Bomberman Design Document

Group #20: Timothy Goon & Jadon Laforest

Initial Developments

Scenario 1:

Scenario 1, Variant 1:

To complete scenario 1 variant 1, we knew that all we had to do was find a path to the exit and follow it. To do this, we used the A* algorithm to find the best path. At first, we used simple cost and heuristic functions just to make sure everything worked accordingly. Our cost function returned infinity if it was a wall and 1 in all other cases. We knew that we *could* use bombs, but believed it was unnecessary for this variant, so we acted like the walls were unpassable. Additionally, we initially treated A* like a Djikstra's algorithm with no heuristic, but we eventually replaced it with a calculation of the Euclidean distance to the exit. With A* only, the character will always follow the path shown in **Figure 1** below.

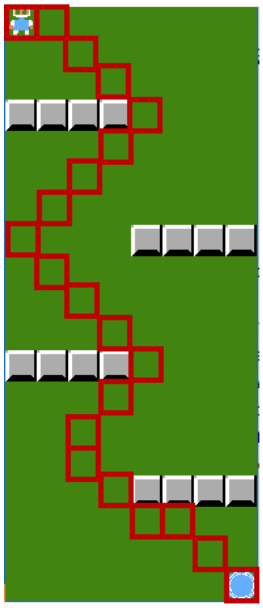

Figure 1: A* default path in Scenario 1, Variant 1

This version of the A* algorithm was then used throughout all our future character iterations in some way. We continued to play with the cost function in future characters and iterations to improve A*'s performance in the different situations.

Scenario 1, Variant 2:

To complete scenario 1 variant 2, the first approach we took was to use an expectimax search to avoid the stupid monster while making progress toward the exit. The expectimax

search was only active when the character got close to the monster. When the character is not close to the monster, it uses A* to progress.

The expectimax search had a heuristic where the score for death was -1000/(depth of the search), exiting was 1000/(depth of the search), and non-terminal worlds had the score of the distance to the exit cell. For the probabilities, all the possible moves for the monster was given an equal probability. The character using expectimax was getting about an ~80% win rate.

We later found out after completing scenario 1 variant 4 that a character using minimax to try and potentially kill the monster performed much better than the expectimax with a ~100% win rate. The details about the minimax character can be found in the section about **Scenario 1, Variant 4**.

Scenario 1, Variant 3:

For scenario 1 variant 3, our first approach was to take the default A* code and make edits to the cost function. We used the cost function to find the location of the monster and check if the move would cause the character to move into the aggression range of the monster. If it did, it would return infinite. Basically, we acted like the monster was a moving block of impassable terrain. While this worked alright, there were multiple times where the character got trapped in corners and died. We looked elsewhere for more consistent ways to deal with the self-preserving monster.

Like variant 2, we next tried with our expectimax character and similarly got about an ~80% winrate. We found that trying to kill the monster instead of trying to move past it led to much more consistent results. This was because it either led to the monster dying to a random explosion, or it opened up the map enough for the character to move past the monster. This finding led us to using the minimax character that we developed later instead. For more details, refer to the section about **Scenario 1, Variant 4**.

Scenario 1, Variant 4:

The first thing we tried to do to complete this variant was reuse the expectimax character created for scenario 1 variant 2. This approach performed badly because in this variant the monster's moves were deterministic when you are being chased. The character was only able to win about 50% of the time.

Our next approach replaced the expectimax with a minimax search. The heuristic function we used at first was the same as the expectimax heuristic descriptive in **Scenario 1, Variant 2**. At this point the character was better at running away from the monster but sometimes got cornered against walls or the edge of the world.

This is where we changed our A* function's cost function. We realized that although we didn't need to blow up walls much in Scenario 1 Variant 1-3, it might be beneficial to us here. We changed it so that it was able to blow up walls in its path and move around the monster. We set the cost for an area around the monster to be infinity and changed the cost of a wall to be (bomb timer + explosion time) which is 12. When a bomb was placed, we also had to force the

character to use minimax so it would avoid blowing itself up. After this change the character did do a better job at avoiding the monster to get to the exit but if it ever moved into detection range the monster would still sometimes kill the character by cornering it.

Our solution to this was to make the character try to kill the monster. This was done by changing the behavior of minimax. First the heuristic function for non-terminal worlds was changed so that instead of the distance to the exit having a higher score, the spaces that were further away from the monster had higher scores. Thus, the minimax causes the character to get as far away from the monster as possible. The second change we made was that if the monster was ever within a certain radius of the character, the character would immediately drop a bomb.

Putting this all together. The resulting behavior is that A* would try to path around the monster using bombs if necessary. Then, if the monster ever gets within a certain radius of the character it would drop a bomb and switch to minimax. The minimax would tell the character to run as far from the monster as possible and avoid death. If the bomb explodes and the monster is not dead the process repeats. Because the timer for the bomb is a long 10 time steps, immediately placing the bomb when the monster is close and hoping the monster gets caught in it is the best we can do in terms of hunting the monster. If the monster does end up dying or the character somehow makes it past the monster, A* would just take the character to the exit cell. After making these changes the win rate of the character was around ~80-90%.

