Testing Documentation

Stargate: Galaxy

Peter Mitchell
Andrew Krix
Karlos White
Phil Lavender
Kane Stone

10/28/2011

Table of Contents

1 Tables and Figures

No Tables of Figures

2 Revision History

Name	Date	Reason For Changes	Version

3 Introduction

3.1 Documentation Purpose

The purpose of this document is to define test cases to evaluate a game called Stargate: Galaxy. This document also provides the results of the execution of these test cases and provides suggestions about issues that have been discovered during testing.

3.2 Testing Strategy

The primary source for the creation of the test cases was the System Requirements Specification created for Stargate: Galaxy. The requirements also helped to define a base for other tests completed as part of the testing process. Other sources for tests were the source code of the game and the System Design Document. The tests were then split into the following categories

Test Category	**Source Document**
System Tests	SRS
Requirements Tests	SRS
Unit Tests	Source Code
Integration Tests	Source Code
Other White Box Tests	Source Code

Tests were completed by individuals and then reviewed by other members of the group; this was done in order to reduce the potential for error during the test execution process. Where possible the tests have included references in order to find the location of related pieces of documentation.

3.3 Limitations of Testing Strategy

During the testing process the test cases were split into three separate categories: critical, core and additional. Test cases in the critical category were cases that must pass in order to proceed with testing, Core tests are tests that have been carried out on important parts of the functionality; additional test cases included minor functionality which was considered important enough to be tested.

It was not the case with the testing of done during the development of this document, but if it had been found that a serious defect existed then code would have been submitted back to the developers for correction.

4 Test-case Specification

4.1 System Tests

Summary		
Critical	1.1	Compile source code
	1.2	Runs on a Windows Environment

ID: 1.1	**Test Name: Compile source code**	
References:	Requirements Documentation Code Other Tests	
Purpose:	To Test that the program compiles correctly within the Unity environment on the windows platform using the default configuration.	
Type:	System	Nature of Test: Black Box
Component to be tested:	Project files	
Pre-conditions:		
Input #	**Input:**	**Expected Output:**
1	Open the project in Unity and select the Build and Run option	The source code is successfully compiled with no errors occurring.
Post-conditions:	The application is compiled	
Environmental needs:	Windows platform Unity Running	
Stubs and Drivers:		
Other Information:		

ID: 1.2	**Test Name: Runs on a windows environment**	
References:	Requirements Documentation Code Other Tests	
Purpose:	To test that the application successfully runs.	
Type:	System	Nature of Test: Black Box
Component to be tested:	Project files	
Pre-conditions:	The application is compiled	
Input #	**Input:**	**Expected Output:**
1	Run the compiled application within the Unity Environment	The application successfully runs with no errors occurring while opening.
2	Run the compiled application from the exe file that is generated by the project	The file executes successfully with no errors occurring while opening.
Post-conditions:	The application window is open	
Environmental needs:	Windows platform Unity Running	
Stubs and Drivers:		
Other Information:		

4.2 Requirements Tests

Summary		
Core	2.1	Ships exist within the game world
	2.2	Planets and other objects (supergate) exist within the game world
	2.3	GUI exists
	2.4	Move to a specific point in space
	2.5	Move to an object within the game world (ship, planet etc)
	2.6	Move to object and orbit
	2.7	Target Object (ship, planet, etc)
	2.8	Move to Target
	2.9	AI ship movement
	2.10	Fire on Target
	2.11	AI ship targeting
	2.12	Select Weapon
	2.13	Engage in combat
	2.14	Escape from combat
	2.15	Main Menu
	2.16	Ingame Menu
	2.17	Save
	2.18	Load
	2.19	Enter Hyperspace
	2.20	Exit Hyperspace
	2.21	Travel through Supergate
	2.22	Ship Destroyed
	2.23	Gain Experience
	2.24	Collect loot
	2.25	Trade Resources
	2.26	Call allies
	2.27	Capture Planet
	2.28	Change Planet Status
	2.29	Track Current Power levels
	2.30	Move Power between systems
Additional	2.31	Ships have different statistics and models depending upon race
	2.32	Weapon deals different damage based on weapon type

ID: 2.1	**Test Name:**	**Ship exists within game world**
References:	Requirements Documentation Code Other Tests	
Purpose:	To test that the ships exist within the game world	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	The application creating ships within the game world	
Pre-conditions:	Application needs to be compiled and running	

Input #	**Input:**	**Expected Output:**
1	Player's ship exists within the game world	Ship exists
2	AI ships exist within the game world	Ships exist

Post-conditions:	Ships exist within the game
Environmental needs:	Instance of the application running. A user
Stubs and Drivers:	
Other Information:	

ID: 2.2	**Test Name:**	**Planets and other objects (supergate) exist within the game world**
References:	Requirements Documentation Code Other Tests	
Purpose:	To test that the planets and other objects exist within the game world	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	The application creating planets and other objects within the game world	
Pre-conditions:	Application needs to be compiled and running	

Input #	**Input:**	**Expected Output:**
1	Planets exist within the game world	Planets exist
2	Supergate exists within the game world	Supergate exists

Post-conditions:	Planets and other game objects exist within the game world
Environmental needs:	Instance of the application running. A user
Stubs and Drivers:	
Other Information:	

