + Functions:
    - Sets
    - Gets
    - UserData (): Creates new house. Might be special interrupt of normal heartbeat. Checks one file’s timestamp?

Databases: Each will generate a single file per object, which will be stored in a file dedicated to that database type.

* Battles: Stores historical battle data.
  + Data:
    - BID (Int): ID for Battle
    - Houses (2d Array of HID/Int/Int): Houses in the fighting/which alliance they belong to/what type of war they are waging
    - Date (Int): Date of battle
* Campaigns: Create new file for every campaign to avoid file bloat. Stores campaign data
  + Data:
    - CamID (Int): ID for the campaign
    - Houses (2d Array of HID/Int/Int): Houses in the fighting/which alliance they belong to/what type of war they are waging
    - Planet (PID): What planet it is on.
    - SDate (Int): Start of the campaign.
    - EDate (Int): When the campaign ended, 0 for still ongoing.
    - Settings (Array of Int): Static array of options for the campaign
    - Battles (Array of BID’s): Stores battles that happened in Campaign
* Characters: Holds character stats, which house they belong to, their biological and social relationships to other characters, current location, and if they hold any key positions in the house.
  + Data:
    - CID (Int): ID of Character
    - Age (Int)
    - Birthday (Int): tick number they were created or backdated too.
    - Gender (Boolean): 1 for female, 0 for male.
    - Health (Boolean): 1 for live, 0 for dead.
    - Exp (Static Array of Int): Experience (Military, Specific Industries, etc)
    - Relations (2d Array of CID/Int): Relations to other characters (Brother, Father, Rival, Friend). Int for type.
    - Preg (CID): Pregnancy. List Father or 0 for not preg.
    - Descript (String): Description of Character
    - Loc (PID): Location
    - Owner (HID): House they belong to.
    - Pos (Int): Position they are in for Owner. Int for type.
* Houses: Misc House stats, character list maybe?
  + Data:
    - HID (Int): ID of House
    - Relation (2d Array of HID/Int): Int for type.
    - Home (PID): Where their center of power is. Like what a capital city is for a nation.
    - Quote (String): A short sentence for the house. Like “Winter is Coming.”
    - Chars (Array of CID): Characters belonging to House
    - Units (Array of MID): What units they control
    - Planets (Array of PID): What planets they have holdings on.
    - Setting (Array of Int): Static array of settings for the house.
* Planets: Location, Houses that have holdings on planet, planet stats (population, planet size, mineral resources, etc).
  + Data:
    - PID (Int): ID of Planet
    - Starlane (Array of SID): What starlanes attach to the planet.
    - Loc (Int, Int): Location of planet on map
    - Size (Int): General size of the planet
    - Terrain (Int): What kind of planet? Desert, Ice, Marine, Temperate, Verdant, Airless
    - SecTerrain (String): A minor tag to put in front of the Terrain, like Cold Desert or Icy Marine. Flavour Text.
    - Descript (String): Sentence that can be blank or give some interesting tidbit about the world, like “3 to 1 gender ratio” or “intoxicating atmosphere” to borrow examples from IN.
    - ExpLabour (Int): Produced by Academic Industry
    - GenLabour (Int): Naturally produced
    - Minerals (Int): Mineral richness of the planet
    - PlanetStock (Array of Ints): Static array that holds the resources for the planet to use.
    - Holdings (2d Array of HoID/Ints): Ints are statically placed, representing various stats of a holding, Eg: Upgrades, Resources, Military Unit ID’s, and industry settings.
    - Total Population (Int)
    - EduLevel (Int)
    - Wealth (Int)
    - PopGrowth (Int)
* Military Units:
  + Data:
    - MID (Int): ID of Military Unit
    - Owner (HID):
    - DefMob (Boolean): Defensive or Mobile Unit.
    - Type (Boolean): 1 for Space, 0 for Land
    - Points (Int): How strong the unit is.
    - Exp (Int): How experienced the Unit is
    - Active (Boolean): Active or Reserve? Maybe change to scaling readiness level.
    - Camp (CamID): Part of which Campaign? 0 for none.
    - Moving (Array of SLID’s): Array of what SLID’s, in order, are to be taken to move locations.
* Starlanes:
  + Data:
    - SLID (Int): ID for Starlanes
    - FPlanet (PID): First planet it attaches too.
    - FPlanet (Array of Int/Int): Stores location of first planet for mapping
    - SPlanet (PID): Second planet it attaches too.
    - SPlanet (Array of Int/Int): Stores location of second planet for mapping
    - Dura (Int): How much health the starlane has, 0 means it dies.
    - Units (Array of MID/Int/Int): What units are currently on the starlane, which direction did they start (1 for First, 0 for second), and what percentage of progress are they.
    - Hidden (Array of HID’s): Which houses can see it?
* User Data: Holds login info, email addresses, preferences, and which house they control. This file needs to be encrypted. This is being kept short, additional content will be needed once forums are implemented.
  + Data:
    - UID (Int): ID of User
    - Login (String): User’s login
    - Pass (String): Password
    - Display Name (String): What the user’s name will be.
    - House (HID): House they control.
* Game: Misc stats for the entire game
  + Data:
    - Current Time

