[MILESTONE 3] PLANTS VS ZOMBIES DOCUMENTATION

Milestone 3 – Snake Squad

## CONTENTS

[DESIGN DECISIONS 1](#_Toc528615767)

[USER MANUAL 2](#_Toc528615768)

[UML 6](#_Toc528615769)

# DESIGN DECISIONS

*Author: Kurt Burton-Rowe*

The design decisions for this milestone involved fixing the implementation of the graphical user interface from the previous milestone where the zombies would walk past plants before eating them. New plants and zombies were added to the game, each having different behaviours to those from the previous iteration. The biggest addition for this milestone was the addition to the undo feature to the game.

The addition of the new types of model characters within the game meant that multiple new classes were created. The new characters that were added are as follows: Double Pea Shooters, Walnuts, Pylon Zombies and Run Zombies. Double Pea Shooters, as their name implies shoots Normal Peas at twice the rate as a Pea Shooter. A Walnut is a plant with a higher max when compared to the rest of the plants—being used to get eaten and buy time for the user to decide to either accumulate sun points or attack the enemy. The Run Zombie is self explanatory, it traverses the board at a much faster rate than a Normal Zombie while having the same level of health. The Bucket and Pylon Zombies are very similar in the sense that they move at the same speed as a Normal Zombie, but they have varying max health variables. All of these classes except walnut used previous plants and zombies as super classes, which places emphasis on the factory design pattern and loose coupling of classes.

The Controller and View packages had to be updated and revamped to not only accommodate this milestone’s requirements but to also refractor our code to fix the logic that worked for the text implementation of the game but did not work reliably for the GUI. If the user attempted to run in eclipse in the previous version, the user would get an ioe exception because when using the Make program, it would create a different bin folder that would not allow the view to access the same image files. This issue is now resolved. The source code was rearranged to make the organization of how our game works a lot simpler. The packages are now Characters, Controller, gameModel, Images, Tests and Views. This made it simple for us to categorize exactly how the classes pertained to the game. The gameModel classes update the Game Board View class, which in turn update the Game Controller class. The Game Controller class updates the gameModel classes with the new information. This is to ensure that we follow the MVC pattern.

The Game Board View class is the same from the previous Milestone, however, the undo and redo features have been refined and correctly added in this iteration. We have added prompts where the user can decide how many zombies will be generated, along with how many waves that they will come in. We decided on this path because we wanted our version of the game to mimic the actual one where it has a lot of variation to their levels rather than hard coding exactly where each zombie will appear from, they will appear at different spots along the y-axis of the game each game played. The Game Object Controller class deals with the spawning logic of the zombies since it directly deals with all the character objects in the game. There was a big emphasis on not cluttering the Game Controller class with tedious or irrelevant functions that could be delegated to other classes.

The Game Object Controller class discards characters that have died and because of this it we are unable to bring said characters back to life. The reason why we did this is because we didn’t want our lists of the characters on the board to become so large that traversing the lists would become wasteful, so we decided to that in order to keep the code nice and tight where the characters that are alive are the ones that are being manipulated. The most complex of the classes of this milestone are the Collision, Move and Game controllers. The reason why we decided to separate the collision and moving detection logic from one another is because many of the plants themselves are stationary but the objects that they shoot are not. Our approach to the entire project was to loosely couple as many classes as possible.

# USER MANUAL

## Running the Game

*Author: Tareq*

For milestone 1:

The output directory contains the compiled code. It must first be added to your classpath.

make compile will compile the code and put the class file in ./output.

export CLASSPATH=$(pwd)/output will add the output folder to the java classpath for **linux** and **macos** machines

make run will start the game.

Once the console starts running the game, it will showcase this:

starting game

running main thread

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

LEVEL: 1, TURN: 0, sunPoints: 10, sf cooldown: 0, ps cooldown: 0

What would you like to do ?

buy sf x y: buy a sunflower for 10

buy ps x y: buy a peashooter for 10

Enter: do nothing

The point of the game is to eliminate all the zombies that get spawned in with the help of plants before they reach the house. The game is situated in a simulated front yard. The front yard is in the shape of a grid, 9x6. The zombies move from the left going towards the right, they only follow the row they get spawned in. The zombies cannot move vertically or diagonally.

