Design Patterns

Singelton Design Pattern

It is applied in the GameController, Game, Player and FactoryOfGameObjects classes. As the application needs only one instance of these classes during the gameplay.

Factory Design Pattern

It is applied in FactoryOfGameObjects class as it randomly returns game objects of different types in the GameController in createGameObjects method.

Command Design Pattern

It is applied to enable the user to give commands without knowing the details of the execution. There are three commands: slice objects, load game and reset game.

Strategy Design Pattern

It is used to change the game objects velocity and the time interval between throwing objects on screen according to the player's score.

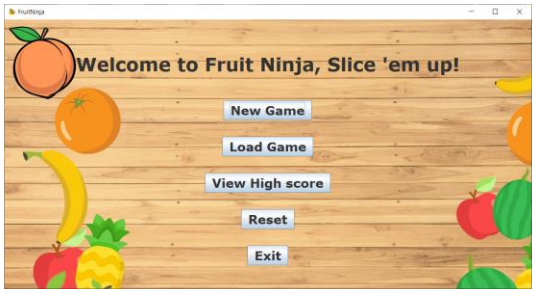
Memento Design Pattern

It is used to take a snapshot of the player's state and save it in the xml file then later on using the memento to load the player's state. The Originator is the Player and the CareTaker is the GameController.

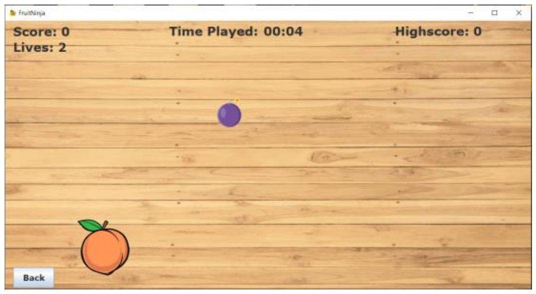
Description of the Design.

This game relies heavily on animation. At first, we were using JavaFX as basis for our animation work but after facing some obstacles along the way and doing some research, we concluded that Swing would be a lot better for us to use in the animation process. We applied the Singleton Design Pattern in the Game class, GameController, Player and FactoryOfGameObjects. As the gam would only use a single instance of each of these classes during operation. Assuming there would only be one player of course. We used the GameActions interface and implemented it in the GameController class. The GameController is a Singleton class which has an instance for the IGameStrategy interfance which is responsible for setting the difficulty for the game. We used the Strategy Design Pattern in deciding the difficulty of the game where it has two main methods. Getting the time interval and getting the game object velocity. And according to the difficulty, the time interval of the game object's motion and its velocity is determined. Each difficulty has its own strategy and it's set according to the score of the player. The GameController class also has an instance of the Memento class which is responsible for the process of saving the state of the player in order to be loaded later on. The state of the player includes his lives remaining, score and total time played. This state is saved through the memento class to be saved in an XML file. This state is to be loaded if the player quit the game during gameplay. The Originator is the Player and the CareTaker is the GameController. There's another Singleton which is the Game class which has an instance of the Singleton Player class and an array list of the game objects to be created. There are two main types of objects in the game which are the bombs and the fruits, each has a parent class representing them and implementing the IGameObject interface, and each of them is inherited by the different types of fruits and bombs each according to their behavior. We used the Factory Design Pattern when creating the game objects in the GameController class where they are created randomly. The animation depends on three main phases, the first phase is when the game object is instantiated through its constructor determining its initial velocity, maximum height, X location and the Y location. The second phase is when all the previous information is used to draw the path in which the game object will take during the gameplay and this path is drawn in the Animation class. The third phasing is displaying the animation itself, where this animation is shown through the main Panel. We also have a Highscore Class where it's used for getting and setting the high score, The High score is separately saved in its own XML file as the High score is not linked to the player's state but to the state of the game itself.

