Software Functional Specification

Revision – 1.0

Last printed 4/13/2010 11:04:00 A4/P4

Duel Reality

Approval Block

Name	Responsibility	Initials
Josh Kilgore (Team Lead)	Game Mechanics & AI	
Obi Atueyi	User Interface	
Tom Calloway	Graphics Window	
Ye Tian	Database	

Abstract (Tom)

The following is a software functional specification document for the Duel Reality personal computer game. The document fully identifies and describes both the high and low level functionality of the software without going into the design details themselves. This document attempts to serve the needs of those looking to understand the functional requirements of the game from both the user and designer perspectives.

Revision History (Tom)

Revision	Date	Revised By	Comments

Table of Contents (Tom)

List of Figures (Tom)

1 Introduction (Obi)

This document describes the functions of the Duel Reality modules, in accordance with its Architectural Specification [2].

Interactions between the user and the game are provided via the User Interface (UI). The UI is the main window that comprises the menu, toolbar, and status bar. It is through this interface that the user sets the desired game play options and receives error messages during battles.

The main window also contains a game view that comprises the map and player unit. This is the Game Graphics and it provides the user a visual representation of the state of the game during a battle.

The Game Mechanics provides the state of the game to the Game Graphics. In response to the user's turns during battle, the Game Mechanics uses the game play options entered at the UI and interactions with the Artificial Intelligence (AI) to define the state of the game at any time.

The AI is the user's opponent that adapts to the game level and the user's units. It retrieves game and user information from the Database and uses this information to model an opponent suitable for the user's experience level.

The Database stores information provided by the AI and UI. It also fetches information on request by the AI and UI. The Database provides permanent storage of such information for use during future program runs.

1.1 Document Purpose (Tom)

The purpose of this document is to provide the functional specifications for the development of the Duel Reality turn based strategy role playing game.

1.2 Product Scope (Josh)

This game is meant to be run on a solitary Windows PC. User interactions will take place with a standard mouse and keyboard. Graphics will be displayed on the screen and sound played through the standard sound output device of the computer.

1.3 Terminology

Sprite	A two-dimensional pre-rendered figure
Unit	An individual infantry
Widget	An interactive feature pertaining to user interface graphics

1.4 Acronyms

AI Artificial Intelligence

AP Action Point

XP Experience Point

GUI Graphical User Interface

UI User Interface

TBS Turn-Based Strategy

2 Overall Description

2.1 Functionality Workflow (Tom)

One method of describing the functionality of the Duel Reality PC game is with a set of state diagrams which depict the various physical states and transitions of the product in use. The functionality of Duel Reality can be broken down into three processes which are the main game process, the level-up process, and the save / load state process.

2.1.1 Main Game Process (Tom)

The main game process refers to the core functionality of the game, excluding the saving / loading, and leveling up processes. The main game process includes running the game, registering and logging in, selecting a map, and participating in a battle. The general flow of this high level functionality can be seen in below.

