Part 2a: Testing Plan & Test Cases

Testing Strategy

Purpose of this Test Plan

The main purpose of this test plan document is to ensure that all testing that takes place on the project is done in a concise, professional and traceable manor. Tests that occur at random intervals during the process of project development can be expensive, unprofessional and are almost certainly going to be undocumented. If all team members follow this test plan, then tests can be traced back to when it was carried out, its results, and whether or not things need to be done in order to get it done.

Validation & Verification

Verification testing is the process of ensuring that the product is being built correctly; these tests are in comparison with the product specification to ensure that what is being built is on the right track.

Validation testing is ensuring that what is being built adheres to the needs of the user or customer; essentially making sure the right product is being built.

The purpose of this section is to ensure that what we are developing is fir for purpose in its intended area of use. Our game has to be playable, able to be completed, and of course enjoyable and polished. The product specification we have for our game will be used to ensure we are on track for verification whilst looking at what an expected user wants, by conducting a user needs analysis, in comparison to what we are supplying.

Unit Testing

Our unit testing will be focussing on a specific part of the game and will be tested separately to the rest of the system. The unit we have decided to test is the Unit Class as it is the core part of the game. Units attack other units and buildings, some units can collect resources and repair structures. They can be moved by the user to specific locations through giving them orders with the mouse and of course can be destroyed when they run out of health.

Integration Testing

The integration testing involves taking two modules which have both been unit tested and test their ability to communicate and interact with each other. This is to see if they behave as expected. Sometimes when combining two modules, they can have unexpected effects on one another causing them to malfunction. For that reason, it is important for us to conduct this type of test before doing system testing to make sure related modules work together. Finding issues at the system level to do with integration is difficult and time consuming, as they are typically not obvious.

System Testing

System testing involves testing the entire solution as a whole; there is no separation of units, or integrating small sections with other small sections which take place. This is – usually – the final phase of testing, as it begins small (with unit testing for example) and ends big. Once all the units are confirmed as working, and units work with other small units through integration, then it

makes logical sense to carry out a full test to see how everything integrates as a whole. In our game, we will test the game from start to finish ensuring that the game responds to the user's input, entities interact with one another, saving and loading occurs as expected, a test to ensure that there is a victory condition, and so on.

Testing Plans

Unit Testing

Unit Importance

The Unit Class is of critical importance to the product, as large parts of the game hinge upon its functionality. If this section fails to work then we will have to remove large sections of the game effectively rolling it back to an alpha state where there was no unit implementation and they would have to be re-implemented.

Test Cases

Test Number	Test Description	Justification	Expected results
1	Hold the "W" key to check if the Tank will move forward	Vehicle movement is an important aspect of vehicle use	The vehicle should move forwards locally to where it is facing
2	Hold the "S" key to check if the Tank will move backwards	Similarly to test1, vehicle movement is critical to the module	The vehicle should move backwards locally to where it is facing
3	Press the "1" key to test whether the Tank can correctly target an object	Vehicle targeting will be needed for vehicle combat in-game	The vehicle's turret should rotate and point towards the front left building
4	Press the "2" key to test whether the Tank can change targets	Swapping targets will be required of the vehicles in game for combat	The vehicle's turret should rotate and point towards the front right building
5	Hold the "W" key to determine whether the Tank can target an object whilst moving	Vehicles will have to be moving whilst in combat, so this aspect will need to be tested	The vehicle's turret should stay pointing at the assigned target
6	Hold the "Q" key to rotate the Tank anti-clockwise	Vehicle movement will rely on its ability to rotate and orientate properly	The vehicle body should rotate anti-clockwise
7	Hold the "E" key to rotate the Tank clockwise	Vehicle movement will rely on its ability to orientate towards it's selected target	The vehicle body should rotate clockwise
8	Hold both the "E" key and the "W" key to test whether the vehicle can rotate and move at the same time	Vehicles will need to rotate and orientate at the same time to better represent realistic movement	The vehicle should move forward locally to where it is facing, and so should move around in a large circle

Contingency Plan

If the units fail to function as expected, and it is in an unfixable state, then we will have to

remove ground functionality and focus on the space unit interactions and combat. If it is in a fixable state then we will reduce functionality in order to present the final working product in a playable state.

Integration Testing

Integration Importance

The interaction between units and buildings is an essential feature of ground combat in our game. Without it, several sections would have to be refactored in order to get them to function as expected. For example, without ground combat, space units would have to be responsible for causing damage to buildings, mainly the Command Centre as that needs to be destroyed for victory to be achieved.