The user can collect sun points which are the currency of this game, they let the user buy the plants they desire. There are two ways, so far, to collect sun points. The first is from the sun, the game will generate 10 sun points every 2 turns which is automatically collected into the user’s collection. The second form is from the sun flower plant. The plant generates 25 sun points every 4 turns, which is also collected.

There are two types plants in this version of the game, sunflowers (sf) and pea shooters (ps). Both these plants cost 10 sun points. To plant one of these plants, there are two conditions that must be met:

1. The user must have enough sun points to purchase them.
2. There must be a space where the user can place the plant. No plant can be planted in a spot in the grid if there is already a plant there.

All plants and zombies have total health, if their health reaches 0 or in the negatives it will die. Once dead, the plant will disappear from the game. The only way that a plant can take damage is if a zombie is on the same grid position as the plant. For every turn a zombie is on the same position as a plant, the plant will take damage. They will stop colliding once one of loses all its health and dies. The only way for a zombie to take damage is from a pea that gets spawned in from pea shooters

The pea shooters are the only offensive type plant currently available to defend against the zombies. Once a zombie is spawned in the same row as a pea shooter, the pea shooter will start shooting a pea every \*\*\*\*\*\* turns. The pea will go through the board, until it hits a zombie. When they collide, the pea will inflict its corresponding damage to the zombie and the pea will disappear.

This game is turn base. The player will choose one thing to do on the board and then a full turn will be done. Once a turn is over, there are a few things that happen before the start of the next turn:

1. The timer goes off by one, which indicates the game turn
2. All entities that can move (zombie and the pea), will move depending if they collide with an enemy or not.
3. If a collision occurs, the plant or the zombie will intake damage and the entities will stop moving.
4. If an entity dies, it will be removed from the game.
5. All sun points that can be accumulated will be added to the user’s database

In the game, zombies spawn in randomly, at random rows in the grid. When a turn starts, all the zombies on the board will move by one space if it’s been on the board for more than 5 turns. It doesn’t move forward only if there’s a plant that collides with it. Zombies do a total of 10 damage per turn. Once they reach below 0 health, they will die and be removed from the game.

Example:

buy ps 4 3

buy ps:4,3

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ SF ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ SF ][ ][ ][ PS ][ ][ ][ ][ ZB ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

LEVEL: 1, TURN: 10, sunPoints: 40, sf cooldown: 0, ps cooldown: 0

What would you like to do ?

buy sf x y: buy a sunflower for 10

buy ps x y: buy a peashooter for 10

Enter: do nothing

To place a plant onto the playing field, the user must input the following:

buy (plant acronym) x position y position

For example:

buy sf 1 1

buy sf:1,1

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ SF ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

LEVEL: 1, TURN: 1, sunPoints: 0, sf cooldown: 2, ps cooldown: 0

What would you like to do ?

buy sf x y: buy a sunflower for 10

buy ps x y: buy a peashooter for 10

Enter: do nothing

The user placed a sun flower onto position 1, 1 of the field.

Once a plant has been used, a cooldown time for that plant will occur so that the player doesn’t continuously spawn in plants every turn. The player can also decide to not do anything by simply inputting enter.

There is a fail-safe precaution in place for the player. Before zombie reaches the home, a lawn mower is placed in front of it. A lawn mower only has a one-time use. It only activates once a zombie collides with it. Once a lawn mower is collided with an enemy, the lawn will go through the entire row, killing all the zombies in its way. However, if a zombie, reaches the home and there’s no lawn mower, then it will be a game over and the zombies will win.

Example if the lawn mower was used:

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

[ NPC ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

LEVEL: 1, TURN: 6, sunPoints: 20, sf cooldown: 2, ps cooldown: 0

What would you like to do ?

buy sf x y: buy a sunflower for 10

buy ps x y: buy a peashooter for 10

Enter: do nothing

The only way to win is if the player succeeds in eliminating all zombies. Once it’s done the game will end and terminate itself.

In future iterations of the game, there will be an added shovel mechanic which will let the user remove a plant while also regaining a certain amount of lost sun points.

For milestone 2:

In this version of the game, tests and gui implementations were added.

For test cases, they check if the lawn mowers, movable, npc, runner, sunflower and zombie work properly.

For the gui implementation, an added view and controller was added for the game. Instead of showcasing everything on the console, the game is played on a separate *view.* The game is then played thanks to the controller.

# UML

*Author: Anthony*

