[MILESTONE 4] PLANTS VS ZOMBIES DOCUMENTATION

Milestone 4 – Snake Squad

## CONTENTS

[DESIGN DECISIONS 1](#_Toc528615767)

[USER MANUAL 2](#_Toc528615768)

[UML](#_Toc528615769) 9

# DESIGN DECISIONS

*Author: Kurt Burton-Rowe*

The design decisions for this milestone involved fixing the implementation and the of the graphical user interface from the previous milestone where the undo and redo buttons unreliably worked. We attempted to save the state of the Game Objects Controller by simply making a stack of the created object in that could be popped and referenced in whenever the buttons were pushed, but it then came to our realization that when the GOC object was being pushed onto the stack that it was being referenced to like a pointer. Since the stack was referencing the original object and was not creating and pushing instances of the same object, it made sense to us that the GOC would be responsible for creating a clone of itself for a stack to reference. Trying to focus each class on a specific responsibility we created a new class that would be responsible for the undo and redo actions within the game, which meant that this new class would be responsible for keeping track of the stacks of the clones and new GOCs. Because the game is based off the Model, we realistically only need to keep track of the Game Object Controller and its clones. This allows us to serialize only one class – the GOC Manager. The level builder: the day/night level selection, the user’s selection of the amount of zombies that that they will fight and the amount of waves said zombies will appear. 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 of the save and load feature to the game and the android conversion to the game.

The character models in the game remain the same-- from the base models, to the additions that were added in last Milestone. The coding for the character models were built using inheritance, loose coupling and the factory design pattern.

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 a stack null pointer exception because the execution of our stack use was flawed. This issue is now resolved. The source code was rearranged to make the organization of how our game works a lot simpler. We have now added a pre-configured and built version of the game, it is a lot simpler to run from the terminal. The packages are still 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 GameBoardView which in turn update the GameController. The GameController updates the gameModel classes with the new information. This is to ensure that we follow the MVC pattern.

The GameBoardView has been refractored to add the load and save features, while expanding the level builder options for the user by adding more prompts to build their game. For the most part that, the GameBoardView is overall the same from the previous Milestone, however, the undo and redo features have been refined and correctly added in this iteration and along with the requirements for this Milestone.

Since the game follows the protect the castle puzzle template, the only decisions that the user must decide are where do they want to plant their plants on the board. We created the Non-Playable Characters class because the game itself, keeps track of each plant’s and zombie’s attack power and health—including how quickly objects traverse the board itself.

The board of the first level for this milestone is a simple 6x10 array since it is the first level of the game and it is intended to get the user used to the controls and the logic of the game. There is only one zombie that will attack and there is only one row that the user’s plants must defend.

The most complex of the classes of this milestone are the GOC Manager and the GOCManagerFileStream classes. The reason why we decided to separate the two is because we wanted to each class to deal with a specific problem and not overcrowd or tightly couple with other classes. A small detail that was added to the game that we are proud of is the fact that our Peashooters don’t shoot unless they detect a zombie in their aisle. Our approach to the entire project was to loosely couple as many classes as possible.

**Important: In order to get the view test to work, you must click the ok button until the game’s frame stops popping up (should only be three).**

# USER MANUAL

## Running the Game

*Author: Tareq*

For milestone 4:

The bin directory contains the compiled java code for the project.

For ease of use a makefile has been provided in the PlantsvZombies directory.

The makefile can compile and run the project with one simple commmand assuming

an a unix based system is in use.

`make run`

will compile the code and put the class files in `bin` directory then it will

run the project.

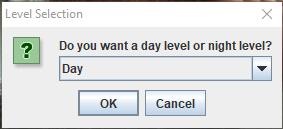
`make runjar`

will run the a version of the game from a jar file inside the /bin directory

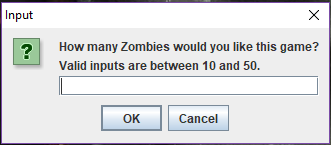
If changes are made to the game a new executable jar file can be created using

`make compile` then `make jar`

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

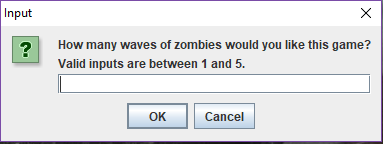


This allows the user to choose between day and night levels. Once the user picks the level design, the user is prompted with this:



User then chooses how many zombies they want inside the game. They can pick between 10 to 50 zombies (anything else out of that range will indicate an error and the user will be able to choose again).

The user then pick how many zombie waves they want to have (between 1-5).



Once all the preliminary settings are in place, the game starts up.



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. There is also a sidewalk 1x6 at the right hand side of the front yard. The side walk is where the zombies spawn in from. 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 four types of plants in this version of the game, sunflowers, pea shooters, double pea shooters and walnuts. These plants cost 10 sun points each. 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 variation of the plant.

The pea shooters are one of the only offensive type plants 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.

