Project Report

(MatchSweeper)

Created by

Chanathip Kittikaiwan 6431310221

Poonyawee Wongsrisujarit 6431330821

2110215 Programming Methodology

Semester 2 Year 2021

Chulalongkorn University

MatchSweeper

Introduction

MatchSweeper is the combination of Matching game and Minesweeper game. The objective is to match all cards and avoid trapping the mine to reach the highest score with the least move.

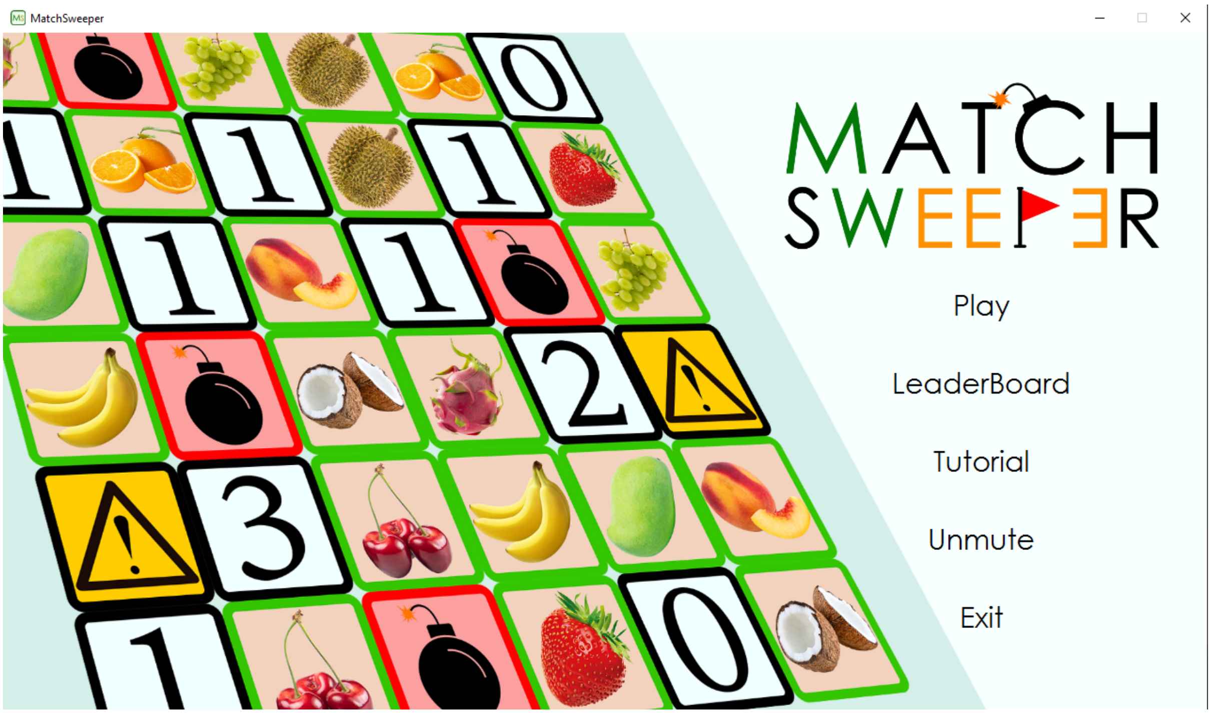
Rules

Players need to flip up the card to memorize the image of each card and match them by the same image. By the way, the game isn’t easy as there are some landmines that infiltrate with the card. Players need to avoid traps on landmines to get the highest score with the least move count and time. When the card is flipped down, it will show the number of landmines around them, not including itself. You can mark the card by right clicking on which you suspect it will be a landmine to prevent from wrong click, also you can’t mark the card that once flipped up.

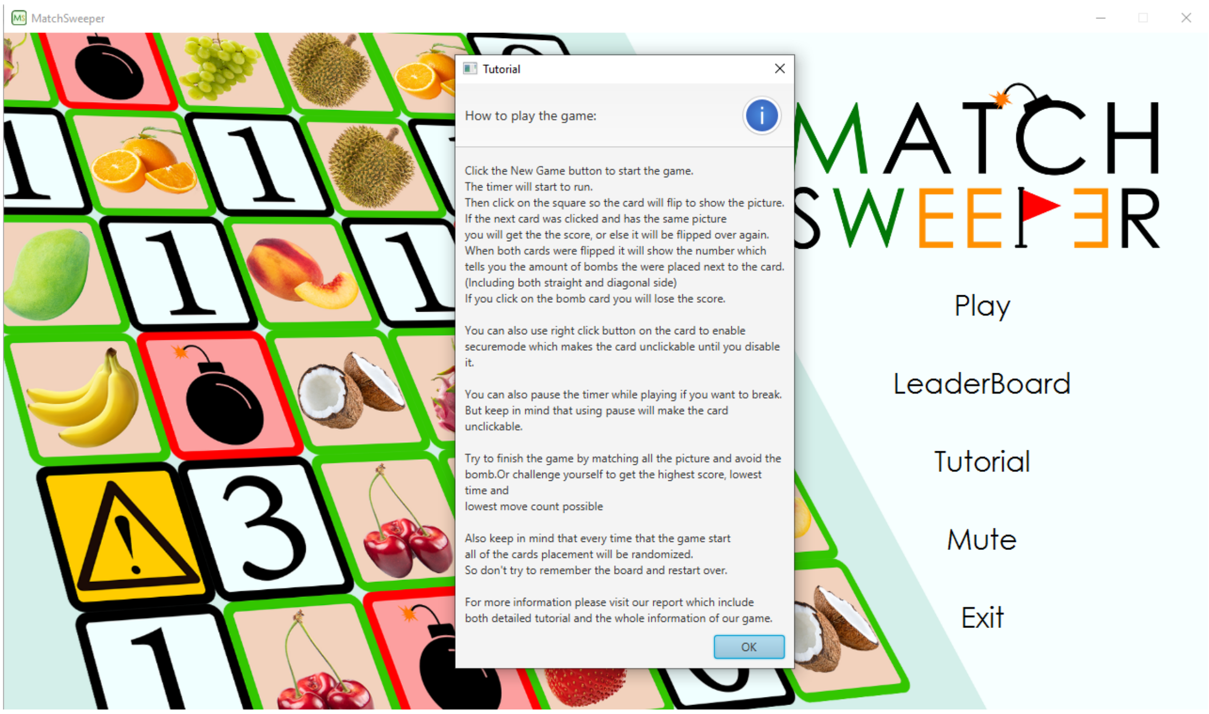
The game will end after you match all the card and mark all the bomb that left on the board.

