Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A21

Game MVC

Team:

Joshua Fearnall - Id: [041019251]

NumPuz Proposal

***This template is suggested (not mandatory) to answer A21 Specification.***

|  |  |
| --- | --- |
| **Part**  **1** | **GUI Definition** |

* 1. **MVC Details**

*Describe the way you can define the MVC components in your game.*

Model: GameModel (Contains data required for the game e.g. time or current moves)

View: GameView (Contains the definition of swing objects, extends JFrame)

Controller: GameController (Contains logic for actions that the player can take in the game)

MVC Components will be instantiated in the main method by calling functions to create objects for each

For example, in the main class:

GameModel model = new GameModel();

GameView view = new GameView();

GameController controller = new GameController(model, view); // since the controller needs to access both

**Example** (from vision “top-down”)

Class: JFrame – Object: “GameFrame”

→ Class: JPanel → Object: “GameuBoard”

→ Class: JButtons → Objects: “BSave”, “BLoad”, etc.

→ Class: JLabel → Objects: “LabOperation”, “LabName”, etc.

…

* 1. **View Component**

*Describe how your interface should be organized using new components. Show the idea about your “top-down” organization.*

*Cont…*

Class JFrame : Object: “gameController”

JDialog : Object : “loadScreenDialog”

JMenuBar : Object : “menuBar”

JMenu : Object : “gameMenu”

JMenuItem : Objects : “newGameItem”, “saveItem”, “loadItem”

JMenu : Object : “optionsMenu”

JMenuItem : Objects : “colorItem”

JPanel : Object : “topPanel”

JPanel : Object : “leftPanel”

JPanel : Object : “centerPanel”

JButton : Object : “buttonGrid” // Contains N\* N buttons

JPanel : Object : “rightPanel”

JPanel : Object : “bottomPanel”

JButton : Object : “logoButton”

JDialog : Object : “aboutDialog” // Opened by the logo button

JLabel : Object : “aboutImage”

JButton : Object : “closeButton”

JLabel : Object : “editorLabel” # 1 Visible during edit mode

JTextField : Object : “editorTextField” # 1

JButton : Object : “editorButton” # 1

JLabel : Object : “pointsLabel” $ 2 Visible during play mode

JLabel : Object : “movesLabel” $ 2

JLabel : Object : “timeLabel” $ 2

* + - ***Example****:*

**Example** (from vision “top-down”)

Class: JFrame – Object: “GameFrame”

→ Class: JPanel → Object: “GameuBoard”

→ Class: JButtons → Objects: “BSave”, “BLoad”, etc.

→ Class: JLabel → Objects: “LabOperation”, “LabName”, etc.

…

* ***Note****: The professor interface continues being a proposal. Focus on your ideas using the best user experience.*
  1. **Controller Component**

*Describe aspects of your controller using, for example, one unique action command. Create the “map” to define functions with actions.*

Object: “newGameItem” – Event: actionPerformed > method: NewGameWindow()

Object: “saveItem” – Event: actionPerformed > method: saveDesign()

Object: “loadItem” – Event: actionPerformed > method: loadDesign()

Object: “colorItem” – Event: actionPerformed > method: chooseColor()

Object: “editorButton” – Event: actionPerformed > method: setSolution()

Object: “buttonGrid” – Event: actionPerformed > method: moveTile()

Object: “logoButton” – Event: actionPerformed > method: openAbout()

Object: “closeButton” – Event: actionPerformed > aboutDialog.dispose()

**Example**

Object: “BSave”

→ Event: actionPerformed → method: saveGame()

Etc.

* 1. **Model Component**

*Finally, what is your idea to define the model to be used in a “default” (randomized) game.*

Default settings would be:

Dimension: 3. solutionData: { {1,2,3}, {4,5,6} {7,8,} }

Upon starting the game, we would initialize the board through the constructor in the GameController; passing data from the model to the view, then shuffling all of the tiles.

User data: Points, Time, Name, and High Score.

int : Dimension : loadDesign()

int[][] : solutionData : loadDesign() // solution saved as a 2d array of integers

int : points : 0 // default for points is 0

float : time : 60 // might be a different value later, maybe dimension^2 \* 5 ?

int : highScore : loadSettings() // this value is set when the user gets a new score after finishing a game, // maybe with a method like updateHighScore()?

String : name : loadSettings() // not exactly sure where this will be set, when a new game is started?

**Example**

Data structure used:

→ Values: gridValue → method: updateData()

|  |  |
| --- | --- |
| **Part**  **2** | **Implementation Design** |

* 1. **Game Evolution**
  + *Considering this new model, explain:*
    - *What are the differences between the original proposal (A11) and the current project to be developed (A21).*
    - *If so, explain why you need to do some adjustments.*

The largest difference is the addition of the MVC pattern and color feature. The color feature will need its own button, but that’s okay because I have a modular menu already, whereas the MVC pattern will require reworking of a few details to make sure each component communicates properly with every other component. I haven’t implemented anything that uses data yet, so changing over to MVC should be easy as I can define the model without worrying about breaking anything in the program.

The splash screen is another feature I hadn’t planned for, but it should be easy to implement as a dialog before launching the main part of the game.

* 1. **Others DP**
     + *Define (at least one) additional DP that you could use in your Game application.*
  + *Explain what is this DP and the reason why it could be recommended.*

We could use the Command Pattern.

The command pattern allows us to bind user input to a given command rather than directly to an action, so that rather than directly moving a tile on the board we would call an action that gives a movement command to the tile.

This would help us implement an undo/redo function to the game. If we store information about the state of the game when the move command is called, we could easily reverse or redo previous actions.

**References**

*[Include eventual references used here]*

Algonquin College

Spring / Summer, 2022