Scenario 1, Variant 5:

The first thing we tried here was to reuse the minimax character from scenario 1 variant 4. The character from scenario 1 variant 4 ended up doing surprisingly well considering it was configured to only consider one monster at a time with a win rate of around 60%.

At this point of the project the variants were being completed non-sequentially, so **Scenario 2, Variant 5** was completed before this one and that character was used to complete this variant. From testing between the variants we found that in Scenario 2, the increase in the number of walls on the map allowed the minimax character to take more risks with how close it could get to the aggressive monster. This was because we observed that for the minimax character the main way the character escaped the aggressive monster was to move around the walls to get out of the chasing range of the monster. Therefore, for scenario 1 with less walls we had to tweak our constant values in the character so that it would try to stay further away from the monster.

One of the pitfalls of our approach to use bombs to kill the monster was that it sometimes ended up destroying too many walls, losing its tools for escaping the aggressive monster. We also had to change some constants in regards to this. In the end we found that the character had around an ~80-90% win rate.

Scenario 2:

Scenario 2, Variant 1:

At first, we thought we could just use A* with a bit of additional code to drop bombs on walls that were in the path. We quickly realized that this didn't work because there was no easy way to move out of the path of the explosion. Also, when surrounded by explosions the character recognized that there was no valid way to get to the exit, so it just walked into the explosion and died. An example of this situation is shown below in **Figure 2**.

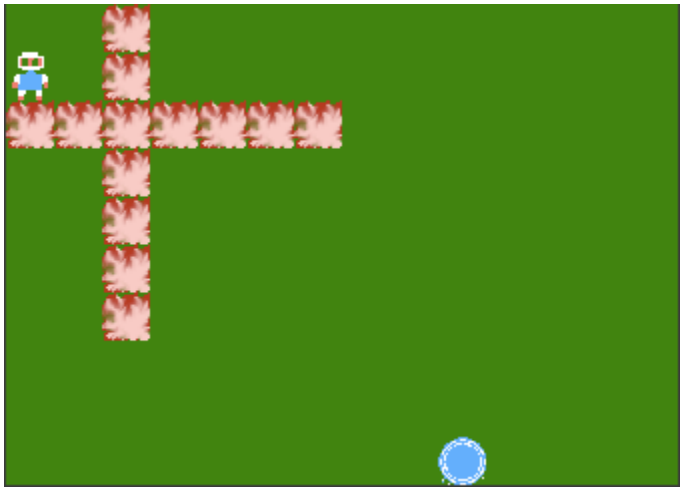

Figure 2: Bomberman trapped in corner, moments away from death!

After realizing this after some testing, we ended up trying to use our previously created minimax character. This way, we ensured that the character wouldn't die to the bomb or explosions while going toward the exit. This character worked just as well without monsters as it did with, and easily achieved a 100% win rate.

Scenario 2, Variant 2:

We attempted to complete this variant after we had already found success in the minimax character created for **Scenario 1, Variant 4**. Our first thoughts were to just try it directly on this variant. To our surprise, it worked ~100% of the time, and didn't show signs of needing much more attention. We were confident that minimax would never let the character die to the stupid monster. At times, using this character takes a long time for it to kill the monster or get to the exit, but we realized there were no real constraints on the score. Since there was no "high score" challenge and our only goal was to live and get to the exit, this character design worked great.

Scenario 2, Variant 3:

Based on our success with the previous variant using our minimax character, we decided to try it on Variant 3 too. Yet again, it worked ~90-100% of the time. Like the last variant, there were many tests that took longer than others because of the many failed bomb attempts to kill the monster. But fortunately, in almost all of those cases, our character was still able to make it to the

exit safely. Again, since we didn't need to pay much attention to our score, our minimax character design worked great, and we were happy with its performance.

Scenario 2, Variant 4:

At first we used the character from **Scenario 1, Variant 4** to complete this variant. We found through testing that in this variant where there are more walls to hide behind, the character using minimax would do better if it was allowed to take more risks and move closer to the monster. This variant performed at about an ~80% win rate. It was not as consistent as **Scenario 2, Variant 2 or 3**, most likely due to the aggressive monster's unique behavior. At times, it seemed like it would corner our character easier since there were more places to get stuck, but other times, it was easy to create distance via random holes that broke the monster's line of sight. Overall, we were okay with its performance using the minimax character.

Scenario 2, Variant 5:

This variant was completed before **Scenario 1, Variant 5**. Here we took the character from **Scenario 1/2, Variant 4** and generalized it to work with multiple monsters. We made 2 main changes to the behavior of our original minimax character.

The first was that the min nodes of the minimax would now consider the movements of both monsters on the board in the search. The second change was made to the minimax non-terminal heuristic function. Instead of the score being the distance from the first monster it found we changed it to be the distance from the closest monster. Although not obvious, this was a simple way for the character to avoid both monsters on the map when it is appropriate to do so and only consider one monster when the other is too far to matter. You can see this illustrated in **Figures 3 and Figure 4** respectively.