Example

1.1 Main menu scene



There are five menus on the main menu. First, “Play” menu will switch the scene to a game. Second, “LeaderBoard” menu will switch the scene to leaderboard.



Tutorial

How to play the game:

Click the New Game button to start the game.
The timer will start to run.
Then click on the square so the card will flip to show the picture.
If the next card was clicked and has the same picture
you will get the the score, or else it will be flipped over again.
When both cards were flipped it will show the number which
tells you the amount of bombs the were placed next to the card.
(Including both straight and diagonal side)
If you click on the bomb card you will lose the score.

You can also use right click button on the card to enable securemode which makes the card unclickable until you disable it.

You can also pause the timer while playing if you want to break.
But keep in mind that using pause will make the card unclickable.

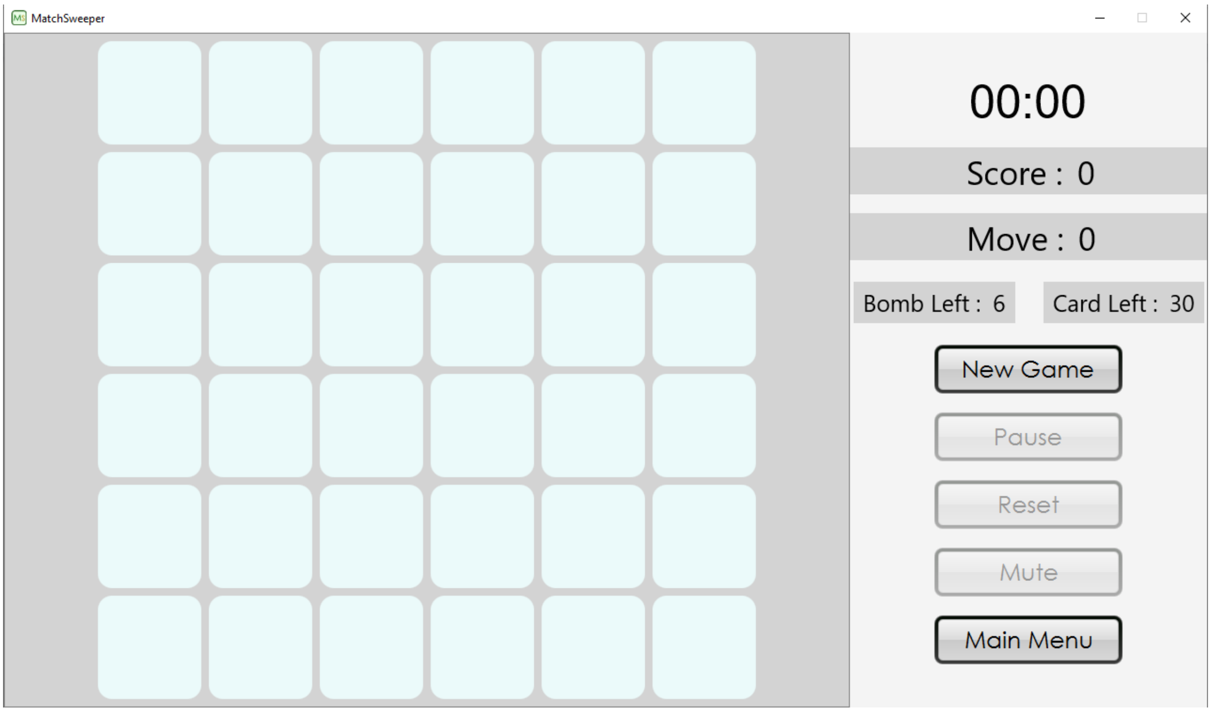
Try to finish the game by matching all the picture and avoid the bomb.Or challenge yourself to get the highest score, lowest time and
lowest move count possible

Also keep in mind that every time that the game start
all of the cards placement will be randomized.
So don't try to remember the board and restart over.

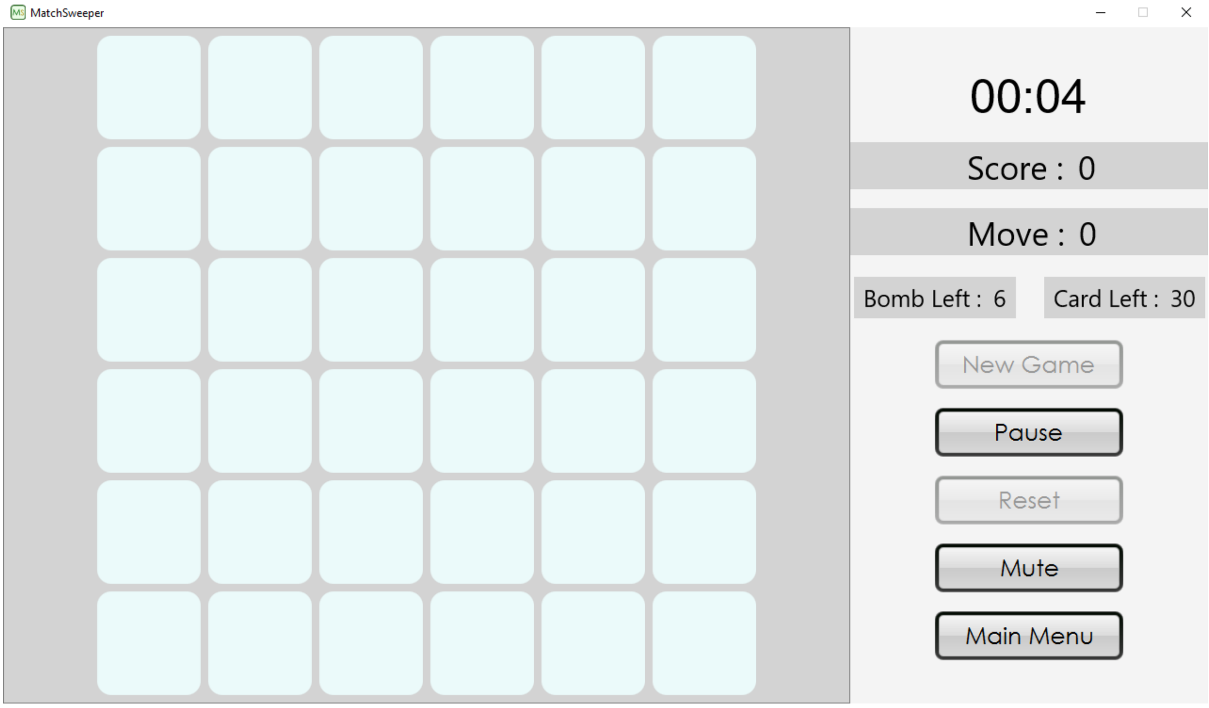
For more information please visit our report which include both detailed tutorial and the whole information of our game.

