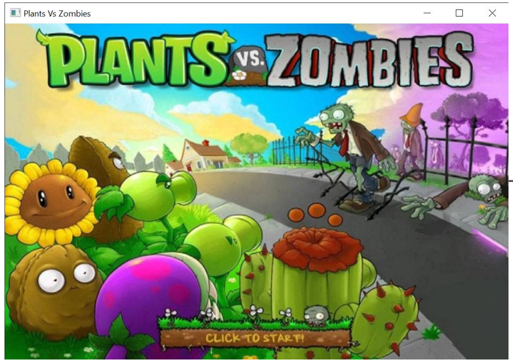
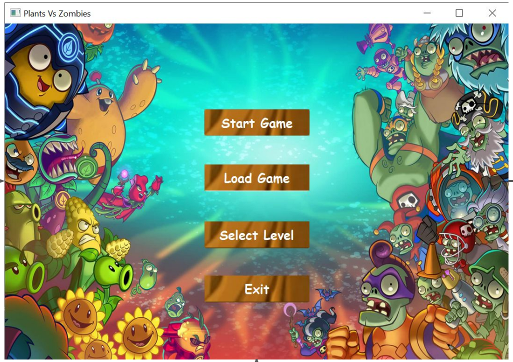
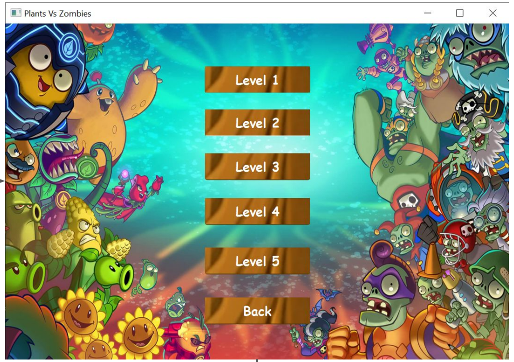
Plants vs Zombies

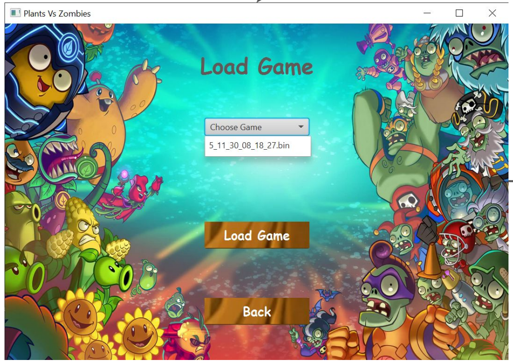
Ayaan Kakkar (2018028)
Shashwat Aggarwal (2018097)

Design and Implementation

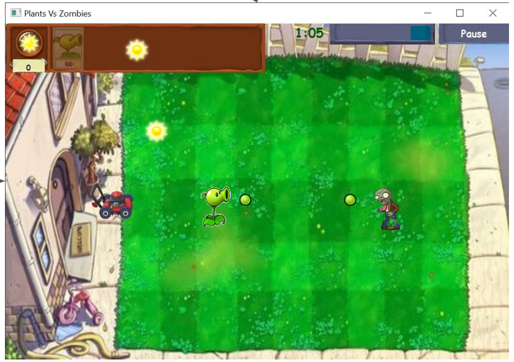








- On starting the game there is an option to start game that starts game from level 1.
- Load game allows player to play an already saved game that he/she might have saved after pause.
- Select level allows user to select one of the 5 levels.



Design and Implementation

The game is built using JavaFX and SceneBuilder. The static GUI components that are always present such as the lawn ImageView, buttons, etc. are all implemented using FXML with the help of SceneBuilder. Each FXML is connected to its controller class that contains all the button listener functions and other attributes/methods to handle the functionality.

The dynamic GUI components such as Zombies, Plants and Bullets are implemented by generating a new ImageView or using an existing ImageView in the game pane and adding making that ImageView visible in the game pane.

The moving/animated GUI components are translated by their respective Timelines. This was tricky to use so as to avoid racing conditions during collision events. This issue was overcome by implementing collision right after translation in the same Timeline. Another issue was for saving the game state, we could not save the ImageViews or Timelines as no JavaFX component are Serializable. This was overcome by saving the coordinates and types in respective data classes that we made for various sprites. The JavaFX components had to be generated once again after deserialization.

Design Patterns:

1. Factory: The intiLevel method in Level class calls various methods such as initZombies method which generates new zombies, createGrid that makes the ImageViews for allowing plants to be placed.
2. Iterator: We use iterators at multiple places to traverse collections such as Arraylists, HashMaps and HashSets.
3. Facade: We just call one function to make the level passing the level number to it. This function is called from the start game button, select level button and load game button by different level values.
4. Composite: We have implemented the composite design pattern in the Zombie class, wherein it holds an ArrayList of its own type.

Work Distribution

The majority of the code was written together during the long hours we spent sitting together at our laptops to code the main functionality of the game such as the level generation and timelines.

Ayaan: Responsible for controller classes, zombie class and generation of zombies during the start of level. Also built the function for generation of sun tokens and bomb blast feature of bomb plants. The pause function and loadGame (deserialization) was also undertaken.

Shashwat: Responsible for making static GUI components in scene builder, plant class and generation of plants on the game pane on drag and drop. Also built the function for shooter in peashooter and special shooter. The resume function and save game (serialization) was also undertaken.

Bonus Feature

We have created a special plant in our game that is the ShooterBombSunflower plant. As the name suggests it is a shooter, bomb and a sunflower. It shoots like a normal peashooter and generates sun tokens at regular time intervals as well. But another feature that this plant has that it bursts as soon as any zombie touches it killing all the zombies in the blast radius. This plant due to its advanced features costs 250 sun tokens which is the highest of the all.This feature makes the game even more interesting by having such a powerful plant making way for more strategies to think of by the player.