Programming II

Fruit Ninja Game

Omar Essam Aboushousha	6492
John Hany	6441
Heba Hassan Elwazzan	6521

Prof. Dr Mervat Mikhail

Description

The Fruit Ninja game is a game where fruits appear on the screen and the user slashes away as many as they can. There are two modes: classic, and arcade. The classic mode does not have a time limit, but missing any fruits or slicing a dangerous bomb loses them a life out of three total lives and subtracts 5 from the score. Slicing a fatal bomb results in losing immediately. The player tries to beat their highscore and maneuvers through the increasing difficulty: in level one the fruits are slow and sparse, the bombs do not appear frequently; in level two, the speed increases as do the density of both fruits and bombs; level three is the absolute challenge as both speed and density are maxed.

The design follows the MVC design. There are 4 total packages: model, view, controller, and main. The model package contains all logic classes. The controller package contains the FXML controllers that control the FXML files created using SceneBuilder and which are stored in the view package.

The design is mainly comprised of a group of main classes in the model package: GameObject child classes such as GameObjectImplementation (where the major common interface methods are overrided), and its inheritors: the Fruit and Bomb classes. The Fruit class is extended by Apple, Banana, Watermelon, and SpecialFruit classes, where method relating directly to the specified object are implemented. Likewise, the DangerousBomb and FatalBomb classes extend the Bomb class.

Main game classes encompass the GameInfo class, which is a singleton class that stores all players information; Player class which contains the players' name, classic mode highscore, and arcade mode highscore; and GameModel class which contains all the information regarding a run of the game in either modes.

Other classes are considered to be utility classes such as Time (which represents the game screen timer), and classes which help with the design patterns used such as FruitWithBonus which extends SpecialFruit, GameState class which is implemented by ArcadeGameState, and ClassicGameState (which in turn has 3 respective classes for the classic mode levels). Other design pattern classes include Observer class and its children, and Subject class, as well as GameObjectFactory. Another utility classes in the model package are the XMLFileHandler class and XmlGame class, which help with marshalling and unmarshalling data from the xml file, GameData.xml. The XmlGame object created is then mapped to the GameInfo instance, as it was not possible for the GameInfo class being marshalled, due to the fact that it is a singleton class whose default constructor is private and therefore cannot be used by JAXB. The XMLFileHandler contains the save and load method, which were originally in the GameActions interface, but as we wished those two methods to be static, we decided a separate class for them would be better.

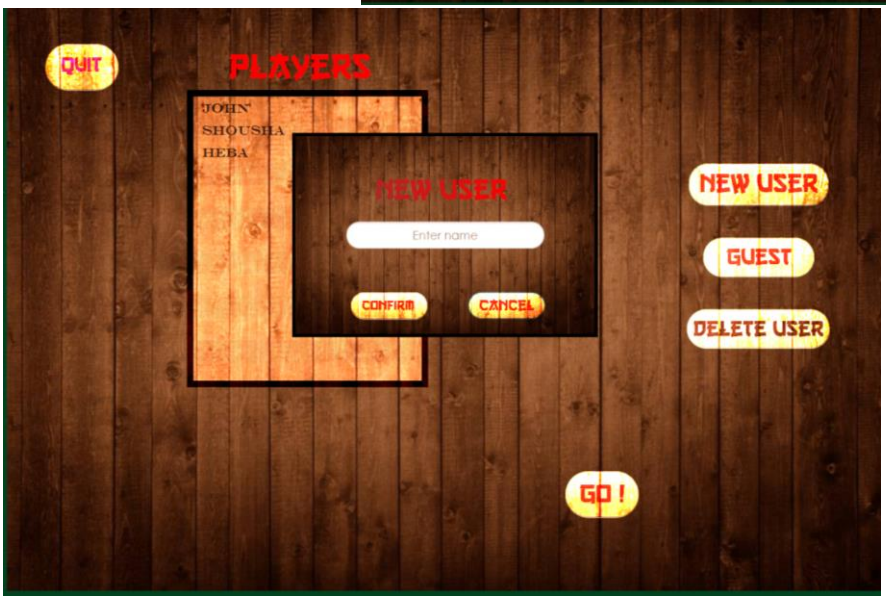
Design Patterns

5 design patterns were implemented:

1. Singleton Design Pattern
   The GameInfo class is a singleton class whose default constructor is private and has one instance which can be fetched using the static getInstance() method. Its intent was to create just one instance of the list of players and use it anywhere in the game code.
2. State Design Pattern
   The GameState class is extended by 4 classes: ArcadeGameState, LevelOneState, LevelTwoState, LevelThreeState. The GameModel class has a state variable which is used to decide things such as speed of objects being thrown, number of simultaneous objects, and the time between each object. The gameState variable is updated when the score increases in the ClassicState instance.
3. Decorator Design Pattern
   To create a special ability to some fruits, a decorator design pattern is used where the FruitWithBonus object wraps a Fruit object, and has a method that doubles the score. There was perhaps room for more decorator classes if needed.
4. Observer Design Pattern
   Observer and Observable design pattern was used to update the game screen labels such as the number of lives, the timer, the level/mode, and the current score. Binding and properties could have been used instead but we decided it was an opportunity to design the observer and subject classes ourselves and use them in order to practice the observer design patter. Inside the GameModel class whenever an update occurs, the notifyAllObservers() method is called and sends to each observer class a GameScreenLabel object, which consists of the current score, the number of lives (in the ArcadeGameState the number of lives remains constant) and the time. Each observer class then in turn updates the labels they control on the game screen. For example the LevelObserver updates the level lable by changing its text; the TimeObserver continuously updates the screen timer as time goes by; the ScoreObserver updates the score label whenever the user scores points and updates the score in the GameModel; and the LivesObserver sets the visibility of the heart icons which represent the lives whenever the player loses a life.
5. Factory Design Pattern
   This design pattern was implemented in the GameObjectFactory class where the control classes request to generate a random fruit or a random bomb, and the factory class returns it. For the fruits, it first generates a random number between 1 and 20, and if that number is 10 for example, it generates a random number between 3 and 5 which represents the special fruits. This means there is a 1 in 20 chance of generating a bonus fruit. If the controller class asks for a bomb, the class generates a random boolean to decide between the fatal bomb or the dangerous bomb. These numbers are then used as indexes to an enumeration which the GameObjectFactory class uses to determine which object to generate.