Third, “Tutorial” menu will show the tutorial like in the picture. Fourth, “Mute/Unmute” menu will mute or unmute BGM. Fifth, “Exit” menu will exit the game.

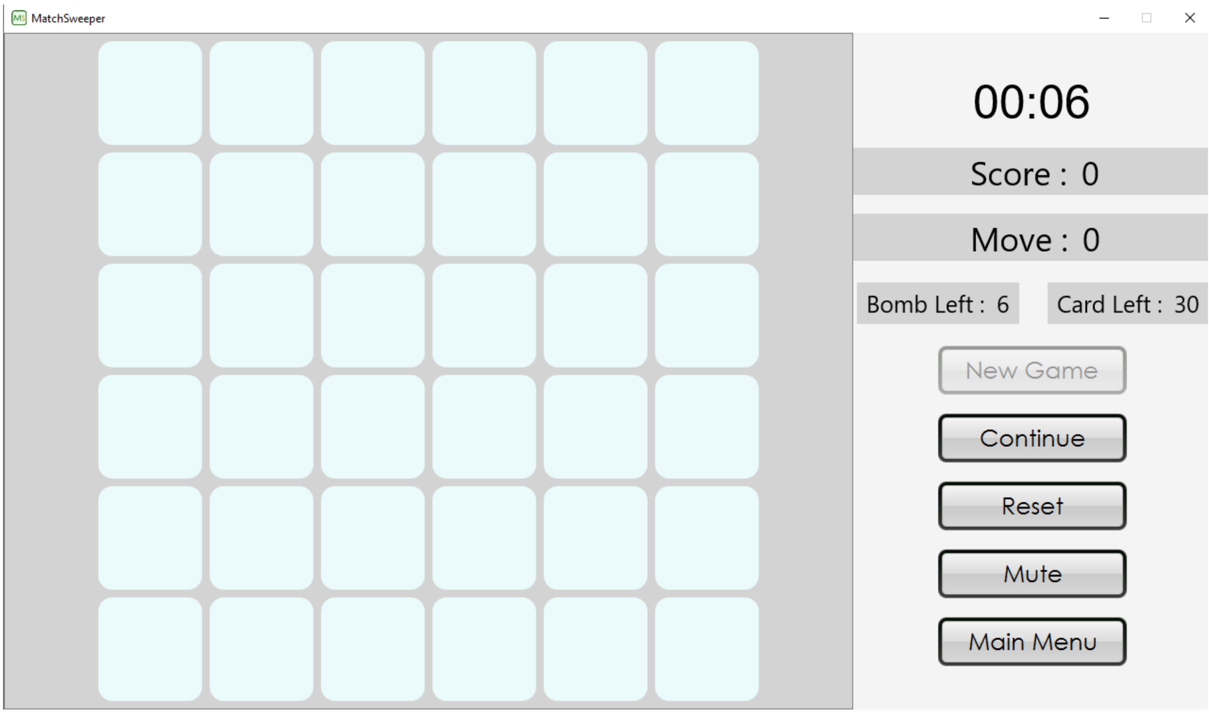
1.2 GamePlay Scene



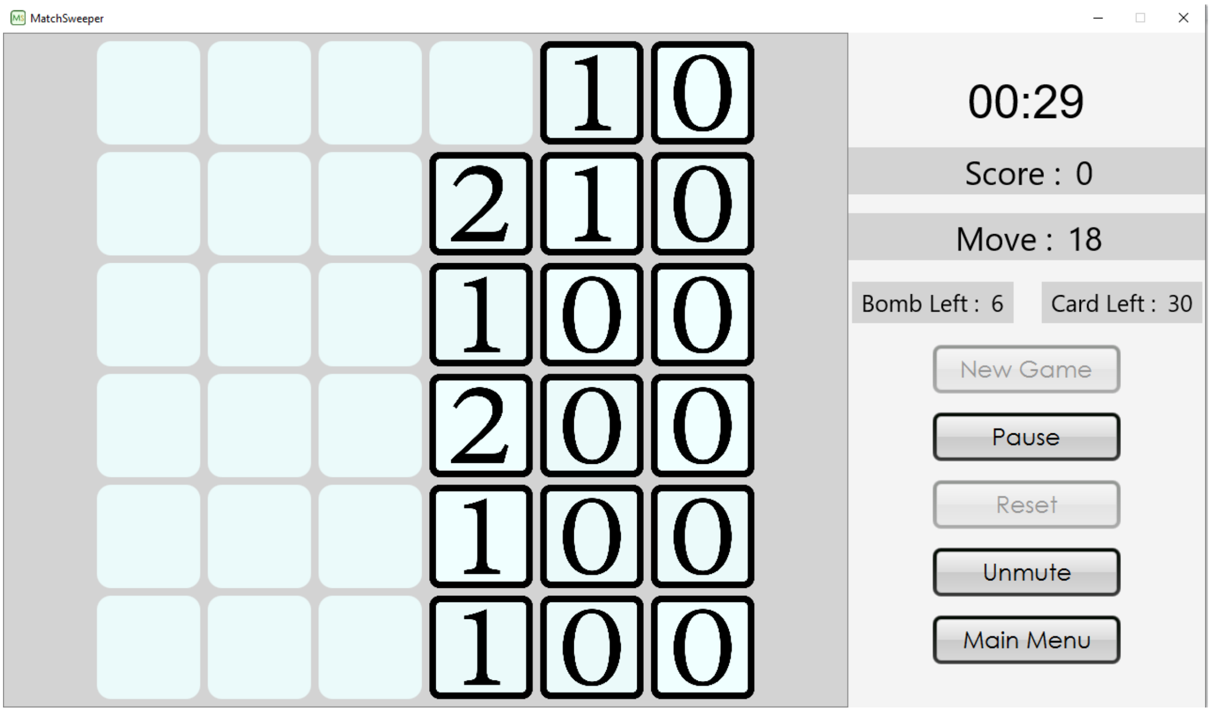
Players can click on the “New Game” button to start the game.



When the game start the timer will start counting and now players can interact with the card



Players can pause the game by clicking on the “Pause” button. After the pause players can continue by clicking on the “Continue” button but if players want to reset the game players can click on “Reset” Button. Also players can mute BGM by clicking on mute.