ID: 2.3	**Test Name: GUI Exists**	
References:	Requirements Documentation Code Other Tests	
Purpose:	To test that the GUI exists while the game is running	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	The application creating ships within the game world	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	GUI exists within the game world	GUI exists
2	Map pane exists within the GUI	Map pane exists
3	Power management pane exists within the GUI	Power management pane exists
4	Weapons pane exists within the GUI	Weapons pane exists
5	Health pane exists within the GUI	Health pane exists
6	Targeting information pane exists within the GUI	Targeting information pane exists
Post-conditions:	GUI and elements of the GUI exist within the game	
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.4	**Test Name: Move to a specific point in space**	
References:	Requirements Documentation Code Other Tests	
Purpose:	To test that the players ship can move to a specific point in space	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Ship Movement	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player wishes to move to a point in between two planets where there are no enemies	The players ship moves to the specified point
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.5	**Test Name: Move to an object within the game world (ship, planet)**	
References:	Requirements Documentation Code Other Tests	
Purpose:	To test that the players ship can move to a specific object	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Ship Movement	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player wishes to move to an enemy ship	The players ship moves to the enemy ship
2	Scenario: The player wishes to move to a planet	The players ship moves to the specified planet
3	Scenario: The player wishes to move to an allied ship	The players ship moves to the allied ship
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.6	**Test Name: Move to an object and orbit**	
References:	Requirements Documentation Code Other Tests 2.5	
Purpose:	Ship moves to an object and orbits it	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Ship Movement	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player wishes to move to an enemy ship and orbit it	The players ship moves to the specified ship and orbits it
2	Scenario: The player wishes to move to an allied ship and orbit	The players ship moves to the specified ship and orbits it
3	Scenario: The player wishes to orbit a planet	The players ship moves to the specified planet and orbits it
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.7	**Test Name: Target Object**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Check that the player can target objects	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Target Object	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player targets an enemy ship	The enemy ship is targeted and the information is shown on the display
2	Scenario: The player targets an allied ship	The allied ship is targeted and the information is shown on the display
3	Scenario: The player targets a planet	The planet is targeted and the information is shown on the display
4	Scenario: The player targets the supergate	The supergate is targeted and the information is shown on the display
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.8	**Test Name: Move to Target**	
References:	Requirements Documentation Code Other Tests 2.7	
Purpose:	The player can move to the target	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Ship Movement	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player wishes to move to a targeted enemy ship	The players ship moves to the targeted ship
2	Scenario: The player wishes to move to a targeted allied ship	The players ship moves to the targeted ship
3	Scenario: The player wishes to orbit a targeted planet	The players ship moves to the targeted planet
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.9	**Test Name: AI Ship Movement**	
References:	Requirements Documentation Code Other Tests 2.5, 2.6, 2.8	
Purpose:	The AI ships can move to a number of targets	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Ship Movement	
Pre-conditions:	Application needs to be compiled and running	

Input #	**Input:**	**Expected Output:**
1	Scenario: The AI ship wishes to move towards the players ship	The ship moves towards the players ship
2	Scenario: The AI ship wishes to move towards an allied AI ship	The ship moves towards the allied ship
3	Scenario: The AI ship wishes to move towards an enemy AI ship	The ship moves towards the enemy ship
4	Scenario: The AI ship wishes to move towards a planet	The ship moves towards a planet
5	Scenario: The AI ship wishes to move towards and orbit the players ship	The ship moves towards and orbits the players ship
6	Scenario: The AI ship wishes to move towards and orbit an allied AI ship	The ship moves towards and orbits the allied ship
7	Scenario: The AI ship wishes to move towards and orbit an enemy AI ship	The ship moves towards and orbits the enemy ship
8	Scenario: The AI ship wishes to move towards and orbit an AI ship	The ship moves towards and orbits the planet

Post-conditions:	
Environmental needs:	Instance of the application running. A user
Stubs and Drivers:	
Other Information:	

ID: 2.10	**Test Name: Fire on Target**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Test that ships can fire upon one another	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Combat	
Pre-conditions:	Application needs to be compiled and running	

Input #	**Input:**	**Expected Output:**
1	Scenario: The player wishes to fire upon a targeted ship using the energy weapon	The players ship fires the energy weapon at the targeted ship
2	Scenario: The player wishes to fire upon a targeted ship using the railgun	The players ship fires the railgun at the targeted ship
3	Scenario: The player wishes to fire upon a targeted ship using the rockets	The players ship fires rockets at the targeted ship
4	Scenario: An AI ship wishes to fire upon the players ship using an energy weapon	The AI ship fires its energy weapon at the players ship
5	Scenario: An AI ship wishes to fire upon the players ship using a railgun	The AI ship fires its railguns at the players ship
6	Scenario: An AI ship wishes to fire upon the players ship using a rocket	The AI ship fires its rockets at the players ship
7	Scenario: An AI ship wishes to fire upon another AI ship using an Energy Weapon	The AI ship fires its energy weapons at the other AI ship
8	Scenario: An AI ship wishes to fire upon another AI ship using a railgun	The AI ship fires its railgun at the other AI ship
9	Scenario: An AI ship wishes to fire upon another AI ship using rockets	The AI ship fires upon the other AI ship using rockets
10	Scenario: A Replicator Ship wishes to fire upon another ship using its special attack	The replicator ship fires upon the targeted ship which then has a chance of being infected

Post-conditions:	
Environmental needs:	Instance of the application running. A user
Stubs and Drivers:	
Other Information:	