Snapshots

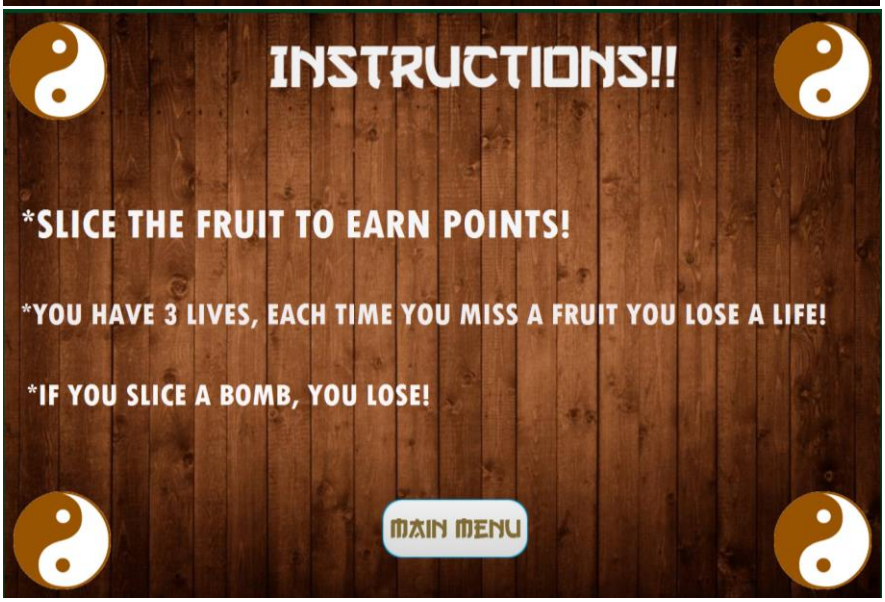
Splash Screen



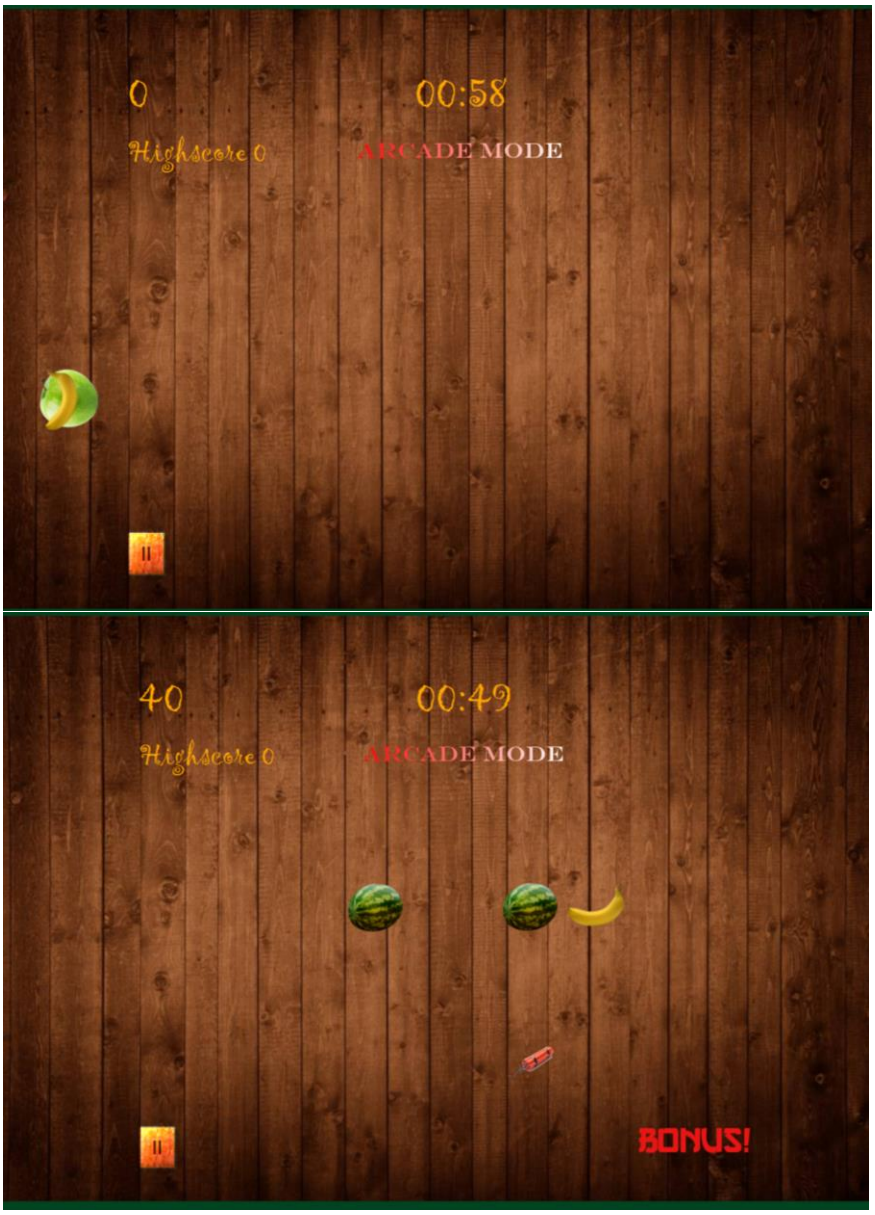
Choosing player's name, play as guest, delete user, or create new user

Main Menu. You can pause theme song here.