00:29

Score : 0

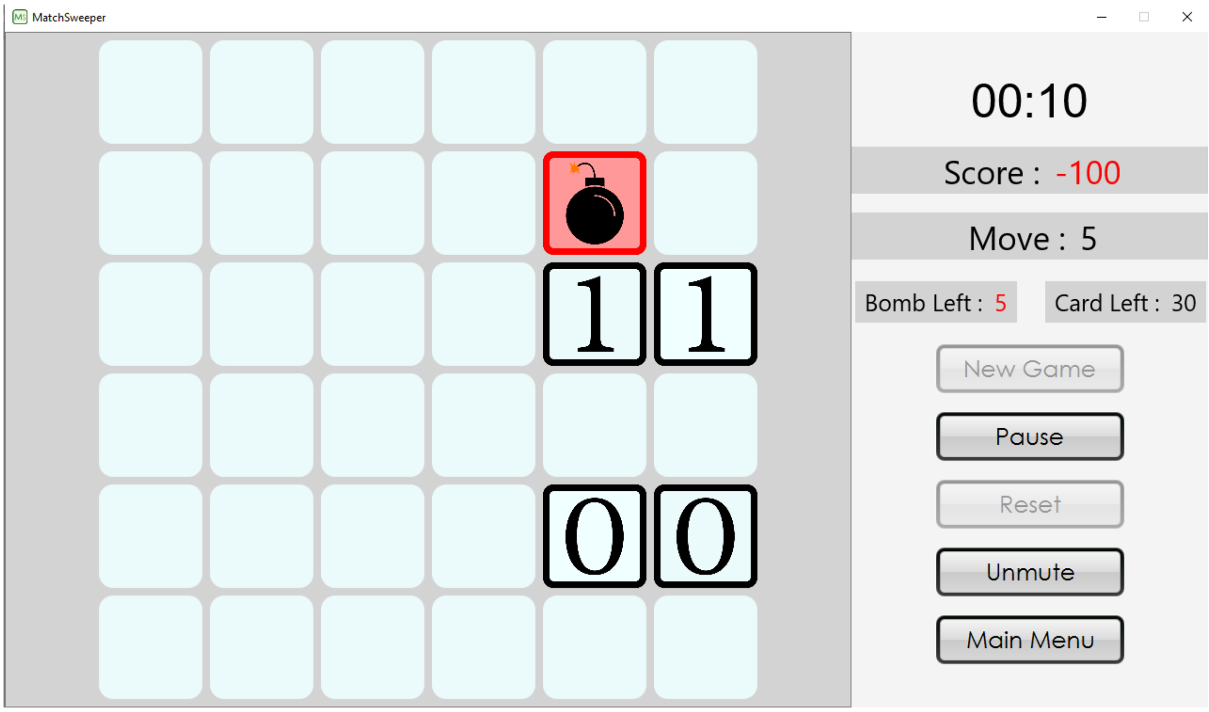
Move : 18

Bomb Left : 6

Card Left : 30

				1	0
			2	1	0
			1	0	0
			2	0	0
			1	0	0
			1	0	0

When the game starts and not in pause players can flip up the card by left clicking on them, if players wrongly match them the card will flip down and show the number of landmines around them, not including itself.



00:10

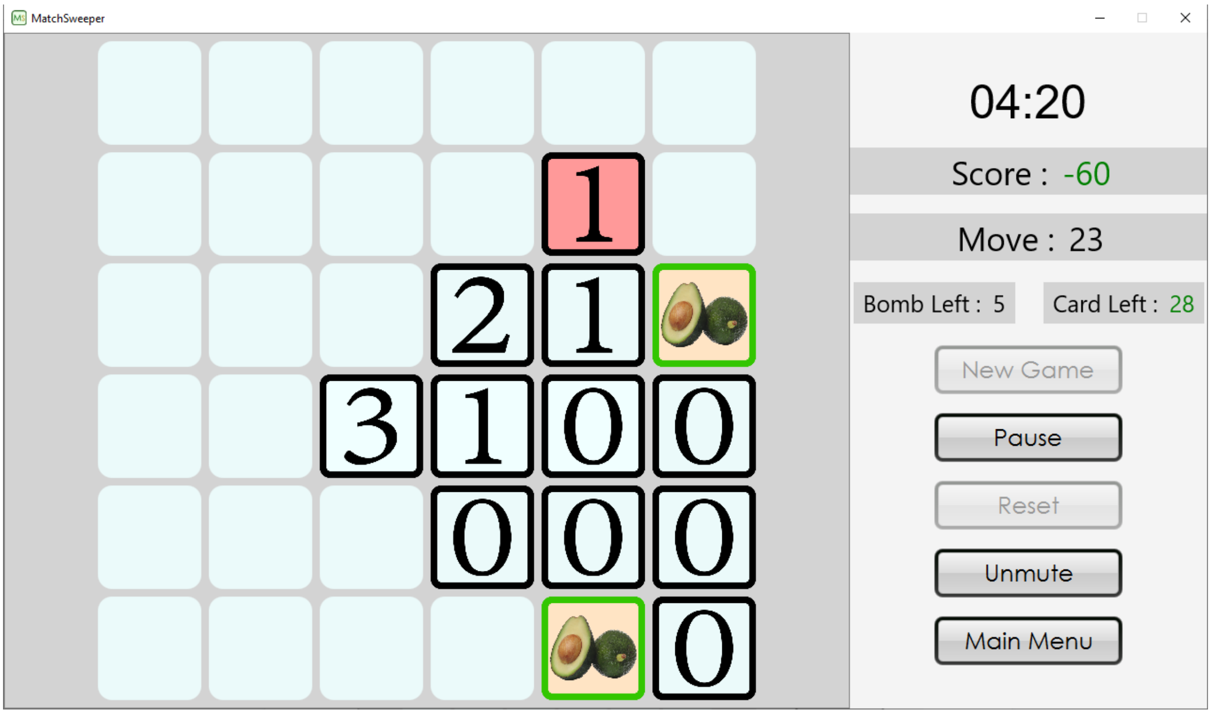
Score : -100

Move : 5

Bomb Left : 5

Card Left : 30

When players trap on landmine, the Score and Bomb Left text will decrease and recolor to red for a second and then change to black like normal.