ID: 2.11	**Test Name: AI ship targeting**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Ensure that the AI can target ship	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Combat	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The AI wishes to target the players ship	The AI ship targets the players ship
2	Scenario: The AI ship wishes to target another AI ship	The AI ship targets the other AI ship
3	Scenario: The AI ship wishes to target a planet	The AI ship targets the planet
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.12	**Test Name: Select Weapon**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Ensure that the player can select between the weapons that they have available	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Combat	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player wishes to use their railgun weapon	The weapon readout reflects the change in weaponry
2	Scenario: The player wishes to use their energy weapon	The weapon readout reflects the change in weaponry
3	Scenario: The player wishes to use their rockets	The weapon readout reflects the change in weaponry
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.13	Test Name: Engage in Combat	
References:	Requirements Documentation Code Other Tests 2.7, 2.8, 2.9, 2.10, 2.11	
Purpose:	Ship moves to target and fires upon it	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Ship Movement and combat	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player wishes to engage an enemy ship in combat	The players ship moves towards the enemy ship and opens fire with the currently selected weapon
2	Scenario: An AI ship wishes to engage the players ship	The AI ship moves to the players ship and then fires upon it
3	Scenario: An AI ship wishes to engage another AI ship	The AI ship moves to the targeted ship and opens fire
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.14	Test Name: Escape from combat	
References:	Requirements Documentation Code Other Tests	
Purpose:	A ship needs to flee from combat	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Ship Movement	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player wishes to escape from combat with an AI ship	The player can move out of combat with an AI ship
2	Scenario: The AI wishes to escape from combat with the players ship	The AI ship attempts to move out of combat with the players ship
3	Scenario: The AI wishes to escape from combat with another AI ship	The AI ship attempts to move out of combat with another AI ship
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.15	**Test Name: Main Menu**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Test that the main menu exists and has the correct functionality	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	GUI	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	The main menu Exists	The menu exists
2	The main menu loads upon game start	The menu loads when the game begins
3	Scenario: The user presses the New Game button and there is no current save	A new game is created
4	Scenario: The user presses the New Game button and there is a current save	The user receives a warning that they may overwrite current save data
5	Scenario: The user presses the Continue Game button and there is no current save	The user receives a warning that there is no current save game
6	Scenario: The user presses the Continue Game button and there is a current save	The user loads the game currently saved
7	Scenario: The user presses the Credits button	The user is shown to the credits screen
8	Scenario: The user presses the Quit Game button	The game closes
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.16	**Test Name: In game menu**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Test that the in game menu exists and has the correct functionality	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	GUI	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	The in game menu exists	The menu exists
2	When the player presses the ESC button the in game menu appears	The in game menu appears correctly
3	Scenario: The user is at Earth, opens the menu and presses the save game button	The user receives a warning that the game will overwrite any current save data
4	Scenario: The user is at Earth, opens the menu and presses the load game button	The user receives a warning that any unsaved data will be lost
5	Scenario: The user is at Earth, opens the menu and wishes to change to a different ship	The users current ship is changed
6	Scenario: The user is at the Asgard Trade Planet and opens the menu	The Trade Menu is visible in addition to the normal menu
7	Scenario: The user opens the menu and selects the resume game button	The user returns to the game
8	Scenario: The user opens the menu and selects the Quit Game button	The user returns to the Main Menu
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.17	**Test Name: Save**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Save the player information so that the game can be loaded in the future	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:		
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player attempts to save the game while at Earth	The game saves correctly
2	Scenario: The player attempts to save the game while not at Earth	The player receives a message telling them that they must be at Earth to save
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.18	**Test Name: Load**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Load the player information previously saved	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:		
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player wishes to load the game currently saved from within the main menu	The game loads the currently saved game
2	Scenario: The player wishes to load the game currently saved from within the game	The player receives a message asking them if they are sure they wish to load a game as any unsaved data may be lost, if the player accepts then the game loads
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.19	**Test Name: Enter Hyperspace**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Ship enters hyperspace	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Ship Movement	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player elects to enter hyperspace and selects a destination	The players ship enters hyperspace
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.20	**Test Name: Exit Hyperspace**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Ship exits hyperspace	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Ship Movement	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player leaves hyperspace at the location they were heading	The players ship exits hyperspace at the specified location
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.21	**Test Name: Travel through Supergate**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Ship travels through the Supergate	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Ship Movement	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player wishes to move through the supergate from the Milky Way to the Ori galaxy	The players ship moves through the supergate and exits in the Ori Galaxy
2	Scenario: The player wishes to move through the supergate from the Ori galaxy to the Milky Way	The players ship moves through the supergate and exits in the Milky Way
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.22	**Test Name: Ship Destroyed**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Test that when a ship reaches 0 hull points it is destroyed	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Combat	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The players ship reaches 0 hull points	The player receives a game over message
2	Scenario: An enemy AI ship reaches 0 hull points	The enemy ship is removed and replaced by wreckage
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.23	**Test Name: Gain Experience**	
References:	Requirements Documentation Code Other Tests	
Purpose:	When the player destroys an enemy ship they gain experience	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:		
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player destroys an enemy ship	The players receives experience based upon the ship destroyed
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.24	**Test Name: Collect Loot**	
References:	Requirements Documentation Code Other Tests	
Purpose:	The player wishes to collect items off of a destroyed ship	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Trade	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player wishes to collect items off of some wreckage	The player receives some scrap material and also has a chance to receive other items based upon the ship that was destroyed
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.25	**Test Name: Trade Resources**	
References:	Requirements Documentation Code Other Tests 2.24	
Purpose:	The player wishes to trade collected resources with the Asgard	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Trade	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player wishes to buy an item off of the Asgard and had the required resources	The player receives the item they purchased
2	Scenario: The player wishes to buy an item off of the Asgard but does not have the required resources	The players receives a message saying that they do not have the required resources
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.26	**Test Name: Call Allies**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Test that the player can call allies when needed	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Combat	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player wishes to call upon allied assistance and has not called upon assistance recently	A number of AI ships appear from hyperspace to assist the player
2	Scenario: The player wishes to call upon allied assistance and has recently received assistance	The player receives a message saying that allied forces are unavailable at this time
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.27	**Test Name: Capture Planet**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Test that a planet can be captured	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Galaxy Control	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player wishes to capture a planet and there are no enemy ships in the vicinity	The players ship captures the planet
2	Scenario: The player wishes to capture a planet while enemy forces are still nearby	The player receives a message that they need to clear the area before they can capture the planet
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.28	Test Name: Change Planet Status	
References:	Requirements Documentation Code Other Tests	
Purpose:	Change Information about the status of a planet	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Galaxy Control	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player captures a planet which was not controlled by hostile forces	The planets status changes from neutral to player controlled
2	Scenario: The player captures a planet which was controlled by hostile forces	The planets status changes from enemy controlled to player controlled
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.29	Test Name: Track Current Power Levels	
References:	Requirements Documentation Code Other Tests 2.3	
Purpose:	Show the current distribution of power	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Ship Movement	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	The power management panel shows the current levels of power	The power management panel shows the current distribution of power
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.30	**Test Name: Move Power between systems**	
References:	Requirements Documentation Code Other Tests	
Purpose:	To test that the player can move power between the different systems	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Power Systems	
Pre-conditions:	Application needs to be compiled and running	

