Appendix A:

* PDD

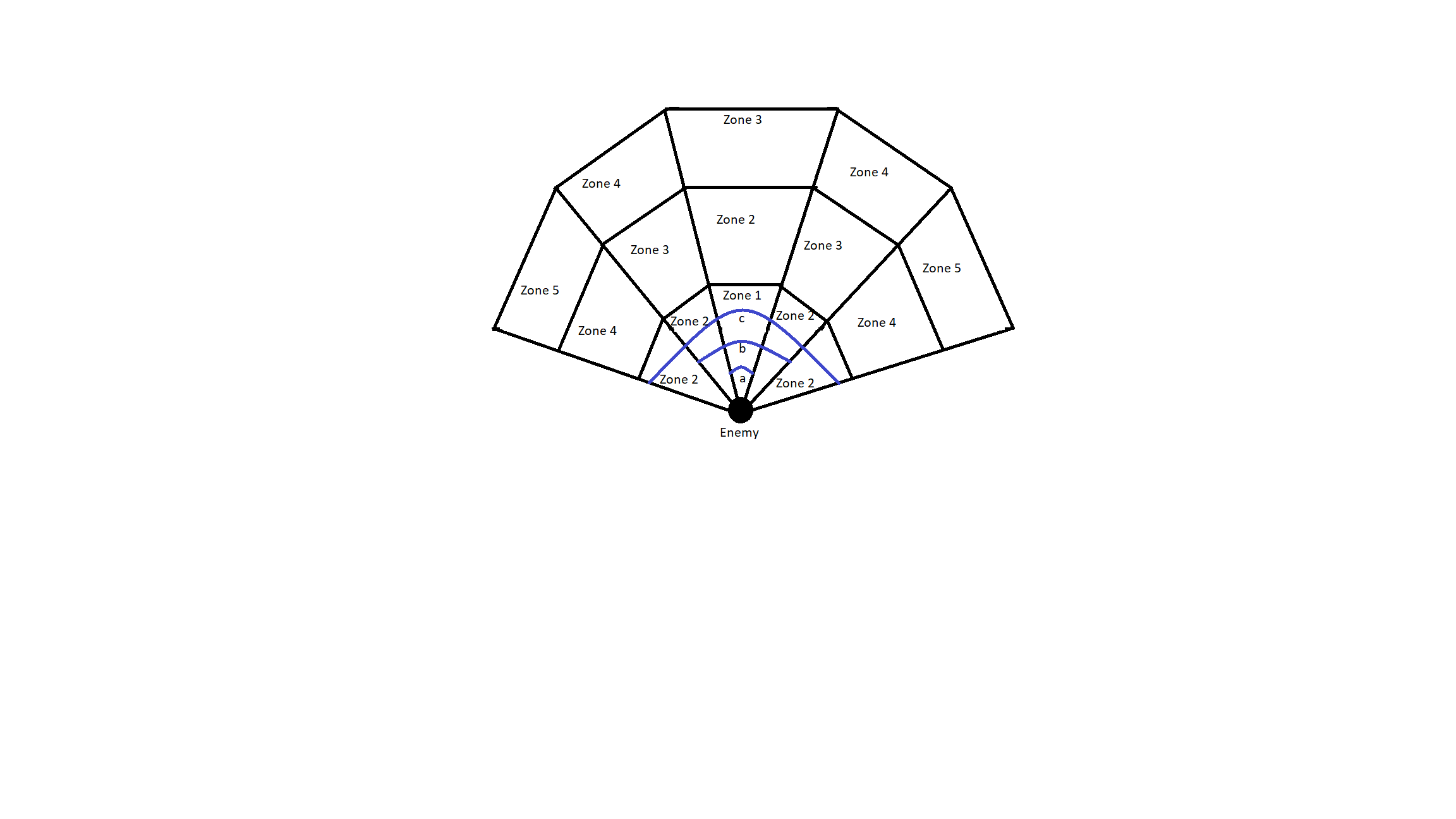
Appendix B:

REUSE SUMMARY

* Unity
* MS Paint
* Visual Studio
* Adobe Behaviour Tree Visual Editor
* Piskel

Appendix C:

MODELS/DIAGRAMS



*An example of an Enemy Vision Cone. 15 vision zones, split into 5 zone types. The higher the zone type, the slower the detection. 'a', 'b' and 'c' are the close, medium and wide angles respectively, displayed with the blue lines.*