04:20

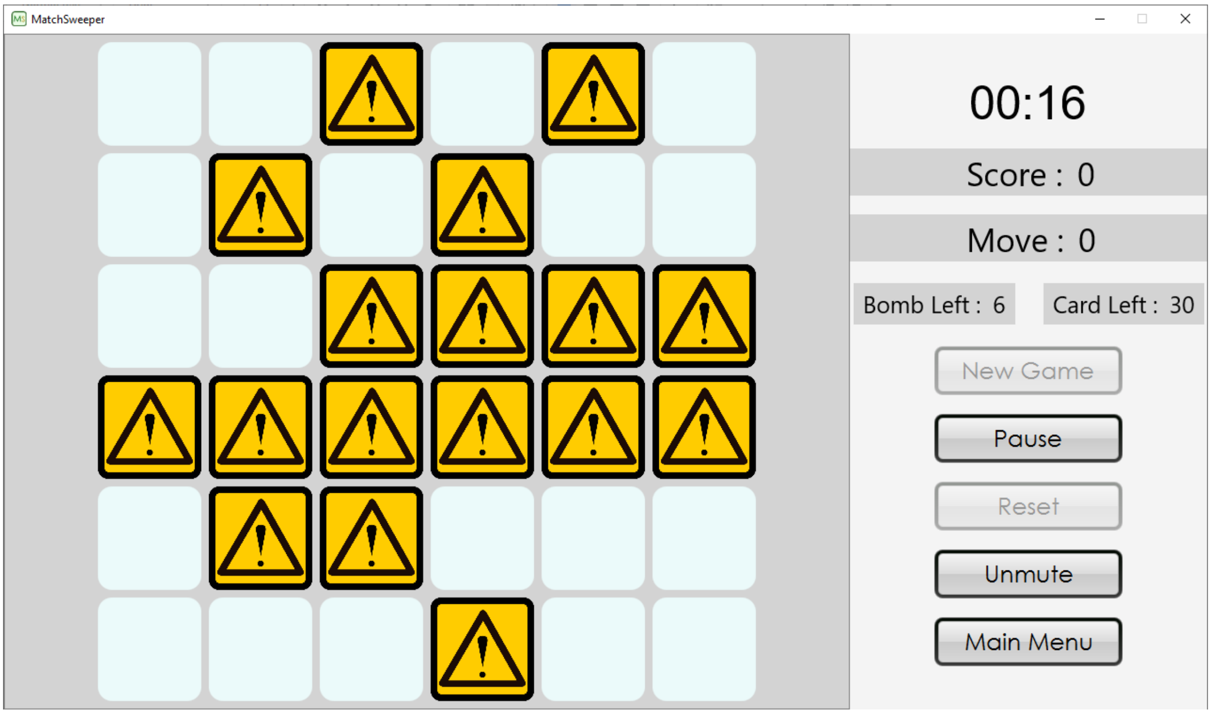
Score : -60

Move : 23

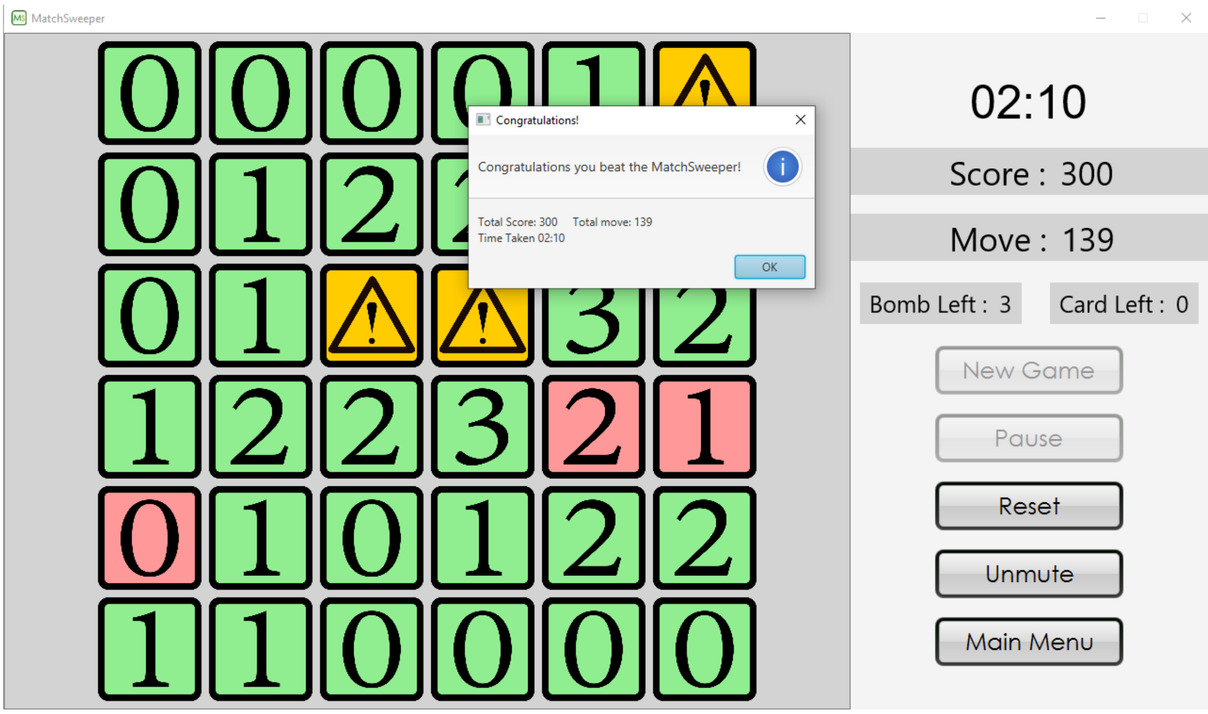
Bomb Left : 5

Card Left : 28

When players match the cards correctly, the Score and Card Left text will decrease and recolor to green for a second and then change to black like normal.



When players want to mark on the suspect card, players can right click them and right click again to unmark them. Players can’t mark on the card once flipped up.



