ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

MÜHENDİSLİK FAKÜLTESİ
FACULTY OF ENGINEERING
ENDÜSTRİ MÜHENDİSLİĞİ BÖLÜMÜ
DEPARTMENT OF INDUSTRIAL ENGINEERING

IE206 Project Report

2016-2017 Spring

The Walking Dead

Academic integrity is expected of all students of METU at all times, whether in the presence or absence of members of the faculty.

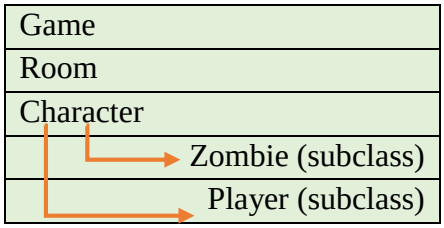
Understanding this, we declare that we shall not give, use, or receive unauthorized aid in this project.

Group ID#	*Student ID Number*	*Full Name*	*Signature*
	2095982	Mustafa Serkan IYIDEMIR	
	2167963	Mehmetcan CANIKLIOĞLU	

Introduction:

In this team Project as a team of two, we were asked to complete three main assignments. First, we completed the script files of the "a simple game: The Walking Dead". The name might be misleading because although the playing part of the game is simple and fun, the codes behind the game are quite complex. Fortunately, our dearest teaching assistants completed most of the complex parts and we only needed to follow their hints and complete the missing parts. On the other hand, the second part of the Project was mostly based on creativity and algorithmic thinking unlike the first part. In part-b we constructed three different algorithms to play the game we created in the part-a just like artificial intelligence. You will find more information about those parts and what we did in those parts as you keep reading this report which is the part-c of the Project.

Part-A:



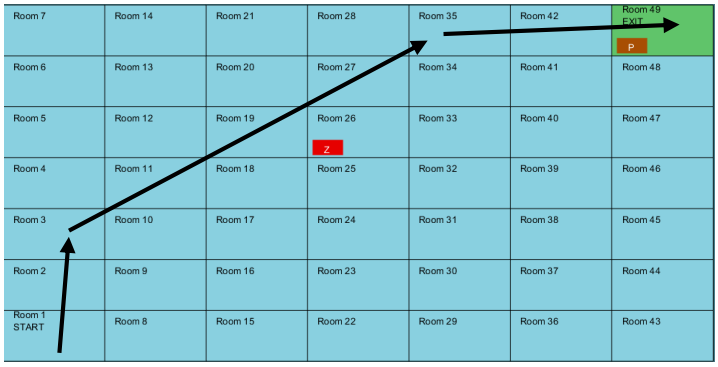
The game is developed with an object-oriented design and we were given five class (Table-1) and two user defined function. The two functions were used by different classes in order to create and organize the visuals of the game on a figure and they were completed by the teaching assistants. What we did in this part is to develop and organize the methods of the objects. The methods we did most in this part are the constructers, which requires error check, getter functions and heritance for objects player and zombie. As we move forward in completing the missions given in the code files, we became more familiar with the dynamics of the game and the ideas about defeating zombie in the game started to shape in our minds. In part-b, we are going to briefly explain those ideas and give you the results of them.

Part-B:

In this part, we will briefly explain the strategies of algorithms and then compare them from different perspectives. Before explaining the algorithms, we developed, let us explain how we integrated them to the game. We observed that in the game class the input (way that player moves) is taken by the build-in function "ginput" and we decide to define our algorithm as a function and replace the ginput with our functions. By this method the move of player would be generated by our functions which takes all the information of the game as input and gives the move that should be done in the light of its strategy.

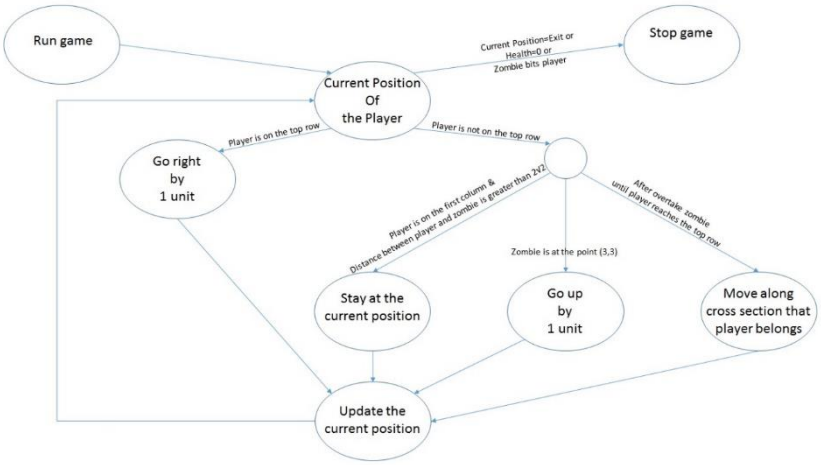
Explanations of Algorithms

Algorithm1 (AlgorithmStay):



