**Bug report**

|  |  |  |
| --- | --- | --- |
| **Bugs** | **How they happened/discovered** | **How they were solved** |
| Blue ghosts eaten sprite would play when in a scatter/chase state | Blue would call the eaten states sprite when moving in certain directions | Recreating each of Blues’ movement animations clips, as the animation controller for one of the clips was assigned to the wrong controller, this meant blue was animating in a strange way. |
| PacMan’s Death animation would not play | PacMan’s death animation would play in the animation window but would not play in game. This was a result of  one of the sprites that were used to make the animation was deleted accidently, this was realised when PacMan was consumed by a ghost. | Creating a completely new death animation. |
| PacMan sprite would turn tiny when moving around the map | PacMan’s sprite would look significantly smaller than other game objects in the map. | Every time a sprite changed locations in the file structure it resets the inspector value of that object, this meant that the sprites that were used for the PacMan’s game object had reset from 16 pixels per unit to its default value of 100 pixels per unit (see figure 5). |
| PacMan would go invisible when reaching a wall | The sprite component in PacMans Game object had gotten reset when the Pacman class had changed, this would result in Pacman going invisible every time he had reached a wall.  (See figure 3) | This was done by attaching the correct sprite to PacMan’s not moving component |
| Ghosts would not leave the house | The pellet component was deactivated instead of the sprite renderer, which meant that the ghosts were unable to find the neighbouring waypoints | An object reference error was displayed, this was strange as all the scripts that were necessary for the ghosts to function were assigned. This helped locate the problem to the maze object, where going through the project hierarchy I was able to see what pellets were not activated. |
| Ghosts would not follow movement pattern | Would not discover waypoints above house | An object reference error was displayed, this was strange as all the scripts that were necessary for the ghosts to function were assigned. This helped locate the problem to the maze object, where going through the project hierarchy I was able to see the if the waypoints above the ghost house had the correct values for the ghosts to start moving. |
| PacMans can go into the house | PacMan was able to discover the waypoints just below the ghost house | This was not fixed in the final version of the game. |
| Portal right would make Pac man get stuck | Through the survey feedback | By creating a new pellet object and assigning it to the correct waypoints. The placements of pellets are also crucial in making this script work. |
| Inserting adjacent waypoints in the maze | This was a difficult mechanic to implement as an error in this section would cause problems further down in the creation of this game, this is the foundation of how objects move in this game.  In making sure that each adjacent waypoint was correctly linked to its counter waypoint and was placed at the correct position on the map took a long period of time. | Testing each Waypoint to see if it works as intended, this was done through the use of the PacMan Game Object, as this object was controlled through the use of keyboards input, which meant testing each waypoint was done by manoeuvring around the map |
| Timing Enumerators to be accurate to real time | The “Get” “Ready” texts that appear before the game were not being accurately displayed, one was disappearing before the other, this made it look unprofessional | By testing multiple different values in the StartCoroutine () method |
| Object reference error (common) | Since there were several scripts that were attached to each game object, it made changing any component in a game object difficult as there were so many pieces that had to be assembled for an object to work efficiently. | Locating which script is causing the error, then adjusting according to this. |
| PacMan not moving efficiently, Gameplay felt slow and choppy. | PacMan’s movement felt choppy at times, in some instances the game would feel smooth but at other times it would feel like there was an input delay, this problem was consistent throughout the project.  This could have been the result of the device that was being used for unity was rather slow, as there were multiple occasions where the laptop had crashed completely | This was solved temporarily by restarting my laptop, this was not a permanent solution. This is because it was not discovered why gameplay sometimes felt choppy. |
| Ghosts not turning blue when in scared state | Through the survey feedback | The eat script had to be attached to the ghost’s game objects in the inspector. |
| Ghosts would consume pellets | The Eat class was attached to each individual ghost which meant that they would also consume ghosts as well Pacman, this bug had occurred due to trying to solve the bug where ghosts would not turn blue when in a scared state as the method to start this was in the eat script. | The eat script just had to be attached to the ghost’s game objects in the inspector. just not activated (see figure 4). |
| Checking to see if Red A.I pathing algorithm works | The function to find PacMan’s was not rounded to the nearest integer, which meant that red’s algorithm was not accurate. | By implementing Unity’s debugger Debug.log () at each instance of the method, it would be possible to see what part of the A.I pathing algorithm is not working. By factoring that this method has a lot of calculations in it, it would be clear to see in the console which part of the method is not working as this debugger could be implanted at specific variable in Red’s A.I class. |
| Checking to see if Pink’s A.I pathing algorithm works | Pink ghosts’ A.I method was not able to find PacMan’s current positions | By implementing Unity’s debugger Debug. Log () at each instance of the method, it would be possible to see what part of the A.I pathing algorithm is not working. By factoring the fact that this method has a lot of calculations in it, it would be clear to see in the console which part of the method is not working as this debugger could be implanted at specific variable in Pink’s A.I class. |
| Checking to see if Blue’s A.I pathing algorithm works | Would not add the red ghosts vector position X to the formula. | By implementing Unity’s debugger Debug. Log () at each instance of the method, it would be possible to see what part of the A.I pathing algorithm is not working. By factoring that this method has a lot of calculations in it, it would be clear to see in the console which part of the method is not working as this debugger could be implanted at specific variable in Blue’s A.I class. |
| Checking to see if Orange’s A.I pathing algorithm works | Orange ghost would have the distance formula the wrong way around so it would only chase PacMan if it was within 8 tiles of its character. | By implementing Unity’s debugger Debug. Log (), at each instance of the method, it would be possible to see what part of the A.I pathing algorithm is not working. By factoring that this method has a lot of calculations in it, it would be clear to see in the console which part of the method is not working as this debugger could be implanted at specific variable in Orange’s A.I class. |
| Manipulating speed of ghosts | The previous move speed was not being called at correct stages, so when a ghost transitioned from one state to another the speed stayed consistent, which should not be the case as a ghost in a scared state should move significantly slower. | Placing Debug. Log () at the instance where a ghost movement speed is instructed to change and trying to see which aspect of the code is not working. |

**Appendix**

Background pattern

Description automatically generated

**Figure 1:** this display the unity console log method, this was placed in both Pinky’s and Reds Target tile function methods to see if they are accurately targeting the player, as you can see Pinky’s target is four tiles ahead of reds which means the A.I algorithm is working.

Text

Description automatically generated

**Figure 2:** This shows the use of the Debug. Log function at the orange Target algorithm, this would display a message every time the orange ghost would decide on when to target PacMan.



**Figure 3:** This was part of the PacMan’s inspector component, it helps describe a bug that was experienced during the creation of the game.

A screenshot of a computer

Description automatically generated with medium confidence

**Figure 4:** this displays the Blue ghost’s Inspector component; this is where all scripts that are assigned to a Game object for that object to use a specific function.

Graphical user interface, application

Description automatically generated

**Figure 5:** This helps display the bug that made PacMan significantly smaller.

A screenshot of a computer

Description automatically generated with medium confidence

**Figure 6:** Showing an example of the sprite “pellet (105)” Adjacent Waypoints