Input #	**Input:**	**Expected Output:**
1	Scenario: The player wishes to move power into the ships weapons and there is power left in the power reserve.	The power readout reflects the movement of power and the strength of the ships weapons increases
2	Scenario: The player wishes to move to power into the ships weapons and there is no power left in the power reserves	The player receives a message saying that no more power is available
3	Scenario: The player wishes to move power into the ships engines and there is power left in the power reserve	The power readout reflects the movement of power and the speed of the ship decreases
4	Scenario: The player wishes to move power into the ships engines and there is no power left in the power reserves	The player receives a message saying that no more power is available
5	Scenario: The player wishes to move power into the ships shields and there is power left in the power reserve	The power readout reflects the movement of power and the strength of the ships shields increases
6	Scenario: The player wishes to move power into the ships shields and there is no power left in the power reserves	The player receives a message saying that no more power is available
7	Scenario: The player wishes to move power from weapons into the power reserves	The power readout reflects the movement of power into the power reserves
8	Scenario: The player wishes to move power from shields into the power reserves	The power readout reflects the movement of power into the power reserves
9	Scenario: The player wishes to move power from the engines into the power reserves	The power readout reflects the movement of power into the power reserves

Post-conditions:	
Environmental needs:	Instance of the application running. A user
Stubs and Drivers:	
Other Information:	

ID: 2.31	Test Name: Ships have different statistics and models based upon race	
References:	Requirements Documentation Code Other Tests	
Purpose:	Test that the ships are different based upon the race	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:		
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	The player ship has a unique model	No models the same as the player model found
2	The wraith ships use two models, the wraith hive ship and the wraith cruiser	Wraith ships are recognisable without having to target them
3	The goa'uld have one ship model for their main fleet. Anubis has a unique ship model	Goa'uld ships are recognisable without having to target them.
4	The asgard have one ship model for their fleet	Asgard ships are recognisable without having to target them
5	The replicators have one ship model	Replicator ships are recognisable without having to target them
6	The Ori have one ship model for their entire fleet	Ori ships are recognisable without having to target them
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

ID: 2.32	Test Name: Weapon deals different damage based on weapon type	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the weapons have different effects on enemy ships based upon type	
Type:	Requirements	Nature of Test: Black Box
Component to be tested:	Combat	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Scenario: The player fires the railgun at an enemy ship	The railgun deals equal damage to both the hull and shields of a ship
2	Scenario: The player fires an energy weapon at an enemy ship	The energy weapons deals more damage to shields and less damage to the hull of a ship
3	Scenario: The player fires rockets at an enemy ship	The rockets deal less damage to the shields and more damage to the hulls of a ship
Post-conditions:		
Environmental needs:	Instance of the application running. A user	
Stubs and Drivers:		
Other Information:		

4.3 Unit Tests

Summary		
Core	2.1	Game Controllers / AI Controller
	2.2	Game Controllers / Data Loader
	2.3	Game Controllers / Data Manager
	2.4	Game Controllers / Game Controller
	2.5	Game Controllers / New Game
	2.6	Ship AI / Lootable Ship
	2.7	Ship AI / Projectile Data
	2.8	Ship AI / Ship AI
	2.9	Other / Hyperspace Window
	2.10	HUD / Ingame Menu
	2.11	HUD / Loading Screen
	2.12	HUD / Main Menu
	2.13	HUD / Object Panel
	2.14	HUD / Player Info
	2.15	HUD / Rep Ali Panel
	2.16	HUD / Screen Messages
	2.17	HUD / Ship Control Panel
	2.18	HUD / Target Select
	2.19	HUD / Top Overlay Script
	2.20	HUD / Weapons Panel
	2.21	HUD / Yes No Window

ID: 3.1	**Test Name: Game Controllers / AI Controller**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the AI Controller script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	AI Controller	
Pre-conditions:	Application needs to be compiled and running	

Input #	**Input:**	**Expected Output:**
1	configureController()	Controllers variables are correctly populated.
2	Update()	Planet capture status is updated and wrecked ships cleared.
3	updateCaptureStatus()	Capturing begun when valid and completion of capturing when correct.
4	selectRandomTarget()	Each variation correctly returns random targets.
5	getEnemyCountofRace()	Number of enemies correct.
6	getEnemyTargetofFriends()	A collection of the enemies that are being targeted by friends.
7	`getAllTargettingShip()`	All ships targeting the specified ship.
8	getEnemyTargetingFriends()	All ships that are targeting friendly ships.
9	selectObjectFromList()	A random object is selected from the list each time.
10	`getAllies()`	All allies of each specified race.
11	getEnemies()	All enemies of each specified race.
12	`getNewTarget()`	A new target.
13	`spawnShip()`	Each variation correctly spawns a ship with or without the hyperspace window and correct stats as required.
14	`addShip()`	Number of enemies and allies increases correctly.
15	`getVector3InRangeOfPoint()`	A random position within the correct ranges.
16	`removePlayerShip()`	Player ship is not found by isPlayerShipHere() anymore after this call.
17	`removeShip()`	Ship's race ship data has been correctly removed.
18	`isPlayerShipHere()`	Correctly returns when the player's ship is actually there.
19	`getRaceForceSizes()`	Correct translated equivalents for each of the races.
20	`setupShipCollection()`	Correctly placed the right numbers of ships.
21	`getRandomShipForRace()`	Correctly gets a ship ID for that type of race.
22	`getBossShipForRace()`	Correctly gets a ship ID for that type of race.
23	`findShipsWithInRangeOf()`	Finds all ships in range of point.
24	`getTargetableObjects()`	Gets all objects that can be targeted in this area.
25	`orderedInsertbyLocation()`	Inserts correctly in order of distance from point.
26	`updateVisionStatus()`	Player's vision status correctly updated.