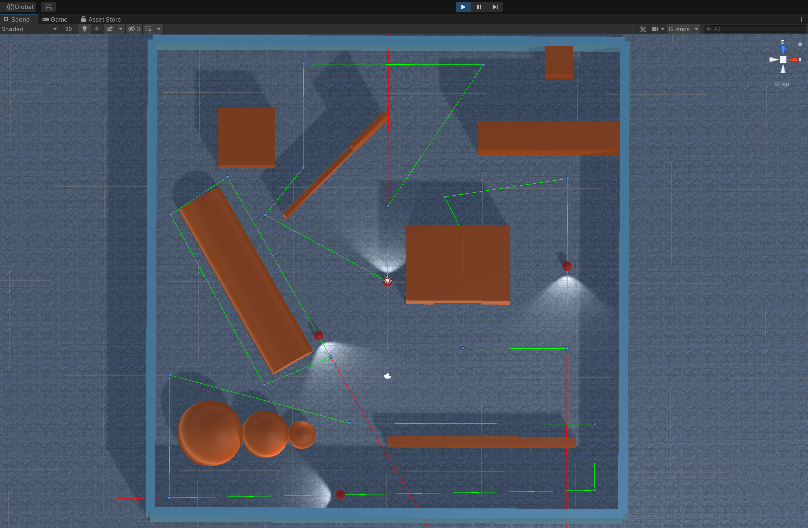
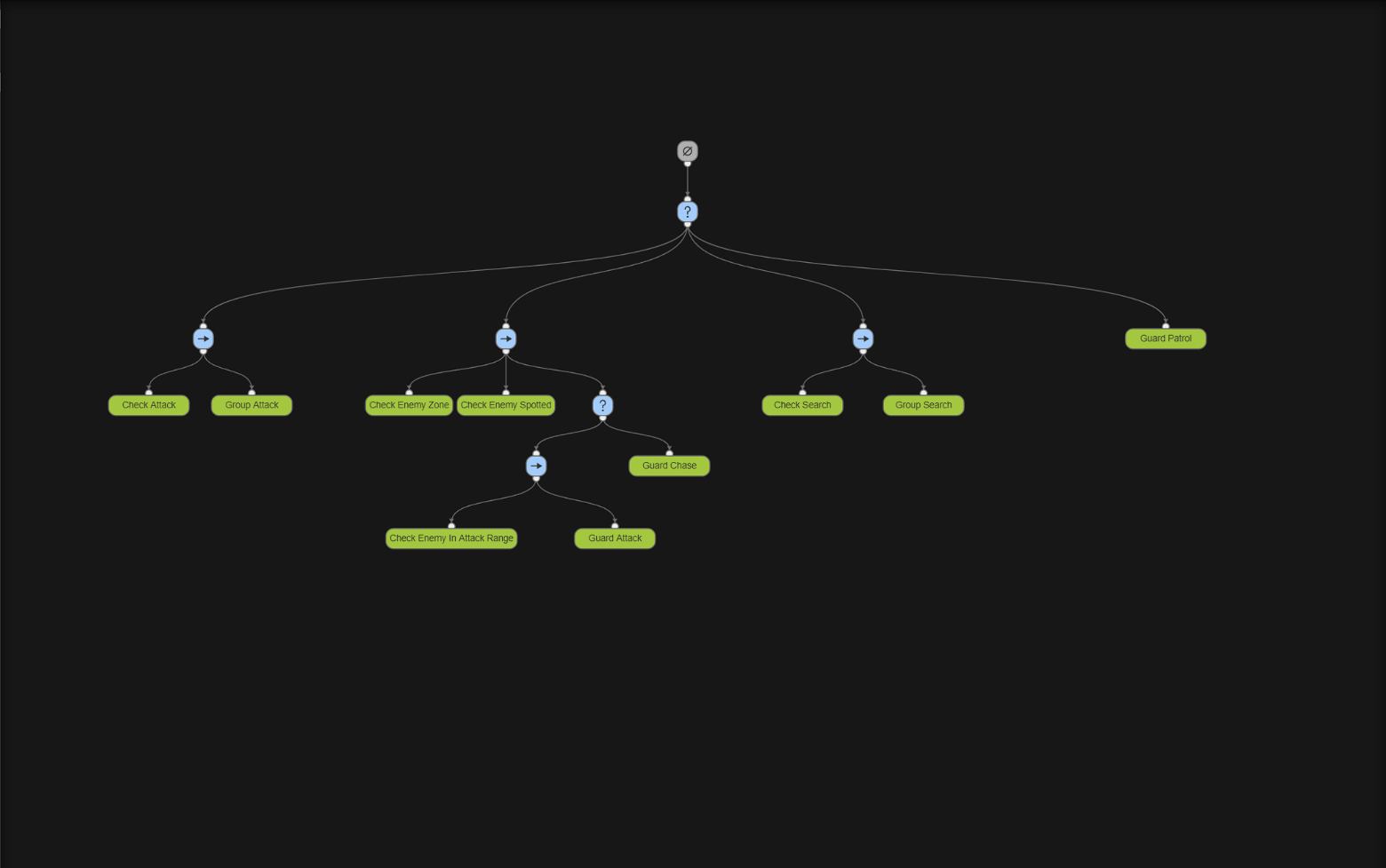


Diagram showing guards on patrol paths indicated by green lines and small blue dots.

Behaviour Tree Diagram showing the Guard Behaviour Tree layout and classes. (The topmost node is the root node. The ‘?’ nodes are selector nodes. The ‘->’ nodes are sequence nodes. The green nodes are leaf nodes.)



The Guard Behaviour Tree specifically is set up as follows.

The first node after the root node is a Selector node (So the first action to return SUCCESS will be the one committed).

From this node, we go to its leftmost child, a Sequence node(all children must return SUCCESS for an action to be committed). This Sequence node has 2 children which are both Leaf nodes. The left-most child checks to see if the Guard should currently be attacking the player with the other Guards by checking a Boolean value in its definition. If the Guard should be attacking, the node returns SUCCESS.

If the node does return SUCCESS the tree looks at the next child. The Group Attack class simply tells the Guard to go to the Player’s current location and then return a FAILURE. To break this down, first, we know the Guard should be attacking, so we tell it to go to the Player’s location. However, we do not want to return SUCCESS at this point as that will cause the tree to keep telling the Guard to go to the Player`s location. What would be better, is for the guard to only be told once to go to the Player`s location but continue down the tree to allow the Guard to spot the Player when the Player is nearby and perform an attack when in range.

Since that last Sequence node returns a FAILURE to the top Selector node, we move on to the next left-most child. This is also a Sequence node. The left-most child of this Sequence node is the Check Enemy Zone class. This class checks to see if the Player is within any vision zone of the Guard. This class returns SUCCESS regardless of the zone the Player is in and even returns SUCCESS if the Player is not currently in a zone.

The next left-most child is the Check Enemy Spotted class. This class checks which zone the Player is in and either increments the timer if they are in the Guard`s vision zone or decrements the timer if they are outside of the vision zone. If the Player is within the Guard’s vision zone for long enough, the class will set the Guard to go to the Player’s position. This class is also responsible for setting the Guard`s attack Boolean to true or false based on how far the Guard is from the Player. This class will only return SUCCESS if the Player has been in the Guard’s vision zone long enough to be detected. The class will return FAILURE if the Player is not in the Guard`s vision zone and the timer to spot the Player is at zero.

The next left-most node is a Selector node. This Selector node has two children. The first child is a Sequence node which also has two children. The left-most child of this Sequence node is the class called Check Enemy in Attack Range. This class return SUCCESS if the player has been spotted nearby the guard and FAILURE otherwise.

