Software Test Report

Project Domino (code name)

Prepared by:

Avik Shenoy, Thanh Tran,

Daniel Sarkisian, Amir

Roochi, Brendan Beagin

05/10/2023

TABLE OF CONTENTS

1 Introduction

1.1 Document overview

This document is the software test report of the final testing phase of the Edmund and Eliza software development project. It contains the results of tests, which were executed from 2022/12/01 to 2023/05/08.

1.2 Abbreviations and Glossary

1.2.1 Abbreviations

UI - user interface
NPC - non-player character
PC - player character
HUD - head-up display
CPU - central processing unit
GPU - graphics processing unit
CTest - CMake Test
GDT - Gameplay-Debugger Tool
N/A - not applicable

1.2.2 Glossary

Unreal Engine - a powerful game-development engine created by Epic Games. Unreal Engine provides a comprehensive suite of tools and technologies that enables developers to create high-quality, immersive, and interactive 3D games, virtual reality experiences, augmented reality applications, and other interactive content.

CTest - a testing framework used in conjunction with CMake which provides a set of utilities and commands that facilitate the execution of tests, the collection of test results, and the generation of test reports.

Gameplay-Debugger Tool - software development tool provided by Unreal Engine which assists game developers by providing a set of visual and interactive tools allowing for developers to monitor and inspect various aspects of the game's runtime behavior.

1.2.3 Project References

#	Document Identifier	Document Title
[R1]	1	Peer Review 3
[R2]	2	Software Test Plan

1.2.4 Standard and regulatory References

N/A

1.3 Conventions

To conduct testing a combination of CTest and GDT was used. First CMake and CTest were set up and configured to generate the build system of Edmund and Eliza. Then all of the test cases were defined in the build-in testing framework of Unreal Engine 5. Most of the test cases utilized checks within them to verify the expected behavior was met. A CTest driver was then set up to execute the test cases automatically. It generated a report of the test results indicating the test outcomes with pass, fail, or error. Real-time debugging was also utilized using the GDT to inspect various aspects of the game during runtime and ensure it was working as intended.

2 Overview of Tests Results

2.1 Tests log

The Edmund and Eliza game was mostly play tested because of its strong dependence on human interaction, although some features were unit tested with the use of CTest and GDT. tests were mostly conducted from the 2022/12/01 to the 2023/05/08. The following features were tested:

1. Movement
2. Combat
    a. Enemy
        i. Patrol
        ii. Target acquisition and navigation
        iii. Attack
        iv. Damage
    b. Player
        i. Attack
        ii. Dodge
        iii. Damage
3. Inventory
    a. Picking up item
    b. Using item
        i. Use effects
4. HUD
    a. Player UI
    b. Inventory UI

Testers where:

- Avik Shenoy,
- Thanh Tran

2.2 Rationale for decision

After executing a test, the decision is defined according to the following rules:

- **OK:** The test sheet is set to "OK" state when all steps are in "OK" state. The real result is compliant to the expected result.
- **NOK:** The test sheet is set to "NOK" state when all steps of the test are set to "NOK" state or when the result of a step differs from the expected result.
- **Partial OK:** The test sheet is set to "Partial OK" state when at least one step of the test is set to "NOK" state or when the result of a step is partially compliant to the expected result. ? Keep it or remove. Source of inconsistencies: criteria to set if result is Partial OK may be qualitative
- **NOT RUN:** Default state of a test sheet not yet executed.
- **NOT COMPLETED:** The test sheet is set to "Not Completed" state when at least one step of the test is set "Not Run" state.

Test results are listed in section 3.

2.3 Overall assessment of tests

Give a qualitative overall assessment of tests.

- All features that were listed in 2.1 were tested and passed.
- features that needed to be unit tested were mostly tested during the development process and then tested together with other features to make sure they are linked together. All above mentioned tests were passed.

Give quantitative results.
Statistics about tests:

- 100 % of tests OK,

Give also statistics about bugs and enhancements (refer to R1):

- Total number: 13
- Number of Critical: 0
- Number of Major: 8
- Number of minor: 5
- Number of enhancements: 8

2.4 Impact of test environment

Since our project is a third person game, the test environment can have a significant impact on the testing process. In the case of Edmund and Eliza the test environment includes a variety of factors such as the hardware platform, operating system, graphics card, network connection and input devices. Although testing in a simulator or software test tools can help to simulate various conditions that may not be possible to reproduce in a real environment these tools are not always accurately reflect the real-world conditions such as network latency and high CPU or GPU usage and it can lead to false positives or negatives in testing results. A good example of this situation is when the hardware used in the real world environment does not meet minimum requirements listed for running the game and it can lead to the game not working as expected.
In summary, the impact of the test environment on any game which is developed using Unreal Engine can be significant and testing in a real environment is essential to ensure that the product works as intended. So our main goal in the testing process was to test the game in real-world conditions whenever possible to achieve the best testing results.