Post-conditions:
Environmental needs:
Stubs and Drivers:
Other Information:

ID: 3.2	**Test Name: Game Controllers / Data Loader**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Data Loader script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Data Loader	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Start()	Variables all have their default states and new data is created on object creation correctly
2	loadPlayerData()	Successfully loads all the data that would be expected.
3	loadShipData()	Successfully loads all the ship database information.
4	saveData()	Correctly saves data to file for each of the two variations.
5	generateBasicData()	Data is generated and correctly saved.
6	generateShipDatabase()	Data is generated and correctly saved.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.3	**Test Name: Game Controllers / Data Manager**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Data Manager script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Data Manager	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	isAlly()	Correctly identifies the difference between allies and enemies.
2	getForceSizeFromCount()	Correctly returns a force size based on the count.
3	getShipCountFromForceSize()	Correctly returns a number in a valid range based on the force size.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.4	**Test Name: Game Controllers / Game Controller**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Game Controller script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Game Controller	
Pre-conditions:	Application needs to be compiled and running	

Input #	**Input:**	**Expected Output:**
1	Start()	Variables all have their default states and new data is created on object creation correctly
2	Awake()	Object correctly stays between scenes.
3	setHyperspaceTarget()	Player begins moving toward a hyperspace window and entering hyperspace occurs.
4	hyperspaceJumpTo()	Current system ID numbers change and switching to the loading screen occurs correctly.
5	completeHyperspaceJump()	Skybox is correctly placed, bosses, planets, and ships are all created.
6	performHyperspaceJump()	When player a loading screen appears, and when AI, the AI should appear at other planet.
7	setAIHyperspaceTarget()	Target is set for the AI and hyperspace entry occurs.
8	`spawnHyperspaceWindow()`	A hyperspace window appears.
9	getVector3InRangeofPoint()	Correctly generates a random position within the specified range.
10	OnLevelWasLoaded()	Correctly switch back to main game scene when loading screen has completed.
11	exitToMainMenu()	Main menu opens.
12	Update()	Changes out of loading screen correctly. Battle simulations update correctly, player is correctly updated for which AIController they should be governed by.
13	updateSimulateBattles()	Battles are simulated and system states change.
14	getPossibleBattles()	Correctly finds possible battles to choose from.
15	storePlayerShipState()	Player ship state variables are all correctly saved.
16	increasePlayerExp()	Ship experience increases and level up occurs when it should.
17	purchaseWeapon()	Weapon is correctly purchased and equipped.
18	buyShip()	Ship is correctly purchased.
19	canBuyShip()	Returns status correctly based on whether the player can currently buy that ship.
20	canBuyWeapon()	Returns status correctly based on whether the player can currently buy that weapon.
21	getPurchasableWeapons()	Shows the correct number of weapons dependent on the ship and level.
22	haveResourcesFor()	Correctly confirms that there are enough resources for a supplied quantity.
23	payResources()	Amount is correctly subtracted.
24	addResources()	Amount is correctly added.
25	setCurrentShip()	Ship changes to different ship.

26	attemptCallAllies()	Allies sometimes come and sometimes don't.
27	claimPlanet()	Planet controller changes.
28	updateStatusPlayerVision()	Update of last seen controller status changes.
29	getTargetableObjects()	All targetable objects including planets are correctly returned.
30	getPlanetList()	The planets in the current system are all returned.
31	getPrefabShip()	Correctly gets the prefabs for each id.
32	getPrefabPlanet()	Correctly gets the prefabs for each id.
33	getPrefabWeapon()	Correctly gets the prefabs for each id.
34	getShipData()	Correctly gets the ship data for each id.
35	getWeaponData()	Correctly gets the weapon data for each id.
36	getPrefabWreck()	Correctly gets the wreck prefab.
37	setSkyboxMat()	Correctly changes the skybox.
38	saveGame()	Game is saved.
39	loadGame()	Current state is lost and previous saved data is loaded.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.5	**Test Name: Game Controllers / NewGame**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the New Game script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	New Game	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Awake()	Object not destroyed on load.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.6	**Test Name: Ship AI / Lootable Ship**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Lootable Ship script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Lootable Ship	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	setResources()	Loot variable changed.
2	getResources()	Equal to content of loot variable.
3	isLooted()	Has already been looted.
4	claimLoot()	Player's resources updated
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.7	**Test Name: Ship AI / Projectile Data**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Projectile Data script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Projectile Data	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Start()	Projectile ignores all other projectiles.
2	Update()	Projectile disappears after period of time.
3	OnTriggerStay()	Collided object takes damage if required and otherwise is destroyed.
4	configureProjectile()	Projectile has the correct weapon data and moves toward the correct target.
5	setupVelocity()	Movement toward the correct target.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.8	**Test Name: Ship AI / Ship AI**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Ship AI script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Ship AI	
Pre-conditions:	Application needs to be compiled and running	