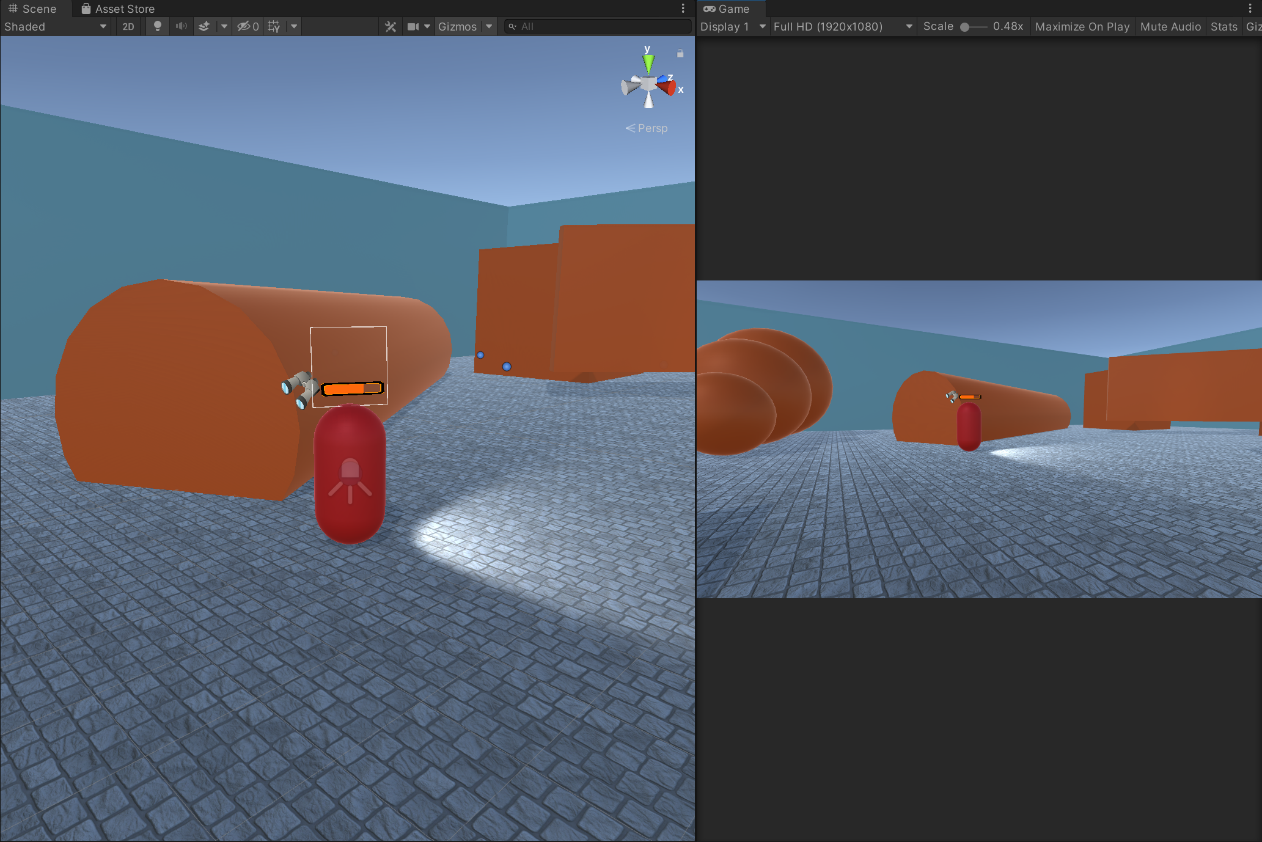
The next child of the same Sequence node is a class called Guard Attack. This class simply gets the Guard to perform an attack on the Player and returns SUCCESS.

If the Guard is not close enough to the Player to attack but the Guard has spotted the Player, the tree goes back to the Selector node after Check Enemy Spotted and traverses down the right child which is the Guard Chase class. This class sets the Guard to go to the Player’s location, imitating the Guard chasing the Player to get within attack range. This class also returns SUCCESS every time it is called.

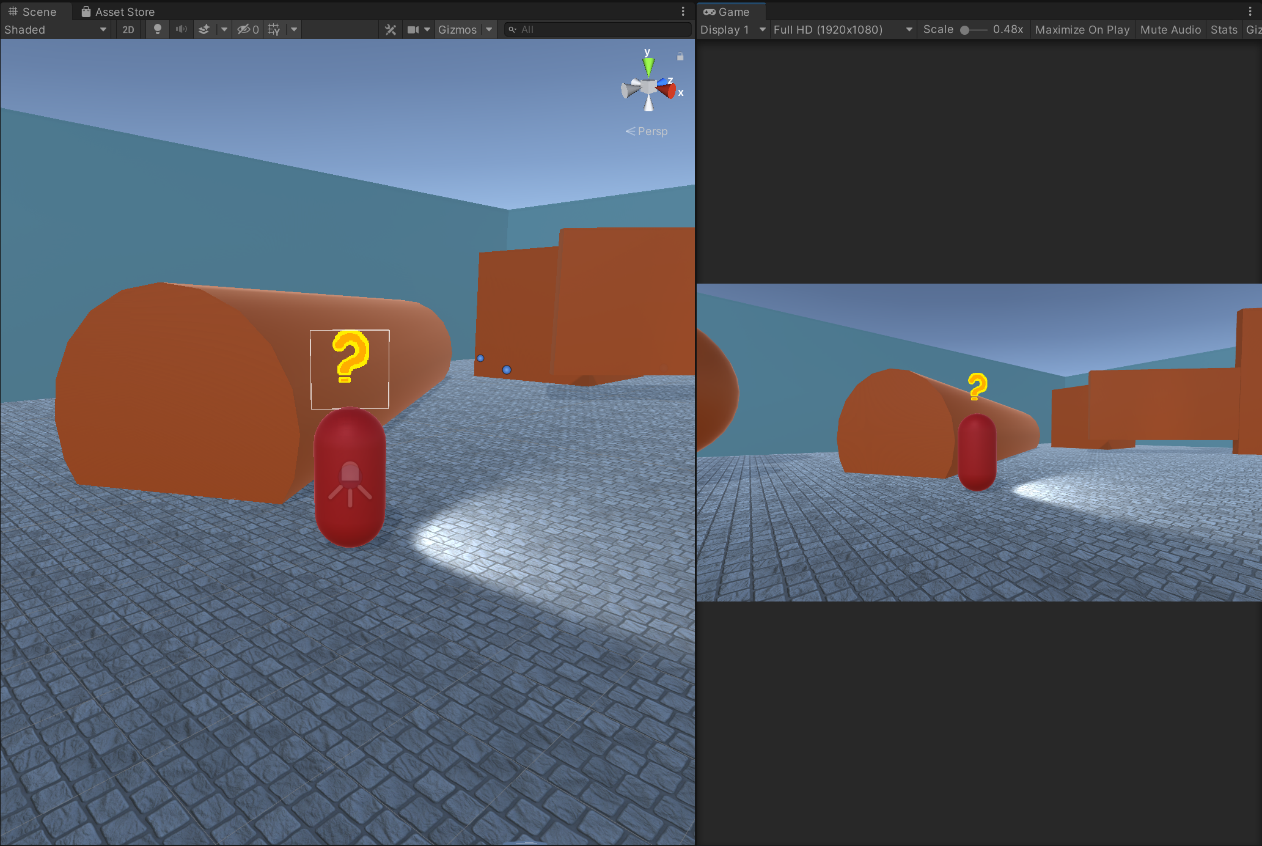
If none of these actions returns SUCCESS, the third child left-most of the top-most Selector node is then called. This child is another Sequence node. The left-most child of this Sequence node is a class called Check Search. This class queries the Guard`s Booleans to see if this particular Guard should be searching for the Player. This Boolean value is set to true or false in the overarching BTGuardGroup class. All that is important at this stage is to understand that if one of the other Guards starts searching for the player, this Guard will also go searching for the player. Making it seem as if the Guards have communicated and are working together.

