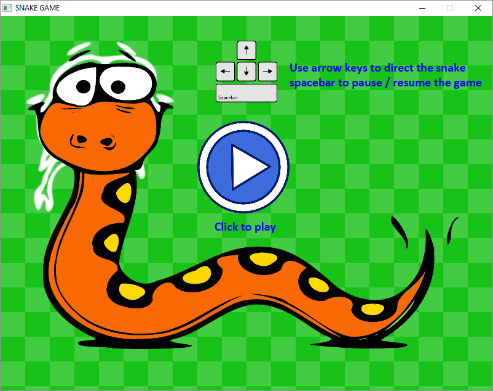
**THE SNAKE GAME**



**FINAL PROJECT**

**COURSE:** COIS 2240H

Professor: Alaadin Addas

Institution: Trent University

**GROUP MEMBERS:**

Fraser Raney

Shivam Parekh

Svyatoslav Loboda

Jiatao Wen

Date Published: April 6th, 2018

**TABLE OF CONTENTS**

**Software Overview**……………………………………………….......................................................................1

* Usage
* Functionality
* Importance

**Software Description to Industry Targeted**…………….................................................................….2

**Scope of Work (SOW)**……….………………….…....................................................................................2

* UML Diagrams/Description:
  + UML 1 (Rename)
  + UML 2 (Rename)
  + UML 3 (Rename)
  + UML 4 (Rename)

**Project Retrospective**.……………………………………...…........................................................................3

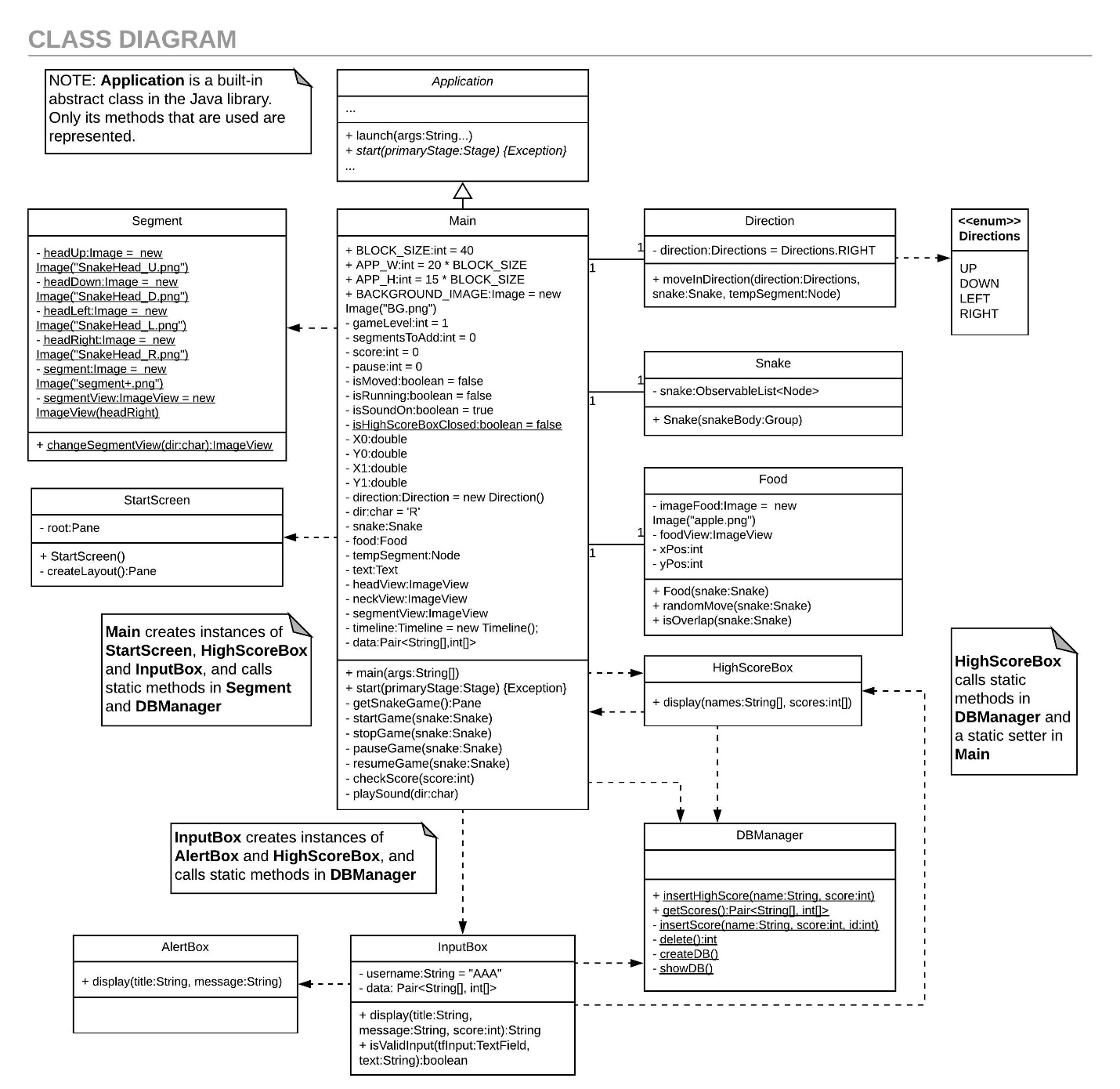
* What was the easiest to implement?
* What was the most difficult?
* What can be done differently?
* What have we learned?