Input #	**Input:**	**Expected Output:**
1	Start()	Variables all have their default states
2	Update()	Ships status appears to change between updates
3	healShip()	Ship heals hull and then shield after damage
4	updateAttackTarget()	inCombat should change and auto target engaging should be configured on timed events. Also random targeting when infected.
5	aiTargetUpdate()	AI target changes.
6	updateMovement()	Ship object moves using correct movement toward points stopping and responding correctly.
7	`updateAIMovement()`	AI chooses valid combinations of movements that make them move around.
8	initiateMove()	Both variations correctly begin movement when triggered
9	enterSupergate()	Ship travels toward supergate.
10	`calculateSpeedMultiplier()`	Speed changes based on the speedpower and level.
11	updateWeapons()	Ship begins firing and continues firing when within 250 unit range.
12	`fireWeapon()`	Different weapon projectiles appear and head toward the correct target.
13	`damageShip()`	Ship takes damage based on the weapon type and shield configuration.
14	`infectShip()`	Ship becomes infected visible by the infected variable.
15	`beginPlayerCancelInfection()`	Cooldown begins on infection cancel cooldown.
16	`damageShield()`	Shield takes damage. Message displayed when shield fails.
17	`damageHull()`	Hull takes damage. Message displayed when hull fails and return to earth initiated.
18	`setMoveMode()`	Mode changes.
19	`getShieldPercent()`	Shield as a percent is returned.
20	`getHullPercent()`	Hull as a percent returned.
21	`getShieldPowerPercent()`	Shield power as a percent of the total returned.
22	`getWeaponPowerPercent()`	Weapon power as a percent of the total is returned.
23	`getEnginePowerPercent()`	Engine power as a percent of the total is returned.
24	`getPowerUsePercent()`	Total power used as a percentage is returned.
25	`powerIncreaseAllowed()`	Correctly returns false when requested power allocation is too large.
26	`configureShip()`	Ship has properties assigned as expected.

27	`equipWeapon()`	Weapons are available on the ship.
28	`enableWeapon()`	One more weapon is enabled.
29	`disableWeapon()`	One weapon is disabled.
30	`canRetreat()`	Returns true when conditions should exist for retreat.
31	`getCountsofWeaponEnabled()`	Correct number of weapons returned based on level of ship.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.9	**Test Name: Other / Hyperspace Window**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Hyperspace Window script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Hyperspace Window	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Start()	ProcessDestroy correctly called.
2	ProcessDestroy	Destroyed after period of tim.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.10	**Test Name: HUD / Ingame Menu**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Ingame Menu script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Ingame Menu	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Start()	Variables correctly initialised.
2	Update()	Shows menu after pressing escape.
3	OnGUI()	Window visible
4	Menu_window()	Button actions work correctly.
5	Toggle()	Disappears and reappears when required.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.11	**Test Name: HUD / Loading Screen**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Loading Screen script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Loading Screen	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Start()	Variables correctly initialised.
2	Update()	Image changes.
3	OnGUI()	Image displayed.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.12	**Test Name: HUD / Main Menu**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Main Menu script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Main Menu	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Start()	Variables correctly initialised.
2	Update()	Does nothing.
3	OnGUI()	Window visible
4	Menu_window()	Button actions work correctly.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.13	**Test Name: HUD / Object Panel**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Object Panel script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Object Panel	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Start()	Variables correctly initialised.
2	Update()	S toggles visibility and the objects are updated.
3	OnGUI()	Window visible
4	objToString()	Correctly gives the string that should be displayed.
5	slideToggle()	Correctly toggles the sliding.
6	mapAction()	Correctly starts the sliding
7	Slide()	Moves the panel.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.14	**Test Name: HUD / Player Info**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Player Info script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Player Info	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Start()	Variables correctly initialised.
2	Update()	Shows/Hides after pressing I.
3	OnGUI()	Window visible
4	objToString()	Correctly gives the string that should be displayed.
5	slideToggle()	Correctly toggles the sliding.
6	mapAction()	Correctly starts the sliding
7	Slide()	Moves the panel.
8	shipDetails()	Ship details visible.
9	Xp_bar()	XP bar visible.
10	gameData()	Updates the current status of information.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.15	**Test Name: HUD / Rep Ali Panel**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Rep Ali Panel script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Rep Ali Panel	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Start()	Variables correctly initialised.
2	Update()	Shows and hides using H
3	OnGUI()	Window visible
4	slideToggle()	Correctly toggles the sliding.
5	mapAction()	Correctly starts the sliding
6	Slide()	Moves the panel.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.16	**Test Name: HUD / Screen Messages**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Screen Messages script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Screen Messages	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Start()	Variables correctly initialised.
2	Update()	Updates correctly.
3	OnGUI()	Window visible
4	Texter()	Added text with colour to array.
5	arrayMove()	Updated the array.
6	setColor()	Set the colour.
7	printText()	Text appears.
8	textScroll()	Text moves.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.17	Test Name: HUD / Ship Control Panel	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Ship Control Panel script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Ship Control Panel	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Start()	Variables correctly initialised.
2	Update()	Values correctly updated.
3	OnGUI()	Window visible
4	sliderCheck()	Sliders updated not beyond maximums.
5	labelUpdate()	Updated labels.
6	shieldStatus()	Visible shield status.
7	Hull()	Visible hull status.
8	Target()	Visible target information.
9	Resource()	Visible resource information.
10	gameData()	Updated current status information.
11	objPosAssign()	Aligned values all correct.
12	getShields()	Gets the shield value.
13	getWeapons()	Gets the weapon value.
14	getEngines()	Gets the engine value.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.18	Test Name: HUD / Target Select	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Target Select script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Target Select	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Start()	Variables correctly initialised.
2	Update()	Values correctly updated.
3	OnGUI()	Window visible
4	Buttons()	Buttons all correctly handled.
5	getData()	Required information all retrieved.
6	callGUI()	Placed the GUI at location.
7	Toggle()	Visibility changed.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.19	**Test Name: HUD / Top Overlay Script**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Top Overlay Script script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Top Overlay Script	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Start()	Variables correctly initialised.
2	Update()	Values correctly updated.
3	OnGUI()	Window visible
4	mapMain()	Map displayed.
5	mapMenu()	Map information shown.
6	tradeMain()	Information displayed.
7	tradeMenu()	Interaction possible.
8	earthMain()	Information displayed.
9	earthMenu()	Interaction possible.
10	Slide()	Sliding handled.
11	Slideisslide()	Begun sliding.
12	weaponList()	Weapon list displayed.
13	planetMapLoc()	Points calculated.
14	objPosAssign()	Objects aligned.
15	Trade()	Trade opened.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.20	**Test Name: HUD / Weapons Panel**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Weapons Panel script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Weapons Panel	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Start()	Variables correctly initialised.
2	Update()	Values correctly updated.
3	OnGUI()	Window visible
4	weaponsMAIN()	All elements displayed.
5	weaponsSUB()	An individual element displayed.
6	getData()	Information correctly updated.
7	buttonSet()	Button skins all matched correctly.
8	mapAction()	Displayed GUI
9	slideToggle()	Hide/Show GUI
10	Slide()	GUI moves
11	objPosAssign()	Object positions aligned correctly.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