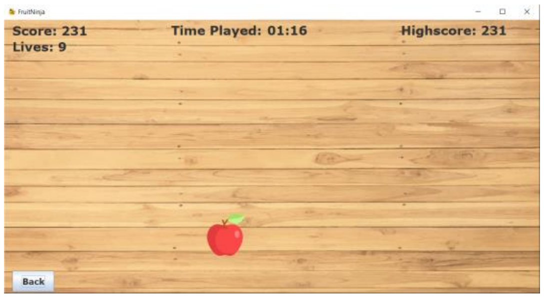
Snapshots :



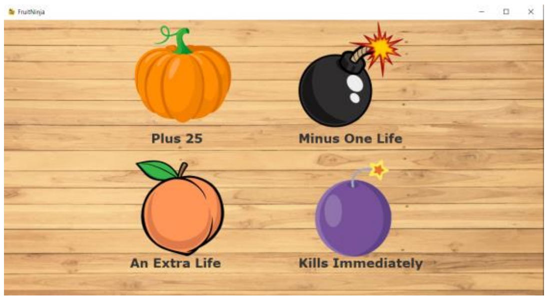
The Main Menu Screen



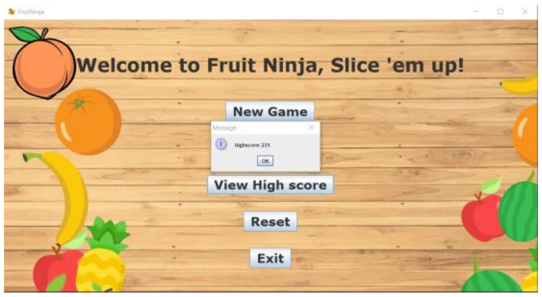
Gameplay Snapshot 1



Gameplay Snapshot 2

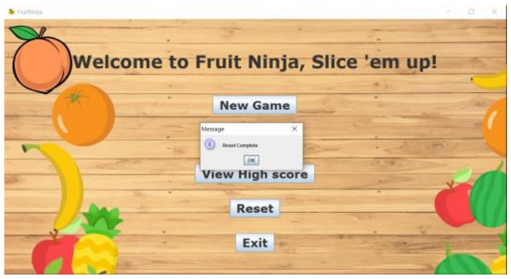


Intermediate Panel Snapshot.



