MINI-PROJECT REPORT

**Group 11:**

**I. Topic:** Construction a simple game.You can freely add/change any features.

Game : Dots & Boxes.

Rule : Players take turns joining two horizontally or vertically adjacent dots by a line. A player that completes the fourth side of a square (a box) colors that box and must play again. When all boxes. have. been colored, the game ends and the player who has. colored more boxes wins.

**II. Mini-project description:**

- Create a game application about ‘Dots & Boxes’.

- On the screen application, player click one of the buttons ‘Play’, ‘Quit’ or ‘Guide’.

+ Quit : out the application.

+ Guide : read the rule of game.

+ Play : play the game.

- After click ‘Guide’, it will show the screen which has guide of the game.

- After click ‘Quit’, you will exit the application.

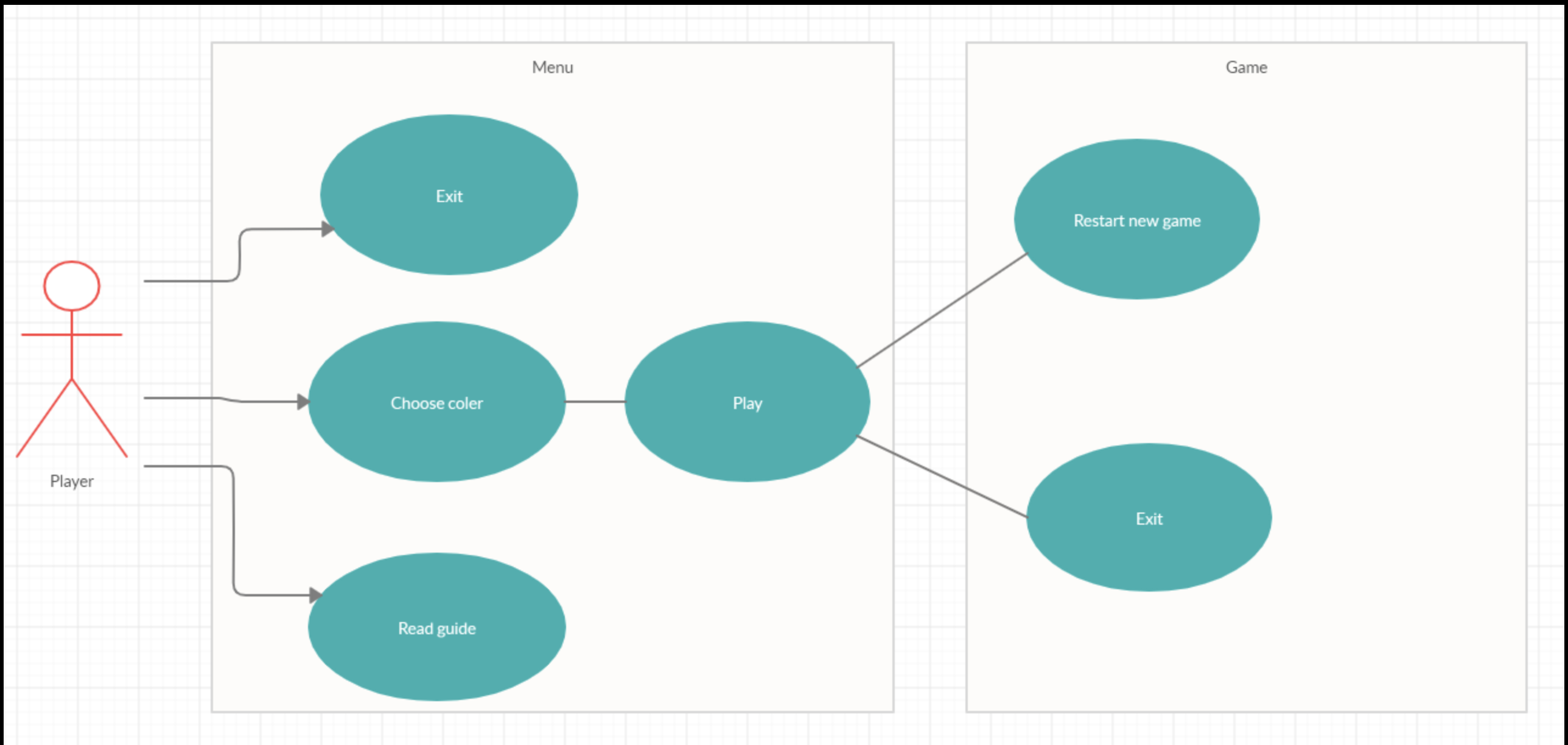
- After click ‘Play’ , it will show the screen to choose color of dots in game.

