Test Plan Documentation

Craig Gorney, Spencer Kruse, Nathan Jonas

Part I. Description of Test Plan

Our project has three main testing components:

1. First person shooting dynamics
2. Reinforcement-based learning for enemy decision-making
3. User flow and experience

The test plan we have created aims to cover all possible outcomes for the three components mentioned above. This will prevent bugs, glitches, and any unexpected results within our video game. By doing so, we will be creating tests before, during, and after the development of work items in our backlog. This will encourage consistency and in-depth testing protocol without having a team member in charge of quality assurance.

Ad Hoc testing refers to a more random approach to testing, and this testing is done throughout development with limited structure. As each work item is developed, we have decided to make sure the functionality of the work item is as expected and consistent with our planning. This decreases the functional tests required after development.

During our planning of work items, we utilize “clean room and tree testing” where we plan out a work item and describe a suitable set of rules and limitations for development. These rules can be described as instructions for development that will guide us when we are stuck or pondering about what is next.

Finally, we will use functional and integration testing for interactions with and between game components. These tests can be seen below and are to be fulfilled by the end of development.

Part II. Test Case Descriptions

1. **AI Control Test 1**
2. This test will ensure that the pawn will not enter a combat related state when there is not a combat situation.
3. This test will be done by ensuring that the AI does not enter any of the combat states using the debug menu (follow, shoot, strafe, approach) when the player character is out of sight and not creating noise. The results are ensured by placing the player character out of bounds and recording the debug decision tree states.
4. Inputs: The inputs to this test will be a test out of bounds platform and a test state recorder both only available in editor. The user starts the player character on the platform and records for 3 minutes.
5. Outputs: The test state recorder logs will not show combat states. (follow, shoot, strafe, approach)
6. Normal
7. Blackbox

8. Functional
9. Integration Test
10. Expected Results: The AI controlled pawn will not sense the player and therefore never enters a combat state.
11. Current Results: The AI is being controlled by blackboard, so test is nullified until reinforcement learning program is linked.

1. **AI Control Test 2**
2. This test will ensure that the pawn will enter a combat related state when there is a combat situation.
3. This test will be done by ensuring that the AI enters any of the combat states using the debug menu (follow, shoot, strafe, approach) when the player character is in sight and/or creating noise. The results are ensured by placing the player character in front of the AI for sight, as well as a separate test with the character shooting behind a nearby wall to create noise. The results of these tests decision tree states will be recorded.
4. Inputs: The inputs to this test will be moving the player to the commented positions in the editor for the test and following the instructions above. The user starts the player with high health and records for 3 minutes.
5. Outputs: The test state recorder logs will show most/all combat states after the initial recognition of player. (follow, shoot, strafe, approach)
6. Normal
7. Blackbox
8. Functional
9. Integration Test
10. Expected Results: The AI controlled pawn will sense the player and enters a combat state until it either loses track of the player or the player dies.
11. Current Results: The AI is being controlled by blackboard, so test is nullified until reinforcement learning program is linked.

1. **AI Growth Test 1**
2. This test will ensure that the AI controlled pawn will find the player character using the non-combat states as well as search the map in an intelligent manner.
3. This test will be done by spawning the player in a random point on the map, and waiting for the AI to come find and kill the player. The time results from spawn to player death averaged every 5 rounds (accounting for random spawns being closer), will be recorded. The results will be the average over time.
4. Inputs: The inputs to this test will be moving the player to random using a debug blueprint in the editor and follow the instructions above.
5. Outputs: The debug blueprint recordings of the average times to find the player throughout the rounds.
6. Normal
7. Blackbox

8. Performance
9. Integration Test
10. Expected Results: The AI controlled pawn will find the player on average faster each 5-round cycle before reaching a plateau average.
11. Current Results: The AI is being controlled by blackboard, so test is nullified until reinforcement learning program is linked.

1. **AI Growth Test 2**
2. This test will ensure that the AI controlled pawn will learn to use the cover and strafe states more upon hearing enemy fire.
3. This test will be done by spawning the player behind the enemy vision and using an editor only debug blueprint having the player instantly fire at the AI pawn. The AI state results will be recorded each round using an editor only blueprint.
4. Inputs: The inputs to this test are playing the game with the auto-shoot and recording debug blueprints enabled.
5. Outputs: The debug blueprint recordings of the AI states over the course of the game.
6. Normal
7. Blackbox
8. Performance
9. Integration Test
10. Expected Results: The AI controlled pawn will use the strafe or cover states more often on average each 5-round cycle.
11. Current Results: The AI is being controlled by blackboard, so test is nullified until reinforcement learning program is linked.

1. **AI Growth Test 3**
2. This test will ensure that the AI controlled pawn will learn to effectively aim over the course of the game.
3. This test will be done by placing the player at a random choice of 9 locations laid out in a grid. The AI will notice the player and start taking shots. The number of shots per round to kill the player will be recorded with the AI blueprint. The number of shots will be averaged every 5 rounds.
4. Inputs: The inputs to this test are running the debug aim training simulator level.
5. Outputs: The average number of shots needed to kill the player every 5 rounds.
6. Normal
7. Blackbox
8. Performance
9. Integration Test
10. Expected Results: The AI controlled pawn will need less shots on average over the course of the game until reaching a plateau.
11. Current Results: The AI is being controlled by blackboard, so test is nullified until reinforcement learning program is linked.

