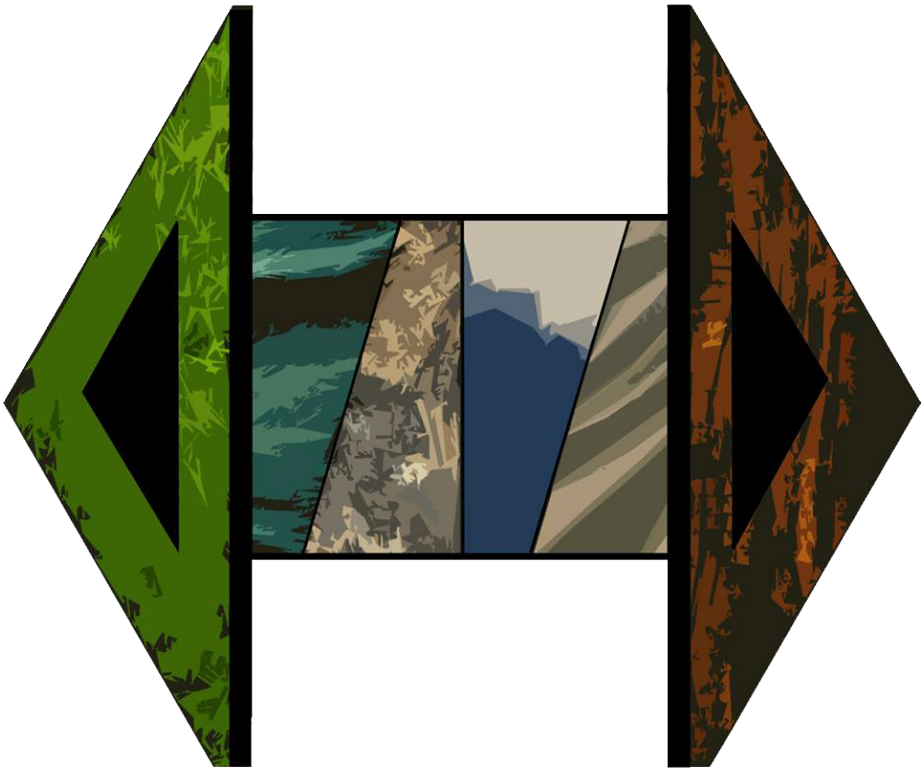
Software Atelier IV - Assignment 3

Vitor Vannuchi Stefanos Gatsios Orestis Melkonian
Lokesh Menghani Hanieh Soleimani

March 9, 2014

1 Hexarchy



---

2 Categories

Categories	Implementor
Map generation	Stefanos
Unit generation	Vitor
UI	Hanieh
Combat Mechanics	Lokesh
Server side	Lokesh & Hanieh
Game Logic	Orestis
Multiplayer	Stefanos
RPG Elements	Orestis

3 Core Requirements

- Map representation
- Units
- Currency
- Turn based Logic
- Combat
- Asychronous Multiplayer Aspect
- Server
- Ethical Constraints
- Legislative Issues

---

4 Core Requirements in Detail

1. Map Existence
   - Type : Functional.
   - Summary : The game environment is represented onto a map.
   - Description : When you start the game, a map composed of hexagons, is generated on the screen and it changes as the game advances
2. Units
   - Type : Functional.
   - Summary : The ability of the player to create and handle units.
   - Description : As a player begins with a certain ammount of currency and resources, the player will be able to create and handle units.
3. Currency
   - Type : Functional.
   - Summary : Each player has his own economic development.

- Description : Each player is allocated with an initial amount of currency which allows, in addition with soldiers, to create a unit of any type.

4. Turn based logic.

   - Type : Functional.
   - Summary : The player will play in turns.
   - Description : Each player has a restricted amount of time to perform his actions.

5. Asynchronus Multiplayer Aspects.

   - Type : Functional.
   - Summary : The game supports many players playing together and no artificial intelligence is involved.
   - Description : Every player has a limited amount of time to coordinate his moves, once every player has completed their moves, the turn is performed, with the exception of a player exceeding the available time.

