**Description of Current Version:**

A MVC model is implemented with the game GUI, controller and model to allow the game to be playable using the mouse. In addition, JUnit tests were added for the model logic testing if the expected results are achieved.

**Changes from Previous Version:**

Milestone 2 consists of reimplemented model with new classes such as Fox, rabbit, Piece, Mushroom, Command, CommandWord, Parser to enable more encapsulation in the model. JUnit tests were added and reimplementation of the game using MVC design pattern.

User Manual:

Run JumpInMVC.java in the JumpIn package, the game is played using the mouse by pressing the animal to be moved first then pressing on the desired location for the animal to be placed.

The obstacles are: Mushroom, Fox, Rabbit, Hole.

The following explains how the obstacles move around the board:

Rabbit:

Rabbits can only move by jumping over one adjacent obstacle, empty holes are NOT obstacles.

Once a rabbit is in a hole, it can be jumped over by other rabbits.

Side note: Rabbits can jump out of their holes to faciliate another rabbit's path.

Rabbits can jump over a fox's waist, or from its head to tail or tail to head.

Foxes:

Foxes can slide depending on their initial direction, however many spots needed.

Mushrooms and holes are stationary.

The objective of the game is to move the rabbits and foxes, through a series of movements

around the obstacles untill all the rabbits are safely in their hole.

In this game, the following abbreviations are used to represent each object on the board:

R1 - Rabbit 1

R2 - Rabbit 2

R3 - Rabbit 3

F1 - Fox 1

F2 - Fox 2

MS - Mushroom

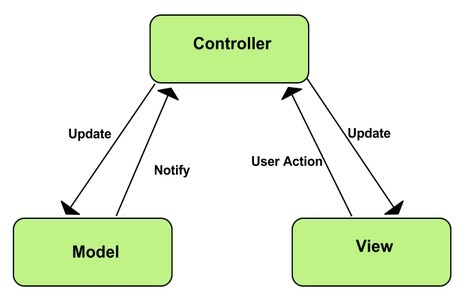
O - hole

The following are the list of menu options you can use:

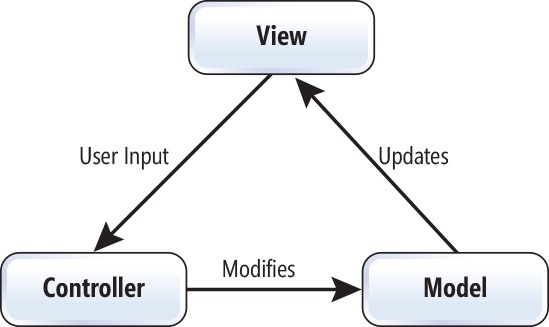
quit help

**Design Decisions:**

The team worked on two ways to implement the MVC design pattern for the game. The two ways are demonstrated in the figures below. The team decided to use the method in Figure 1 since it was found to be simpler and the other way is in the HassanController branch. Also, for the model classes were made for most of the pieces on the board to help differentiate between them efficiently, Command, CommandWord and Parser class were implemented to get commands from the user using the GameView(Graphical User Interface) or the console.



Figure



Figure

**Known Issues:**

None.

**Team memeber roles:**

Mariam Almalki: Reimplemented the model, worked on connecting the model, view and controller

Nazifa Tanzim: Worked on JUnit testing, helped in reimplementation of the model

Taher Shabaan: Deisgned the View (GUI), Worked on JUint testing

Hassan Hassan: Worked on connecting the model, view and controller, Worked on JUnit testing

Abdulla Al-wazzan: Did not help the team with Milestone 2

**Things to Note:**

1.The design not used for connecting the MVC together was worked on in the HassanController branch.

2.Images are in bin folder. sometimes they disappear depending on what files are being tracked.

**Roadmap Ahead**

Add Undo/Redo features to the game to help the user get back to the initial position of the pieces on the board, as well as, helping the user solve the game giving hints on possible moves to solve the game. Also, adding Save/Load features using a database to the game to allow the user to save the positions of the pieces in the game and come back to it later to continue playing the game.