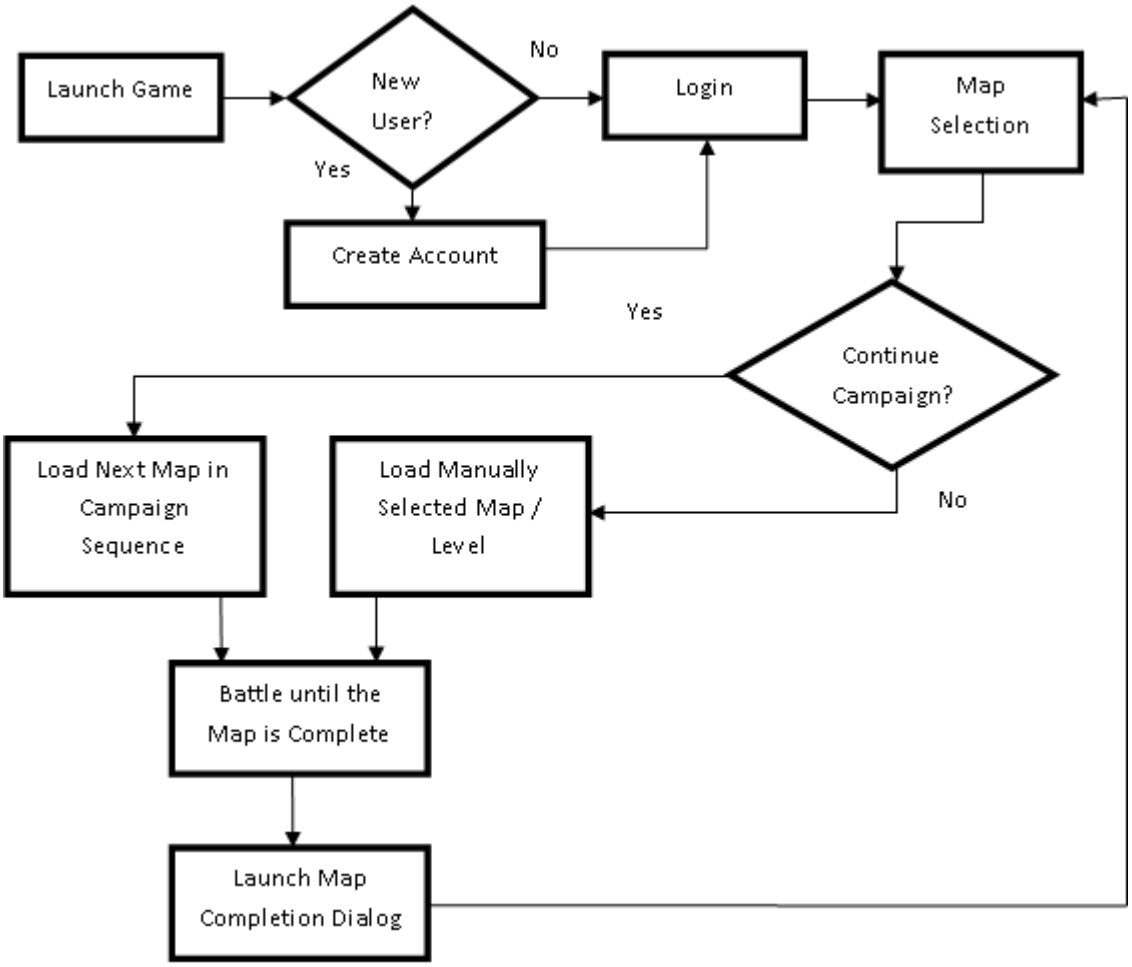

Figure 2-1: Main Game Process (Tom)

2.1.2 Battle Process (Josh)

A subset to the main game process, the battle process will take place between a human player and an opponent who is either human or computer controlled. The battle will start with player one and each turn of the players will consist of movement and actions until the number of action points is spent. The players will then switch turns. This will continue until one team has defeated all units of the opposition, as shown below in Figure 2-2.

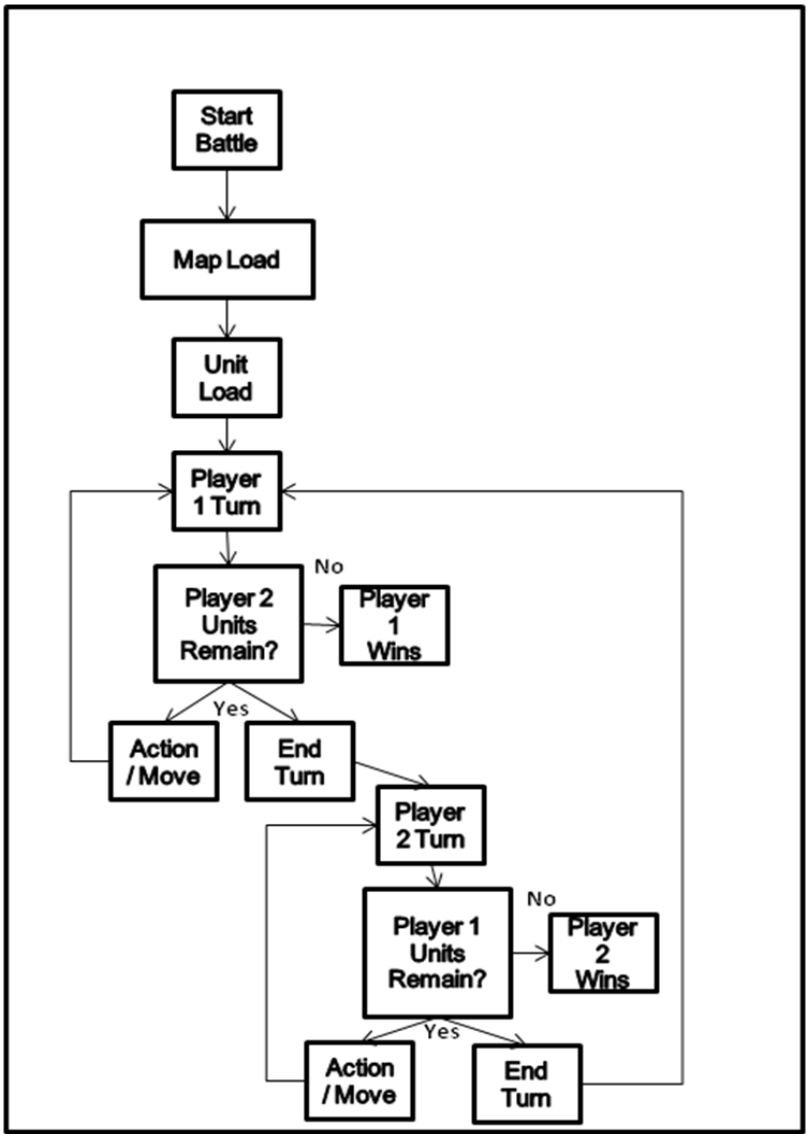

Figure 2-2: Battle Process

2.1.3 Level-Up Process (Tom)

The level-up process refers to the ability for users to spend points earned by completing battles and advancing through the main campaign. This process is activated at specific times during the gaming process as detailed in the User Interface section of this document. The leveling up process flow is illustrated in below.