**Software Overview**

**Software Description to Industry Targeted**

**Scope of Work (SOW)**

## Classes

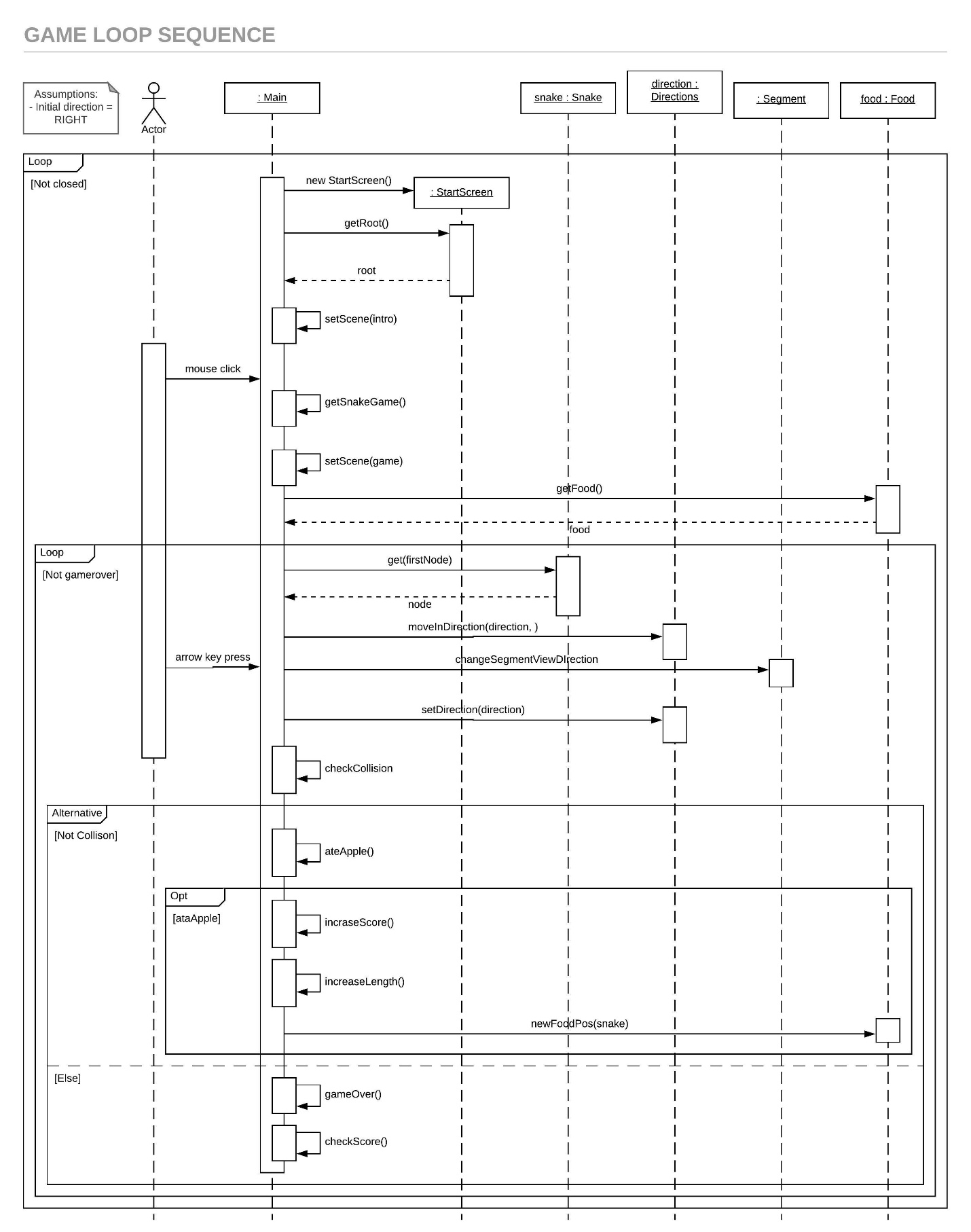


The Main class extends the built-in Application class in order to run the JavaFX program. Main handles all events and manages the game animation and game state. The Main class stores one instance of the Food class that it uses to render and track the position of the food in the game. Similarly, the Main class also stores an instance of the Snake class to track the snake in the game, however, its ImageViews are managed through the Segement class. The Main class also stores an instance of the Direction class which it uses to update the snake’s position according to key events.

The Main class also creates an instance of StartScreen in order to display the introductory scene in the game, and instances of HighScoreBox and InputBox in its checkScore method when the game ends.

Main, InputBox, and HighScoreBox require data from the scores database and access it through the methods of the DBManager class. The InputBox creates an instance of AlertBox during input validation.

