In the default map, the agent cannot avoid the radiation. As the agent gradually learns, it finds out that for every human it saves, it receives a positive reward per turn. This reward increases the score of the bot, and it learns that it is better to take the risk of going through radiation to get the human and save it rather than leave it. In the default map, the robot starts on the left hand side of the map. The robot moves right and goes straight into radiation. After playing a few hundred iterations, the bot figures out the pattern of the enemy bot and realizes it is better to keep the enemy alive and go through radiation to pick up the first human. Instead of losing points each turn by killing the enemy, it can go through radiation once and pick up the human faster which will increase its score. Further, to pick up the second human, there is no way for it to avoid the radiation. There is a wall and two blocks of radiation that blocks the agent bot from the picking up the second human. Again, after a few hundred iterations, the agent figures out to go past the radiation and pick up the human because the bot will receive a bigger reward for saving said human. If one wanted the bot to avoid the radiation, one of two things would need to happen to the map. The human on the right side of the map would need to be on the left hand side of the radiation so that the bot does not need to cross over the radiation. Additionally, the bot would be able to avoid much more radiation if there was a gap between the initial location of the bot and the second human. To be clear, this means there must be some sort of path for the bot to reach the second human without touching the radiation. This can be done by removing either of the two blocks of radiation the separate the human and bot or by removing the wall above the two blocks of radiation. Lastly, if the reward for saving the human is negative, the agent would also avoid the radiation. I set my reward for saving humans to -500. This actually forces the agent to not worry about the humans and the agent goes straight to the goal.

The smallest value for enemyDead that would make it so that the bot is willing to kill the enemy if they cross paths is -1.375. In order to determine this value, we need to do a little bit of math. If we ignore saving human rewards and we assume the bot is staying on the goal state because it has killed all enemies, then the reward it can get is 10 minus the value of the killing the enemy. Secondly, if the enemy was still alive, and we were on the goal state, then out of seven turns, we would have to move one turn in order to avoid the enemy bot. We are able to get 7 times 10 reward for seven turns when agent stays on goal and we have to lose a point when we are not on the goal. On average of eight turns, our reward turns out to be 8.625. We have to find a value for enemyDead now. We can solve the inequality system here: $10 - \text{enemyDead} > (70 - 1) / 8$. This means our value for enemyDead is 1.375. We have to make sure it is negative since if it were positive, the agent would begin to seek out the enemy instead of just if they cross paths. This is explained later on in the answer. The smallest value for enemyDead that would induce the bot to seek out the enemy and kill it is anything positive. If we were considering integers as rewards, then the smallest value would be 1. If we consider floating point numbers, then it can be the smallest floating-point number greater than zero. Once this value is greater than zero, the agent decides it is better to kill the enemy and it begins to seek it out to kill it as soon as it can. The reasoning behind this is that the bot will receive a better reward per turn for killing the enemy bot. If it can get a better reward per turn for killing the enemy bot, then it will increase its overall final score, which is good for the bot. Earlier in the answer, I mentioned that the smallest value for enemyDead to be so that the bot is willing to kill the enemy if they cross paths is -1.375. To solve the inequality, it's necessary to make enemyDead negative since if it were positive, then it would immediately seek out the enemy and kill it, ensuring higher rewards per turn.

When the enemy mode is 1, the enemy follows an influence map and the bot learns this pattern after a while. When the enemy mode is changed to 2, the enemy starts moving randomly and it is hard for the bot to tell where the enemy is going to go. I did a couple of tests to see how the bot would react to the enemy moving randomly. Firstly, I ran the simulation with 500 training episodes. I didn't see much of a difference besides the bot still trying to avoid the enemy in a similar fashion to when enemy mode was 1. I thought I might get different results with 100 training episodes. The results still turned out the same. Finally, I tested with 1000 training episodes and saw significant differences. Now the bot tried to dodge the enemy even when the enemy wasn't there. For example, if the bot was on the right side of the map (past the wall and two radiation blocks), sometimes, it would move up in one time slot and move down in the consecutive time slot or vice versa. Because the bot moved randomly, it would always keep itself on its toes in order to avoid the bot. Once the enemy bot actually came close to the agent, the agent would start moving to the left of the two blocks of radiation. It had figured out where the enemy bot most likely was and moved based on this assumption. If the enemy bot started moving farther to the right, the agent bot would start moving to the left to avoid the enemy bot. It would then make its way to the base so that it could gain a better score. If the enemy bot randomly moved into the spot where the agent bot was at, (at this point it is the goal spot), then the agent bot moved up one spot in one time slot and moved down in the consecutive time slot so it ended up the goal location. The agent bot had learned that the enemy was probably only going to be in its spot for one time slot which allowed it to gain less reward for that iteration, but still continue to gain an equal amount of reward for future iterations. The agent had taken into account the random actions of the enemy and moved so that it would not hurt the enemy and gain less reward. Finally, because of learning that the enemy bot was moving randomly, the max reward for the agent bot came out to be bigger than the normal max reward for the enemy bot following the influence map.