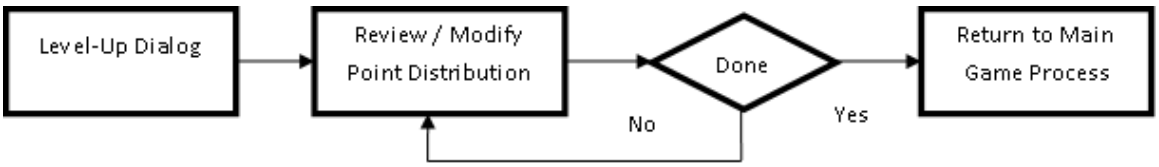

Figure 2-3: Level-up Process (Tom)

2.1.4 Load / Save Game Process (Tom)

When the load / save game dialogs are launched, users shall have the option to either save the state of their unit levels, level-up point distribution, and current battle, or they may choose to load a previously saved state. This decision process is detailed in below.

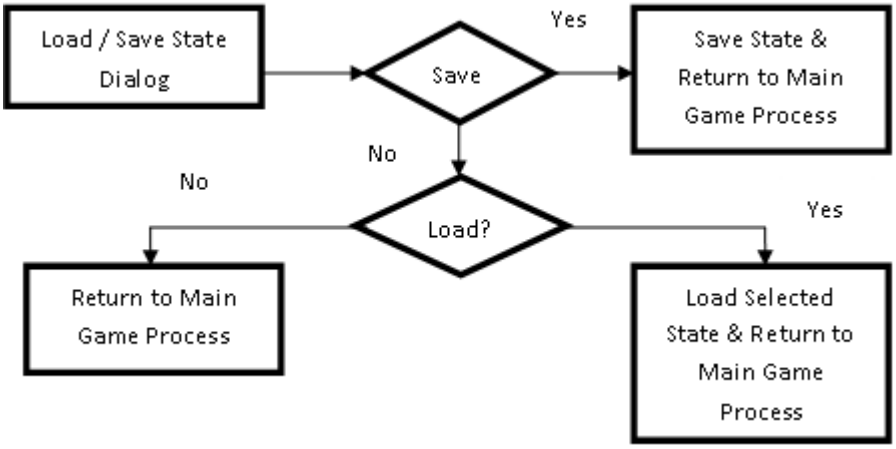

Figure 2-4: Load/Save Game Process (Tom)

2.2 Design Constraints (Josh)

Some of the limiting factors in our design and execution of this project include the time limit of having the project due on a certain date and being limited to C++ for development language.

2.3 Assumptions and Dependencies (Josh)

We assume that C++ object oriented programming would be sufficient to do game design. That using QT tools adds value to this process, that time constraints would be sufficient to make a working game, and that we were assuming a Windows environment for deployment.

3 Functional Requirements

3.1 System Requirements (Tom)

3.1.1 Duel Reality ***shall*** be distributable via a single folder, containing all files and sub-folders required to make full use of the game.

3.1.2 Duel Reality ***shall*** run on Windows PCs running the XP, Vista, or Win 7 operating systems.

3.2 User Interface (Obi)

3.2.1 Description

Although the UI module is, in general, the application display to the user, it will be separate from the game graphics. The UI enables the user to configure game options for each level of the game. The menu options are used to start, load, save or quit a game. During a battle, the user uses the tool-bars to move or perform actions on units. Any invalid moves or actions during battle are indicated on the status bar.

3.2.2 Functional Requirements

3.2.2.1 New Game Dialog

3.2.2.1.1 User ***shall*** be able to start a new game only at program start or after winning a battle.

3.2.2.1.2 On New Game request, user ***shall*** be prompted to enter player name.

3.2.2.1.3 If player name exists, user ***shall*** have the option to either continue campaign from saved game data; or start a new campaign.