3 Detailed Tests Results

3.1 Movement (OK)

Test ID	**3.1 Movement.**	**Comment**	**Decision**
Test description	This test is designed to verify that the movement system and third-person camera system behaves as expected in response to user input.		
Verified Requirement	Func-001.1		
Initial conditions	Player Character standing still		
Tests inputs	Hardware: A, W, S, D, Spacebar keyboard input, Mouse inputs		
Data collection actions	N/A		
Tests outputs	Player Character moves accordingly to inputs Camera moves according to input		
Assumptions and constraints	N/A		
Expected results and criteria	1. The camera should move smoothly and responsively, and should not get stuck or move erratically. 2. PC moves according with the inputs tightly		
Test procedure	Tester runs debug build of the game		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**
1	Operator moves the mouse to the left	Camera rotates smoothly to the left, maintaining a clear view of the player character	OK
2	Operator moves the mouse to the right	Camera rotates smoothly to the right, maintaining a clear view of the player character	OK
3	Operator moves the mouse up	Camera tilts up, maintaining a clear view of the player character	OK
4	Operator moves the mouse down	Camera tilts down, maintaining a clear view of the player character	OK

5	Operator presses the 'A' key	Camera moves to the left, maintaining a clear view of the player character	OK

3.2 Enemy Patrol (OK)

Test ID	**Description**	**Comment**	**Decision**
Test description	This test verifies that the enemy patrol system moves the NPC character smoothly and consistently along a predetermined path in the patrol area.		
Verified Requirement	Func-001.3		
Initial conditions	1. Player character is standing near the enemy patrol area 2. Enemy character is patrolling between predetermined points in the patrol area		
Tests inputs	N/A		
Data collection actions	N/A		
Tests outputs	Enemy character moves smoothly and consistently along its patrol path in the patrol area		
Assumptions and constraints	N/A		
Expected results and criteria	Enemy character moves smoothly and consistently along its patrol path in the patrol area		
Test procedure	Tester runs debug build of the game		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**
1	Operator observes the enemy character's patrol path between predetermined points	Enemy character moves smoothly and consistently along its patrol path	OK

2	Operator moves the enemy character to a different predetermined point in the patrol area	Enemy character moves smoothly to the new predetermined point	OK
3	Operator moves the enemy character towards a nearby obstacle	Enemy character navigates smoothly around the obstacle and resumes patrolling along its path	OK
4	Operator moves the enemy character towards the edge of the patrol area	Enemy character turns around and resumes patrolling along its path without leaving the designated area	OK

3.3 Enemy Target acquisition and navigation (OK)

Test ID	**Description**	**Comment**	**Decision**
Test description	This test verifies that the enemy can successfully acquire and navigate towards a target within the game environment.		
Verified Requirement	Func-001.3		
Initial conditions	1. PC is located in game environment 2. Enemy is patrolling around		
Tests inputs	N/A		
Data collection actions	N/A		
Tests outputs	Enemy character acquired PC as target and moves toward to engage		
Assumptions and constraints	N/A		
Expected results and criteria	Enemy character successfully acquires the target object and navigates towards it within the game environment		
Test procedure	Tester runs debug build of the game		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**

1	Operator observes the enemy character's behavior before encountering the target object	Enemy character moves randomly within the game environment, not actively targeting the player character or any other object	OK
2	Operator places the target object within the enemy character's range	Enemy character detects the target object and begins moving towards it	OK
3	Operator moves the target object away from the enemy character	Enemy character adjusts its path and continues moving towards the target object	OK
4	Operator moves the player character within the enemy character's range	Enemy character detects the player character and begins moving towards it instead of the target object	OK
5	Operator moves the player character out of the enemy character's range	Enemy character loses track of the player character and resumes moving towards the target object	OK

3.4 Enemy Target Attack & Damage (OK)

Test ID	**Description**	**Comment**	**Decision**
Test description	This test verifies that the enemy can successfully attack a target and inflict damage.		
Verified Requirement	Func-001.3		
Initial conditions	PC is targeted by enemy character		
Tests inputs	N/A		
Data collection actions	N/A		
Tests outputs	Enemy attacks Character, inflicting damage		
Assumptions and constraints	N/A		
Expected results and criteria	Enemy character successfully attacks the target object and inflicts damage		

Test procedure	Tester runs debug build of the game		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**
1	Operator moves the player character within the enemy character's range	Enemy character targets the player character	OK
2	Operator stands still with the player character within the enemy character's engage range	Enemy character decides to either attack or circle around the player character	OK
3	Operator stands still with the player character and allows the enemy character to attack	Player character loses health points as a result of the enemy character's attack	OK