If the Check Search class returns SUCCESS, then the next class to be checked is the Group Search class. This class performs no action but only returns SUCCESS. This is because every Guard’s searching pattern is being handled by an aforementioned overarching BTGuardGroup class. All that is required is that the Behaviour Tree does nothing else while this BTGuardGroup class is being executed to coordinate the Guards` search. Therefore, this class returns SUCCESS and performs no actions.

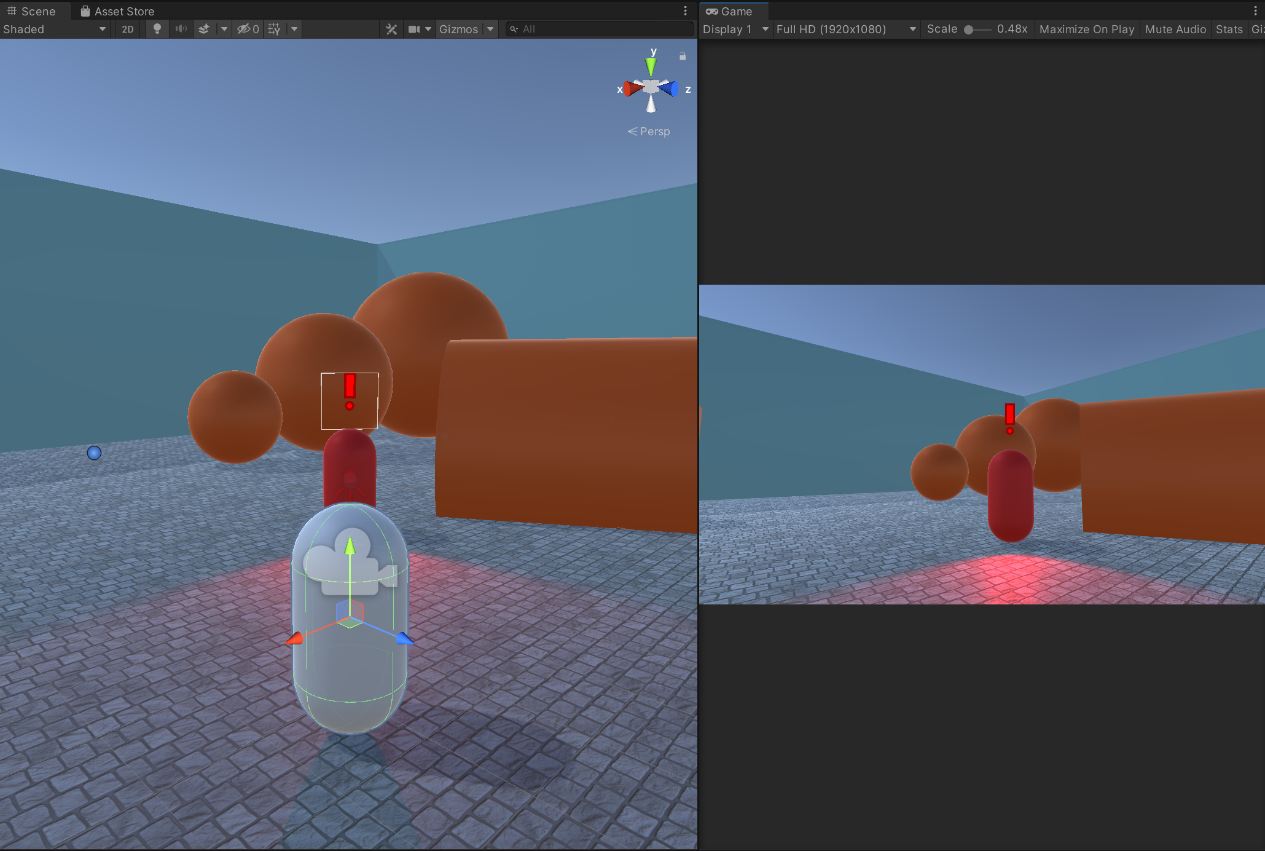
Lastly, if none of the first 3 children of the top-most Selector node returns a successful action, the Guard Patrol class is called. This class firstly ensures that the Guard should not be currently attacking. The reason for this is that in the Group Attack class at the beginning, the class returned failure, even if the Guard should be attacking the Player with all of the other Guards. This was done to allow for the rest of the tree to be run and for the Guards to still be able to perform individual actions such as actually attacking or chasing the Player. However, at this stage, the class ensures that the Guard isn’t coordinating an attack on the Player with the other Guards. And only sets the Guard to his patrolling route if this is the case. The patrolling class itself is relatively simple. It takes in a set of waypoints which are just empty game objects placed around the map. And sets the Guard to go to one of the waypoints and wait for a set amount of time. Then, after waiting, the Guard proceeds to the next waypoint and so on, continuously looping while the class is run. This class always returns RUNNING.