3.2.2.1.4 If new player, user ***shall*** create a new account.

3.2.2.1.5 User ***shall*** be prompted to select unit count and types.

3.2.2.1.6 Dialog ***shall*** complete after user has entered unit count and types.

3.2.2.1.7 After prompts are complete, battle ***shall*** start according to data entered in New Game dialog.

3.2.2.2 Save Game Dialog

3.2.2.2.1 User ***shall*** be able to save a game only during a battle.

3.2.2.2.2 Only game data for battle in session ***shall*** be saved.

3.2.2.2.3 On Save Game request, previously saved game data ***shall*** be overridden with game data for current game.

3.2.2.2.4 After game is saved, battle ***shall*** resume in the exact battle configuration prior to Save Game request.

3.2.2.3 Load Game Dialog

3.2.2.3.1 User ***shall*** be able to load a saved game only if there is no battle in session.

3.2.2.3.2 On Load Game request, user ***shall*** be prompted to select from a list of saved games.

3.2.2.3.3 After user selects game, user ***shall*** have the option to level up.

3.2.2.4 Battle Over Dialog

3.2.2.4.1 If battle was won, user ***shall*** have the options to quit game, restart game, or continue to next level.

3.2.2.4.2 If battle was lost, user ***shall*** have the options to quit game or restart game.

3.2.2.5 Level-up Dialog

3.2.2.5.1 User ***shall*** be able to level-up after loading a saved game or winning a battle.

3.2.2.5.2 On Level-up request, map ***shall*** advance to next level.

3.2.2.5.3 User ***shall*** be able to advance units by spending experience points (XP).

3.2.2.5.4 After level-up complete, battle ***shall*** resume according to data entered in Level-up dialog.

3.2.2.6 Restart Game Dialog

3.2.2.6.1 User ***shall*** be able to restart a game at the end of a battle.

3.2.2.6.2 On Restart Game request, user ***shall*** be prompted to select unit count and types.

3.2.2.6.3 After prompts are complete, battle ***shall*** start according to data entered in Restart Game dialog.

3.2.2.7 Game Over Dialog

3.2.2.7.1 Game ***shall*** complete after user wins battle in the last level.

3.2.2.7.2 After game is complete, a congratulatory message ***shall*** be maintained on the display.

3.2.2.7.3 User ***shall*** have the option to exit program, start new game or load saved game.

3.2.2.8 Quit Game Dialog

3.2.2.8.1 User ***shall*** be able to quit a game during a battle or at the end of a battle.

3.2.2.8.2 On Quit Game request, game ***shall*** end and user ***shall*** have the options to exit program, start new game or load saved game.

Figure 3-1 shows a continuous flow of the game with points at which user dialogs are accessible. The Quit Game dialog and Game Over dialogs are not shown since they are precursors of the game end.