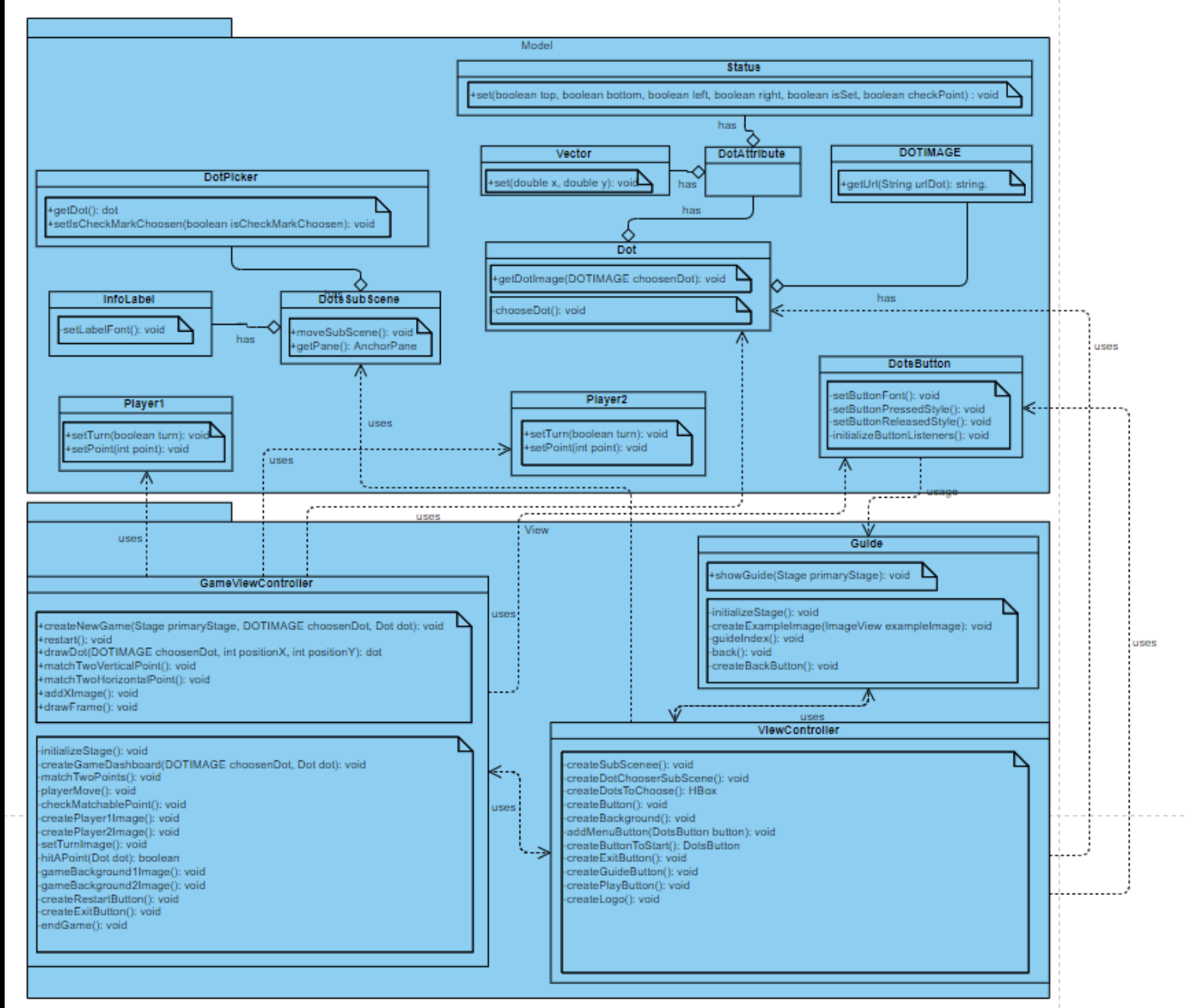
- After choose the color, show the screen to play game includes:

+ The center place is the dots in order to play game.

+ The left & right place show the signal (X-O) , point and turn of each player.

- When playing game, we can ‘Restart’ or ‘Quit’ the game by the corresponding buttons.