**5. COMPONENT DESIGN**

Classes:

* Battle: Stores historical battle data.
  + Sets
  + Gets
  + Battle (PID): Constructor
    - Pulls required data from other classes.
    - Find last BID number in database, set BID to that+1
    - Calls CalculateBattle ().
  + CalculateBattle (): Determines outcome of battle.
    - Immediate check, if one participant has no units, has upgrades left, they have no allies, and the war type allows their destruction, immediately destroy the upgrades. Skip to last step.
    - Random number between 1 and 3, for small, medium, or large battle size. This effects the scale of all of the below.
    - Determine how many points from each fighting unit will contribute based on size of battle and how many points are on each side.
    - Determine who the aggressor is, determined by type of campaigns.
    - Based on aggressor, points on each side, defensive upgrades on defenders holdings, space forces if campaign type allows, and character leader experience, calculate who wins. Can be draw.
    - If aggressor wins, make more upgrades be lost.
    - If defender wins, make fewer upgrades be lost.
    - If draw, do a medium level.
    - Lost points determined. Defensive upgrades can make aggressor loses far greater. Experience of units increases opposing sides point loss while decreasing own loss.
    - Calculate chance of collateral damage to houses not involved in fighting.
    - Record loses, update other classes.
* Campaign:
  + Sets
  + Gets
  + Campaign (): Constructor
    - Pulls required data from other classes.
    - Sets SDate as Current Date
    - Find last CamID number in database, set CamID to that+1
    - Creates Battles array
    - Creates Settings array
    - Calls Valid()
    - Generates Battle if Valid() is true.
  + BattleChance (): Calculates odds of battle happening, creates battle if true.
    - Call Valid(), if true, continue. If false, put an end date to campaign. Update military units.
    - Use a formula to calculate a threshold based on number of unit points involved in the fighting. The more points, the lower the value.
    - Check to see if (current date - last battle date) is equal or above formula. If above, trigger a new battle.
    - Call Valid() if true, end. If false, put an end date to campaign. Update military units.
  + Valid (): Checks if there are houses who are fighting still.
    - Go through each house in array, check if any units or upgrades left. If true, return true. If false, return false.
* Character:
  + Sets
  + Gets
  + CreateNew (HID): Create new character for house creation. Backdate the birthdate based on generated age.
    - Find last CID number in database, set CID to that +1
    - Assign Birthday to be Current Date
    - Random 1 or 0 for gender
    - Set Health to 1
    - random age 1 to 120 in character years (depends on tick amount for time conversion)
    - Random exp, with tendency for lower for young characters and very old characters
    - Calculate relations, this will be complex.
    - Set Preg to 0.
    - Generate short description of character.
    - Set Loc to House capital
    - Set owner to inputted HID
    - Set Position to 0 for none.
  + BirthNew (CID, CID): Using Mother and Father for genes.
    - Find last CID number in database, set CID to that +1
    - Assign Birthday to be Current Date
    - Random 1 or 0 for gender
    - Set Health to 1
    - Set Age to 0
    - Set exp to 0’s
    - Set relations to Mother and Father.
    - Set Mother and Father’s relation to child
    - Set Preg to 0
    - Generate short description of character of what child will look like when grown up.
    - Set Loc to House capital
    - Set owner to mother or father, depending on house settings.
    - Set Position to 0 for none.
  + BirthNew (CID): Mysterious father, use Mother for genes.
    - Find last CID number in database, set CID to that +1
    - Assign Birthday to be Current Date
    - Random 1 or 0 for gender
    - Set Health to 1
    - Set Age to 0
    - Set exp to 0’s
    - Set relations to Mother and Father.
    - Set Mother and Father’s relation to child
    - Set Preg to 0
    - Generate short description of character of what child will look like when grown up.
