CENG 492

TEST SPECIFICATION REPORT FOR THE ELERON

ÇAĞLAR GÜNDOĞDU - 1679000

SAMET CAN - 1678812

EMRE AKSAN - 1678655

OKAN HARPUT - 1679042

14.04.2013

Contents

1. Introduction

1.1 Problem Definition

A lot of humans, animals, vehicles are interacting with each other on a daily basis and these interactions may vary in a large-scale. Often understanding and estimating the behaviors of these agents is a hard task. To understand, estimate the interactions of these agents in a large crowd or even visualization is a common problem due to some reasons such as cost of real demonstrations, security issues, etc. For these very specific cases where visualization of the behaviors of agents is important simulation technology helps us.

While imitating the real world and gathering realistic responses are possible, simulating many agents can be costly and usually ineffective with common simulation techniques. The need for crowd simulation arises when a scene calls for more characters than can be practically animated using conventional systems.

To meet these needs, we are going to develop the software called “The Eleron”. This software is basically a game which includes crowd simulation, artificial intelligence and multi-player concept.

In this game, military vehicles such as combat aircrafts, helicopters, and maps will be supported in 3D.

In the game, two or more players will fight against each other as two teams. These teams also include AI controlled units. In pre-game, the players are expected to specify some predefined rules. These rules refer map of the game, weather conditions, formation of the AI units, aircraft types and behaviors of the AI units.

1.2 Purpose

The purpose of the test process is to reveal critical bugs, misbehaviors and performance problems of The Eleron efficiently and as early as possible, so that they can be corrected with minimum effort. The testing process will allow us to improve our system to become user-friendly, free of bugs and performance problems as much as possible.

1.3 Scope of Document

This document briefly describes the testing process of The Eleron project. As in fact the testing process is continuous from the starting date of the project to the end date, this document also includes some information about how the testing has been done up till now in the project. However, the main aim and the scope of this document is to present the testing plan for the remaining part of the project in which the implementation is considered to be nearly complete.

2. Test Plan

2.1 Test Environment

During the testing following softwares and minimum system requirements will be used.

2.1.1 Minimum System Requirements

1. Windows: XP SP2 or later; Mac OS X: Intel CPU & "Snow Leopard" 10.6 or later. Note that the Eleron was not tested on server versions of Windows and OS X.
2. Java(TM) Platform, Standard Edition Runtime Environment 7
3. Graphics card with DirectX 9 level (shader model 2.0) capabilities. Any card made since 2004 should work.
4. Using Occlusion Culling requires GPU with Occlusion Query support (some Intel GPUs do not support that).
5. Keyboard and Mouse

2.1.2 Softwares Used

2.1.2.1 Unity

Unity (also called Unity3D) is a cross-platform game engine with a built-in IDE developed by Unity Technologies. It is used to develop video games for web plugins, desktop platforms, consoles and mobile devices, and is utilized by over one million developers.Unity is primarily used to create mobile and web games, but can also deploy games to consoles or the PC. The game engine was developed in C/C++, and is able to support code written in C#, JavaScript or Boo. It grew from an OS X supported game development tool in 2005 to the multi-platform game engine that it is today.

2.1.2.2 MonoDevelop

MonoDevelop is an open source integrated development environment for the Linux platform, Mac OS X,and Microsoft Windows, primarily targeted for the development of software that uses both the Mono and Microsoft .NET frameworks. MonoDevelop integrates features similar to those of NetBeans and Microsoft Visual Studio, such as automatic code completion, source control, a graphical user interface (GUI) and Web designer. MonoDevelop integrates a Gtk# GUI designer called Stetic. It currently has language support for C#, Java, Boo, Visual Basic.NET, Oxygene, CIL, Python, Vala, C and C++.

2.2 Features to Be Tested

Here are the main features of individual parts of “The Eleron” system to be tested.

2.2.1 Pre-game tests : These tests involves the tests should be done before playing the game.

1. Configuring buttons
2. Selecting spacecrafts, environment and number of agents
3. Transitions between create game, start game, join game screens
4. Player name, game name specifications
5. Interactions of pre-game actions with portico

2.2.2 In-game tests: These tests involves the tests should be done when playing the game.

1. Spacecraft movement control
2. Fire, hit, collision detection
3. Pause, zoom, health bar, minimap, log information control
4. Controlling AI units actions such as follow, attack, patrolling

2.2.3 RTI tests: These tests involves the tests should be done both before the game and while playing the game in the RTI side of the “The Eleron”.