Test Cases

Test Number	Test Description	Justification	Expected results
1	Press the "1" key to assign a target to the tank	The integrated modules will affect the Tank's firing function, so this will need to be re-tested	The Tank's turret should rotate towards the building forward and to the right relative to the Tank
2	Once the target has been locked, wait for the tank to fire a shot at the target	The integrated module includes a process of vehicle attacks, and so must be tested with the vehicle module	The tank should fire a small pellet from its turret towards the target. Once the pellet reaches the target, it should create an explosion particle system, emitting from the target
3	One the target has received an attack from the vehicle, it should emit smoke to indicate damage	Part of the damage model will involve smoke to indicate damage, and so is a part of the particle module which will be tested	A smoke system should emit from the target, and should increase in intensity over a short length of time
4	Press the "2" key to change the target	Multiple particle systems are critical to the system, as many different game agents may have active particle systems	Like test 1, the tank turret should rotate to point at the target, which this time is back and to the right of the tank
5	Once the target has been locked, the tank should fire a shell at the new target	Similarly to the justification to test 4, the system needs to be tested for multiple particle system instances	The outcome should be identical to the expected results of test 2, with a pellet from the turret and an explosion system upon impact
6	Once the second target has been shot,	This test is to ensure that multiple instance	Smoke particle systems should be

	a smoke system should emit from the target	of multiple particle systems are supported, and that the performance of the program is not affected	updating on both targets without a significant hit to frame rate
7	Hold the “W” key to check if Tank movement is still supported	Despite the second module not effecting the vehicle movement, this test is to ensure that movement has not be compromised during integration	The vehicle should move forward locally to where it is facing
8	Hold the “Q” key to ensure that the Tank rotation is still functioning	Similarly to test 7, the particle system module integration should not have affected tank rotation, however this test is to check this is the case	The vehicle should rotate anti-clockwise

Stubs and Test Harnesses

As the module for vehicle movement has unit testing, we will need to construct a test stub. This will incorporate the features of the vehicle module, which will include the ability to move the vehicle, as well as rotating the vehicle and turret separately in order to demonstrate realistic targeting and movement.

In addition to a vehicle test harness, a test stub will need to be developed in order to test the functionality of particle systems module. The harness will be able to demonstrate the various particle system types that the game will implement, as well as test the performance of the program with multiple particle system instances.

Finally, a combined test stub for the two modules will need to be developed for integration testing. This test harness will be constructed in order to facilitate testing of both the vehicle and particle system functionality with each other, for example explosion systems resulting from vehicle attacks, as well as ensuring that modules have not affected each other in unforeseen way (e.g. vehicle attack particle systems affecting movement).

Test Scripts

Test Scripts Importance

It is important that the system functions as outlined by the client. Test Scripts will help to

ensure that what we have produced is in-line with what has been specified in the design documentation. The design documentation in question is centred on the Test Cases from the Use Case diagram.

Place Building Use Case Test Script

Case Number	Test Case	Expected Result
1	User selects a building to place	A 'ghost' of building appears and moves with player's mouse position
2	The user has a building selected	A 'ghost' of building appears and moves with player's mouse position
3	Comparison between cost of building and minerals of the player	When player clicks, the game should branch down one of two paths; sufficient or insufficient funds
4	Player has sufficient funds	If player has sufficient funds, no error message is shown and game progresses to check surrounding area
5	Game checks surrounding area to ensure it is clear of other structures	If surrounding area is clear, 'ghost' of building is shaded green
6	Building is not overlapping edge of the grid	If the building is not overlapping, it should again appear green
7	Game creates building at chosen location	Building no longer tracks the mouse, and the ghost appears white. Can no longer build in that location until building is deleted.
8	Game deducts cost of the building from the player	The player should have less minerals relevant to the price of the building
2A1	Insufficient minerals with error message	An alert should display stating that the player does not have enough minerals
4A1	Surrounding area used with error message	An error message should display stating that the area is in use
5A1	Building overlaps the edge with error message	An error message should display that the building is not in a valid location

Build Units Use Case Test Script

Case Number	Test Case	Expected Result

1	User selects building they want to purchase a unit from	Information for the current building, along with buttons for constructing units, are shown
2	Button for the unit the user wants to build is clicked	The game begins checks for the specific unit consistent with the button that is being clicked
3	Game compares cost of unit to the funds of the player	If the player has enough funds, the checks progress
5	Game checks to make sure unit production queue for that building is not full	If the production queue for the building is not full then the purchase of the unit will be allowed
7	Game deducts the cost of the unit from the player	The cost of the unit is deducted from the player
8	Building begins production of the unit	The timer until the unit is finished building begins counting down
3A1	An error message should display to show that the player does not have enough minerals for the unit purchase	The unit cannot be purchased and an error message is displayed
5A1	An error message should display to state that the queue for this building is full	The unit cannot be purchased and an error message appears specifying that the unit queue is full

Launch Attack Use Case Test Script

Case Number	Test Case	Expected Result
1	User chooses to launch an attack	The game should begin its transition into space state
2	The Game Checks to see if the player has any ships.	If The player has sufficient ships they are launched into space
3	The game asks how many ships the player wants to commit	A menu should appear to give the player options with regards
4	The game loads in the ships	The game should load in the ships into rows facing one another
5	The game enters the attack stage.	The ships should begin attacking one another
2A1	The player does not have any ships	The game should return to the world state and output an error message
3A1	The player chooses to cancel the attack	The game should return to the world state