6. Combat.

   - Type : Functional.
   - Summary : Players can attack other players.
   - Description : By using their units to attack, players diminish opponents' strength and eventually defeat them to win the battle.

7. Server.

   - Type : Functional.
   - Summary : The server implements the decisions made by the players.
   - Description : The server validates the actions of the users, performs all the actions and informs the outcome of every turn to all the players.

8. Goal of the game.

   - Type : Non-functional.
   - Summary : The goal of the game is to destroy the enemy's HQ.

- Description : In order to motivate players a primary goal is provided; which is to dominate or destroy the enemy's headquarters.

9. Server Response Time.

   - Type : Non-functional (Performance).

   - Summary : After players submit their moves, server should implemenet the action in maximum of 30 seconds.

   - Description : When the actions are submited the server must make all computations (moves,change of values,attacks and battle outcomes) within a logical time. e.g. (per say 30 seconds)

10. Ethical Constraints

    - Type : Non-functional

    - Summary : The game should comply with the current ethical standards of the society.

    - Description : The game must not conflict with any ethical constraints of today's society (e.g. PEGI content rating system so as children can be users).

11. Legislative Issues

    - Type : Non-functional

    - Summary : Avoid all law violations.

    - Description : The game should not violate any law, such as copyrights etc.

---

5 Stefanos Gatsios

5.1 Map Generation

Basic

1. Hexagon Map

   - Type : Functional.

   - Summary : The based will be composed of hexagons.

   - Description : Each of the map cells is a hexagon, inheriting all the properties of the hexagon, like nearby cells, etcetera.

2. Different Type of Cells

    - Type : Functional.
    - Summary : Each of the hexagons will be of some type.
    - Description : The hexagons on the map will have a pre-assigned type, like land, sea, lake, mountain, forest, etc. . .

Advanced

1. Prototype maps

    - Type : Non-functional
    - Summary : Having a bank of prototype maps for each of the sizes
    - Description : For each size of the maps some basic named prototypes, probably in ASCII code will be pre-made.

Complex

1. Random Map Generation based on Prototype

    - Type : Functional
    - Summary : Random map each new game
    - Description : A map based on a prototype will be generated every time a new game is started.

2. Parametrized Map Generation

    - Type : Functional.
    - Summary : Control on the geometry of the map and elements via variables.
    - Description : The created of the map is based upon values, e.g. I want at least two lakes and mountains around these lakes.

3. Pure random map generation

    - Type : Functional.
    - Summary : Generating the map without any use of prototype.
    - Description : Pure random map generation without any prototype influence.

---

6 Vitor Vannuchi

6.1 Unit Generation

Basic

1. Types of units

   - Type : Functional.
   - Summary : The units will be separated into dynamic unit and static unit
   - Description : When create an unit you can just choose between dynamic or static unit. Dynamic are all the units that you can move through the map, while the static are the fixed one that give you an kind of bonus per turn.

2. 1 Types of units

   - Type : Functional.
   - Summary : After being separated between dynamic and static units, the units will be "separated" into air unit, land unit and sea unit.
   - Description : Each unit will have an specific type, that will limit this unit actions to his kind of terrain/type of hexagon.

3. Size of units

   - Type : Functional.
   - Summary : The units will be separated into small, medium, big units.
   - Description : you can have three sizes of dynamic units. Small that will have good movimentation and bad power of attack (soldier capacity). Medium that will will have relevant movementation and relevant attack power (soldier capacity) and Big that will have bad movimentation and good power of attack(soldier capacity).

4. Creation of Unit

   - Type : Functional.
   - Summary : Each unit will be created from some especial hexagon in the map.
   - Description : The units to be generated on the map, need to be created based on an special hexagon considered the main base of the player, Every unit need and minimal ammount of soldiers and an specific value of resource.

5. Movement of Unit

    - Type : Functional.
    - Summary : Each unit can move inside an radius by turn.
    - Description : After an unit be created you need to wait one turn and then youre able to move this unit in the radius of this unit only once by turn.