1. Federation creation
2. Joining federation
3. Controlling data flow between unity and portico.

3. Test Procedures

3.1 Main Scenario

We prepared a generic scenario that involves all specific test cases we have written. By simply following this scenario, a tester can easily check all of the test cases.

1. The user starts “theeleron.exe”.
2. The user changes screen resolution, graphics quality and button configuration on the opened screen.
3. The user starts game by pressing “Play!”
4. Main game menu screen is displayed.
5. If user is host player:
    a. User clicks new game.
    b. In create game menu screen host player enters game name, player name, specifies agent number, chooses environment and spacecraft type.
    c. User clicks create game.
    d. Screen is changed to wait menu and the federation is created in RTI with the name of game name, and the federate named with player name joins the federation.
    e. Host player waits for other players to join the federation and click the ready button.
    f. Host player clicks start game button.
6. If user is joining player:
    a. User clicks join game.
    b. In join game menu, specifies player name and selects spacecraft, and enters game name.
    c. User clicks join game button.
    d. Screen is changed to wait menu and if there is a federation with the corresponding game name, the user federate joins the federation and a success message is displayed on the screen. If the federation does not exist, an error message is displayed.
    e. The user clicks ready button.
7. The game starts.
8. The player orders a bot to patrol around the team base.
9. The player moves to the enemy base by controlling spacecraft. All enable movements are applied.
10. The player interacts with an enemy.
11. The player starts attacking and hits the enemy and destroys the enemy.
12. In the log screen, the hit and destroyed events are displayed.
13. An enemy player moves towards the ally base.
14. The bots interact with the enemy, and notify the real players.

3.2 Test Cases

3.2.1 Environment Determination

Test Case ID	TC1
Test Case Name	Environment Determination
Test Case Description	The user can select the game environment, before the game started.
Test Case Objective	To check that game starts with the selected game environment according to the user selection in pre-game menu.
Pre-conditions	The user must start the game.
Steps	1. The user must be in pre-game menu. 2. The user must select “New Game. 3. The user must select one of the available environments from “Map ” section.
Expected System Response	1. Game starts with the selected environment in pre-game menu. 2. If user does not select an environment game starts with the default environment “Deep Space Green With Planet”.

3.2.2 Agent Number Specification

Test Case ID	TC2
Test Case Name	Agent Number Specification
Test Case Description	The user can set the number of agents in game, before the game started.
Test Case Objective	To check that game starts with the selected number of agents according to the user selection in pre-game menu.
Pre-conditions	The user must starte the game.
Steps	1. The user must be in pre-game menu. 2. The user must select “New Game”. 3. The user must enter the number of agents to “Agent Number” section.
Expected System Response	1. Game starts with entered number of agents in pre-game menu. 2. If the user does not enter number of agents game starts without agents. 3. If the user enters a number greater than 10 and tries to start game, game does not start and warns the user that limit is 10 and waits user to re-enter number.

3.2.3 Game Name Specification

Test Case ID	TC3
Test Case Name	Game Name Specification
Test Case Description	The user can set the game name, before the game started.
Test Case Objective	To check that game starts with the selected game name according to the user selections in pre-game menu.
Pre-conditions	The user must start the game.
Steps	1. The user must be in pre-game menu. 2. The user must select "New Game". 3. The user must enter the name of game to "Game Name" section.
Expected System Response	1. Federation with the entered game name is created in RTI side of game. 2. If the user does not enter a name and starts the game, player name is set to default as "Game (Number)".

3.2.4 Player Name Specification

Test Case ID	TC4
Test Case Name	Player Name Specification
Test Case Description	The user can set the player name, before the game started.
Test Case Objective	To check that player starts the game with the selected player game according to the user selections in pre-game menu.
Pre-conditions	The user must start the game.
Steps	1. The user must be in pre-game menu. 2. The user must select “New Game” or “Join Game”. 3. The user must enter the number of agents to “Agent Number” section.
Expected System Response	1. Federate with the entered player name is created in RTI side of game. 2. If the user does not enter a name and starts the game, player name is set to default as “Player (Number)”.