This algorithm is our laziest algorithm. Its strategy is mainly based on the same thing as the others. All of our algorithms goal is to somehow become closer to the exit than the zombie is so that the zombie will stop chasing the player and run to the exit. In this algorithm player waits for zombie in the starting room to come closer. The starting room is the best spot to wait in this game because this is the only room that there is no risk to be poisoned or trapped. When the zombie reaches a certain distance, player uses its advantage of move number and gets closer to the exit room than the zombie.

Flowchart of Algorithm 1

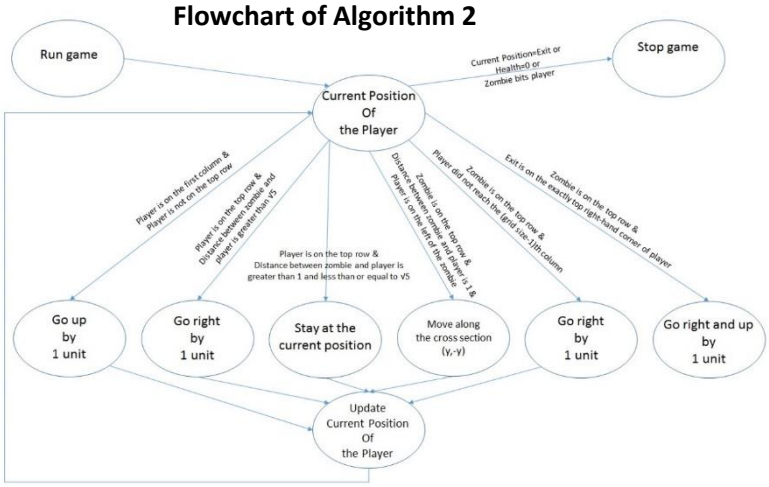


After this move both player and zombie rushes for the exit but since player makes two moves while zombie is making one move it is impossible for zombie either to catch player or get to the exit Room before player. But this algorithm has disadvantages too. For the grid sizes, smaller than 5X5 zombie can get to the exit room just before the player so our poor player meets zombie at the exit room and become a zombie. (lose the game) The algorithms path is give in above.

Algorithm2 (AlgorithmSide):

This is our most ambitious and unfortunately least successful (explained later on) algorithm. This algorithm's strategy is to meet and pass the zombie in the upper side of grid. So, player starts and climbs to the top of the grid bravely and when it reaches to the top it checks its distance with zombie and walks towards zombie but leaves the last move to zombie. When the zombie gets to the neighbour room of player it passes zombie and rushes toward exit.

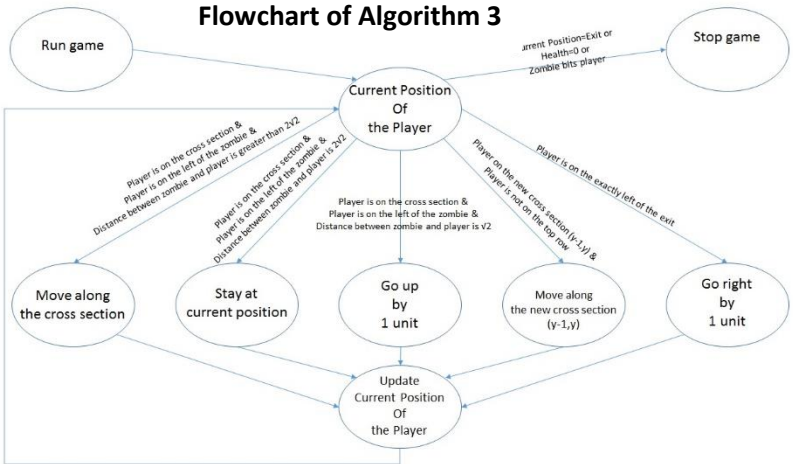
Flowchart of Algorithm 2



Algorithm3(AlgorithmDirect):