3.2.2.9 Actions and Movement Buttons

3.2.2.9.1 User ***shall*** be able to act or move a unit only during a valid turn.

3.2.2.10 Error Message

3.2.2.10.1 Any invalid action or movement ***shall*** be indicated in a pop-up on the status bar.

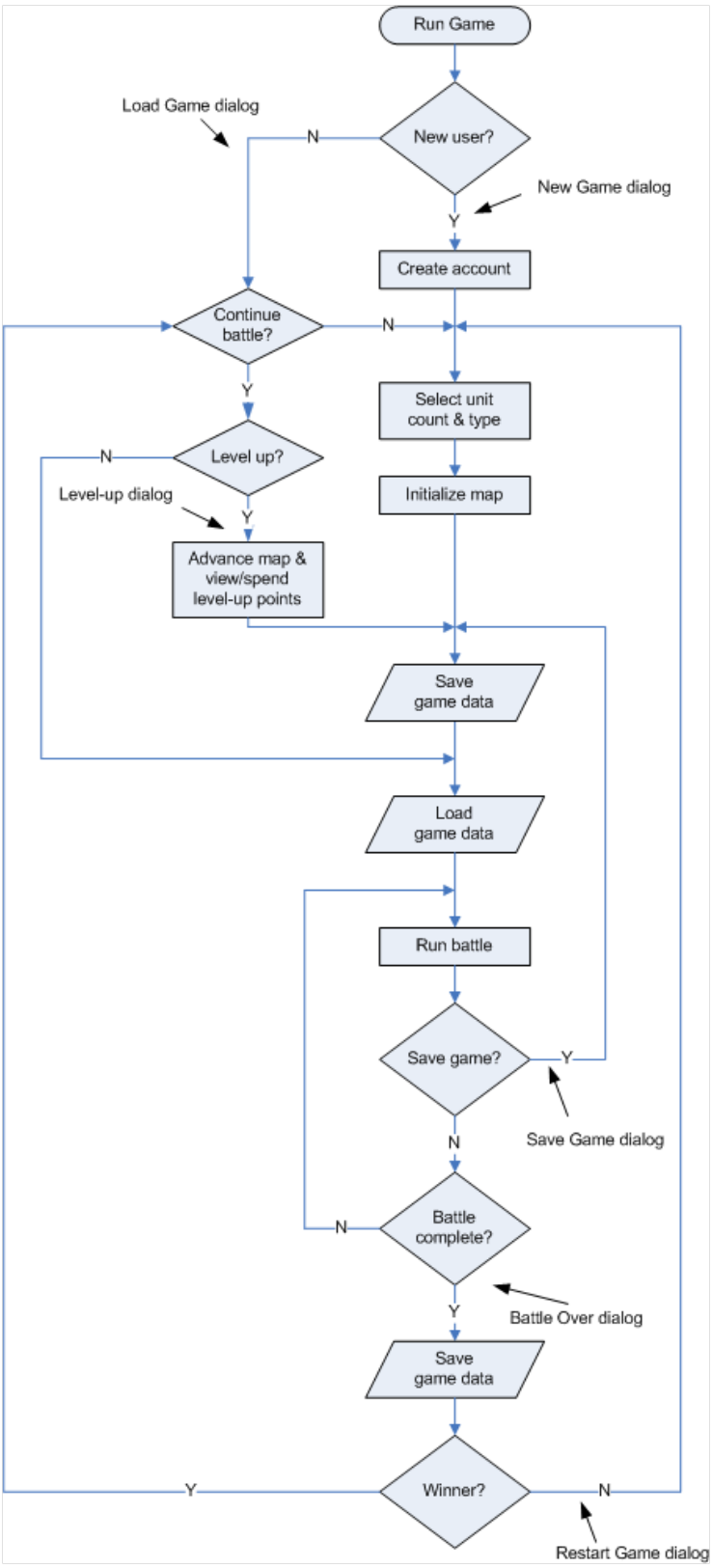

Figure 3-1: Game flowchart Showing Points of User Dialogs

3.3 Game Graphics (Tom)

3.3.1 Description (Tom)

The main dialog of the Duel Reality PC game features a game graphics window, which represents the map and units currently in play. As such, the graphics window is the user's primary feedback regarding the current state of the game. The graphics window will include a background image, two-dimensional sprites to represent units, and a grid representing the discrete locations of the map. The game graphics window will also feature sounds and visual effect to represent significant game events.

3.3.2 Graphics Window Functional Requirements (Tom)

What follows is a list of functional requirements for the game graphics module.

3.3.2.1 Duel Reality ***shall*** feature a graphics window in the main dialog which takes up at least 75% of the main dialog's user interface

3.3.2.2 The Duel Reality graphics windows ***shall*** accurately reflect the current state of the game as determined by the game mechanics and AI modules as well as the users themselves.

3.3.2.3 When a battle is not in progress, the Graphics Window background ***shall*** be a static image.

3.3.2.4 When a battle is in progress, a grid corresponding to the current map ***shall*** be overlaid on top of the battle background.

3.3.2.5 The graphics window ***shall*** resize itself when the parent form is resized such that it always comprises the same percentage of the parent form's total height and width.

3.3.2.6 When the graphics window is resized, all graphics and images contained within it ***shall*** scale and resize smoothly and proportionately.

3.3.2.7 The graphics windows shall represent every unit present with a graphical "sprite".

3.3.2.8 All unit sprites within the graphics window ***shall*** fit horizontally within one grid cell and be less than 2 grid cells tall.

3.3.2.9 During battle game play, when a grid cell is clicked, the grid cell ***shall*** be selected.

3.3.2.10 Selecting a grid cell ***shall*** cause additional information about the contents of that cell to be displayed along the top of the graphics window.

3.3.2.11 The graphics display ***shall*** handle "move" events by moving units from one cell to another and triggering a visual and auditory event.

3.3.2.12 The graphics display ***shall*** handle "attack" events by reducing the displayed health of the target of an attack and triggering a visual and auditory event.

3.3.2.13 The graphics display ***shall*** display the current action points and hit points for every unit on the map at all times during battle.

3.3.2.14 The graphics display ***shall*** handle “defeated” events by removing defeated units from the map and triggering a visual and auditory event.

3.3.2.15 The graphics display ***shall*** handle “load map” events by loading the selected background, adding the correct grid, and adding the correct player and AI units.

3.4 Game Mechanics (Josh)

3.4.1 Description

The Game Mechanics module comprises the rules about how the different parts of the program work together to present the player with a full experience. In this section are defined the units, their interaction with the board and the players, the basic rules of the game including movement and actions and the battle process as shown in Figure 2-2.

3.4.2 Functional Requirements(Josh)

3.4.2.1 Units

3.4.2.1.1 A battle ***shall*** begin once the player has launched the game from the Settings dialogue.

3.4.2.1.2 A battle ***shall*** conclude when all the units of one of the players are 'dead', or the user chooses to end the game.

3.4.2.1.3 If the player has won, the player ***shall*** be shown the Upgrade screen.

3.4.2.1.4 If the player has lost, the player ***shall*** be asked if they want to exit program, start new game or load existing game.

3.4.2.1.5 The game ***shall*** last for 7 boards.

3.4.2.1.6 The 7th level ***shall*** be a boss fight, after which the player will have won the game.

3.4.2.1.7 Upon winning the game, the player ***shall*** be shown the Game Won screen.

3.4.2.1.8 Units ***shall*** not be able to occupy the same game space with obstacles or other units.

3.4.2.1.9 Attempting to occupy the same game space as an obstacle or other unit ***shall*** not result in a penalty, simply a return to the starting position of the move and post an error to the screen.