6. Attack

    - Type : Functional.
    - Summary : Each unit can attack an enemy unit in an small radius.
    - Description : Each turn the player has the option to attack an enemy, if one of the rules of the combat happen. If two units are placed at the same hexagon on the map an attack occurs. If two units get neighboor in the hexagon a battle also occurs. if one enemy is inside the attack range of an unit is option of the player start an attack or not.

Advanced

1. Merge Of Units

    - Type : Functional
    - Summary : If you have two units damaged (without all the soldier) you can merge both.
    - Description : If you have two units inside the range of each other of movement and both are damaged (with less soldiers) and the capacity of one of this units if big enought to hold the sum of both units soldiers you can just merge this units and then "destroy" the "smaller" unit.

2. Refill Units

    - Type : Functional.
    - Summary : Refill an damaged unit with more soldiers to recover the power of each single unit.
    - Description : When an damaged unit has at least one soldier inside and the player want to keep that position he can just refill the tank with more soldier, and with an small ammount of resource, this action cost more than one turn because is the time of send an helicopter with soldiers to refill the tank with how many soldiers the player want and then keep playing.

Complex

1. Follow and destroy (Attack 2.0)

    - Type : Functional/Non-functional
    - Summary : To engage an battle with an enemy unit that run away.
    - Description : Every unit can attacl an enemy unit if the enemy unit is inside his range of attack, but sometimes the enemy can just run away to any other direction, avoiding the enemy and leaving the fog/attack area of the "attacker" unit. For this we create an Follow command, that if you find an enemy unit inside your range of attack you can just say follow, and then your tank will move over the enemys unit or to the maximum of his movement range, in the direction that the other unit has moved to run away. If the enemy stay inside the attack area an battle occurs.

---

7 Stefanos Gatsios

7.1 Multiplayer

Basic

1. Grouping in teams / co-op

    - Type : Functional.
    - Summary : Able to cooperate with your fellow players
    - Description : Using shared resources, donating units and helping assaults from other friendly players.

Advanced

1. Chatting

    - Type : Functional.
    - Summary : Ability to chat with co-players
    - Description : Each player will be able to chat with his fellow players during the game, and with all participating in the game (including enemy) after the end of the game.

Complex

1. Inappropriate words in chats

    - Type : Non-functional
    - Summary : Be able to detect rude players and ban them from the game.
    - Description : Rude players who violate rules and regulations will be either frozen account for a period of time, or banned indefinitely from the game.

Optional

1. Revenge players (TBD)

    - Type : Functional.
    - Summary : If a player loses to another player he will have option for revenge.
    - Description : After a game has been complete, the lost players will have the ability to challenge them on a rematch.

8 Orestis

8.1 Game Logic

Basic Requirements

1. Decision Making

    - Type : Non-functional
    - Summary : Decisions must be provided to the player
    - Description : In order for meaningful strategy to exist, multiple choices must be present.It provides motivation and,above all, fun.

2. Understandable game mechanics

    - Type: Non-functional
    - Summary: Reducing game mechanics complexity
    - Description: If the game mechanics are difficult to comprehend players will immediately get bored. This involves a limited amount of unit types(three), clear combat mechanics, etc.

3. Offence-Defence Aspect

- Type: Functional
- Summary: Clear differentiation of attack and defence.
- Description: When a combat occurs, one player is attacking and the other one is defending. This has impact on the damage factors, soldiers at stake. This will be decided by the previous movement (that brought the units to the point of battle) and the position on the map (according to headquarters location).

4. Punishment for non-strategic play
   - Type: Non-functional
   - Summary: Strategy is mandatory!
   - Description: When a thoughtless move is made by a player, bad consequences will arise. For example, if a player just produce tanks with no forethought and sends them to slaughter, the other player will just produce a few anti-tank units and win.