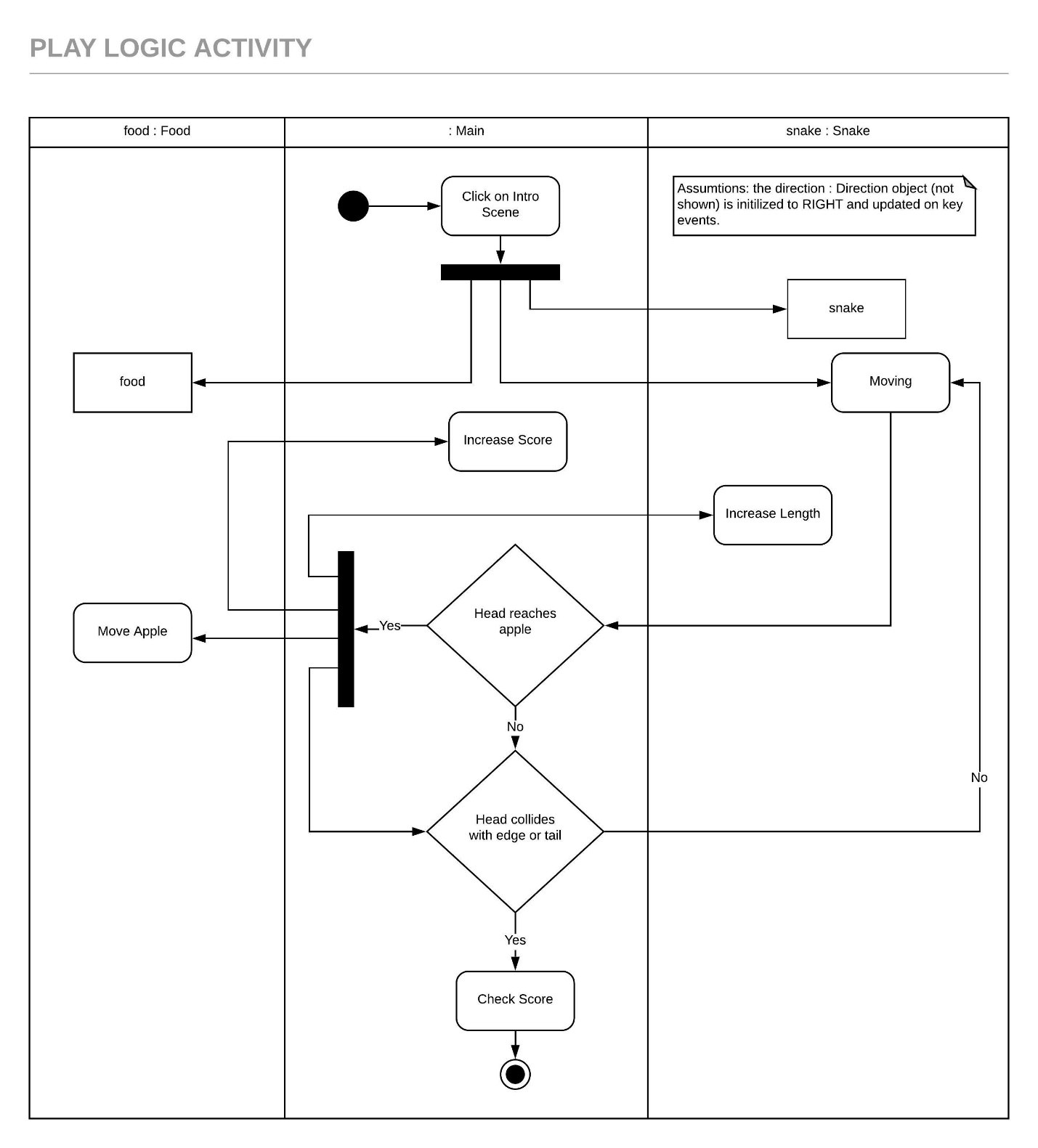
## Game Loop



The program is essentially two event driven loops. The fist loop consists of sequence of events once the game is opened and until the user closes the window. The introductory screen is shown, when player clicks the scene it switches to the game Scene, and when the game ends the Scene returns to the introductory screen.

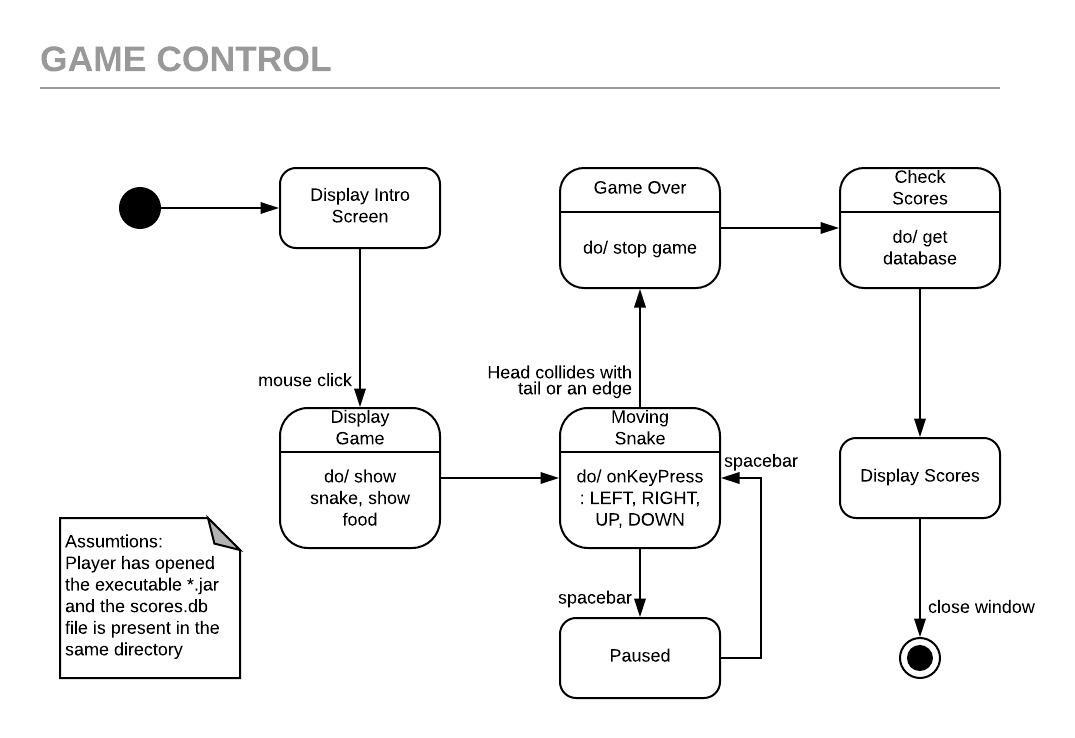
The game is the sequence of events in which the player interacts with the program through the snake object and the arrow keys (also spacebar for pause, not shown). The snake changes position on the screen according to the players selected direction and grows when the snake’s head reaches the food object’s positon. When the snake’s head and food collide the food changes positon. The game ends if the snake’s head collides with an edge of the screen or one of its body members.