ID: 3.21	**Test Name: HUD / Yes No Window**	
References:	Requirements Documentation Code Other Tests	
Purpose:	Tests that the Yes No Window script does what it should	
Type:	Unit Test	Nature of Test: White Box
Component to be tested:	Yes No Window	
Pre-conditions:	Application needs to be compiled and running	
Input #	**Input:**	**Expected Output:**
1	Start()	Variables correctly initialised.
2	Update()	Values correctly updated.
3	OnGUI()	Window visible
4	Menu_window()	Menu displayed.
5	Option()	Correctly indicates the returned option.
6	Toggle()	Shows/Hides the interface.
7	sendMsg()	Changes the message on the dialog.
Post-conditions:		
Environmental needs:		
Stubs and Drivers:		
Other Information:		

4.4 Integration Tests

Not included in document due to time constraints.

4.5 Other White Box Tests

Not included in document due to time constraints.

5 Test Execution and Outcome

5.1 Overall report on testing

Systems Tests

The system tests were completed first because if the system test cases did not complete successfully then the application would not run.

Requirements Tests

Requirements testing followed the test plan which was designed for many of the test cases. Due to time constraints, the open ended nature of the game and the complexity of the tests it was not possible to test everything, however the tests created covered an area broad enough to ensure that the game was playable. During the testing no major issues were found, however due to the limited size of the testing team and the limited size of the testing equipment it is possible that certain actions or hardware configurations may cause issues with the game.

Unit Tests

The unit tests were performed on most of the public interfaces of the application. The list of tests had been created from a summary list of the complex set of public interfaces by identifying those that would be testable without the need for too much complex analysis. The tests that have been completed may also be largely considered integration tests. They were included as part of the integration tests as the unit tests were conducted on the methods rather than testing the integration directly. The majority of tests passed as expected

Integration Tests

Tests within the integration section were not written up and tested as there was not time to accommodate the testing. Much of the integration testing was carried out as part of the Unit Testing. Testing would have been completed to ensure that the project operated correctly under various conditions. The integration tests could be completed if further testing was required.

Other White Box Tests

Tests within the Other White Box tests category was not carried out due to time constraints. The tests would have consisted of code walkthroughs ensuring that there were no regions of unreachable code, no unused variables or any ensure that the code was as optimal as possible. These tests would have also been used to ensure that future developers would have easily been able to work with the code when used in conjunction with completed documentation.

5.2 Test Execution

5.2.1 System Tests

ID: 1.1	Test Name:	Compile source code
General Report on Execution: The source code successfully compiled with no errors.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 1.2	Test Name:	Runs on a windows environment
General Report on Execution: The application successfully executed in both cases.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
Impact on Subsequent Tests		
Recommendation		

5.2.2 Requirements Tests

ID: 2.1	Test Name:	Ships exist within the game world
General Report on Execution: The ships were created correctly		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.2	Test Name:	Planets and other objects (supergate) exist within the game world
General Report on Execution: Planets and objects created correctly		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.3	**Test Name:**	**GUI exists**
General Report on Execution: All elements of the GUI displayed correctly		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.4	**Test Name:**	**Move to a specific point in space**
General Report on Execution: The ship successfully moved to the specified point		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.5	**Test Name:**	**Move to an object within the game world (ship, planet etc)**
General Report on Execution: The ships successfully moved to the specified object		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.6	**Test Name:**	**Move to object and orbit**
General Report on Execution: The ship successfully moved to the object and orbited it		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.7	**Test Name:**	**Target object (ship, planet, etc)**
General Report on Execution:	The ship was successfully targeted and the information displayed within the interface	
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.8	**Test Name:**	**Move to target**
General Report on Execution:	The ships successfully moved to the target	
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.9	**Test Name:**	**AI ship movement**
General Report on Execution:	The source code successfully compiled with no errors.	
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
7	Yes	
8	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.10	Test Name:	Fire on target
General Report on Execution: The ships successfully fired on the target with the correct weapon		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
7	Yes	
8	Yes	
9	Yes	
10	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.11	Test Name:	AI ship targeting
General Report on Execution: The AI ships target ships correctly		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.12	Test Name:	Select weapon
General Report on Execution: The player could successfully swap between weapons		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.13	Test Name:	Engage in combat
General Report on Execution: The player and the AI were successfully able to engage in combat		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.14	Test Name:	Escape from combat
General Report on Execution: The player and the AI could successfully escape from combat		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.15	Test Name:	Main Menu
General Report on Execution: The main menu was created correctly and had the correct functionality		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
7	Yes	
8	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.16	Test Name:	In game menu
General Report on Execution: The in game menu was created and had the correct functionality		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
7	Yes	
8	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.17	Test Name:	Save
General Report on Execution: The player was able to save successfully		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	