1. **Map Control Test 1**
2. This test will ensure that the map is a stable gameplay environment.
3. This test will be done by manually searching for physics bugs involving the launchers, and anything else that gives speedups to ensure the player cannot clip through objects. This test also involves rechecking the navmesh in the editor for bad geometry.
4. Inputs: The inputs to this test will be a manual player using each meaningful combinations of movement to approach the launcher. (W+A, W+D, …) Also involves the player using the in editor camera to check for navmesh errors.
5. Outputs: A list of navmesh and physics errors causing an unstable game environment.
6. Abnormal
7. Whitebox
8. Functional
9. Unit Test
10. Expected Results: The level will be reworked until this test returns no results.
11. Current Results: One physics bug exists. Shooting a projectile downwards while on the launcher causes the player to halt momentum as the projectile has physics and the engine considers the player to have hit a solid object.

1. **Map Control Test 2**
2. This test will ensure that there are no dark areas to give the AI and advantage over the player inside the house.
3. This test is done by moving around inside the house at the center of the map and making sure you can see all obstacles and entities in the house
4. Inputs: Simple movement commands from the player
5. Outputs: Visual light levels that make it so the user can traverse the house
6. Abnormal
7. Blackbox
8. Functional
9. Unit Test
10. Expected Results: Player can see the walls and objects clearly in order to make an informed decision about their movement
11. Current Results: Some lighting is a little low and will need to be enhanced

1. **Map Control Test 3**
2. This test will verify the landscape boundaries for user movement in x and y directions.
3. This test is accomplished by walking around the expected landscape boundaries and preventing the user from moving into unexpected terrain. The landscape is large; however, a small portion of the map is available for gameplay.
4. Inputs: manual player, button combination W+A while walking along the entire perimeter, and button combination W+D while walking along the entire perimeter

5. Outputs: List of boundary bugs. Failure leads to an attempted fix and a re-run of the test until success.
6. Abnormal
7. Whitebox
8. Functional
9. Unit Test
10. Expected Results: Empty list of boundary bugs.
11. Current Results: Empty list of boundary bugs.

1. **Map Control Test 4**
2. This test will verify the landscape boundaries for user movement in the z direction.
3. This test is accomplished by walking and jumping simultaneously around the expected landscape boundaries and preventing the user from moving into unexpected terrain.
4. Inputs: manual player, button combination W+A+SPACE while walking along the entire perimeter, and button combination W+D+SPACE while walking along the entire perimeter
5. Outputs: List of boundary bugs. Failure leads to an attempted fix and a re-run of the test until success.
6. Abnormal
7. Whitebox
8. Functional
9. Unit Test
10. Expected Results: Empty list of boundary bugs.
11. Current Results: One boundary bug involving slope near negative positive axis boundary.

1. **Map Control Test 5**
2. This test will verify the building boundaries for user movement in the x and y direction.
3. This test is accomplished by walking around each room (and objects within each room) in the building and preventing the user from moving in unexpected ways.
4. Inputs: manual player, button combination W+A while walking along the entire perimeter and objects in each room, and button combination W+D while walking along the entire perimeter and objects in each room
5. Outputs: List of boundary bugs. Failure leads to an attempted fix and a re-run of the test until success.
6. Abnormal
7. Whitebox
8. Functional
9. Unit Test
10. Expected Results: Empty list of boundary bugs.
11. Current Results: Empty list of boundary bugs.

1. **Map Control Test 6**

2. This test will verify the building boundaries for user movement in the z direction.
3. This test is accomplished by walking and jumping simultaneously around each room (and objects within each room) in the building and preventing the user from moving in unexpected ways.
4. Inputs: manual player, button combination W+A+SPACE while walking along the entire perimeter and objects in each room, and button combination W+D+SPACE while walking along the entire perimeter and objects in each room
5. Outputs: List of boundary bugs. Failure leads to an attempted fix and a re-run of the test until success.
6. Abnormal
7. Whitebox
8. Functional
9. Unit Test
10. Expected Results: Empty list of boundary bugs.
11. Current Results: Empty list of boundary bugs.

1. **Map Control Test 7**
2. This test will ensure that each of the walls and floors in the house have the correct collision meshes.
3. This will be accomplished by having the tester fire into each of the walls and floors to make sure no projectiles can pass through a wall or floor
4. Inputs: Basic movement commands and fire command
5. Outputs: The projectile will either bounce off the wall/floor indicating a correct collision mesh, or it will pass through indicating a missing collision mesh
6. Normal
7. Blackbox
8. Functional
9. Unit Test
10. Expected results: All projectiles bounce off of the walls and floors.
11. Current Results: All projectiles were repelled as expected.

Part III. Test Case Matrix:

	Normal/Abnormal	**Blackbox/Whitebox**	**Functional/Performance**	**Unit/Integration**
AC1	Normal	Blackbox	Functional	Integration
AC2	Normal	Blackbox	Functional	Integration
AG1	Normal	Blackbox	Performance	Integration
AG2	Normal	Blackbox	Performance	Integration
AG3	Normal	Blackbox	Performance	Integration
MC1	Abnormal	Whitebox	Functional	Unit Test
MC2	Abnormal	Whitebox	Functional	Unit Test
MC3	Abnormal	Whitebox	Functional	Unit Test
MC4	Abnormal	Whitebox	Functional	Unit Test
MC5	Abnormal	Whitebox	Functional	Unit Test

MC6	Abnormal	Whitebox	Functional	Unit Test
MC7	Normal	Blackbox	Functional	Unit Test