Advanced Requirements

1. Risk/Reward Aspect
   - Type: Non-functional
   - Summary: Every decision has its price
   - Description: When a player is brought to the point of a choice (e.g. selecting between two equally priced units), this decision will help him only in a few situations, but prove a drawback in others.
2. Strategy VS luck
   - Type: Functional
   - Summary: Tradeoff between predefined and random elements.
   - Description: As we gain luck, using some random generation in the mechanics' computations, we lose the element of strategy and vice versa. Of course, randomness means more fun to the game. A perfect balance must be implemented for the game.
3. Less variables, more choices
   - Type: Functional
   - Summary: Not too many entities in addition with many possible moves.

- Description: Having a great number of different units, confusion will arise. Emphasis on multiple choices within those limited units is mandatory.

4. Drawbacks coupled with benefits

   - Type: Functional

   - Summary: Every choice should have different impact on different situations.

   - Description: When a player is required to select one of many choices, each one will be providing positive statistical feedback on different aspects of the game (e.g. anti-ship tank VS anti-plane tank).

5. Restricted player knowledge

   - Type: Functional

   - Summary: Concealing implementations of the system and certain elements of the environment.

   - Description: For interest to be continuously present in the game flow, information should appear in different time positions (e.g. fog of war to conceil map / reveal when units provide visibility).

Complex Requirements

1. Diversity VS Homogeny

   - Type: Functional / Non-functional

   - Summary: Stable/unstable element in the available paths.

   - Description: Supported by the rpg elements of the game, each player will be able to create a unique path that will have a huge roll on team-based battles (e.g. a team of a navy admiral and an explosives expert would benefit from an economy expert and achive great balance). There will be stable paths that can withstand any diverse situations and unstable ones, which will provide the player freedom to decide how he will adapt to different situations.

2. Long-term Goals

   - Type: Functional / Non-functional

   - Summary: Development elements outside of one single battle

- Description: There should be a constant improvement of a player as long as he plays the game, which will be mainly supported be the rpg system outside a battle. Of course, the previous greatly influence functional things inside the game (additional units, different variables' percentages).

Optional Requirements

1. Multiple roads to victory
   - Type: Functional
   - Summary: Multiple strategy methods may lead to victory
   - Description: There shouldn't be a single set of moves that are guaranteed to succeed. Instead, many possible moves will generate a different strategy context.
2. No obvious choices
   - Type: Functional
   - Summary: Omission of totally determined parameters
   - Description: If a player has a single option for one thing, it shouldn't be an option, but implemented by the system.
3. Clear feedback of choices
   - Type: Functional
   - Summary: Positive/negative feedback to player choices
   - Description: A choice must provide feedback in short time, so players will be able to develop a concrete,a assured strategy.

8.2 RPG Elements

Basic Requirements

1. Unit experience
   - Type: Functional
   - Summary: Upgrade units after certain achievements
   - Description: By winning combats and discovering environment entities, units will gain levels, providing additional bonuses to their properties (soldier capacity, movement points, statistical specific variables, etc. . .).

2. Player overall development system

    - Type: Functional
    - Summary: Character experience system
    - Description: Apart from the battles themselves, a constant entity representing the player will gain leves, which will give him access to new units, give him certain percentages for multiple variables and build a unique path for the player.

3. Randomness in combat

    - Type: Functional
    - Summary: Selection of a range of quantities(w/ randomness), instead of predefined ones.
    - Description: A lot of outcomes in the game will depend on a random element (rolling a dice), so as not all choices are possible to calculate. That is mandatory for a game of one-day turns, as so many hours will give the opportunity to calculate every parameter, if all information is revealed.

Advanced Requirements

Complex Requirements

1. Unique player paths

    - Type: Functional
    - Summary: Different talent trees
    - Description: When a player levels up, talent points will be awarded that can be spent for evolution in a different tree. That will provide many possible talent trees with different pros and cons. It also produce great interest in team-based games where the choice a which players should cooperate will provide many meaningful aspects to analyse.

2. Player-Character Identification

    - Type: Non-functional
    - Summary: Make the user play the role of his character
    - Description: The player should identify himself with his character, so as more sentimental connection is present in the game. Gaining levels will be equivalent to improving oneself's personality (restricted by ethical constraints)

3. Unique hero units

   - Type: Functional
   - Summary: Existence of very strong single units with special abilities
   - Description: Single and unique strong units can be produced that can execute very special actions and give a nice variety on the gameplay (e.g. a diplomat who can use “Ceasefire” , making battles impossible for X turns).