    - Set Loc to House capital
    - Set owner to mothers owner HID
    - Set Position to 0 for none.
  + Experience (Int): Int is type of experience, typically used by Industry and Military to potentially raise characters experience.
    - Roll random chance, younger has a lower threshold to beat.
    - If threshold beaten by roll chance, increase experience in relevant type.
    - Roll random chance, younger has lower threshold to beat. Threshold should be very high regardless, very low chance of being beaten
    - If threshold beaten by roll chance, increase experience in random type.
  + Age (): Ages character by a tick, chance of death based on age.
    - Increase age by 1.
    - Roll to see if character dies to natural causes. Higher age has greater chance to beat threshold.
    - If threshold beaten, character dies.
    - Set health to 0.
    - Remove from position.
  + Pregnancy (): Checks if character becomes pregnant.
    - If lacks lover or married relationship, 0 chance.
    - If married, roll random to see if a threshold is beaten, affected by age of both partners.
    - If beaten, set female pregnant to Married’s CID
    - Repeat until out of married
    - If has lovers, roll random to see if a threshold is beaten, affected by age of both partners.
    - If beaten, set female pregnant to Lover’s CID
    - Repeat until out of lovers.
* House:
  + Sets
  + Gets
  + House (): Constructor
    - Find last HID number in database, set HID to that +1
    - Create house family
    - Determine where house starts in galaxy
    - Create holding, assign it as capital
    - Give a few upgrades.
    - Give holding starts resources
    - Set default settings.
* Planet:
  + Planet (): Constructor
    - Find last PID number in database, set PID to that +1
    - Create starlane from planet to another planet.
    - Generate stats
    - Derive stats from generated stats.
  + Consumption (): Consumes resources.
    - Use Total Population, Wealth, and EduLevel to determine resources the planet needs to consume.
    - Attempt to consume resources from planets.
    - If not enough resources, Call BuyResources().
    - Attempt to consume resources from planets.
    - If not enough resources, decrease stats accordingly.
  + BuyResources (): Buys resources from selling houses.
    - Checks planets current stock of resources
    - If resources lower than (2 \* next consumption cycle), attempts to buy enough to meet that amount.
    - Gives something in return (cash equivalent)
  + PopGrow (): Adds to GenLabour.
    - If surplus food, population grows decently. If not, pop grows slowly.
  + PopDie(): Lowers GenLabour and ExpLabour
    - If surplus food, medical, etc, lower death rate. If lacking, death rate increases.
  + TotPop(): Calculates total population
    - Calculate by (GenLabour+ExpLabour)\*X%
  + EducatePop (): Converts GenLabour into ExpLabour based on education facilities on planet.
  + EducateLvl (): Calculate education level of planet.
  + MineralDecay (): Chance Mineral richness decreasing based on mining occurring on planet.
  + Calculate Wealth (): Calculates wealth
* MilitaryUnit:
  + Sets
  + Gets
  + MilitaryUnit (HID, Boolean, Boolean, Int): Constructor
    - Find last MID number in database, set MID to that +1
    - Set DefMob to first Boolean
    - Set type to second Boolean
    - Set points to Int
    - Set Exp to 0, Active to true, Camp to 0, Moving to 0.
  + Experience (): If active, chance to gain exp. If reserve, chance to lose exp.
* Starlane:
  + Sets
  + Gets
  + Starlane (PID, PID): Constructor
    - Find last SLID number in database, set SLID to that +1
    - Set FPlanet to first PID
    - Set SPlanet to second PID
    - Randomly generate Dura, with a static min.
    - Set Units to NULL
  + MoveUnit (): Moves unit along lane
  + MoveResource(): Moves Resources along lane
  + CheckCollision (): Checks if two units will collide and if so, if they are hostile. Trigger battle if Hostile
* UserData:
  + Sets
  + Gets
  + UserData (): Creates new house. Might be special interrupt of normal heartbeat. Checks one file’s timestamp?
    - Find last UID number in database, set UID to that +1