## Play Logic Activity

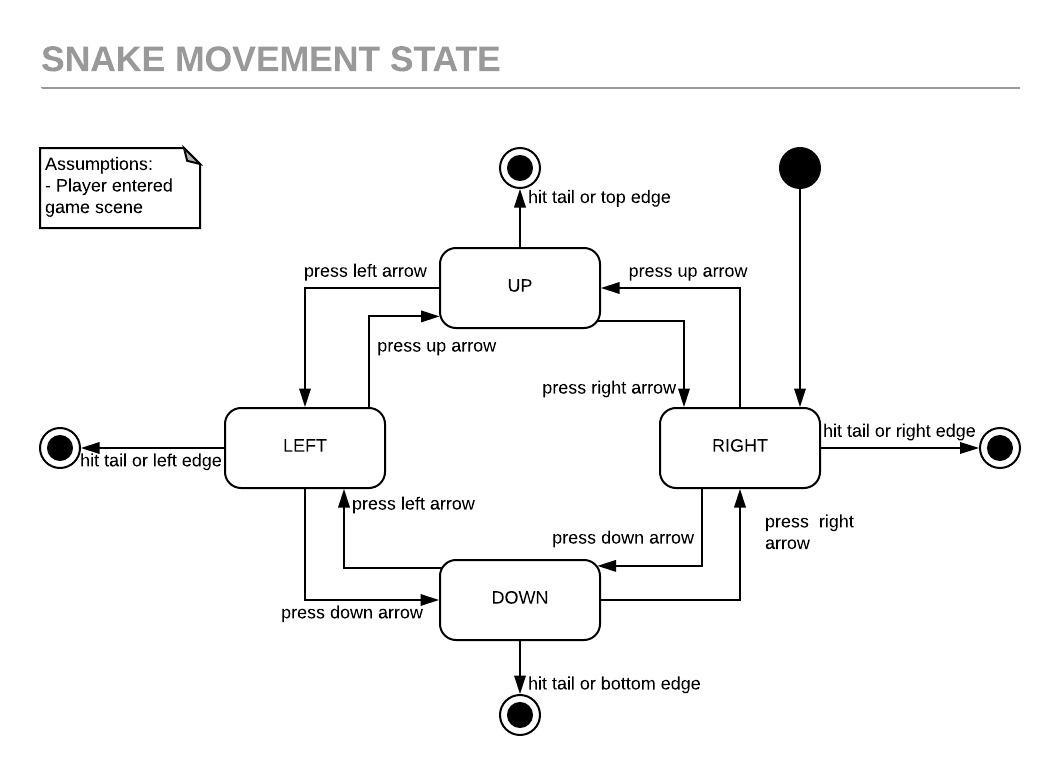


The program shows the introductory Scene initially and starts the game once the player clicks on the screen. The Main class instantiates the food and snake game objects and the snake moves on the screen. The player controls the movement (not shown). If the head reaches the food the score and length of the snake is increased. If the head collides with an edge of the window or one of its body members, the game ends and the score is checked against the high scores database (not shown).