3.4.2.1.10 Units ***shall*** begin the game at random locations on that player's respective side of the screen.

3.4.2.1.11 Players ***shall*** take turns moving units.

3.4.2.1.12 Each Player ***shall*** end the turn they are currently on, then control will switch to the other player to make their turn.

3.4.2.1.13 A turn ***shall*** consist of moving and using actions of one unit until the player ends the turn.

3.4.2.1.14 Clicking on a unit ***shall*** cause information about that unit to be displayed, as well as to visually indicate that it has been selected.

3.4.2.1.15 Units ***shall*** have the following attributes: Health, attack power, action points, attack range and experience points(XP).

3.4.2.1.16 Health and action points ***shall*** be represented on the screen.

3.4.2.1.17 Health ***shall*** decrease when a unit is attacked successfully.

3.4.2.1.18 Health ***shall*** increase when a unit is healed, up to the maximum of the unit.

3.4.2.1.19 When health reaches zero, that unit is 'dead' and ***shall*** not be playable for the remainder of the round.

3.4.2.1.20 Attack Power ***shall*** determine how much an attack from a unit will do to an enemy unit.

3.4.2.1.21 Units ***shall*** only be able to move and do actions such as attack, up to their allotted amount of action points each turn.

3.4.2.1.22 Unit movement and actions ***shall*** decrease the number of action points for the unit that turn.

3.4.2.1.23 Action Points ***shall*** be replenished at the start of each turn.

3.4.2.1.24 Attack range ***shall*** determine how many game squares away an opponent has to be before the attacking unit can attack the target.

3.4.2.1.25 Units ***shall*** have an associated amount of experience points (XP).

3.4.2.1.26 The value of XP ***shall*** be different for various unit types and levels.

3.4.2.1.27 The player ***shall*** gain XP by killing each enemy unit and having their units alive at the end of the match.

3.4.2.1.28 XP ***shall*** be calculated and awarded at the end of a match.

3.4.2.1.29 Total XP gained over the course of the game shall determine player ranking and ***shall*** not be affected by XP spent on unit upgrades.

3.4.2.1.30 XP ***shall*** also be used to buy upgrades for units.

3.4.2.1.31 Available upgrades ***shall*** be: increased attack power, increased health, and increased action points per round.

3.4.2.1.32 Upgraded attributes ***shall*** be available for all player units in subsequent rounds.

3.4.2.1.33 Each time a unit upgrade is purchased, the cost for subsequent upgrades ***shall*** increase.

3.4.2.2 Movement

3.4.2.2.1 Units ***shall*** be able to move vertically and horizontally between game squares on the grid of the game board.

3.4.2.2.2 Movement ***shall*** be initiated by clicking on the unit to be moved, the move button, and then the desired destination.

3.4.2.2.3 If the destination is occupied, then the unit ***shall*** not move.

3.4.2.2.4 If the destination is beyond the number of action points available to the unit, then the unit ***shall*** not move.

3.4.2.2.5 If the destination is within the action points available to the unit and is unoccupied, then the unit ***shall*** move.

3.4.2.2.6 Units ***shall*** not move diagonally between game squares.

3.4.2.2.7 Moving between game squares ***shall*** cost a unit a set number of action points.

3.4.2.2.8 Units ***shall*** not be able to move beyond the limits of the game board.

3.4.2.2.9 Units ***shall*** not be able to loop from the top row to the bottom or from the first column to the last.

3.4.2.2.10 A unit ***shall*** not move through an obstacle or other units.

3.4.2.2.11 Units ***shall*** have the option of not moving.

3.4.2.3 Actions

3.4.2.3.1 Units ***shall*** only be able to attack units of the other team.

3.4.2.3.2 Units ***shall*** only be able to heal units of the same team.

3.4.2.3.3 Actions ***shall*** be initiated by clicking on the unit to do the action, the button for the action type and the target unit.

3.4.2.3.4 Units ***shall*** only be able to attack/heal units within their attack range as determined by their attack range attribute.

3.4.2.3.5 Units ***shall*** only be able to attack while the have sufficient action points.

3.4.2.3.6 Units ***shall*** not be able to attack through obstacles unless they have an attack range greater than 1, and are therefor ranged units like archers.

3.5 AI (Josh)

3.5.1 Description

As shown in Figure 3-2, the AI will consist of a set of decisions and actions which will mimic the actions of another human opponent for the single Player to play against. The AI will operate under the same constraints as the Player in terms of gameplay, but will not have the ability to automatically upgrade its capabilities. This AI model will be sufficient for the purposes of this game, and can scale easily into a medium complexity.

.

