Oliver Collins-Cope

2102775@rutc.ac.uk

Learning Aim B & C

Carry out a range of testing methods on a software product to meet a client’s needs & Review and present the results from software tests to meet a client’s needs and suggest improvement

Unit 13 Software Testing

Assignment 2

Contents

[Introduction 2](#_Toc136331683)

[Game Description 2](#_Toc136331684)

[Using Agile 2](#_Toc136331685)

[Activity diagram 2](#_Toc136331686)

[Basic Test Plan 2](#_Toc136331687)

[Improved Test Plan 3](#_Toc136331688)

[User acceptance testing against requirements 3](#_Toc136331689)

[Analysis of test results 3](#_Toc136331690)

[Basic test plan 4](#_Toc136331691)

[Test 1 etc 4](#_Toc136331692)

[Improved test plan 4](#_Toc136331693)

[Test 1 etc 4](#_Toc136331694)

[Regression based testing for the future 4](#_Toc136331695)

[Evaluation 4](#_Toc136331696)

[Testing quality 4](#_Toc136331697)

[Justification for failed tests 4](#_Toc136331698)

[Valid reasons for skipped tests and not failed 4](#_Toc136331699)

[Product against tests 4](#_Toc136331700)

[System testing 4](#_Toc136331701)

[Test plan 4](#_Toc136331702)

# Introduction

This case study aims to depict the impact and importance of testing in different project management methodologies, and how choosing a different project management methodology can heavily influence the testing, ranging from the quality of tests, the tests used, and the final outcome of the tests. This will be documented in an orderly fashion in order to maintain a clear documentation of the work. Additional sources and projects used include a games development project, and can be found linked in the bibliography.

# Game Description

The game that I had decided to make for this was a 3D puzzle game that was supposed to meet a number of requirements. These requirements were various and include:

* 8+ years old
* Collision detection
* Score
* Levels
* Lives

In order to satisfy these requirements, I found unique ways to implement them into my game, like adding in a timer that counts down until the player loses their one life and restarts. Furthermore, I aimed to add two levels, therefore meeting another requirement that was provided to me. This is an example of the different types of requirements and how I worked to implement them in the game.

To summarise, I had a 3D puzzle game that met the above requirements, and these successfully made a completed game.

Game descriptions – agile test method activity diagram and *why* activity diagram was used

## Using Agile

When using Agile within my game I found that it was essential due to its iterative and flexible approach to development. When it came to testing the actual game, I found that the principles ingrained into Agile development were essential to the development of my game to a high quality. The software testing was seamlessly integrated throughout the whole of the development process, allowing for different iterative phases of testing and development. Testing was often run in tandem with development in order to ensure that all of the introduced features were fully functional, and any developments made had instant feedback on their impact and implementation.

Furthermore, the incremental nature of Agile allowed for smooth implementation and testing of new features, ensuring that all of the features were added correctly and there were no bugs within the software. The emphasis on prioritisation of tasks led to an increase in productivity when it came to resolving issues as they were put at the top of the to do list in order to ensure that final product was a bug free as possible. Regular self-reviews, as a placeholder for meetings, also allowed me to identify better what was a priority, i.e., a new feature vs bug fixes, allowing for a high-quality final product.

## Activity diagram

A picture containing text, diagram, screenshot, plan

Description automatically generated

A picture containing text, screenshot, diagram, parallel

Description automatically generated

An activity diagram is an essential component in game planning as it allows for the developers, like me, to create a clear understanding of how I wanted the games structure and progression to exist. The emphasis on using the activity diagram here is due to the illustrative ability it provides, displaying logical processes that must occur and that enable the player to move forward through these different points.

Through the visual display of these different points, like receiving a key, this offers a comprehensive overview of the players potential journey through the level, and what those objectives signify to the development/programming team, allowing for them to create and develop around those ideas and ensure a smooth playing experience where everything the player does makes sense.

This allows a clear view at the game’s progressions, while also having the additional benefit of identifying where the game can have key moments, like introducing different features or aspects at moments where they are needed to solve a problem, therefore teaching the player through doing things.

Finally, the incorporation of the activity diagrams in the final plan of the game has aided in creating a streamlined experience for the player where the game progresses linearly and in a way that provides the best experience.