3.5 Player Attack & Damage (OK)

Test ID	**Description**	**Comment**	**Decision**
Test description	This test verifies that the player character can successfully attack and damage the enemy character.		
Verified Requirement	Func-001.3		
Initial conditions	1. Player character is located in the game environment 2. Enemy character is located in the game environment		
Tests inputs	Hardware Input: "Left-Click" to initiate player character's attack		
Data collection actions	N/A		
Tests outputs	1. Enemy character loses health points as a result of the player character's attack 2. Player character successfully attacks and damages the enemy character		
Assumptions and constraints	N/A		
Expected results and criteria	Player character successfully attacks and damages the enemy character		

Test procedure	Tester runs debug build of the game		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**
1	Operator moves the player character to engage enemy character	OK	Ok
2	Operator initiates the player character's attack using the "Left-Click" keyboard input	Animation plays. Enemy character loses health points as a result of the player character's attack	OK

3.6 Player Dodge (OK)

Test ID	**Description**	**Comment**	**Decision**
Test description	This test verifies that the player character can successfully dodge enemy attacks and that dodging consumes player stamina.		
Verified Requirement	Func-001.3		
Initial conditions	1. Player character is located in the game environment 2. Enemy character is located in the game environment		
Tests inputs	Hardware input: Player pressed key for dodge.		
Data collection actions	N/A		
Tests outputs	Player character successfully dodges enemy attacks and takes no damage		
Assumptions and constraints	N/A		
Expected results and criteria	Player character successfully dodges enemy attacks, takes no damage, and consumes stamina		
Test procedure	Tester runs debug build of the game		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**

1	Operator moves the player character within the enemy character's range	Enemy character targets player	OK
2	Enemy character initiates an attack on the player character	Player character takes damage	OK
3	Operator initiates the player character's dodge using the dodge keyboard input	Enemy character's attack misses the player character, and player character's stamina decreases	OK

3.7 Inventory Picking up Item(OK)

Test ID	**Description**	**Comment**	**Decision**
Test description	Inventory system testing, picking up an item and storing it inside the inventory		
Verified Requirement	Func-001.6		
Initial conditions	Player character is standing near an item		
Tests inputs	Keyboard input to interact with the item		
Data collection actions			
Tests outputs	Item is added to the player character's inventory		
Assumptions and constraints			
Expected results and criteria	Item is successfully picked up and added to the player character's inventory		
Test procedure	Tester runs debug build of the game		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**
2	Interact with item using keyboard input	Item is added to player character's inventory	OK
3	Check player character's inventory	Item is present in player character's inventory	OK

3.8 Using Item(OK)

Test ID	**Description**	**Comment**	**Decision**
Test description	Inventory system testing, Using an item to heal		
Verified Requirement	Func-001.6		
Initial conditions	Player character has a heal item in their inventory		
Tests inputs	Keyboard input to use the item		
Data collection actions			
Tests outputs	Item effect is applied (heal)		
Assumptions and constraints			
Expected results and criteria	Item effect is successfully applied (heal)		
Test procedure	Tester runs debug build of the game		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**
1	Tester opens the player character's inventory	Inventory should be displayed	OK
2	Tester clicks on an item in the inventory to use it	Item effect should be applied	OK
3	The item effect is applied, resulting in the player character being healed	Player character should be healed by the item	OK
4	Tester checks the player character's status to ensure that the healing effect has been applied	Player character's health should be increased	OK

3.9 HUD(OK)

Test ID	**Description**	**Comment**	**Decision**
Test description			
Verified Requirement	Func-001.4		
Initial conditions	Player character is in-game		
Tests inputs	Keyboard input		
Data collection actions			
Tests outputs	1. Health and stamina bars are displayed and are updated when appropriate 2. Inventory is opened when player presses inventory key		
Assumptions and constraints			
Expected results and criteria	The HUD should accurately display the player character's health and stamina levels and the inventory.		
Test procedure	Tester runs debug build of the game		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**
1	Tester starts the game	Game should load successfully and the HUD should be displayed	OK
2	Tester verifies that the health and stamina bars are displayed	Health and stamina bars should be visible on the HUD	OK
3	Tester verifies that the health and stamina bars accurately reflect the player character's current levels	Health and stamina bars should decrease when the player character takes damage or performs actions that use stamina, and should increase when the player character is healed or rests to recover stamina	OK

4	Tester verifies that the health and stamina bars are updated in real-time	Health and stamina bars should update immediately when the player character's health or stamina changes	OK
5	Tester performs a dodge maneuver	The stamina bar should decrease and then gradually recover over time	OK
6	Tester allows the player character to be attacked by an enemy	The health bar should decrease accordingly	OK
7	Tester opens the player character's inventory screen	The inventory screen should be displayed	OK