02:10

Score : 300

Move : 139

Bomb Left : 3

Card Left : 0

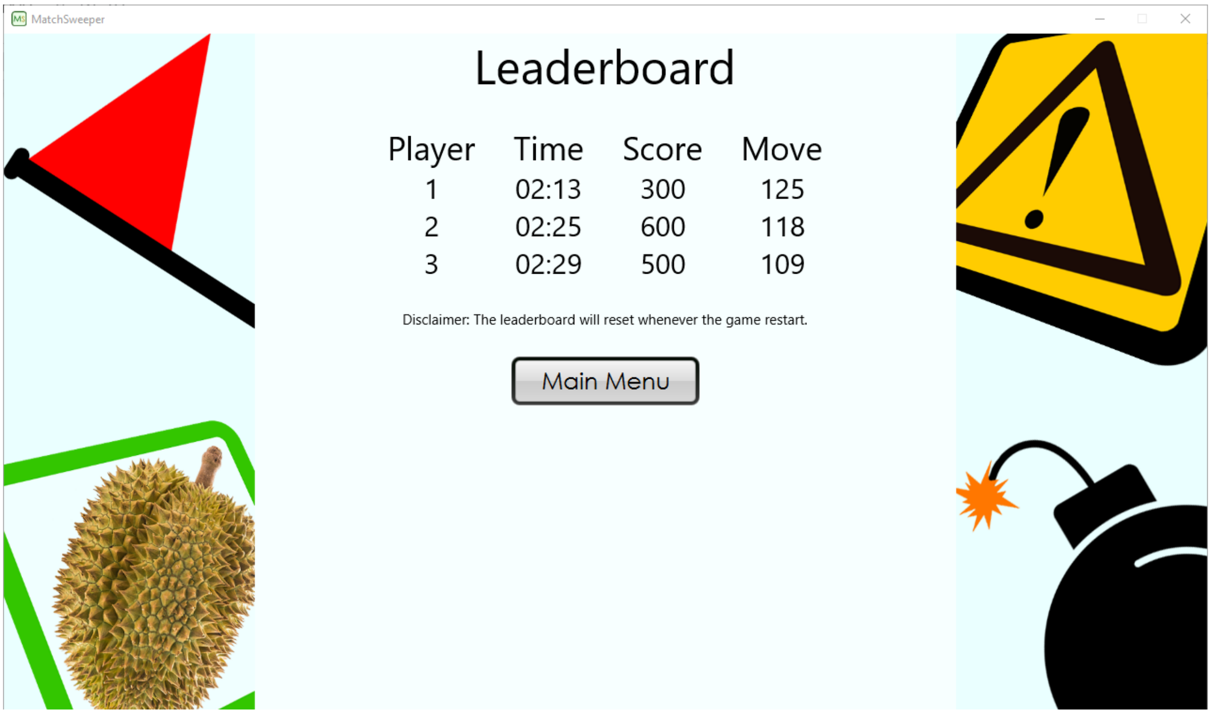
Congratulations!

Congratulations you beat the MatchSweeper!

Total Score: 300 Total move: 139
Time Taken 02:10

When players complete the game, it will show the alert like in the picture and inform the status of players.

1.3 Leaderboard Scene



Leaderboard

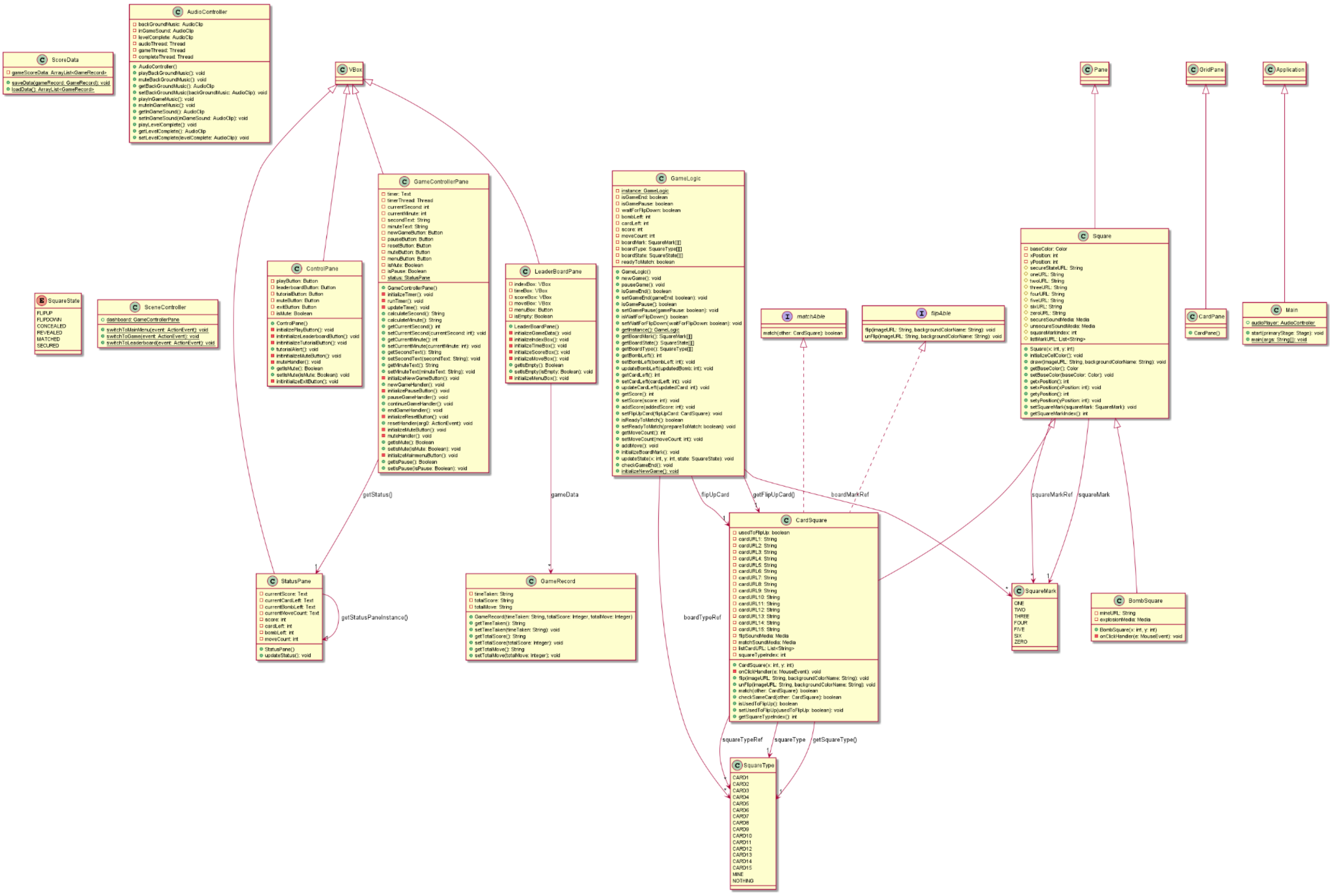
Player	Time	Score	Move
1	02:13	300	125
2	02:25	600	118
3	02:29	500	109

Disclaimer: The leaderboard will reset whenever the game restart.