# Basic Test Plan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Test case** | **Type of Test** | **Expected outcome** | **Actual outcome** | **Result** | **Evidence** |
| 1. | Player will be able to pick up objects (i.e., flashlight) | Unit testing | The player will be able to pick up the object. | The object could not be picked up. | Fail |  |
| 2. | Player being able to turn a flashlight on and off by pressing a button. | Unit testing | The player will be able to turn the flashlight on and off. | The player turned the flashlight on and off. | Pass |  |
| 3. | The player is able to move around and pick up the flashlight, and then turn it off and on. | Element testing | The player will be able to move around and turn it off and on after picking it up. | The player was able to move and interact with the flashlight, picking it up and turning it off and on. | Pass |  |
| 4. | The player will input a password into a keypad, and it will open a door next to it, both having different scripts. | Integration testing | The door will open after the password is verified. | The password flashed green and then the door opened | Pass |  |
| 5. | The game will run smoothly without discrete graphics card. | Performance testing | There game will run without issue on both levels and function well. | The second level encountered some performance issues. | Failed |  |
| 6. | The cube, when interacted with, will load the next level, and put the player there. | Element testing | The next level will load without issue. When the player interacts with the cube. | Interacting with the cube did nothing. | Failed |  |
| 7. | The player will be able to pick up the key and use it to interact with the door on the first level. | Integration testing | The key can be picked up and the door is opened. | The door was opened perfectly fine after the key was picked up. | Pass |  |
| 8. | The player will be able to pick up the books and put them in the bookshelf. | Element testing | The player will be able to interact and pick up the books, and then put them in the shelf. | The player was able to pick up the books and put them in the shelf. | Pass |  |

Test plan – Features etc/ test number, test case, type of testing, result, expected vs actual outcome, evidence, result = pass/failed/skipped

**Unit testing, functional testing, integration testing**

QA processes

# Improved Test Plan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Test case** | **Type of Test** | **Expected outcome** | **Actual outcome** | **Result** | **Evidence** |
| 1. | The player will be able to pick up objects. | Unit testing | The player will be able to interact and pick up objects. | The player picked up the objects. | Pass |  |
| 2. | The game will run smoothly without a discrete graphics card. | Performance testing | The game will function well on both levels smoothly. | Both levels are running smoothly now. | Pass |  |
| 3. | The cube, when interacted with, will load the next level, and put the player there. | Integration testing | The next level will load without issue. When the player interacts with the cube. | The next level is loaded when interacting with the cube. | Pass |  |

Test plan 2 (test plan improvements) – for failed tests/ improved test plan

# User acceptance testing against requirements

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Test case** | **Type of Test** | **Expected outcome** | **Actual outcome** | **Result** | **Evidence** |
| 1. | There are multiple levels in the final game. | Acceptance testing | There will be more than one level in the final game. | There are two levels in the game, not including the main menu. | Pass |  |
| 2. | There is a life system and life deduction in the game. | Acceptance testing | The player will die if the timer runs out. | The player restarts from the beginning if the timer runs out. | Pass |  |
| 3. | Collision detection will be present. | Acceptance testing | The player will not fall through the world and be able to collide. | The collision detections is there and the player cannot fall through the floor. | Pass |  |
| 4. | There will be score and adding score. | Acceptance testing | The score will increase when the level progresses. | The player gets +1 score when going to level 2. | Pass |  |
| 5. | The game is suitable for 8+. | Acceptance testing | The levels will be safe for all ages and the puzzles will not be too simple to prevent older audiences playing. | The levels are designed that although literally dark, are not creepy or gory, making it suitable for 8+. The puzzles are solvable for all ages, although some will take a little more time. | Pass |  |

Versions of unity

UAT test/ acceptance testing user requirements

Potential errors

# Analysis of test results

Analysis of test results (result 1 etc, result 2)

## Basic test plan

### Test 1 etc

## Improved test plan

### Test 1 etc

## Regression based testing for the future

# Evaluation

Evaluation (high level testing, product testing, system testing) constructive/positive

## Testing quality

### Justification for failed tests

### Valid reasons for skipped tests and not failed

## Product against tests

## System testing

## Test plan

what would you do next time

what did you do now

how well will did I do it

Referencing work constantly

|  |  |  |
| --- | --- | --- |
| **Features** | **Test results** | **Test evidence** |
|  |  |  |
|  |  |  |
|  |  |  |