Impact on Subsequent Tests
Recommendation

ID: 2.18	**Test Name:**	**Load**
General Report on Execution: The player was able to load successfully		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.19	**Test Name:**	**Enter Hyperspace**
General Report on Execution: The player could successfully enter hyperspace		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.20	**Test Name:**	**Exit Hyperspace**
General Report on Execution: The player could successfully exit hyperspace		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.21	**Test Name:**	**Travel through supergate**
General Report on Execution: The player could successfully pass through the supergate both ways		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.22	Test Name:	Ship destroyed
General Report on Execution:	Ships were successfully destroyed once they reached 0 hull points	
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.23	Test Name:	Gain experience
General Report on Execution:	The player was successfully able to gain experience	
Input #	**Passed?**	**Problems Discovered**
1	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.24	Test Name:	Collect loot
General Report on Execution:	The player was successfully able to collect materials off of the destroyed ship	
Input #	**Passed?**	**Problems Discovered**
1	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.25	Test Name:	Trade resources
General Report on Execution:	The player was successfully able to trade with the Asgard	
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.26	Test Name:	Call allies
General Report on Execution:	The player was successfully able to call allies	
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.27	**Test Name:**	**Capture Planet**
General Report on Execution: The player was able to capture a planet		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.28	**Test Name:**	**Change planet status**
General Report on Execution: The planet status was successfully changed		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.29	**Test Name:**	**Track current power levels**
General Report on Execution: The player was able to track the current power levels		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.30	**Test Name:**	**Move power between systems**
General Report on Execution: The player was able to move power between systems		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
7	Yes	
8	Yes	
9	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.31	**Test Name:**	**Ships have different statistics and models based upon race**
General Report on Execution: The ships had varied models and statistics		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 2.32	**Test Name:**	**Weapon deals different damage based on weapon type**
General Report on Execution: The weapons dealt different damage based upon the weapon type		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
Impact on Subsequent Tests		
Recommendation		

5.2.3 Unit Tests

ID: 3.1	**Test Name:**	**Game Controllers / AI Controller**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
7	Yes	
8	Yes	
9	Yes	
10	Yes	
11	Yes	
12	Yes	
13	Yes	
14	Yes	
15	Yes	
16	Yes	
17	Yes	
18	Yes	
19	Yes	
20	Yes	
21	Yes	
22	Yes	
23	Yes	
24	Yes	
25	Yes	
26	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.2	**Test Name:**	**Game Controllers / Data Loader**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.3	**Test Name:**	**Game Controllers / Data Manager**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.4	**Test Name:**	**Game Controllers / Game Controller**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
7	Yes	
8	Yes	
9	Yes	
10	Yes	
11	Yes	
12	Yes	
13	Yes	
14	Yes	
15	Yes	
16	Yes	
17	Yes	
18	Yes	
19	Yes	
20	Yes	
21	Yes	
22	Yes	
23	Yes	
24	Yes	
25	Yes	
26	Yes	
27	Yes	
28	Yes	
29	Yes	
30	Yes	
31	Yes	
32	Yes	
33	Yes	
34	Yes	
35	Yes	

36	Yes	
37	Yes	
38	Yes	
39	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.5	**Test Name:**	**Game Controllers / NewGame**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.6	**Test Name:**	**Ship AI / Lootable Ship**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.7	**Test Name:**	**Ship AI / Projectile Data**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.8	**Test Name:**	**Ship AI / Ship AI**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	

5	Yes	
6	Yes	
7	Yes	
8	Yes	
9	Yes	
10	Yes	
11	Yes	
12	Yes	
13	Yes	
14	Yes	
15	Yes	
16	Yes	
17	Yes	
18	Yes	
19	Yes	
20	Yes	
21	Yes	
22	Yes	
23	Yes	
24	Yes	
25	Yes	
26	Yes	
27	Yes	
28	Yes	
29	Yes	
30	Yes	
31	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.9	**Test Name:**	**Other / Hyperspace Window**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.10	**Test Name:**	**HUD / Ingame Menu**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
Impact on Subsequent Tests		

Recommendation

ID: 3.11	**Test Name:**	**HUD / Loading Screen**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.12	**Test Name:**	**HUD / Main Menu**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.13	**Test Name:**	**HUD / Object Panel**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
7	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.14	**Test Name:**	**HUD / Player Info**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	

7	Yes	
8	Yes	
9	Yes	
10	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.15	**Test Name:**	**HUD / Rep Ali Panel**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.16	**Test Name:**	**HUD / Screen Messages**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
7	Yes	
8	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.17	**Test Name:**	**HUD / Ship Control Panel**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
7	Yes	
8	Yes	
9	Yes	

10	Yes	
11	Yes	
12	Yes	
13	Yes	
14	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.18	**Test Name:**	**HUD / Target Select**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
7	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.19	**Test Name:**	**HUD / Top Overlay Script**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
7	Yes	
8	Yes	
9	Yes	
10	Yes	
11	Yes	
12	Yes	
13	Yes	
14	Yes	
15	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.20	**Test Name:**	**HUD / Weapons Panel**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
7	Yes	
8	Yes	
9	Yes	
10	Yes	
11	Yes	
Impact on Subsequent Tests		
Recommendation		

ID: 3.21	**Test Name:**	**HUD / Yes No Window**
General Report on Execution: Successful.		
Input #	**Passed?**	**Problems Discovered**
1	Yes	
2	Yes	
3	Yes	
4	Yes	
5	Yes	
6	Yes	
7	Yes	
Impact on Subsequent Tests		
Recommendation		

5.2.4 Other White Box Tests

Other white box tests were not completed due to time constraints.

5.3 Overall Recommendation

The application Stargate: Galaxy has been reviewed and tested using a moderate number of test cases. The test cases that have been executed were successful in verifying that the implementation is correct in its design. Ideally more testing would have been completed so that issues could be identified and then any issues discovered could be prioritised. Although there were no major issues found within the program it is possible that there are issues with the game. The products documentation was well presented and thorough. All together the testing team is happy that the product implemented satisfied the requirements.