## Control State

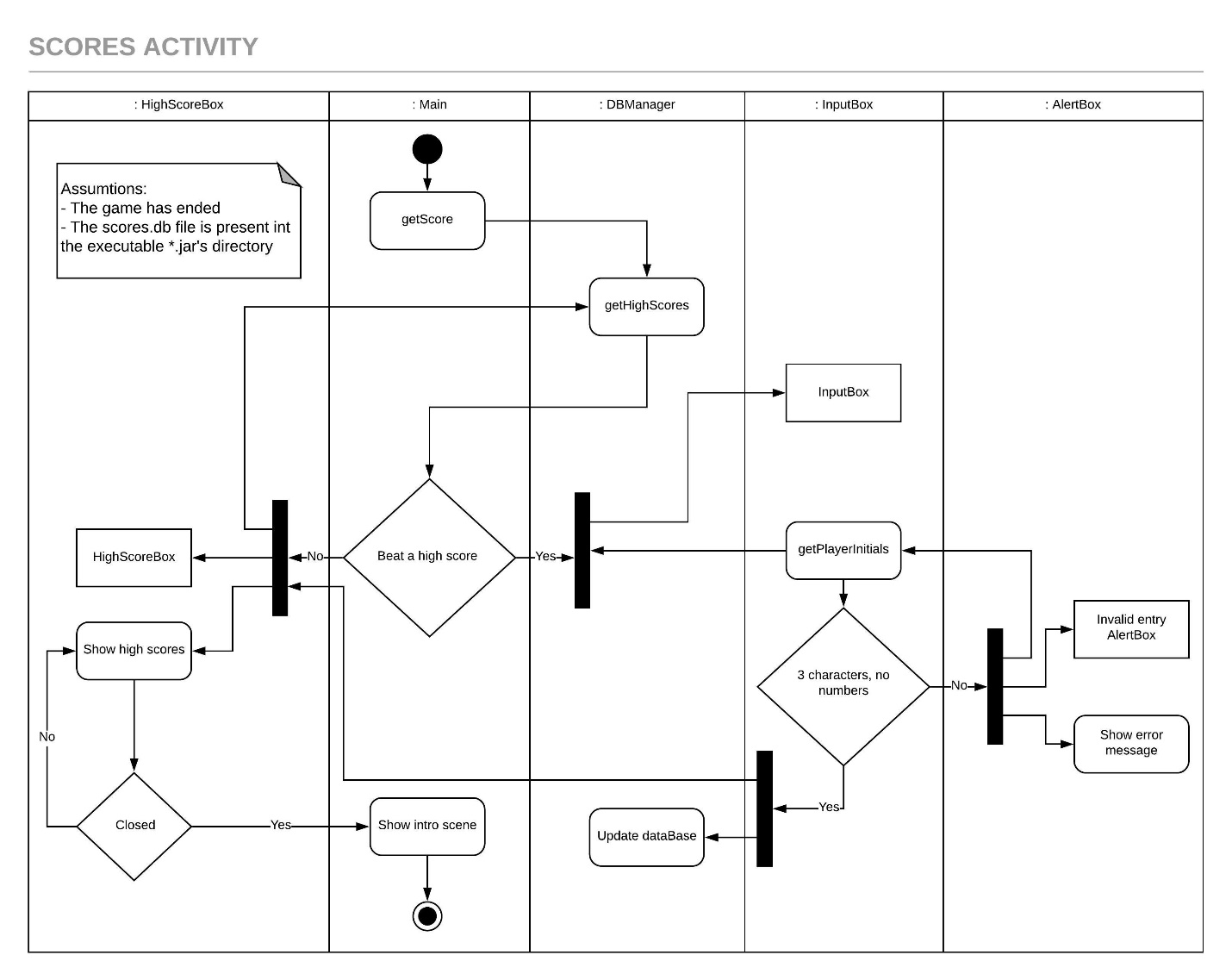


When the game window opens the introductory scene is displayed. When the player clicks on the scene, the scene switches to the game scene. The food and snake objects are shown and the snake is moving. When the player presses spacebar the game is paused, if the game is already paused, the game is resumed. The game ends if the snake collides with its tail or an edge of the window. The player’s score is check and the high scores are shown.



Since the game ends when the snake collides with one of its body members, the player’s choice of direction is limited to two choices at any point in the game. As an analogy, if we consider that the snake is always moving forward from the snake’s perspective, it can only choose to go left or right. That is the snake cannot move backwards. The analogy does not correspond to the arrow keys that player uses to control the game, it is only an explanation of the control logic.

## Scores Activity



When the game ends, the high scores are retrieved from the database file using the DBManager class and the player’s score is checked against the high scores. If the player has not beat a high score, the high scores are displayed in an instance of the HighScoreBox class. If the player beat a high score an instance of InputBox is created in order to get the player’s initials. If the player enters characters other than letters or less than 3 characters an error message is displayed in an instance of the AlertBox class. Once the player has successfully entered their name the database is updated using the DBManager class and an instance of HighScoreBox is create and the new high scores are shown. When the HighScoreBox is closed, the introductory scene is shown.

**Project Retrospective**

What was the easiest to implement?

The current version of the Snake game is based primarily on Svyatoslav’s efforts. We had made a graphical version of the game earlier, however, shortly after its development, we decided to go with Svyatoslav’s idea as the input handling was substantially better, that is, more-user friendly. The easiest part of the earlier game to develop was when Fraser once had his head around the Canvas object, which was the Food class. It is a very simple sprite that finds a random position and stores an image. As far as the final version, the simplest class that we contributed was the “AlertBox”, which is used for the input validation.

What was the most difficult?

Building the “DBManager” class was challenging. However, we were able to find a helpful tutorial and apply what was necessary to implement and manage the high score database for the game. The earlier version of the game had unresolved control issues where the user could “turn too quickly” by pressing three of the arrows rapidly causing the snake to fold back on itself in one game position. Also, the snake movement was slow to respond, that is, the player would have to choose the next direction slightly before the snake entered a game position.

What can be done differently?

Time was the biggest challenge in the project. We started early, but the development of the game took longer than expected. In retrospect, setting more realistic goals at the inception of the project would have been sensible; however, we are all pleased with the resulting game.

What have we learned?

JavaFX and JDBC represented huge learning curves, but the knowledge that we gained from the project is invaluable. Using GitHub to coordinate the project development was again something that was completely new to us and the experience of developing the game as part of a team was a great experience.