3.2.5 Button Configuration

Test Case ID	TC5
Test Case Name	Button Configuration
Test Case Description	The user can configure the buttons, before the game started.
Test Case Objective	To check that game starts with the selected button configuration according to the user selections in pre-game menu.
Pre-conditions	The user must be started the game.
Steps	1. The user must run the “The Eleron.exe” 2. The user must click the “Input“ tab of the opened configuration window. 3. The user must double click the button configuration which he/she wants to change. 4. The user must enter a key from keyboard to set button function.
Expected System Response	1. Game starts according to button configuration done by the user. 2. If the user does not configure the vuttons before game starts default button configurations are used. Which are: a. D/d – Right b. A/a – Left c. W/w – Up d. S/s – Down e. Left Mouse – Fire f. Mouse Scroll – Zoom In/Out g. Q/q – Rotate Left Around Forward Axis h. E/e – Rotate Right Around Forward Axis

3.2.6 Graphics Configuration

Test Case ID	TC6
Test Case Name	Graphics Configuration
Test Case Description	The user can configure the graphic settings, before the game started.
Test Case Objective	To check that game starts with the selected graphic configurations according to the user selections in pre-game menu.
Pre-conditions	The user must be started the game.
Steps	1. The user must run the “The Eleron.exe” 2. The user must click the “Graphics“ tab of the opened configuration window. 3. The user must set the screen resolution from the dropdown menu in “Screen Resolution” section. 4. The user must set the quality of the graphics from the dropdown menu in “Graphics Quality” section. 5. The user must check the “Windowed” check box from the “Graphics” tab.
Expected System Response	1. Game starts according to graphics configuration done by the user. 2. If user does not set the graphics settings before game started game starts with the default settings which is “640x480” resolution and “Good” quality and “Windowed options”

3.2.7 Graphics Configuration

Test Case ID	TC7
Test Case Name	Space Craft Selection
Test Case Description	The user can set the his/her own space craft model, before the game started.
Test Case Objective	To check that game starts with the selected space craft model according to the user selection in pre-game menu.
Pre-conditions	The user must be started the game.
Steps	1. The user must select “New Game” or “Join Game”. 2. The user must be in pre-game menu. 3. The user select the one of the available models from “Space Craft” section.
Expected System Response	1. Game starts with selected space craft model in pre-game settings. 2. If user does not select a space craft model it starts with the default space craft model which is “Sci-Fi Hammer Head Patrol Cruiser”.

3.2.8 Spacecraft Selection

Test Case ID	TC8
Test Case Name	Spacecraft Selection
Test Case Description	The users simply select one of the provided spaceship models.
Test Case Objective	To check that the users are controlling the spaceships that they selected in pre-game screen.
Pre-conditions	1. User is in create or join game menu.
Steps	1. Some spacecraft models are provided in boxes. 2. The user clicks one of these boxes to select one.
Expected System Response	1. Selected spaceship model is bound to the user specific game object. 2. If the user does not make any selection, the default (first one) spacecraft is used. 3. When the game starts, the user controls selected spaceship.

3.2.9 Game Log Content

Test Case ID	TC9
Test Case Name	Game log content
Test Case Description	During game flow, some specific events appear in game log screen.
Test Case Objective	To check whether the predefined events are announced in the log panel.
Pre-conditions	1. Game is started. 2. Log panel is not closed.
Steps	1. The users encounter with events: a. Hit the enemy, b. Destroy the enemy 2. Or the bots: a) Detect enemy ships around, b) Hit an enemy, c) Destroy the enemy
Expected System Response	1. An event specific message is appeared in the log panels of all players such that “Player X destroys player Y”, “Player X is damaged by Y” and etc.

3.2.10 AI Patrolling

Test Case ID	TC10
Test Case Name	AI Patrolling
Test Case Description	The ai-controlled units may patrol in a specific region to detect enemy ships or to defend that spot.
Test Case Objective	To check whether the bots act according to the given commands during patrol.
Pre-conditions	1. Game is started. 2. There are bots in the game.
Steps	1. Bot is given order to patrol around a region. 2. Bot is ordered to report an enemy when it encounters with or attack to the encountered enemy. 3. Bot starts patrolling.
Expected System Response	1. Bot flight around. 2. If bot is encountered with an enemy ship: a) If it is ordered to attack, it reports the coordinates and simply attacks. b) If it is only ordered to report, it reports the event and avoids fighting.

3.2.11 Federation Initialization

Test Case ID	TC11
Test Case Name	Federation Initialization
Test Case Description	Federation creation is the key role of the game creation and joins. In the first place, the federation corresponding to the game name is initialized.
Test Case Objective	To handle prospective errors, unexpected situations/inputs from the client side.
Pre-conditions	1. The user is in the create game screen.
Steps	1. The user sends game creation request with a specific game name by clicking the “Create Game” button.
Expected System Response	1. Portico side accepts the request and starts the runtime module to create federation. a) If there is an already existing federation with specified game, return the user an alert to choose different game. b) If there exists portico specific network errors, notify the user about errors. c) If the federation is successfully created, send the client side a confirmation. 2. In the client side, the screen is changed and the response message is appeared on the new screen.