Show the statistics of every round players complete the game. Disclaimer, the leaderboard will reset whenever the game restarts.

UML Class Diagram

For detailed version please visit: https://drive.google.com/file/d/1b8_ytJKbVaSTXD0USHSkamJrbzQYA7t1/view?usp=sharing


ScoreData
gameScoreData: ArrayList<GameRecord>
saveData(gameRecord: GameRecord): void
loadData(): ArrayList<GameRecord>
AudioController
backGroundMusic: AudioClip
inGameSound: AudioClip
levelComplete: AudioClip
audioThread: Thread
gameThread: Thread
completeThread: Thread
AudioController()
playBackGroundMusic(): void
muteBackGroundMusic(): void
getBackGroundMusic(): AudioClip
setBackGroundMusic(backGroundMusic: AudioClip): void
playInGameMusic(): void
muteInGameMusic(): void
getInGameSound(): AudioClip
setInGameSound(inGameSound: AudioClip): void
playLevelComplete(): void
getLevelComplete(): AudioClip
setLevelComplete(levelComplete: AudioClip): void
VBox
Pane
GridPane
Application
SquareState
FLIPUP
FLIPDOWN
CONCEALED
REVEALED
MATCHED
SECURED
SceneController
dashboard: GameControllerPane
switchToMainMenu(event: ActionEvent): void
switchToGame(event: ActionEvent): void
switchToLeaderboard(event: ActionEvent): void
ControlPane
playButton: Button
leaderboardButton: Button
tutorialButton: Button
muteButton: Button
exitButton: Button
isMute: Boolean
ControlPane()
initializePlayButton(): void
intinitializeLeaderboardButton(): void
intinitializeTutorialButton(): void
tutorialAlert(): void
intinitializeMuteButton(): void
muteHandler(): void
getIsMute(): Boolean
setIsMute(isMute: Boolean): void
intinitializeExitButton(): void
GameControllerPane
timer: Text
timerThread: Thread
currentSecond: int
currentMinute: int
secondText: String
minuteText: String
newGameButton: Button
pauseButton: Button
resetButton: Button
muteButton: Button
menuButton: Button
isMute: Boolean
isPause: Boolean
status: StatusPane
GameControllerPane()
initializeTimer(): void
runTimer(): void
updateTime(): void
calculateSecond(): String
calculateMinute(): String
getCurrentSecond(): int
setCurrentSecond(currentSecond: int): void
getCurrentMinute(): int
setCurrentMinute(currentMinute: int): void
getSecondText(): String
setSecondText(secondText: String): void
getMinuteText(): String
setMinuteText(minuteText: String): void
initializeNewGameButton(): void
newGameHandler(): void
initializePauseButton(): void
pauseGameHandler(): void
continueGameHandler(): void
endGameHandler(): void
initializeResetButton(): void
resetHandler(arg0: ActionEvent): void
initializeMuteButton(): void
muteHandler(): void
getIsMute(): Boolean
setIsMute(isMute: Boolean): void
initializeMainmenuButton(): void
getIsPause(): Boolean
setIsPause(isPause: Boolean): void
LeaderBoardPane
indexBox: VBox
timeBox: VBox
scoreBox: VBox
moveBox: VBox
menuBox: Button
isEmpty: Boolean
LeaderBoardPane()
initializeGameData(): void
initializeIndexBox(): void
initializeTimeBox(): void
initializeScoreBox(): void
initializeMoveBox(): void
getIsEmpty(): Boolean
setIsEmpty(isEmpty: Boolean): void
initializeMenuBox(): void
GameLogic
instance: GameLogic
isGameEnd: boolean
isGamePause: boolean
waitForFlipDown: boolean
bombLeft: int
cardLeft: int
score: int
moveCount: int
boardMark: SquareMark[][]
boardType: SquareType[][]
boardState: SquareState[][]
readyToMatch: boolean
GameLogic()
newGame(): void
pauseGame(): void
isGameEnd(): boolean
setGameEnd(gameEnd: boolean): void
isGamePause(): boolean
setGamePause(gamePause: boolean): void
isWaitForFlipDown(): boolean
setWaitForFlipDown(waitForFlipDown: boolean): void
getInstance(): GameLogic
getBoardMark(): SquareMark[][]
getBoardState(): SquareState[][]
getBoardType(): SquareType[][]
getBombLeft(): int
setBombLeft(bombLeft: int): void
updateBombLeft(updatedBomb: int): void
getCardLeft(): int
setCardLeft(cardLeft: int): void
updateCardLeft(updatedCard: int): void
getScore(): int
setScore(score: int): void
addScore(addedScore: int): void
setFlipUpCard(flipUpCard: CardSquare): void
isReadyToMatch(): boolean
setReadyToMatch(prepareToMatch: boolean): void
getMoveCount(): int
setMoveCount(moveCount: int): void
addMove(): void
initializeBoardMark(): void
updateState(x: int, y: int, state: SquareState): void
checkGameEnd(): void
initializeNewGame(): void
matchAble
match(other: CardSquare): boolean
flipAble
flip(imageURL: String, backgroundColorName: String): void
unFlip(imageURL: String, backgroundColorName: String): void
Square
baseColor: Color
xPosition: int
yPosition: int
secureStateURL: String
oneURL: String
twoURL: String
threeURL: String
fourURL: String
fiveURL: String
sixURL: String
zeroURL: String
secureSoundMedia: Media
unsecureSoundMedia: Media
squareMarkIndex: int
listMarkURL: List<String>
Square(x: int, y: int)
initializeCellColor(): void
draw(imageURL: String, backgroundColorName: String): void
getBaseColor(): Color
setBaseColor(baseColor: Color): void
getxPosition(): int
setxPosition(xPosition: int): void
getyPosition(): int
setyPosition(yPosition: int): void
setSquareMark(squareMark: SquareMark): void
getSquareMarkIndex(): int
CardPane
CardPane()
Main
audioPlayer: AudioController
start(primaryStage: Stage): void
main(args: String[]): void
getStatus()
gameData
flipUpCard
getFlipUpCard()
boardMarkRef
squareMarkRef
squareMark
1
*
StatusPane
currentScore: Text
currentCardLeft: Text
currentBombLeft: Text
currentMoveCount: Text
score: int
cardLeft: int
bombLeft: int
moveCount: int
StatusPane()
updateStatus(): void
getStatusPaneInstance()
GameRecord
timeTaken: String
totalScore: String
totalMove: String
GameRecord(timeTaken: String, totalScore: Integer, totalMove: Integer)
getTimeTaken(): String
setTimeTaken(timeTaken: String): void
getTotalScore(): String
setTotalScore(totalScore: Integer): void
getTotalMove(): String
setTotalMove(totalMove: Integer): void
boardTypeRef
CardSquare
usedToFlipUp: boolean
cardURL1: String
cardURL2: String
cardURL3: String
cardURL4: String
cardURL5: String
cardURL6: String
cardURL7: String
cardURL8: String
cardURL9: String
cardURL10: String
cardURL11: String
cardURL12: String
cardURL13: String
cardURL14: String
cardURL15: String
flipSoundMedia: Media
matchSoundMedia: Media
listCardURL: List<String>
squareTypeIndex: int
CardSquare(x: int, y: int)
onClickHandler(e: MouseEvent): void
flip(imageURL: String, backgroundColorName: String): void
unFlip(imageURL: String, backgroundColorName: String): void
match(other: CardSquare): boolean
checkSameCard(other: CardSquare): boolean
isUsedToFlipUp(): boolean
setUsedToFlipUp(usedToFlipUp: boolean): void
getSquareTypeIndex(): int
SquareMark
ONE
TWO
THREE
FOUR
FIVE
SIX
ZERO
BombSquare
mineURL: String
explosionMedia: Media
BombSquare(x: int, y: int)
onClickHandler(e: MouseEvent): void
squareTypeRef
squareType
getSquareType()
SquareType
CARD1
CARD2
CARD3
CARD4
CARD5
CARD6
CARD7
CARD8
CARD9
CARD10
CARD11
CARD12
CARD13
CARD14
CARD15
MINE
NOTHING