Screenshot showing Detection Bar ¾ full. (Red Capsule is the Enemy. The left view is the Scene view. The right view is the Game view/Player’s view)



Screenshot showing Enemy ‘Searching’ state sprite. (Red Capsule is the Enemy. The left view is the Scene view. The right view is the Game view/Player’s view)



Screenshot showing Enemy ‘Alerted’ state sprite. (Red Capsule is the Enemy. The left view is the Scene view. The right view is the Game view/Player’s view)

Appendix D:

USE CASE REQUIREMENTS

|  |  |
| --- | --- |
| **Use Case:** Enemy Detection | **ID:** 1C |
| **Description:** An enemy will be able to detect the player slowly over time | |
| **Primary Actors:** Enemy | **Secondary Actors:** |
| **Preconditions:**   1. Player is in an undetected state | |
| **Main Flow:**   1. The use case will begin when the player enters the enemy`s field of view 2. A timer will start to check how long the player is in the enemy`s field of view 3. If the timer reaches a pre-determined endpoint, the player will be spotted 4. The timer should have a shorter pre-determined endpoint if the player is closer and more central in the enemy`s line of sight | |
| **Postconditions:**   1. The player is spotted by the enemy | |
| **Alternative Flows:** The player escapes the enemy`s field of view before the timer ends | |
| **Preconditions:**   1. The player is in the enemy`s field of view AND the timer has not ended | |
| **Alternative Flow:**   1. The player escapes the enemy`s field of view 2. The timer starts counting back down until 0 3. The player is not spotted and the timer reaches 0 | |
| **Postconditions:**   1. The enemy does not spot the player | |

|  |  |
| --- | --- |
| **Use Case:** Enemy Pathfinding | **ID:** 1B |
| **Description:**  The enemies will use a pathfinding algorithm to get to the player’s last known location. The enemies will then path find from that point outwards to try to locate the player. | |
| **Primary Actors:** Enemy | **Secondary Actors:** |
| **Preconditions:**   1. The player has been spotted by an enemy AND has since escaped | |
| **Main Flow:**   1. All of the enemies nearby, convene on the player`s last known location 2. The enemies use a pathfinding algorithm to traverse different parts and corners of the map | |
| **Postconditions:**   1. The enemies end their search after a short search and return to their pre-determined patrol paths | |
| **Alternative Flows:** N/A | |

|  |  |
| --- | --- |
| **Use Case:**  Enemy Behaviour Tree | **ID:** 1A |
| **Description:**  The enemy’s behaviour will be dictated by a behaviour tree containing 4 states. The 4 states include Patrolling, Chasing, Attacking, Searching | |
| **Primary Actors:** Enemy | **Secondary Actors:** |
| **Preconditions:**   1. There are instances of guards using the behaviour tree within the game | |
| **Main Flow:**   1. If the player has not been spotted, the guard will patrol a pre-determined route 2. When a guard spots the player, all of the guards will chase the player 3. When the guards are within range, they will attack the player 4. If the player escapes, the guards will search the area 5. If the player is found again, they will chase and attack 6. If the player is not found again, they will go back to their patrol paths | |
| **Postconditions:**   1. The states will reset back to what they were before the player was spotted | |

|  |  |
| --- | --- |
| **Use Case:** UI Depicting Enemy States | **ID:** 2A |
| **Description:**  The enemies will have a small UI element/sprite above their heads depicting their current state. The 4 states the enemy can be in include, Patrolling, Chasing, Attacking and Searching. These will be split into 3 groups. One for Patrolling which will have no UI element. One for Searching will have a UI element/ Finally, one for Chasing and Attacking, which will have the same UI element. | |
| **Primary Actors:** Enemy | **Secondary Actors:** |
| **Preconditions:**   1. The game will be running | |
| **Main Flow:**   1. The guards will have no UI element above their heads when patrolling 2. The guards will have a small exclamation mark icon above their heads when the player has been detected and the guards are either chasing or attacking 3. The guards will have a small question mark icon above their heads when the player has been lost and they are searching | |
| **Postconditions:**   1. The guards will return to having no icons above their heads if the player has not been found and they return to patrolling | |
| **Alternative Flows:** N/A | |

|  |  |
| --- | --- |
| **Use Case:** UI Depicting Enemy Detection Level | **ID:** 2B |
| **Description:**  When the player is visible to the enemy, a small UI element will appear above the head of the enemy. This bar will fill up slowly based on close the enemy is to detect the player. The closer the enemy is to detecting the player the further the bar will fill up. The quicker the enemy is detecting the player, the faster the bar will fill up. | |
| **Primary Actors:** Enemy | **Secondary Actors:** |
| **Preconditions:**   1. The game will be running and the UI element for detection will not be above the enemy`s head. | |
| **Main Flow:**   1. The player will start in an undetected state. 2. The player will then enter the enemy’s vision zone. The detection bar should take the same amount of time to fill up as it does for the enemy to detect the player in that zone. E.g. it should take 1 second for zone 1, 5 seconds for zone 5, etc. 3. Once detected, the detection bar should disappear and reveal the alerted UI symbol from objective 2A. | |
| **Postconditions:**   1. The detection bar will only reappear when the guard is not in an alerted state and the guard is in the process of spotting the player. | |
| **Alternative Flows:**  The player goes into hiding while the enemy`s detection bar is half full. | |
| **Preconditions:**   1. The player is in an undetected state | |
| **Alternative Flow:**   1. The player will go into the enemy`s vision cone. 2. The enemy will begin to detect the player and the detection bar should start to fill up 3. Before the enemy fully detects the player, the player will leave the line of sight. 4. The detection bar should start going back down to empty if the player is not visible to the enemy. | |
| **Postconditions:**   1. The detection bar return to empty and subsequently disappears. | |