3.2.12 Pause

Test Case ID	TC12
Test Case Name	Pause
Test Case Description	The user pauses game (simulation).
Test Case Objective	To check that game paused or not.
Pre-conditions	The game is already running.
Steps	1. The user presses “ESC” button of keyboard.
Expected System Response	1. The game should be paused. 2. Spacecraft should not be able to move. 3. AI users also should be paused. 4. Menu panel(Continue, Restart, Quit options) should be shown.

3.2.13 Mini Map

Test Case ID	TC13
Test Case Name	Mini map
Test Case Description	The system shall provide the user with the functionality that he/she can see current locations of all planes in game.
Test Case Objective	To check that the system provides current locations of all planes in game in a mini map.
Pre-conditions	The game is already running.
Steps	1. When the User starts game the in left corner of GUI map will be opened automatically.
Expected System Response	1. In this map, teammates, opponents and battles will be shown according to these specifications. 2. Team mates will be represented by green arrow. 3. Opponents will be represented by red cross. 4. When a battle begins exclamation point of battle location. 5. Skull for a destroyed plane coordinates.

3.2.14 Attack Enemy

Test Case ID	TC14
Test Case Name	Attack Enemy
Test Case Description	When the enemy is in attack range, the AI units shall fire to the enemy.
Test Case Objective	To check that the AI units fire to enemies which are in attack range.
Pre-conditions	1. The game is already running. 2. An enemy unit must be in attack range.
Steps	1. The user gets closer to the enemies with AI-agents.
Expected System Response	1. An AI-agent should attack to the enemy. 2. If the enemy gets out of range, AI-agents stop following and attacking.

3.2.15 Join to federation

Test Case ID	TC15
Test Case Name	Join to federation
Test Case Description	The new federate joins the federation.
Test Case Objective	To check that the new federate is able to join federation or not.
Pre-conditions	1. One of the user creates the game (Federation is created.) 2. Creator user's network information must be known. (Federation name and its network information (IP, port) must be known.) 3. The user who wants to join a game sees the menu panel (Create, Join menu).
Steps	1. The user pressed join button.
Expected System Response	1. Unity side sends information about that user to the RTI side (Portico). 2. In portico, federate join the federation. 3. When federate joins the federation, portico side sends this information to unity side. 4. If join operation is Successful: The user should join the game. Unsuccessful: error message should send the unity side, and error message is shown by the user in unity screen.

3.2.16 Control of Movements

Test Case ID	TC16
Test Case Name	Control of Movements
Test Case Description	The user can control the movements of the character from the keyboard.
Test Case Objective	To check that according to user inputs from the keyboard, controller movements are correct.
Pre-condition	The game has already started.
Steps	1. The user presses “left arrow” button on the keyboard. 2. The user presses “right arrow” button on the keyboard. 3. The user presses “up arrow” button on the keyboard. 4. The user presses “down arrow” button on the keyboard. 5. The user presses “space” button on the keyboard. 6. The user presses “q” button on the keyboard. 7. The user presses “e” button on the keyboard
Expected System Response	1. When user presses “left arrow” button, the spacecraft turns left. 2. When user presses “right arrow” button, the spacecraft turns right. 3. When user presses “up arrow” button, the spacecraft ascends. 4. When user presses “down arrow” button, the spacecraft descends. 5. When user presses “space” button, the spacecraft accelerates. 6. When user presses “q” button, the spacecraft rotates around its forward direction to the left. 7. When user presses “e” button, the spacecraft rotates around its forward direction to the right.

3.2.17 Fire

Test Case ID	TC17
Test Case Name	Fire
Test Case Description	The user can fire from the spacecraft.
Test Case Objective	To check that when user presses mouse left button, lasers are generated and can be seen on the screen.
Pre-condition	The game has already started.
Steps	1. The user presses mouse left button.
Expected System Response	1. On left and right sides of the aircraft, lasers are generated. 2. Lasers move forward in the forward direction of the spacecraft. 3. If lasers hit some object, they explodes. 4. If they do not hit any object, they are destroyed after 10 seconds.