Highscore: 231

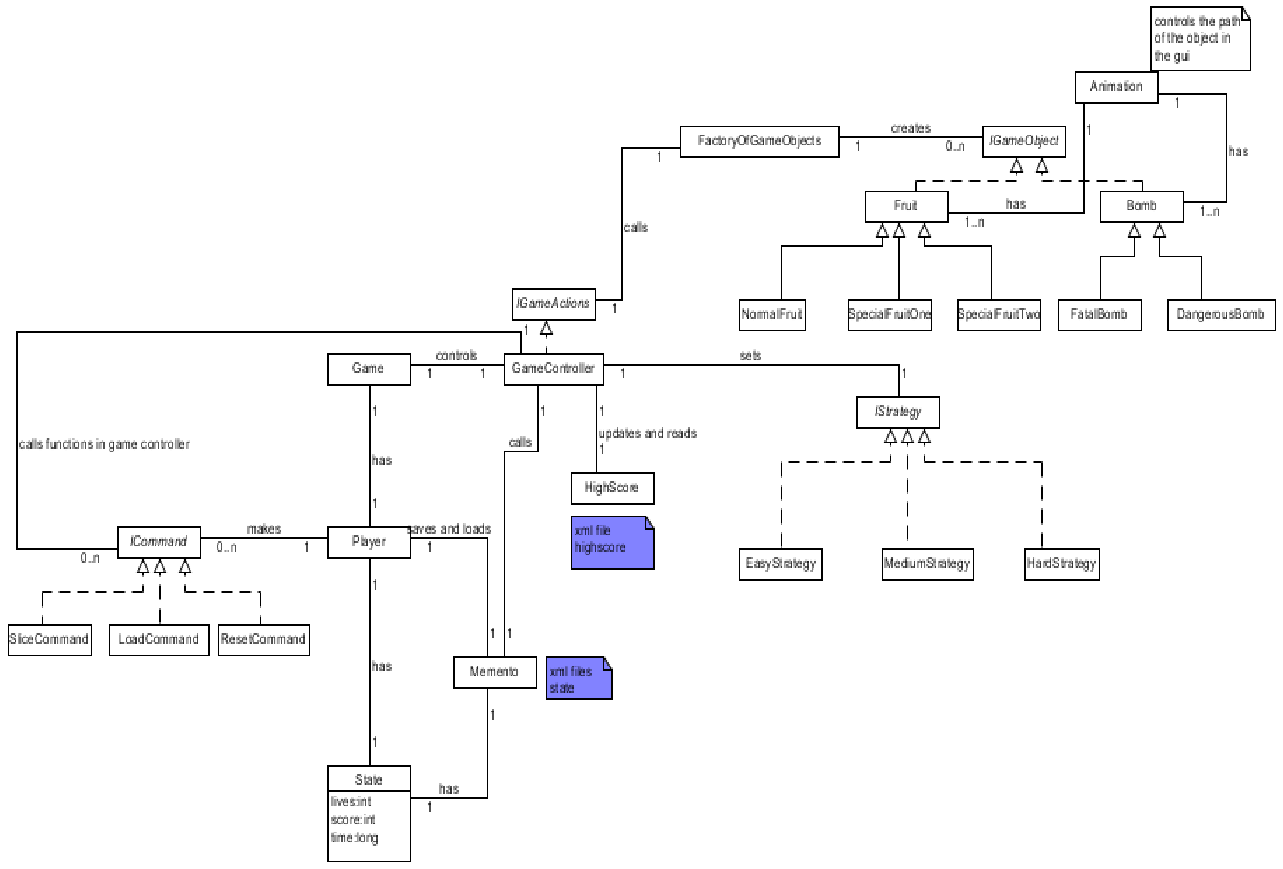
Viewing High score Snapshot



Reset Complete

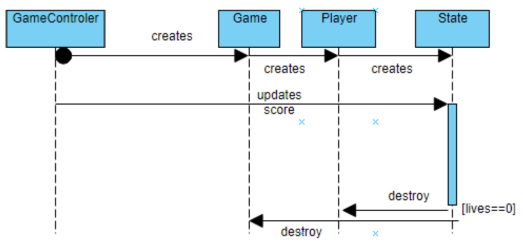
Resting the game Snapshot

Class Diagram:

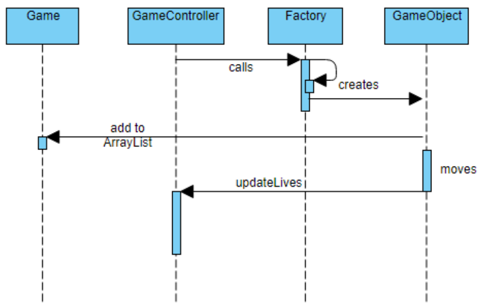


Sequence Diagrams:

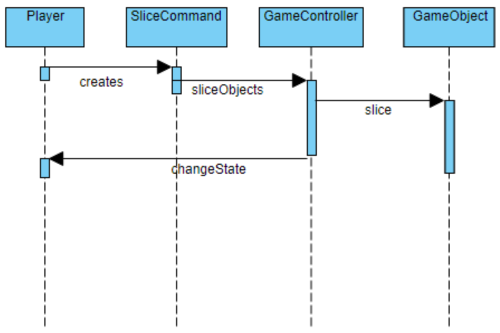
Initializing the game sequence diagram:



Creating game objects sequence diagram:



Slicing Objects sequence Diagram:



User’s manual:

1) Opening the game you will be greeted in the main menu screen which has several options to choose from either starting a new game, loading your last game, view your high score, resetting the game or exiting the application.

2) Pressing new game button starts a new game with your high score to beat.

3) Load game button gives you the ability to continue your last game.

4) High score button lets you know the high score that you want to beat.

5) The reset button gives you an opportunity for a fresh start, by pressing this button all your old games are deleted and the high score is set to zero again.

6) Before you start playing you will be given some information about which fruit is special, which bomb is fatal and so on.

7) When playing you have a timer, score and high score (the score that you want to beat), you also have a return button which gives you the ability to return to the main menu (pressing load game will make you return to your last point in the game.

8) Improving your score depends mainly on two things slicing as much fruits as you can and avoiding bombs, especially fatal ones.

9) You have three lives you lose one of them when you miss a fruit or slice a bomb

10) You lose immediately when slicing a fatal bomb.

11) Your game is saved automatically given that you did not lose.