|  |  |
| --- | --- |
| **Use Case:** Smoke Bomb Mechanic | **ID:** 3Ai |
| **Description:**  The player will be able to use a smoke bomb in order to evade the enemies. The smoke bomb will temporarily blind any enemies within the smoke bomb radius and all enemies will lose sight of the player while the player is within the smoke bomb radius. | |
| **Primary Actors:** Player | **Secondary Actors:** Enemies |
| **Preconditions:**   1. The player will have been spotted by the enemies and attempting to escape | |
| **Main Flow:**   1. The player will press a button on the keyboard to drop a smoke bomb at their feet 2. The smoke bomb will explode soon after it hits the ground 3. The smoke bomb will have a radius much larger than the player 4. The smoke will obstruct the vision of any enemy 5. Any enemy within the smoke will lose sight of the player and remain stationary until the smoke clears | |
| **Postconditions:**   1. After the smoke clears the enemies will go into a search state if they cannot immediately see the player | |
| **Alternative Flows:** The smoke bomb explodes while the player has not been spotted by the enemies | |
| **Preconditions:**   1. The player will be hidden from the guards | |
| **Main Flow:**   1. The player will drop a smoke bomb 2. If the guards see the smoke bomb they will investigate it 3. After the smoke clears and the guards have stopped investigating the smoke bomb, they will enter a search pattern | |
| **Postconditions:**   1. The guards will search for the player as normal | |

|  |  |
| --- | --- |
| **Use Case:** Hiding Mechanic | **ID:** 3Aii |
| **Description:**  The player will be able to escape the enemy’s line of sight by entering an object such as a bush or tall grass, etc. | |
| **Primary Actors:** Player | **Secondary Actors:** Enemy |
| **Preconditions:**   1. The player will be visible to the enemy | |
| **Main Flow:**   1. The player will then move into the bush/tall grass 2. The enemy will lose sight of the player and move to the bush/tall grass to investigate | |
| **Postconditions:**   1. The enemy will search for the player as normal if the player cannot be found inside the bush | |
| **Alternative Flows:** N/A | |

|  |  |
| --- | --- |
| **Use Case:**  Limiting Use of Smoke Bomb Mechanic | **ID:** 3Bi |
| **Description:**  The player will only be able to use a set number of smoke bombs before they run out and the player should not be able to use them repeatedly in quick succession. | |
| **Primary Actors:** Player | **Secondary Actors:** |
| **Preconditions:**   1. The game will begin as normal | |
| **Main Flow:**   1. The player will drop a smoke bomb and have to wait a specified cooldown period before dropping another 2. Once the cooldown timer has ended, the player can drop another smoke bomb 3. Once the player has dropped all of their specified smoke bombs, they cannot drop any more | |
| **Postconditions:**   1. The player has no remaining smoke bombs and cannot drop any more | |
| **Alternative Flows:** N/A | |

|  |  |
| --- | --- |
| **Use Case:** Limiting Use of Hiding Mechanic | **ID:** 3Bii |
| **Description:**  The player should not be able to rely solely on the game's hiding mechanic. | |
| **Primary Actors:** Player | **Secondary Actors:** |
| **Preconditions:**   1. The game will begin as normal | |
| **Main Flow:**   1. The player will have many hiding options when on the outskirts of the level, allowing the player to safely perform reconnaissance on the level. 2. As the player moves closer to main part of the level, the number of hiding options will drastically reduce and be significantly smaller than those on the outskirts. | |
| **Postconditions:**   1. The player continues the game as normal | |
| **Alternative Flows:** N/A | |

Appendix E:

USE CASE TESTING

|  |  |
| --- | --- |
| **Use Case:** Enemy Detection | **ID:** 1C |
| **Test Number:** 1 | |
| **Objective:**  To test whether a player will be detected by the enemy and be detected at different speeds based on where they are in the enemy’s field of view | |
| **Set up:**  The player will take turns standing in the 5 different vision zones the enemy has. The player will start outside of the enemy’s field of view AND ensure the timer has not started. The player will then move to a vision zone and test how long it takes for the player to be spotted. The enemy’s spotlight will change to a different colour based on which zone the player has been spotted in, making it easy to visualise this test. | |
| **Expected Results:**  The player should be spotted after spending 1 second in zone 1 AND the enemy`s spotlight should go red.  The player should be spotted after spending 1.5 seconds in zone 2 AND the enemy`s spotlight should go magenta.  The player should be spotted after spending 2 seconds in zone 3 AND the enemy`s spotlight should go yellow.  The player should be spotted after spending 3 seconds in zone 4 AND the enemy`s spotlight should go green.  The player should be spotted after spending 5 seconds in zone 5 AND the enemy`s spotlight should go blue. | |
| **Test:**  The player will enter zone 1, check how long it takes for the enemy`s spotlight to go red and then leave the enemy`s field of view and wait for the timer to reset.  The player will enter zone 2, check how long it takes for the enemy`s spotlight to go magenta and then leave the enemy`s field of view and wait for the timer to reset.  The player will enter zone 3, check how long it takes for the enemy`s spotlight to go yellow and then leave the enemy`s field of view and wait for the timer to reset.  The player will enter zone 4, check how long it takes for the enemy`s spotlight to go green and then leave the enemy`s field of view and wait for the timer to reset.  The player will enter zone 5, check how long it takes for the enemy`s spotlight to go blue and then leave the enemy`s field of view and wait for the timer to reset. | |
| **Test Record:** Expected results observed | |
| **Date:** 23rd March 2023 | **Tester:** Tayyab Hussain |
| **Result:** Passed | |