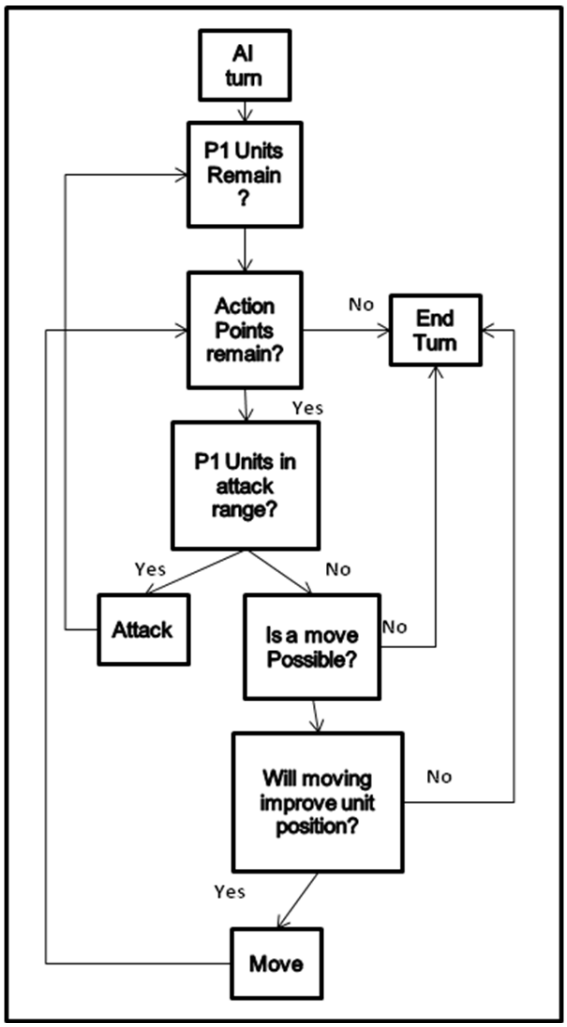

Figure 3-2: AI flow

3.5.2 Functional Requirements

3.5.2.1 The AI ***shall*** be able to maneuver around obstacles.

3.5.2.2 The AI ***shall*** react to the players actions and movements.

3.5.2.3 The AI shall have ***shall*** have the same, or similar resources with respect to Action Points as the player.

3.5.2.4 The moves and actions of the AI ***shall*** not be hidden to the player.

3.5.2.5 The AI ***shall*** not upgrade units at the end of the round, instead shall have scaling units based on the level played.

3.6 Database (Ye)

3.6.1 Description (Ye)

The game database contains all of the tables and data records, which function like a background support to other modules. It is a collection of the names, parameters, status of all units, and the game content such as maps. The data is retrieved and overwritten in real-time game going and asynchronous backup.

The database module interacts with other modules, but does not affect them. It ***shall*** provide operation function by other modules.

3.6.2 Functional Requirements (Ye)

3.6.2.1 Typical character

3.6.2.1.1 The database ***shall*** include a set of characters.

3.6.2.1.2 Every character ***shall*** have a set of necessary and advanced parameters such as health, attack power, attack range, level, and movement type.

3.6.2.1.3 Three kinds of characters ***shall*** be supported\in the database. The different styles shall be realized by defining three special set of parameters including attractive, defensive and middle.

3.6.2.1.4 Several unit copies of one kind of character ***shall*** be initialized in the database table.

3.6.2.1.5 Every character ***shall*** have a level-up mechanism. Every level corresponds to matched set of parameters.

3.6.2.1.6 The database ***shall*** include a top10 score list that records the highest statistics.

3.6.2.2 Start/Restart Game

3.6.2.2.1 If new game is started or restarted with a map background and a set of opponents and player units, the database table ***shall*** be initialized randomly from the database defined characters parameters.

3.6.2.2.2 The database table ***shall*** record current map pointer.

3.6.2.3 Storage/Retrieval of Map and Unit Data

3.6.2.3.1 The game content such as map and sprite data as well as player unit statistics that synchronized with the database ***shall*** be saved as a data file in defined folder when the game is running.

3.6.2.3.2 In the other way round, all the saved information including the map, sprite data and statistics ***shall*** be reloaded for continuing the game.

3.6.2.3.3 Some shortcut key ***shall*** active the save/load function.

3.6.2.4 Units Status Synchronization

3.6.2.4.1 Database table ***shall*** provide current status to mechanics module.

3.6.2.4.2 During the game running, the resulted parameters of all units in the datable table ***shall*** be updated after any game actions.

3.6.2.4.3 During the game running, the resulted position of all units in the datable table ***shall*** be updated after any unit movement.

3.6.2.4.4 After updating the parameters, special values like XP ***shall*** be compared with the level-up mechanism values.

3.6.2.4.5 Level-up parameters ***shall*** be applied when the comparison is positive. High level enemy units maybe allowed to be used in next battle.

3.6.2.4.6 Database ***shall*** be synchronized with the real-time game evolving all the time.

3.6.2.4.7 Database table ***shall*** be released in the condition of game over.

3.6.2.5 Module interaction

Database module mainly interacts with User Interface module and Game Mechanics module. Some database operation functions such as read function, write function, release function, load function, and level-up function ***shall*** be provided to the other modules. The database utilization flow is shown in Figure 3-3.