In this algorithm, player starts to move towards to end with cross step. While moving, zombie is also moves towards to player by using the same way. Player and zombie keeps moving until the distance between player and zombie becomes $2\sqrt{2}$. At that point, if x coordinate and y coordinate of the player equal to each other and at x axis, player is on the left of the zombie, player waits at the current position till the distance between player and zombie is $\sqrt{2}$. During this waiting process, zombie keeps moving to the player. Thanks to that, at some point the distance becomes $\sqrt{2}$ and player move up by increasing y coordinate value by 1. Then, player keeps moving on the cross section where its x coordinate is equal to y coordinate – 1. Zombie tries to follow and catch the player but it cannot because of the difference of the number of steps. When player reaches the top row, this x=y-1 cross sectional movement stops. The exit is on the right of the player. All the player must do is moving to right and player wins the game.

Flowchart of Algorithm 3



Discussion of Algorithms

Now, let's compare our algorithms by playing 100 games!

For this purpose, we created 3 different game models and gave them names as Small, Medium and Large. Small game has 5x5 game area and %70 hazard chance. Medium game's game area is bigger but as we expect that we will need more steps to win, we decreased the hazard chance. Thus, medium game has 10x10 game area and %50 hazard chance. Likewise, we have the same approach for large game. Its game area is 15x15 and hazard chance is %30. By making MATLAB play these 3 different games 100 times for each, we obtained the following data;

Small Game 5x5 - Hazard Chance = 0,7

	# of wins out of 100 plays	**Average # of steps to win**	**Average # of steps to lose**	**Average # of health points in the case of win**	**Computational time**
Algorithm-1	100	10	-	53,2	0,1336 s
Algorithm-2	74	10	7,346	33,514	0,1319 s
Algorithm-3	0	-	5	-	0,1065 s

Medium Game 10x10 – Hazard Chance = 0,5

	# of wins out of 100 plays	**Average # of steps to win**	**Average # of steps to lose**	**Average # of health points in the case of win**	**Computational time**	
Algorithm-1	74	25	22,769	38,649	0,3220 s	
Algorithm-2	64	19	12,389	10,781	0,2655 s	
Algorithm-3	84	11	8,625	42,5	0,2336 s	

Large Game 15x15 – Hazard Chance = 0,3

	# of wins out of 100 plays	**Average # of steps to win**	**Average # of steps to lose**	**Average # of health points in the case of win**	**Computational time**	
Algorithm-1	80	40	37,6	44,125	0,6163 s	
Algorithm-2	56	28	19,023	13,214	0,4891 s	
Algorithm-3	84	17	12,250	43,214	0,4351 s	

For small game;

Our first algorithm has 100 wins out of 100 plays. About number of wins, first algorithm is the best option because second one has 74 wins. In addition, third algorithm won none of the game. For this reason, third algorithm is the worst for this case. About average number of steps to win, first and the second algorithm have 10 steps as they are the best since algorithm 3 cannot win. In the average steps to lose case, algorithm 2 has higher value than algorithm 3 which means that algorithm 3 loses the game in a shorter path. Algorithm 2 proceeds more than algorithm 3. When we compare those according to average health to win, algorithm 1 has the highest value. Although having the same average step to win and hazard chance, algorithm 1 wins with higher average health than algorithm 2. In terms of computational time algorithm 3 is the fastest one although it cannot win. Algorithm 1 and 2 nearly have the same time. As a conclusion, we can say that in small grid game algorithm 1 is the best option.

For medium game;

In terms of number of wins' algorithm 3 is the best this time. In addition, it wins with the least number of steps. Similarly, while losing, it loses in a shorter path. Algorithm 1 loses after a long path. Again in terms of average health points to win, algorithm 3 is the best. 1st algorithm uses a longer path to win, this can be the reason why its average health to win is less. In that case algorithm 2 is the worst one. When compare our algorithms about computation time, algorithm 3 is the best and algorithm 1 is the worst option. In that type of the game, algorithm 3 is definitely the best choice.