|  |  |
| --- | --- |
| **Use Case:** Enemy Pathfinding | **ID:** 1B |
| **Test Number:** 2 | |
| **Objective:**  To test whether the enemies are successful in independent pathfinding in order to search for the player. | |
| **Set up:**  The enemies should be going along their patrol paths as normal, to begin with. After they spot the player, they should path find to the player’s location. Once the player has escaped and the enemies can no longer see the player, they should path find to search for the player. | |
| **Expected Results:**  The enemies should all congregate at the player’s last known location. They should then each spend 15 seconds searching a pre-determined position on the map using the pathfinding algorithm to traverse to that location. After 15 seconds they should then path find to another location on the map and search there for 15 seconds. After 30 total seconds of searching, they should path find back to their patrol paths and continue patrolling. | |
| **Test:**  The player will start in an undetected state and check that the enemies are following their patrol paths.  The player will then enter an enemy’s vision zone and be spotted.  The player will then run and hide and be completely outside of any enemy vision zone until the enemies have completed both searches and returned to their patrol paths. | |
| **Test Record:** Expected Results Observed | |
| **Date:** 22/04/23 | **Tester:** Tayyab Hussain |
| **Result:** Passed | |

|  |  |
| --- | --- |
| **Use Case:** Enemy Behaviour Tree | **ID:** 1A |
| **Test Number:** 3 | |
| **Objective:**  To test whether the behaviour tree can successfully implement the 4 relevant actions defined in the requirements specification and do so at the correct time based on the current game state. | |
| **Set up:**  The player will begin the game outside of the map to test the enemy patrol paths. The player will then be placed into the map and play in a way that will result in the enemy AI having to use all of the 4 different actions within the game and use them at the correct time. | |
| **Expected Results:**  The enemies should patrol when the player has not yet been spotted.  If a guard spots the player, all the other guards should be alerted and should pathfind to the player’s location. If the guards are close enough to the player they should attack. If they have spotted him but are too far away, they will chase him. If they all lose sight of him after having previously seen him. They will enter a search pattern. They will search 2 randomly assigned locations on the map until they have been searching for 40 seconds. Then they should return to their patrol paths. If they find the player whilst searching, they should return to either chase or attack the player based on the distance between them and the player. | |
| **Test:**  The player will start in an undetected state and ensure that the enemies all follow their patrol paths.  The player will then appear in the vision cone of one enemy and be subsequently spotted by the enemy.  The player will wait there until all of the guards have traversed to the player’s location.  The player will then slowly walk around the map. The tester will ensure that all of the guards attack the player when they are close and chase the player when they are far away.  The player will then go into hiding.  The tester will ensure that all guards enter their search pattern.  During the searching, the player will then attempt o be spotted again by the guards to ensure they switch states from searching to either attacking or chasing.  The player will then go back into hiding.  The tester will ensure once again that the guards enter their search pattern.  The tester will then observe if the guards will return to their patrol paths after 40 seconds. | |
| **Test Record:** Expected Results observed | |
| **Date:** 05/05/23 | **Tester:** Tayyab Hussain |
| **Result:** Passed | |

|  |  |
| --- | --- |
| **Use Case:** UI Depicting Enemy States | **ID:** 2A |
| **Test Number:** 4 | |
| **Objective:**  To test whether the UI elements depicting the enemy`s state are working correctly based on which state the guard is currently in. | |
| **Set up:**  The player will play the game as normal ensuring to go through all 4 states for each enemy and checking if the UI is correctly depicting each guard’s state. | |
| **Expected Results:**  The enemies should not have a sprite above their heads when patrolling.  The enemies should have an exclamation mark sprite over their heads when the player has been spotted.  The enemies should have a question mark sprite above their heads when in a search pattern.  The enemies should return to having no UI element above their heads when they return to a patrolling state after the search is complete. | |
| **Test:**  The player will start the game outside of the map. The tester will ensure the guards have no UI elements above their heads while patrolling.  The player will then be put inside the map and try to be detected by a guard.  The tester should only see an exclamation mark over the guard’s head when the player has been completely spotted by the guard.  The player will then go into hiding and the guards should enter a search pattern.  During this search pattern, the tester should check that the guards no longer display an exclamation mark sprite over their heads and now display a question mark sprite.  The player will attempt to be spotted again by a guard while the search is ongoing to check if the UI elements can change back from a question mark sprite to an exclamation mark sprite.  The player will then leave the map entirely and the tester will observe if the guards all change their UI elements to a question mark sprite while in a search pattern.  Once the search pattern has concluded, the tester will observe if the guards have returned to patrolling and no longer have any UI element above their heads. | |
| **Test Record:**  Expected Results observed | |
| **Date:** 08/05/23 | **Tester:** Tayyab Hussain |
| **Result:** Passed | |

|  |  |
| --- | --- |
| **Use Case:** UI Depicting Enemy Detection Level | **ID:** 2B |
| **Test Number:** 5 | |
| **Objective:**  To test whether the UI correctly depicts how close the enemy is to detecting the player. | |
| **Set up:**  The test version of the game will have only one enemy on the map and the enemy will not be able to move or look around. This will make it easier to check the detection level. | |
| **Expected Results:**  The enemy`s detection bar should not appear above their head until the player walks into the enemy`s vision zone.  The enemy will then have a detection bar above their head and it should slowly start filling up at the same rate the enemy would detect the player.  Once the enemy`s detection bar is full and the enemy detects the player, the detection bar should disappear and the Alerted UI element should appear instead. | |
| **Test:**  The player will start outside of the enemy`s vision cone.  The player will then walk into the vision cone and the tester should ensure the detection bar appears above the enemy and starts to fill up.  The player will stay in the zone until the bar is half full and then step out of the zone. The tester should ensure that the bar starts to slowly decrease and once empty the bar should disappear.  The player will then re-enter the enemy`s vision zone and stay in the zone until the bar is completely full. The tester should observe that the bar disappears and the Alerted UI element appears above the enemy`s head. | |
| **Test Record:**  The test was largely successful. However, there was one small pitfall. When the player leaves the enemy’s vision zone, the bar will jump down initially before then slowly decreasing. This is due to the way the bar`s detection amount is coded. This is discussed further in the Results section. | |
| **Date:** 11/05/23 | **Tester:** Tayyab Hussain |
| **Result:** Passed, but a slight adjustment is needed to make the feature more comprehensive | |