In this section, we take a closer look at what each component does in a more systematic way. If you gave a functional description in section 3.2, provide a summary of your algorithm for each function listed in 3.2 in procedural description language (PDL) or pseudocode. If you gave an OO description, summarize each object member function for all the objects listed in 3.2 in PDL

or pseudocode. Describe any local data when necessary.

**6. HUMAN INTERFACE DESIGN**

**6.1 Overview of User Interface**

The user should be able to complete every task from the browser that they need to accomplish. The left side of the screen will display a map of the galaxy normally, letting them select objects (planets, spacelanes namely) to interact with. Clicking an object will bring up data and options to interact with it on the right side of the screen. A back button and home button will return the right side of the screen to the respective state.

Users will be able to order upgrades to holdings, order units around, start/end campaigns, transfer resources from one planet’s holding to another, affect settings for various tasks.

The map will likely be replaced with forums when the user is checking them or writing, letting them reference info from the data frame.

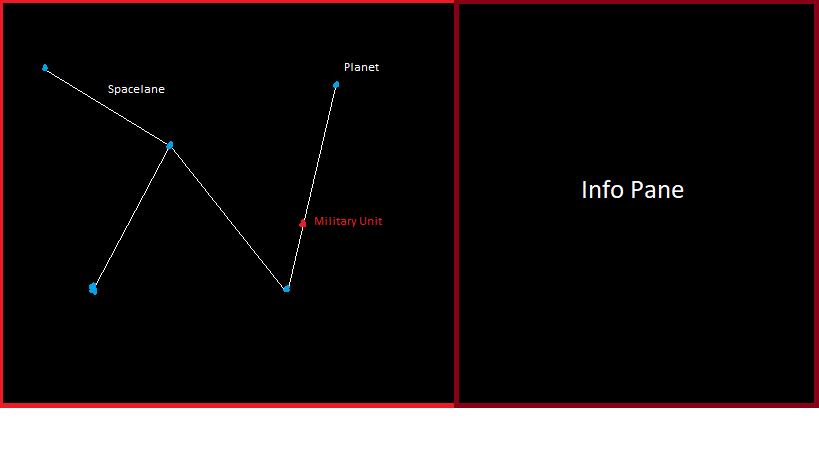
**6.2 Screen Images**

Browser UI Interface

Data

Map

2nd Browser Interface Image



**6.3 Screen Objects and Actions**

On the left will be the map, which will display planets, spacelanes, as well as other items like units traversing spacelanes. Planets and spacelanes may have icons around them to indicate different things to the player about them, such as having a campaign on them or being blockaded.

The right side contains data. The first page will be general info about the users House. Clicking an entity in the map frame will change the data frame to display info about it. Sub-pages may be accessed from the data frame, such as checking on another players assets on a planet while in the planet menu.

Somehow, the user will be able to switch the map to display the forums of the game, letting them write their stories, read other players work, or simply talk to other players. This is not a live messaging system, but bulletin board style forums.

Specifics have not been decided.

**7. REQUIREMENTS MATRIX**

User Interface: Satisfied by Browser UI

Speed: Not satisfied, prove by testing. May be dealt with otherways.

Easy of Use: Conditionally satisfied by Browser UI, pending testing.

Portability: Satisfied by Browser UI

Expandable: Satisfied by modular design of classes and databases.

Security: Unsatisfied, pending more research into browser security.