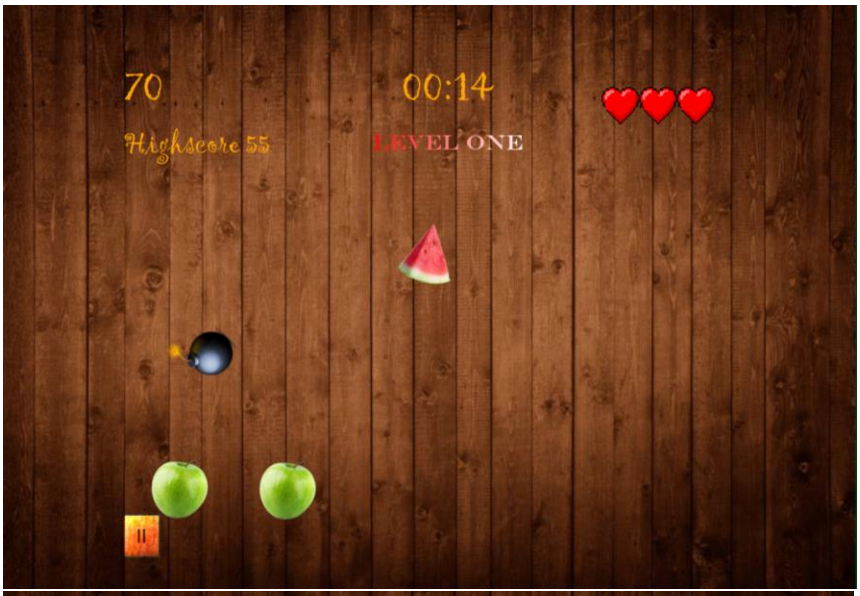
Leaderboard



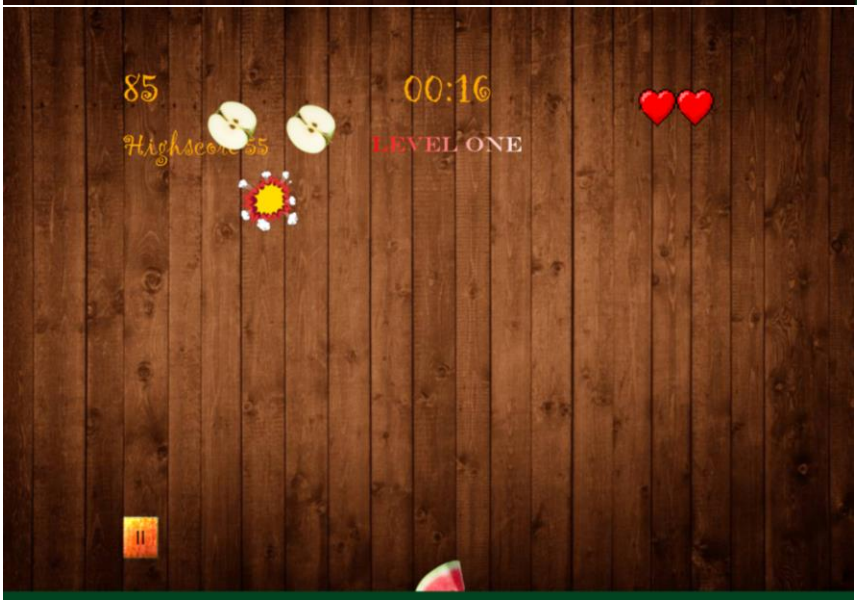
Help menu



Arcade Mode game screen

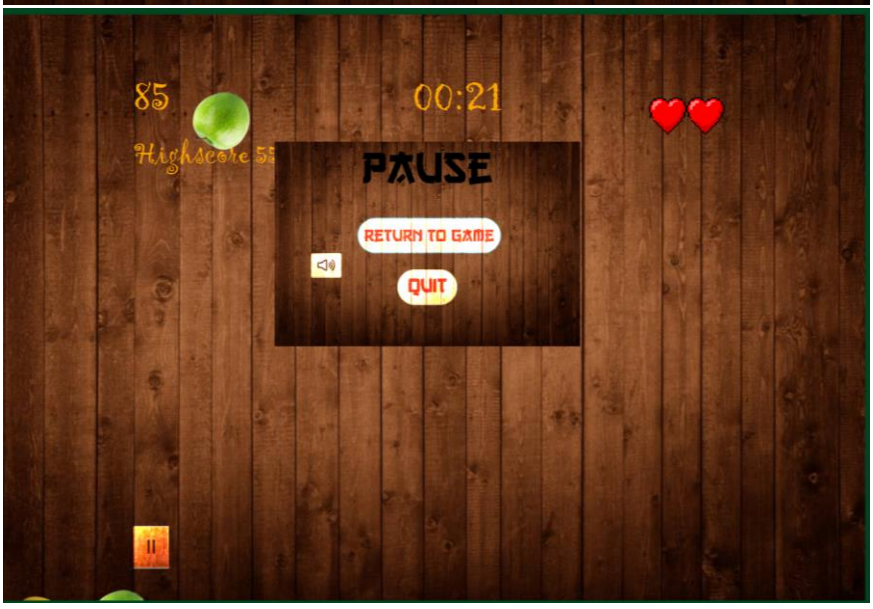


Classic Mode game screen



Losing a life by detonating a bomb or missing a fruit

Game Over when you detonate a fatal bomb, lose all your lives, or if time runs out.



Pause menu. You can mute theme song here too.

Quit confirmation menu

User Guide

Starting the Game

When you click the game, a loading screen will appear and it will take a few seconds to start the game. You will find a screen where you can either choose a preexisting player, create a new one, or play as a guest. You might also want to delete users if you wish.

Main Menu

In the main menu you will find the Leaderboards button, where you will be able to see who has the highest score in each category: level one classic mode, level two classic mode, level three classic mode, and arcade mode.

You will also find the Change Player button which will transport you to the previous screen where you can switch players.