|  |  |
| --- | --- |
| **Use Case:** Smoke Bomb Mechanic | **ID:** 3Ai |
| **Test Number:** 6 | |
| **Objective:**  To test whether the smoke bomb deploys correctly, obstructs the view of the Enemies, stops the enemies from moving when inside the smoke and is correctly detected by an enemy without the presence of the player. | |
| **Set up:**  To test the smoke bomb a test version of the game will be set up with only the player and a test enemy who cannot move. This is so that the tester can verify whether the enemy can see the player through the smoke or not and can also verify the basics such as the deployment of the smoke and the visuals.  Then the normal game will be played to check the enemies` reaction to the smoke. | |
| **Expected Results:**  When the player hits the ‘G’ key on the keyboard, the smoke bomb canister will spawn in front of them and drop to the ground.  After a 1-second delay, the smoke should be instantiated and the canister should be culled.  The enemies will not be able to see the player if they are obstructed by the smoke.  The enemies will not move if they are caught in the smoke.  The enemies will search for the player if they cannot find them after the smoke clears  If a smoke bomb appears but the player has not been spotted by the enemies, the enemies should move towards the smoke but not go in the smoke and search for the player after the smoke dissipates. | |
| **Test:**  The player will start in the test version and be spotted by the test enemy.  The player will then press ‘G’ and drop the smoke bomb.  The tester will ensure the canister will behave as a physics object and drop to the ground and have a 1-second delay before being culled from the level.  The tester will also observe the smoke being instantiated at the location of the now-culled canister.  As the smoke billows, the tester will observe the enemy and ensure that the enemy can no longer see the player when the player is obstructed by the smoke.  Once the smoke clears, the tester will ensure that the enemy can now see and spot the player.  Then the normal version of the game will be played.  The player will be spotted by all of the enemies and drop the smoke.  The tester will ensure that the enemies do not move while in the smoke.  The player will leave the map and the tester will ensure that the enemies start searching for the player and complete a full search pattern before returning to their patrol paths.  The player will then drop smoke in front of one of the enemies on the level without being spotted themselves.  The tester will ensure that the enemy spots the smoke even without seeing the player and moves all of the enemies to the location of the smoke.  The tester will ensure that when the smoke dissipates that the enemies enter a searching pattern again. | |
| **Test Record:**  The tests related to the smoke itself passed. This includes the smoke being released from the canister after a delay and the canister being culled. The smoke also blocks the enemy’s vision and the enemies cannot spot the player through the smoke which works as intended.  However, the remaining parts of the test all failed. This was predicted of course as they were not implemented successfully into the build. The enemies did not spot the smoke, they only saw it as an obstacle in their way and therefore they did not stop when inside the smoke or move to the smoke when they spotted it without seeing the player. This will be explored further in the Results section. | |
| **Date:** 20/05/23 | **Tester:** Tayyab Hussain |
| **Result:** Failed | |

|  |  |
| --- | --- |
| **Use Case:** Hiding Mechanic | **ID:** 3Aii |
| **Test Number:** 7 | |
| **Objective:**  To test whether the player can be spotted when by the enemy when the p[layer is inside a bush | |
| **Set up:**  A test version of the game will be set up. This will include, the player, the bush and a test enemy. | |
| **Expected Results:**  When the player stands next to but not in the bush, the enemy should be able to spot the player and this should be reflected by the enemy`s UI symbols.  When the player enters the bush, the enemy should lose sight of the player.  The enemy should then move to where the player was last spotted.  If the player cannot be found the enemy should enter a search pattern as usual. | |
| **Test:**  The player will begin next to the bush and let the enemy spot them.  Once the enemy has spotted the player, they will move into the bush.  The tester will ensure that the UI elements above the enemy`s head indicate that the enemy has lost sight of the player.  The tester should ensure that the enemy moves towards the player’s last known location but cannot spot the player through the bush.  The player will remain in the bush and when the enemy is also in the bush the tester will ensure that the enemy has spotted the player. | |
| **Test Record:** Expected Results Observed | |
| **Date:** 27/05/23 | **Tester:** Tayyab Hussain |
| **Result:** Passed | |

|  |  |
| --- | --- |
| **Use Case:**  Limiting Use of Smoke Bomb Mechanic | **ID:** 3Bi |
| **Test Number:** 8 | |
| **Objective:**  To test whether the smoke bombs have a finite use and ensure they cannot be used repeatedly in quick succession. | |
| **Set up:**  A test version of the game will be set up with just the player and no enemies. | |
| **Expected Results:**  When the player presses the button for the smoke bomb, a smoke bomb will deploy.  There will then be a delay before the player can drop another smoke bomb.  The player also should only be able to drop a specified number of smoke bombs | |
| **Test:**  The player will drop the first smoke bomb.  After the first smoke bomb is dropped, the player will continue to press the button to drop another smoke bomb.  The tester should ensure no more smoke bombs are dropped until the specified cooldown timer for the smoke bomb has ended.  Once the timer has ended, the player will be able to drop another smoke bomb.  They will continue to press the smoke bomb button continuously.  The tester will ensure that when the specified number of smoke bombs have been dropped, no more smoke bombs are dropped by the player, despite them still pressing the smoke bomb button. | |
| **Test Record:** Expected Results Observed | |
| **Date:** 22/05/23 | **Tester:** Tayyab Hussain |
| **Result:** Passed | |

|  |  |
| --- | --- |
| **Use Case:**  Limiting Use of Hiding Mechanic | **ID:** 3Bii |
| **Test Number:** 9 | |
| **Objective:**  To test whether the player is able to solely rely on the hiding mechanic in the game. | |
| **Set up:**  The normal game will begin | |
| **Expected Results:**  The player will be able to hide in several locations on the outskirts of the map.  Once the player gets closer to the centre of the map, the number of hiding locations will reduce significantly. | |
| **Test:**  The player will begin by going to each of the hiding areas on the outskirts of the map.  The player will then go closer to the main part of the level and utilise all of the hiding areas there.  The tester will ensure that there are significantly fewer places for the player to hide in the main area of the level as opposed to the outskirts. | |
| **Test Record:** Expected Results Observed | |
| **Date:** 22/05/23 | **Tester:** Tayyab Hussain |
| **Result:** Passed | |