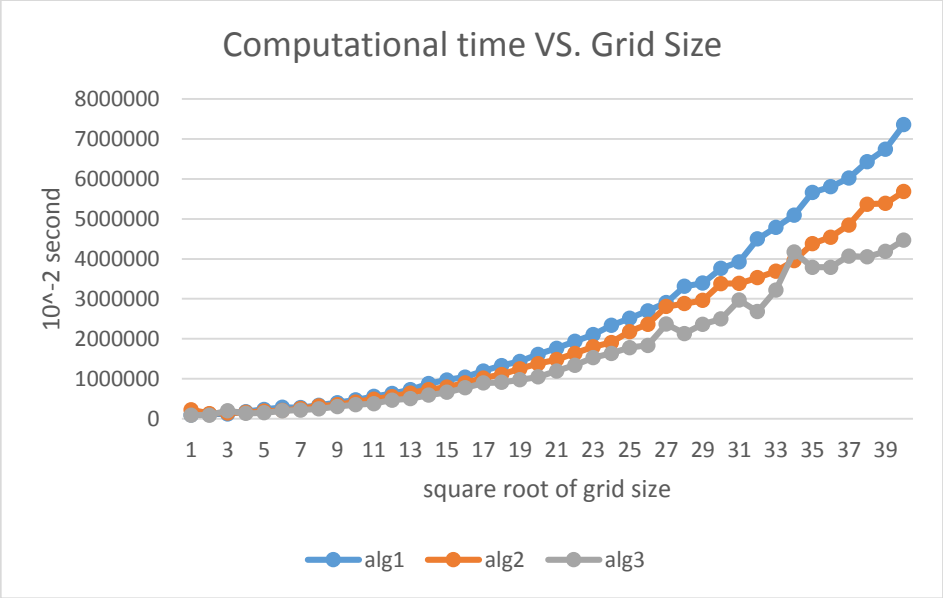
For large game;

Algorithm 1 has higher number of wins although 1st and 3rd have similar number of wins out of 100 plays while algorithm 2 is the worst. While winning, algorithm 3 uses least steps and 1st one uses the most. Similarly, while losing, 3rd algorithm has the shortest path in average and 1st one has the longest. At both of the step cases, 2nd algorithm is in the middle, not the worst not the best. As in the number of win case, there are again a competition between 1st and 3rd algorithm while 2nd one is the worst. They have similar values but 1st algorithm is the best with a little difference. Comparison about computation time, 3rd algorithm is the fastest and 1st one is the slowest. Algorithm 3 loses only 1 comparison with a very little difference so we can say that for the large game 3rd algorithm is the best option.

General discussion;

First, we checked the computational times of each algorithm. We saw that for any kind of grid there is an order of algorithms about their computational time. The third algorithm beats the others in the category of computational time in all grid sizes. Also, the first one is the middle in the list in each grid. The reason of this result may be explained by their strategies. As it is seen in tables there is a positive correlation between number of wins and computational time. (take the average steps to lose into account for algorithm-3in small grids) While the algorithm-3 is focused on reaching exit as soon as possible and moves toward that way, the two others focused to play with the zombie, that's why they lose time and play more steps. Specially "number of wins out of 100 plays" and "Average number of health points in the case of win "are quite close for algorithm-1 and 3 because they follow a similar path and play with similar net steps.

By net steps we want to mean the steps made in grid (not in beginning room) Since algorithm 1 choses to wait in the beginning room safely its total step exceeds algorithm-3's and its computational time peaks for larger grids. (see figure in right)



After comparing the algorithms in the perspective of computing time, we also need to add the following remarks. Maybe for smaller grid sizes computational time does not have such influence so we need to give you the following information. We mentioned that the algorithm-3 is seems to be the best algorithm but its number of wins out of 100 is zero as it is seen so we can say that this algorithm is suitable for larger grids. Also, the algorithm-1 can win in 5X5 girds but is fails in 4X4 and smaller grids. On the other hand, algoritm-2 can beat the computer in every size of grids including 3X3 and 4X4.

Conclusion:

To sum up, for this team project we focused on object oriented design and completed the game. Then we created different algorithms which both designed to win the game. While designing the algorithms we tried to look to the game from different perspectives and develop different strategies to win. After constructing the algorithms, we observe their efficiencies by obtaining different data from the game. After careful considerations of different aspects, as we mentioned in the discussion part repeatedly, we nominated algorithm-3 as our most efficient algorithm to play and win the game.