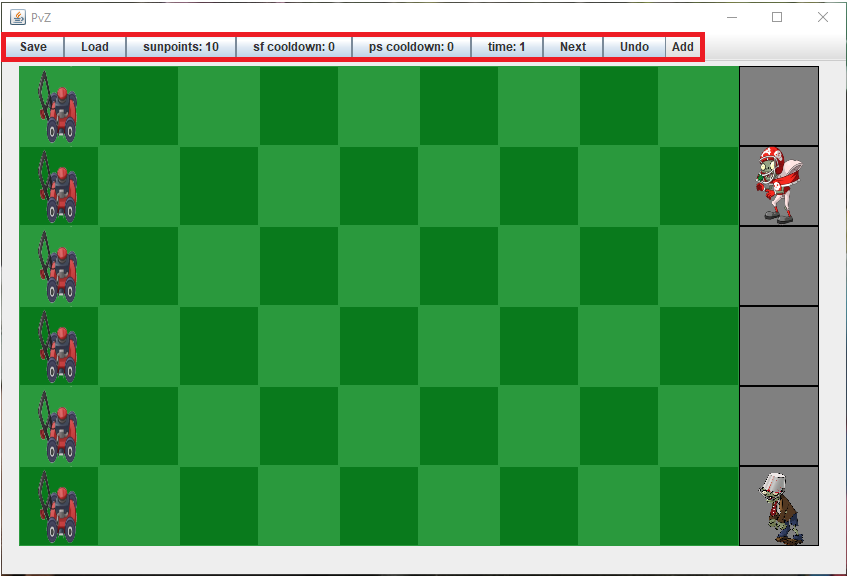
The double pea shooter works the same way as the regular pea shooter, except it’s firing rate increased by double. The walnut plant is a defensive plant that prevent zombies from accessing the lane. This plant can take has 150 health.

This game is turn base. The player will choose certain things 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 damages vary depending on the type of zombie it is. Once they reach below 0 health, they will die and be removed from the game.

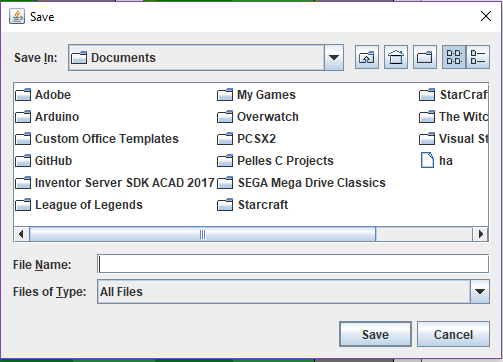
The bucket head zombie and the pylon zombie are two upgraded versions of the regular zombie, the only difference being the items they wear on their heads, bucket and pylon. The pylon zombies have higher health than regular zombies and bucket head zombies have higher defense than pylon zombies. The running zombies are faster and healthier zombies than the regular zombies. These zombies all deal 10 damage per turn.



There are multiple buttons at the of the display; Save, Load, sunpoints:, sf cooldown:, ps cooldown:, time:, Next, Undo and Add.

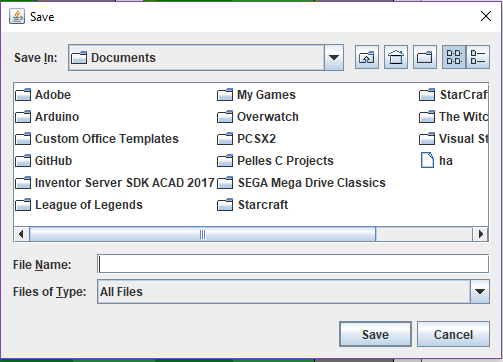


The save button saves the game at the exact moment of when the user clicks the button. The user is then prompted to a document to name and save their save file.





The load button loads up the save file that the user saved earlier. Once clicked a browser will appear for the user to choose their save file.





The sunpoints button is a dead button that only indicates how many sun points the user has at the given turn.



These cooldown buttons indicate how many turns are left for the user to use a plant again.



The time button indicates what turn the user is on.



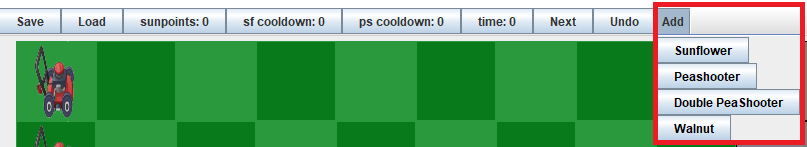
The next button is the button the user clicks when they are done setting up their turn. The user can add plants, look at their stats, save, load and undo before they click the button. This button is how the player progresses through the game. When the it’s initiated, the time button increments by 1, the sun points are tallied up and the cooldowns are reworked for the situation.



The undo button allows the user to go back to their previous turn. Once they go back to their previous turn, they regain all their bought sun points, the plants planted the turn before disappear and all movement that took place go back to their original position. The user can repeatedly click this button to go back to the start of the game. If the game is at the start of the game and they click the button, nothing will occur.



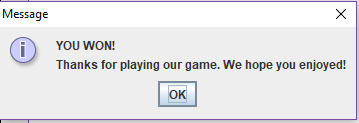
The add button is a drop down menu style button that allows the user to choose between the 4 plants (sun flowers, pea shooter, double pea shooter and walnut).



To add a plant, the user must click their desired plant and then click the location of where they want to place it. A plant cannot be planted at the same location as a planted that’s already been planted. A plant cannot be placed if the player doesn’t have the necessary amount of sun points or if the cool down is insufficient.

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.

Once the player either beats all the zombies or if a zombie enters the home a pop up will appear indicating that the game is over.



# UML

*Author: Anthony*