**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.

The documentation for this version include the sequence diagrams, an updated UML, CRC cards, design decisions, team member roles and responsibilities, a user manual , and the road map ahead.

**Changes from Previous Version:**

The feedback from milestone 1 was implemented and the changes can be found on branch “Milestone1V2”

Milestone 2 consists of reimplemented model with new classes such as Fox, rabbit, Piece, Mushroom, Command, CommandWord, Parser to enable more encapsulation and decoupling in the model. JUnit tests were added and reimplementation of the game using MVC design pattern. A sequence diagram was created for milestone 1.

User Manual:

The main class for this version of the project is called 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 following are the list of menu options you can use:

quit help

quit will end the game. Help will display the following instructions:

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 facilitate 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 until all the rabbits are safely in their hole.

**Design Decisions:**

Throughout the research process for this milestone, the team found two different ways to implement the MVC pattern. The first way is shown in figure 1 and the second is shown in figure 2. The team worked on both ways in order to see which is best suited for the task. The team decided to use the method in Figure 1 since it was found to be simpler. The implementation of the second way can be found in the HassanController branch.

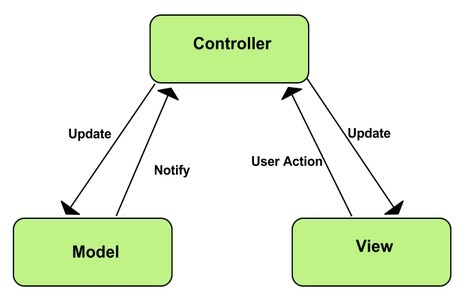


Figure 1

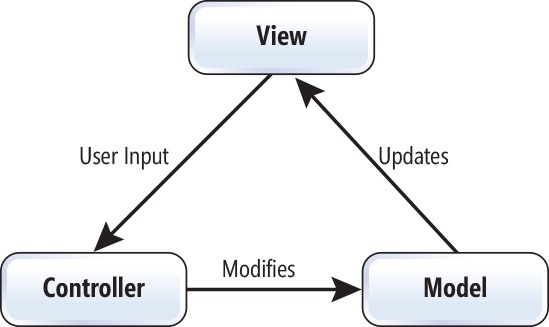


Figure 2

It was decided to pass the model to the view so that the view can be constructed based on the objects in the model. This makes the code more cohesive and the classes work together, and so that nothing exists in the model that doesn’t exist in the view or vice versa.

Also, for the model, classes were made for most of the pieces on the board to reduce the smell of the code and help differentiate between the different components of the code. Command, CommandWord and Parser class were implemented to get commands from the user using the GameView(Graphical User Interface) or the console. This decision was based on the feedback received for Milestone 1.

We decided to change the implementation of Fox such that a fox object can represent either a fox’s head or a fox’s tail, and each object is associated with it’s corresponding piece. This implementation proved to be better because then each part can be mapped to its appropriate image.

Similarly, we added a colour attribute to Rabbits, so that the right image can be used for each.

**Known Issues:**

None, although potentially: images disappearing from bin folder. The images used in the GUI are located in the project bin folder and often need to be re-added to that folder after pulling the project.

**Team member roles:**

Mariam Almalki: Added command classes and functionality to the code, reimplemented the model to work with MVC, worked on connecting the model, view and controller. Created sequence diagram for Milestone 2 and helped compose design decisions.

Nazifa Tanzim: Worked on JUnit testing, split old gameEngine class to create a piece abstract class from which foxes, rabbits, and mushrooms can be made. Worked on reimplementing the model. Created CRC cards and helped with final testing of Milestone2.

Taher Shabaan: Designed and implemented the look of the View (GUI), wrote JUint tests for various classes. Created the UML diagram for milestone 2.

Hassan Hassan: Worked on connecting the model, view and controller, Worked on JUnit testing. Composed all other documentation elements for milestone 2.

Abdulla Al-wazzan: Unfortunately did not help the team with Milestone 2, although he did create the originally missing sequence diagram for milestone 1.

**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.