Optional Requirements

1. Unique appearance

   - Type: Non-functional
   - Summary: Player can parametrize the visual aspect of the game
   - Description: The player is able to decide the appearance of his units,building,etc... completely independent from the gameplay. This is an additional patch to the requirement ‘Player-Character Identification’

9 Lokesh

9.1 Combat Mechanics

Basic Requirements

1. Winner/Loser Decision

   - Type: Functional
   - Summary: When a combat has ended the player with more soldiers wins and the other players loose.
   - Description: Each player involved in a combat will have an amount of soldiers, according to their unit, and each one will be lost each time the player looses a round in the combat. Therefore the player remaining with soldiers will be the winner and the other players will be the loosers. The decision of the loss of soldiers is computed by the dice algorithm.

2. Combat initialization

   - Type: Functional
   - Summary: When two or more players are in a certain range a combat is initialized.

- Description: When two or more players are within a certain range(i.e., in the same hexagon), each player will be notified that they are about to participate in a combat.

3. Attacker/Defender choice

    - Type: Functional

    - Summary: When a combat is initialized one player will be the attacker and one will be the defender.

    - Description: Depending on some factors(i.e., map position, which player is more closer to his base, and previous movements) it will be chosen which player is defender and which is the attacker and the dice algorithm will be run to decide which player loses a soldier as this algorithm will be partially random.

Advanced Requirements

1. Attack to bases

    - Type: Functional

    - Summary: A player can attack an opponent player's base.

    - Description: If a player has the base of an opponent player's base within the range of their attack unit, then the player will be able to attack the base and again the dice algorithm will be runned to decide who looses a soldier.

Complex Requirements

1. Alliance option (TBD)

    - Type: Functional

    - Summary: During the start of a combat if there are more than 2 players involved then alliances can be made.

    - Description: If more than 2 players are involved in a combat then all the players will have an option to choose for an alliance and if 2 players choose each other as their alliances then they both will attack or defend as a team.

Optional Requirements

1. Prize(currency) if a battle is won.

    - Type: Non-functional

- Summary: The player who wins a battle will receive a prize.

- Description: After each battle the winner of the battle will receive a prize(i.e., more currency) which will act as motivation factor in the game.

10 Lokesh and Hanieh

10.1 Server Side

Basic Requirements

1. Player statistics (LOKESH)

   - Type: Functional

   - Summary: All information about each player will be stored.

   - Description: All information(i.e., the player's battles, combats, current experience level and personal details) about each player will be stored in a database on the server. The statistics regarding the gameplay about each player will be public so that other players can see for example the rank of a player or the number of battles won by a player.

2. User Notification (LOKESH)

   - Type: Functional

   - Summary: Players will receive a notification about the game.

   - Description: Players will receive notification(i.e., by email, facebook or whatsapp) informing them when ever a movement or action is taken place in the game.

3. Performing moves and actions (LOKESH)

   - Type: Functional

   - Summary: All moves and actions submitted by the players will be carried out by the server.

   - Description: So after each turn all the submitted moves and action will be performed by the server(i.e., destroying a unit if a player has lost a combat, moving units and etc. . .)

4. Maintaining Server up all the time (HANIEH)

   - Type: Functional

- Summary: keep the server in the stand-up mode .
- Description: since this game is asynchronous, the server should be kept in this mode for ever if not players will face difficulties.

Advanced Requirements

1. Login/Account management (HANIEH)
   - Type: Functional
   - Summary: having the user login.
   - Description: Each user can register by creating username and password.
2. Battle replays storage (HANIEH)
   - Type: Functional
   - Summary: Battles will be recorded to be replayed
   - Description: Since the beginning of the battle until the end, the whole battle will recorded saved with all its statistics.