3.6.2.5.1 Basic user interface menu such as “save game”, “load game”, and “restart game” ***shall*** read the database predefined parameters or backup file to initialize game ongoing and a synchronized database table.

3.6.2.5.2 If the player fails the battle, the database table ***shall*** be released.

3.6.2.5.3 If some mechanics action or movement happens, the database table ***shall*** be updated.

3.6.2.5.4 If success happens in the battle, some pop-up menus like “level up” dialog ***shall*** also update player units’ status in database table.

3.6.2.5.5 High level parameters of enemy units ***shall*** be allowed to use.

3.6.2.5.6 The updated or level-up database table ***shall*** be saved as backup files anytime during the battle by menu or shortcut key.

3.6.2.5.7 The backup files ***shall*** be loaded into game ongoing and synchronized database table again.

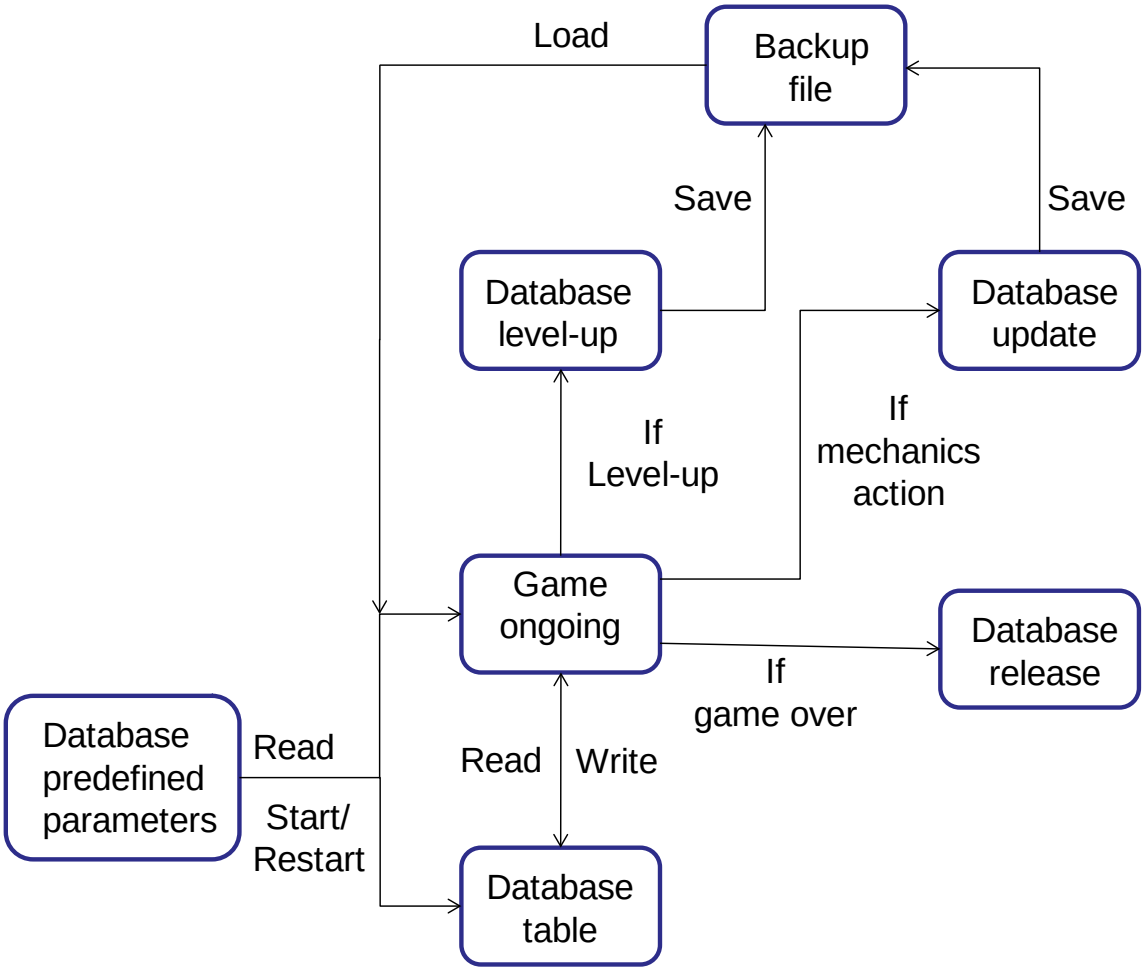

Figure 3-3: Database Utilization Flow

4 References (Ye)

[1] Team Gold: Josh Kilgore, Obi Atueyi, Thomas Calloway, Ye Tian, "Duel Reality: A Turn-Based Battle Strategy Game", Proposal, 02/19/2010.

[2] Team Gold: Josh Kilgore, Obi Atueyi, Thomas Calloway, Ye Tian, "Duel Reality: A Turn-Based Battle Strategy Game", Software Architecture Specification, 03/20/2010.