3.2.18 Hit

Test Case ID	TC18
Test Case Name	Hit
Test Case Description	Objects get hit when interact with lasers.
Test Case Objective	To check when lasers hit some objects, that objects have little explosions.
Pre-conditions	Any user fires.
Steps	1. Lasers hit some objects.
Expected System Response	1. If the object lasers hit is not spacecraft, they have little explosions on their surfaces. 2. If the object lasers hit is spacecraft, spacecraft has explosions on its surface and health value of the spacecraft on the health bar is decreased. Also log information about which spacecraft is hit has shown on the log screen. 3. If health value of the spacecraft equals 0, the spacecraft is destroyed and dissappears. “spacecraft x is destroyed” log information shown on the log screen.

3.2.19 Collision

Test Case ID	TC19
Test Case Name	Collision
Test Case Description	The spacecrafts collides some objects.
Test Case Objective	To check whether collision detection works, when spaceraft collides some objects.
Pre-conditions	The user is playing game.
Steps	1. The user controls spacecraft 2. The spacecraft collides other spacecrafts , meteors or user bases.
Expected System Response	1. When spacecrafts collides any objects in the game, spacecraft has exploded and destroyed.

3.2.20 Zoom In/Out

Test Case ID	TC20
Test Case Name	Zoom In / Out
Test Case Description	The user can zooms to see the objects in the game closer or far away.
Test Case Objective	To check whether zoom operation works when user use zoom option.
Pre-conditions	The user is playing game.
Steps	1. The user rotates mouse wheel forward. 2. The user rotates mouse wheel backward.
Expected System Response	1. When user rotates mouse wheel forward, the objects can be seen in more details and closer. 2. When user rotates mouse wheel backward, the objects can be seen in less details and far away.

3.3 High-Order Testing

3.3.1 Stress Testing

Stress testing involves two main approaches:

3.3.1.1 Game engine test

There will be many game objects such as spaceships, team bases, meteors, planets, skybox and lasers etc. There will also be some special effects to provide virtual reality. Because there is a limit for the Unity3d game engine to render the scene, the amount of these objects and effects must be well defined. We plan to test the scene with 3 basic scenarios:

1. There will be 2 players and small number of meteors. There will not be any team bases, lasers and effects.
2. There will be 6 players and small number of meteors, team bases and planets in the environment. The players will fire each other ¼ of the game time. Also, there are some virtual effects for movement of the spacecrafts.
3. There will be 10 players and large number of meteors, team bases and planets. The players will fire each other ¾ of the game time. Therefore, there will be huge amount of fire lasers in the game screen. The collision detection script also runs and on a collision or laser hit there will be explosion effects.

The aim of the scenario one is to see how the game engine renders a basic scene and the quality of the game environment. In the second and third scenario we increase the load on the game engine.

The number of the players in the scenarios may change. However, the edge for the other objects will be determined from these scenarios for the final game.

3.3.1.2 Portico test

The number of the players is limited to 10 due to the network issues. Since we use Portico RTI software for communication between players and AI server, there will be some synchronization problems as the player number increases. Therefore, to test whether the synchronization is okay, we simply test with different number of players. As in the graphics test we will run the system with 2, 6 and 10 players with the same data is passing for each case.

3.3.2 Performance Testing

We aim to reach at least 24 fps. However, there are some constraints before this goal. In each frame, in the unity game engine, the data of the other players is retrieved and set to the corresponding player object.

We are planning to observe performance of the game engine via Lumos[1] software. Lumos software is a tool that runs within Unity3d to analyze resource use, load and etc. It also provides statistics and error-log files. The results of the scenarios mentioned in the stress testing will be analyzed with Lumos tool. However, since we use external Portico software, there is a possibility that Lumos tool will not works. In that case, we will have to observe the results by manually. We will try to check and match incoming & going data, and with bare eye we will try to catch whether there are any delay or lag because delays and lag are the main problems in a multiplayer game.

[1] http://www.uselumos.com/

3.3.3 Alpha & Beta Testing

For alpha testing we will pick ten people with various levels of experience with computer applications. In order to get clear feedback, a face-to-face satisfaction questionnaire that is prepared for the testers will be used. We will ask them to try The Eleron and to give us feedback about mostly GUIs.

Beta testing will be entirely public. We will publish our product at http://code.google.com/p/randomsoft along with installation instructions. Users will use the issue tracking system provided by Google to submit defects and their opinions. We are expecting to get feedback not only on GUIs, but also on easiness of installation and usage of functionalities.