Complex Requirements

1. Inactive Accounts (LOKESH)
   - Type: Functional
   - Summary: If accounts that have not been logged in since a long time will be declared inactive.
   - Description: Users that don't log in their accounts for a long time, then this account will be declared inactive and no details about this account(player) will be public.

Optional Requirements

11 Hanieh

11.1 User Interface

Basic Requirements

1. Keyboard control
   - Type : Functional

- Summary : is one way to provide input to this video game.

- Description : typically to control an object or character in the game. When a user presses the button corresponding to a certain movement, the system shall bring her to that position.

2. Mouse control

   - Type: Functional

   - Summary: ability to play the game with mouse.

   - Description: user can move and control his game by mouse.
     user can pinpoint his/her shots better and with a high performance.

3. Graphics

   - Type: Non-functional

   - Summary: visualization of the game.

   - Description: each entity is a sort of image.this requirement is consisted of all units according to their resources have specific graphic view. for example building(metal), forest(woods), sea and lake(water).

4. Basic user interface window

   - Type: functional

   - summary: revealing basic information about each unit.

   - Description: when user creates units from his headquarters and main base, he can see all of the available actions of unit according to that specific unit.

5. Animation of performed actions

   - Type: Functional

   - Summary: visualization of execution of turns.

   - Description: when both players have submitted their moves,the server implements all necessary actions and animates them in front of the user's eyes.

6. Map screen

   - Type: Functional

   - Summary: user can see the whole map of the game.

- Description: user can have the whole map during the game, then he has general overview of the tactical situation. It allows user to jump to every position where he clicks on the map.

Advanced Requirements

1. Sounds
   - Type: Functional
   - Summary: put sounds in every functionality in the game.
   - Description: Audio feedback of user interaction.
     for example: when a player clicks on a building its unique sound is heard.
2. Background music
   - Type: functional
   - Summary: having music on the background while the game is running.
   - Description: to create a nice atmosphere for the user ambient music should discretely exist.
3. Main menu
   - Type: Functional
   - Summary: basic operations of the game, such as: quit,resume, change controls.
   - Description: at any moment of the game user can quit the game, change controllers and resume the game.
4. Usability
   - Type: non-functional
   - Summary: user friendly.
   - Description: delivering a better and deeper experience with less unnecessary interruptions for user during the game.
     and make playing more fun for users.

Complex Requirements:

1. Unit grouping
   - Type: Functional

- Summary: Visual grouping of many kinds of units.

- Description: if user selects many kinds of units, he can have the set of all these units in one group.

Optional Requirements:

1. Help menu

   - Type: Functional

   - Summary: allows user to solve his doubts about game.

   - Description: allows the user to search the all rules in order how to play the game.

2. pop-up window

   - Type: functional

   - Summary: having the menu by hovering the mouse pointer above a unit.

   - Description: user can see all the informations and all the properties of the unit which the mouse pointer is over on. (for example: amount of soldiers, types of unit)

12 Trace Matrix

	Map Package	Hexagon	Unit Package	Combat	Resources	Gameplay	Server	Database	Player	Client.UI	Client.Assets
Map Existance	X										
Units			X	X							
Currency					X						
Turn Based Logic						X					
Combat				X							
Server							X				
Server Response Time							X				
Map of Hexagons	X	X									
Different Type Hexagons		X									
Prototype Maps	X										
Random Prototype maps	X										
Parametrized random maps	X										
Pure random	X										
Groups							X	X			
Chatting							X	X			
Chat filter							X				
Revenge											
Offense Defense				X							
Str vs Luck				X	X						
Restricted Knowledge	X										
Unit XP			X						X		
Player XP									X		
Random Combat				X							
Unique player path									X		
Unique Hero units			X								
Win/Lose Decision				X							
Attack Action				X							
Combat initialization				X							
Attack Base				X							
All option				X							
Prize				X	X						
Player Stat							X	X	X		
User Notification							X	X	X		
Battles Storage							X	X			
Login							X	X			
Error Handling							X				
Keyboard										X	
Mouse										X	
Graphics										X	
Basic UI window										X	
Meno										X	
Sounds											X
Pop-ups										X	
Help Menu										X	