The help button contains some general instructions for the game and the quit button lets you exit the game.

On the left you will find the sound button which can be used to switch the sound on and off.

To start playing you may choose one of two modes: arcade mode, or classic mode.

Classic Mode

Start playing by swiping over the fruits to slice them. Here there is no time limit, **however** if you miss a fruit you will lose a life. If you lose all lives, it's game over. If you slice a dangerous bomb (the round one) you will lose a life and 5 points off your score. If you slice a fatal bomb (the dynamite one) you lose all your lives and it's game over. Sometimes you will come across a bonus fruit which will give you double the score.

Dangerous Bomb

Arcade Mode

Here you have 60 seconds to slice as many fruits as you can. There are no lives and you do not get taxed if you miss a fruit. However, if you slice a dangerous bomb, you will still lose 5 points, and if you slice a fatal bomb you lose immediately. The bonus fruits may also appear here.

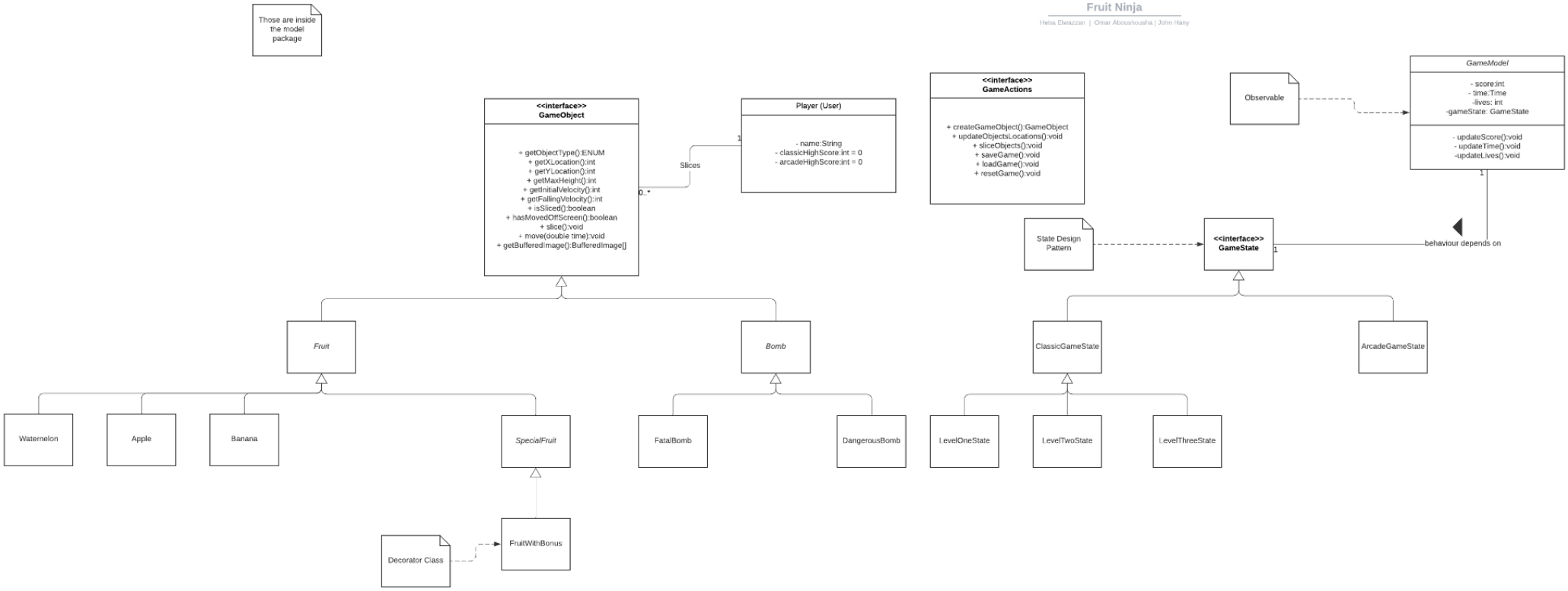
Fatal Bomb

Pause Menu

If you wish to pause the game you can click on the pause icon on the bottom left corner of the screen. Here you can either go back to the main menu, resume playing, or pause the music. WARNING: if you quit mid-game, your score will NOT be saved.

Note if you exit the game suddenly from the system without clicking quit, your data might not be saved, so be sure to close the game from inside.

UML Diagrams



Fruit Ninja

Heba Elwazzan | Omar Aboushousha | John Hany