**III. Explanation of the design ideas****:**



**IV. Assignment of members :**

\* **Nguyen Hong Phuong: Design object.**

Classes usage:

1. DOTIMAGE: enum class. Use to define the image of dots ( 2 colors to choose: green and blue).

- getUrl(): function to get the url of image.

2. DotAttribute: class to define the attribute of dots. There are 2 attributes: position and status. Get those attribute from class Vector and Status.

3. Vector: define the position (x, y) of dots.

4. Status: define status of dots. There are 3 main status: direction, isSet, checkPoint.

4.1: direction(boolean):

A dot has 4 direction: top, bottom, left, right.

4.2: isSet(boolean):

To check if a dot is select by player.

4.3: checkPoint(int):

After player get a point, the upper left corner dot’s checkPoint attribute will be set to 1 if player1 hit that point, 2 if player2 hit that point.

5. DotsAction: An abstract class to define the method chooseDot() of object Dot.

6. Dot: design the dot of the game.

The dot will get image from class DOTIMAGE and attribute from class DotAttribute. We also define action of the dot in function chooseDot().

7. DotsButton: design button of the game. The font of button will set in setButtonFont(), the action will set in setButtonPressedStyle() and setButtonReleaseStyle() and initializeButtonListeners.

8. DotPicker: set the checkmark while player choose color of dots.

9. DotsSubScene: design the sub scene of main menu. The sub scene will set default outside of the game screen, and move into screen if player choose “Play”. It will move out of the screen if player choose “Play” again.

10. InfoLable: The title “CHOOSE DOT COLOR!!!” and its background in the sub scene.

11. Player1: An abstract class to define the method setTurn() and setPoint() of object Player 1 and Player 2.

12. Player1 and Player2: design player1 and player2. The ảe tưo attribute: turn and point.

13. View: parent class of 3 view controller classes. Create static attributes and characteristic.

\* **Nguyen Thi Thanh Hoai: menu + guide view controller**

1. MenuViewController: When the program running, the main function will get the parentPane in ViewController to parent pane, so this pane will shown to the screen while the program start.

There pane will show:

1.1: Logo. Display to the screen the logo of game by function createLogo().

1.2: Button. There are 3 button in the screen: Exit, Guide and Play.

Exit: use method .close() to exit the program.

Guide: Display the guide scene while player click on Guide button.

Play: When player click Play, a sub scene will shown for player to choose the color of dots. Player must choose the color before play game.

Methods in ViewController:

+createLogo(). Create the logo image.

+createPlayButton(): Create the play button.

+createGuideButton(): Create the guide button.

+createExitButton(): Create the exit button.

+createButtonToStarT(): Create the button to start in sub scene.

+addMenuButton(): add the button in mainPane to an arrayList to use operation to design the position of the button.

+createBackground(): create the main menu background.

+createButton(): create all button.

+createDotChooserSubScene(): create all element in sub scene.

2. Guide: A scene to display the guide of the game.

+guideIndex(): show the text of the guide.

+createBackButton(): create the back button.

+back(): The action while click back button.

+createExampleImage: create the example image.

+showGuide: hide the primaryStage and show the Guide stage.

**\* Le Tuan : Logic & GameViewController**

GameViewController: Logic and Display of game board.

* Design the logic in game:

If player select a dot, the attribute isSet of that dot will set to !isSet, so the program can check if the dot is selected or not.

A dot will have 4 direction: top, left, bottom, right. It also have position (x, y) attribute. We use position attribute to identify two adjacent dot. Player can just match to dot if they are adjacent.

If two adjacent dots is selected, it will be matched. After an action of player, the frame will be clear and draw again. For the compiler know if two dots is matched, we use the direction. For example, if two dots is matched (fig 1.1), the attribute right of the left dot will set to true, and the attribute left of the right dot is also set to true.

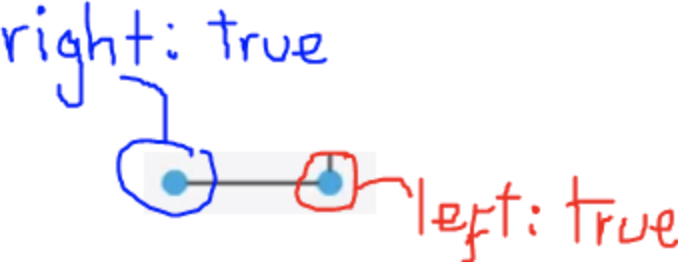


Figure 1.1

We will create 40 dots, which will saved in an arrayList name dotList. Each dot is an element of dotList, and we use that to check the logic of the game.

