## Introduction

### Goals

The goal for this test plan is to thoroughly test the resulting project’s software production to check consistency, balancing and for any errors.

### Scope

The test plan will cover all code written, there are no requirements to test any part of the graphical elements of the end software product. The test plan will not test things like user feedback because ultimately there are currently no plans for the project to reach the public.

### Schedule

#### Ad-Hoc

Ad-Hoc testing will be conducted throughout the software development process (being superseded by White and Black box testing at the end of the development cycle, therefore the schedule for ad-hoc testing is unknown and depends heavily on the situation.

#### White Box

White box testing will comprise 75% of the time allocated to software testing, this will provide an adequate chance to document any known bugs and fix them before release.

#### Black Box

Black box testing will comprise the remaining 25% of the allocated software testing time (seen in the project Gantt chart), the reason for this is that black box testing won’t cover quite as much as White box and is more a “surface” check to ensure everything is working fine on the consumers side.

## Objectives

### Ad-hoc

To test all code/software developed or applicable to the end software before release, this will happen during development.

### White Box

* Test all interactions between entities to ensure algorithms/interactions are performed correctly
* Test Initial setup values to ensure everything is initialised correctly
* Test Direct3D functions output to ensure correct models/graphics/sounds are being set correctly
* Test threading for each sub-process (see overarching flow diagram)
* Test functions with mock data to ensure all output/errors are process correctly/caught (below min, min, average, max, above max, obscure)

### Black Box

* Check graphics are loaded & displayed correctly in general
* Test movement controls
* Test collisions
* Check UI/UX for errors
* Test sounds
* Test win/lose
* Test High scores

## Dependencies & Risks

### Microsoft (GitHub)

* GitHub goes down
* Microsoft shut GitHub down
* Data is leaked and data protection laws are broken
* Files are lost

### Microsoft (Word)

* Microsoft Word crashes and corrupts the test results file resulting in a complete restart being required

### Microsoft (Visual Studio)

* A patch is released which corrupts files, resulting in a complete restart
* File not saved correctly, resulting in lost work
* Microsoft stops its free software program

### Microsoft (DirectX)

* An error in DirectX results in skewed test results
* Direct3D version is updated, and everything changes, resulting In the requirement to re-learn the language used

## General Risks

* Software product released without adequate testing, resulting in errors for the end user
* Not enough time to test everything to the level of detail expected
* Things like sickness/family issues which prevent full testing being completed
* Function testing not done correctly, and errors are released with the product
* White Box testing reveals massive issues with the programs core functionality which Ad-Hoc testing didn’t reveal

## Pass/Fail Criteria

### Ad-Hoc

As there are no clearly defined criteria for this type of testing (the tests are so varied and random that the criteria would cover literally everything in the other two types).

### White Box

#### Type of Test

1. Function Testing
   1. Minimum
   2. Below Minimum
   3. Average/Normal
   4. Maximum
   5. Above Maximum
   6. Obscure Value
2. Class Testing
   1. Initialisation
   2. Variable/Function Availability
   3. Inheritance
   4. Polymorphism
   5. Overloading

#### Pass Criteria

1. Function Testing
   1. The minimum value is entered into the function and the function provides correct response/output.
   2. Below the minimum value is entered into the function and the function throws an error/returns correctly incorrect response.
   3. The Average/Normal value is entered into the function and the function provides correct response/output.
   4. The maximum value is entered into the function and the function provides correct response/output.
   5. Above the maximum value is entered into the function and the function throws an error/returns correctly incorrect response.
   6. An obscure value is entered into the function and the function deals with this obscure value correctly without returning/providing an incorrect response or output.
2. Class Testing
   1. Initialisation sets the class up in the desired setting.
   2. Each function call or variable use is available/restricted depending on design specification.
   3. Inheritance for the class is setup correctly and child classes can access all relevant functions/variables of their parents.
   4. Functions accept required polymorphed variable parameters.
   5. Any overloads written output correctly when using controlled tests

### Black Box

#### Type of Test

1. Movement testing
2. Screen size testing
3. UI/UX testing
4. Shooting test
5. Collision testing
   1. Moving object
   2. Unmoving object
6. Win/Loss tests
7. High scores test
8. Quit game test

#### Pass Criteria

1. Moving the controls moves the player’s ship
2. Resizing using the options menu sets the game to fullscreen/windowed
3. UI is coherent and scales to screen size
4. Projectiles shoot when pressing the fire button, projectiles visibly travel towards target and if hit do damage
5. Collision testing
   1. Collision with the object makes both objects bounce off of each other (as well as plays sound effects and/or animation)
   2. Collision with the object makes the player ship bounce off of the unmoving object
6. Winning the game displayed the win screen, with appropriate play again/high scores/quit buttons, similarly losing the game displays the loss screen, with the same buttons
7. High scores are in order and seem to work correctly when testing the game consistently
8. Quitting the game at any point doesn’t break the program

## Results Documentation

The results of the test phase of the project will be culminated in a final document which has a grid of: **Type of Test (White, Black, Ad-Hoc)**, **Test performed**, **Expected result**, **Actual Outcome**, **Pass/Fail**, **Screenshot**, **Fix** (In brief). This will give us the required results in an easily viewable format with little to go wrong.