Intentionally Left Blank

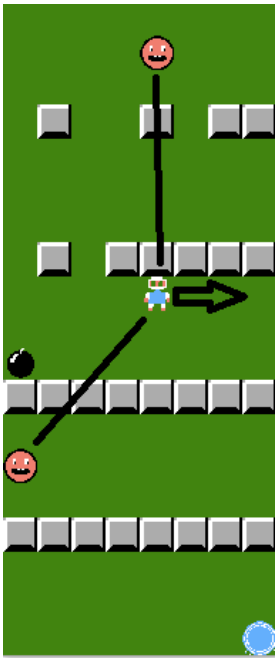

Figure 3: The character is about equidistant from either monster and moves to a location furthest away from both. Minimax is active because there is a bomb placed.

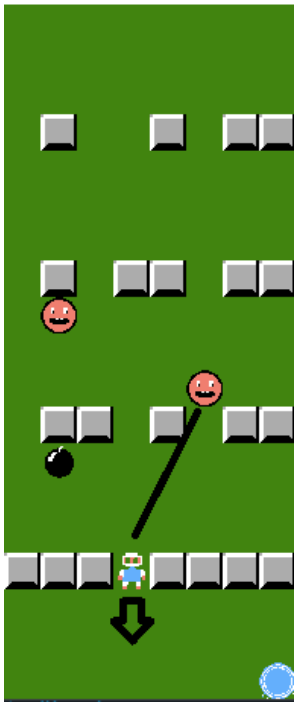

Figure 4: Both monsters are on one side of the character. The character only needs to consider running away from the closest one. A bomb is placed so minimax is active.

Testing Methods & Further Improvements

For the most part, we manually tested our characters on the given variants and scenarios. At the beginning, we used set seeds to see our consistent improvement until we won. Then, we exclusively would run random seeds to see more diverse possibilities. Watching each of the steps of the character helped us determine if things were working the way we intended. On runs where the character died, we looked back at the previous steps to see where something went wrong and if it was something we could have avoided.

Early on, we found through manual testing that using A* in the minimax/expectimax to get the shortest distance to the exit cell made the algorithm very slow. Therefore, we swapped it out with an expression to just calculate the euclidean distance between the character and the exit cell.

One of the major bugs that we found early on was in regard to our character that swapped from minimax and A*. We noticed that there were times where the character would have a clear path from their current location and the exit (with no threat from the monster), but instead of running toward the exit, it would move in place/a circle, allowing the monster to get close enough to corner it or attack. We were able to find the issue and resolve it by only running minimax if there were active bombs or explosions or a monster was 3 or less spaces away (outside of the most aggressive monster's attack range). Previously it was at a higher monster range and caused our character to play too cautiously.

Many of the constants we used throughout our characters' code were fine tuned after many (many) tests. We would change a value, see if it seemed to affect anything, and if it did, we would run ~10 tests on the new and old value to see if it resulted in significant improvement. If it did, we would replace the values and continue testing.

Final Characters

Although there were *many* iterations of our characters, we broke them down and condensed them to three main characters to be used for all variants.

Character 1:

This character includes our default A* code, and is only used by **Scenario 1, Variant 1**. Characters 2 and 3 also work with this variant, but we wanted to show a bit of the evolution of our character iterations.

Character 2:

This character includes our minimax character. This character is only designed to be used against variants with 1 monster. It is used by **Scenario 1, Variant 2-4** and **Scenario 2- Variant 1-4**. Character 3 did work well with a few of these variants, but we had slightly greater success rates

using Character 2 on them. We operated minimax at depth 5, as it took normal time and produced consistent results.

Character 3:

This character includes our minimax character with modifications. This character is designed to be used against variants with multiple monsters. We found after extensive testing that Scenario 1 and 2's Variant 5 each worked better with a few different constants. We decided to split them into two different characters to give each the best chance of success. **Character 3 One** works with **Scenario 1, Variant 5** and **Character 3 Two** works with **Scenario 2, Variant 5**.

Future Improvements

In **Scenario 2, Variant 5**, the presence of the stupid monster close to the player results in the player placing down many bombs that produce no result because there are walls in the way. We could add in a feature that detects if a bomb placed in order to kill a monster has any chance of killing that monster in the first place.

We could add a feature that uses the first few moves to detect the behavior of each monster and determine which one is which. With more testing, we could perfect our constants to give our character the best chance versus each type of monster instead of using the same general constants. Then, using the monster detection, we could use some kind of state machine to swap character behaviors, or even combine them in cases where multiple monsters are in close proximity.

We also could spend more time troubleshooting weird edge cases that led Variants 4 and 5 to unnecessary or abnormal deaths. Of course, with more time we would have loved to continue investigating and iron out all the small issues in our code and design, but we are quite happy with where we ended up.

Special Note to the Professor/TAs:
Thanks for a great term- we really enjoyed these projects, learned a lot, and had a ton of fun!
-Jadon & Tim :)