A player hit a point when he match a square, which is made by 4 dots. And we will check the score of player by saving it to the attribute checkPoint of the top left corner dot. For example, if player hit a point, the attribute of the circled dot (we call that is dot1 for example) in figure 1.2 will be:

dot1.attribute.status.checkPoint = 1 if player1 hit that point. And dot1.attribute.status.checkPoint = 2 if player2 hit that point.

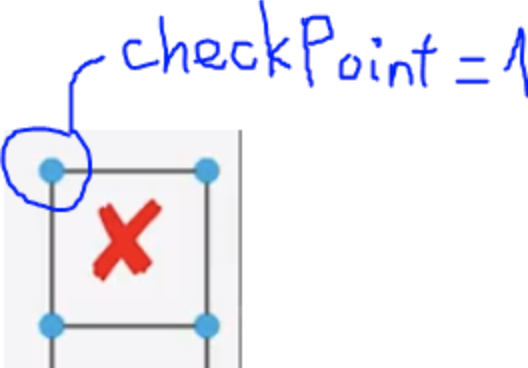


Figure 1.2

Check if player hit a point: we will check 4 dots if those direction attribute is satisfy that:

dot1.right && dot1.bottom == true.

dot2.left && dot2.bottom == true.

[dot3.top](http://dot3.top/) && dot3.right == true.

[dot4.top](http://dot4.top/) && dot4.left ==true.

You can see it clearly in figure 1.3.

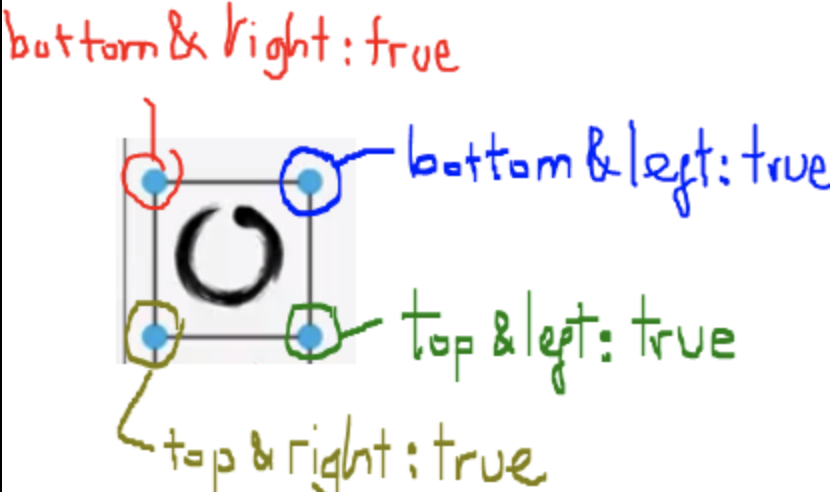


Figure 1.3

Turn: if player complete their move and do not hit any point, the turn will be switched.

End game: The game will end while total point of two player is equal or more than 28.

* Function usage:

+createNewGame(): create new game. Will hide the main menu scene and show the game scene.

+restart(): restart the game. Set player turn and point to default.

+createGameDashboard(): use for loop 2 times to create 5x8 dots table (equal to 40 dots).

+drawDot(): draw the dot. createGameDashboard will use this function to draw dot to the screen.

+matchTwoVerticalPoint() and matchTwoHorizontalPoint(): use vertical and horizontal image to match two valid point that player select.

+hitAPoint(): check if player hit a point.

+addXImage() and addYImage(): draw the image X if player1 hit points or draw the image O if player2 hit points.

+drawFrame(): clear and draw image of objects again.

+matchTwoPoints(): check valid point and use matchTwoVerticalPoint, matchTwoHorizontalPoint to match point.

+playerMove(): check action of dot.

+checkMatchablePoint(): check if two dots can match together.

+createPlayer1Image() and createPlayer2Image(): create two image of player.

+setTurnImage(): show the image and animation while player switch turn.

+switchTurn(): switch player’s turn.

+checkPoint(): calculate points off player1 and player2.

+gameBackground1Image() and gameBackground2Image: create two background image of player1 and player2.

+createRestartButton(): create the restart button.

+createExitButton(): create the exit button.

+endGame(): The effect while the game is ended.