1.Package gui.Game

1.1 class Square

1.1.1 Fields

- Color baseColor	Base color of the square
- int xPosition	Coordinate of x in CardPane
- int yPosition	Coordinate of y in CardPane
# final String secureStateURL	URL of secure state image
# final String oneURL	URL of square with mark one state image
# final String twoURL	URL of square with mark two image
# final String threeURL	URL of square with mark state image
# final String fourURL	URL of square with mark four image
# final String fiveURL	URL of square with mark five image
# final String sixURL	URL of square with mark six image
# final String zeroURL	URL of square with mark zero image
# final Media secureSoundMedia	Media sound when square turn into secured state
# final Media unsecureSoundMedia	Media sound when square turn back from secured state
# SquareMark squareMark	Enum SquareMark of this square
# final int squareMarkIndex	Index of SquareMark of this square in squareMarkRef
# List<String> listMarkURL	listMarkURL = new ArrayList<>()

	- List of URL of square with mark image
# final List<SquareMark> squareMarkRef	squareMarkRef = Arrays.asList(SquareMark.ZERO, SquareMark.ONE, SquareMark.TWO, SquareMark.THREE, SquareMark.FOUR, SquareMark.FIVE, SquareMark.SIX); - Use to referent index of SquareMark of this square

1.1.2 Constructor

+ Square(int x, int y)	- Receive coordinate of x and y in CardPane - Set preference - Set URL of each mark image - Set listMarkURL - Set squareMark from GameLogic - Set squarMarkIndex from squareMark in squareMarkRef - Set secureSoundMedia from URL - Set unsecureSoundMedia from URL

1.1.3 Methods

+ void initializeCellColor()	Set style
+ void draw(String imageURL, String backgroundColorName)	Set background image and style
+ void setSquareMark(SquareMark squareMark)	Set squareMark

+ int getSquareMarkIndex()	Get squareMarkIndex
Generate getter/setter of baseColor, xPosition, yPosition	

1.1.4 Classes

# class PlaySound implements Runnable	Play the sound when the card is clicked or matched.
# class DrawThread implements Runnable	Using thread to draw

1.2 class BombSquare extends Square

1.2.1 Fields

- final String mineURL	URL enum SquareType MINE
- final Media explosionMedia	Media sound when BombSquare revealed

1.2.2 Constructor

+BombSquare(int x, int y)	- Initialize superclass with value x and y - Load and initialize the picture URL of mine - Initialize explosionMedia - Set on mouse clicked event

1.2.3 Method

+Void onClickHandler(MouseEvent e)	Changing bomb square state and color. Then play a sound.

1.3 class CardSquare extends Square implements flipAble, matchAble

1.3.1 Fields

-boolean usedToFlipUp	Check that this square used to flip up or not
-final String cardURL1-cardURL15	URL of each SquareType image
-final Media flipSoundMedia	Media sound when flip up this square
-final Media matchSoundMedia	Media sound when match with other square correctly
-final List<SquareType> squareTypeRef	squareTypeRef = Arrays.asList(SquareType.CARD1, SquareType.CARD2, SquareType.CARD3, SquareType.CARD4, SquareType.CARD5, SquareType.CARD6, SquareType.CARD7, SquareType.CARD8, SquareType.CARD9, SquareType.CARD10, SquareType.CARD11, SquareType.CARD12, SquareType.CARD13, SquareType.CARD14, SquareType.CARD15); - Use to referent index of SquareType of this square
-List<String> listCardURL	listCardURL = new ArrayList<>(); - List of URL of SquareType image
-SquareType squareType	Enum SquareType of this square
-final int squareTypeIndex	Index of SquareType of this square in squareTypeRef

1.3.2 Constructor

+CardSquare(int x, int y)	- Initialize superclass with value x and y - Load and initialize the picture URL of each card - Initialize listCardURL - Load and Initialize flipSoundMedia - Load andInitialize matchSoundMedia - Set on mouse clicked event

1.3.3 Methods

-void onClickHandler(MouseEvent e)	- Changing normal card square state and color. Then play a sound.
+void flip(String imageURL, String backgroundColorName)	- Set new style with imageURL and backgroundColorName when flip up this square - setUsedToFlipUp to true - create and start PlaySound thread with flipSoundMedia
+void unFlip(String imageURL, String backgroundColorName)	- Set new style with imageURL and backgroundColorName when this square flip down by using Thread
+boolean match(CardSquare other)	- Check weather the card is match with another card or not
+boolean checkSameCard (CardSquare other)	- Check if the selected card is the same or not
Generate getter/setter of isUsedToFlipUp, squareType	

1.4 class CardPane extends GridPane

1.4.1 Constructor

+CardPane()	- Set up the pane - Load the card into the pane

1.5 interface flipAble

1.5.1 Methods

+void flip(String imageURL, String backgroundColorName)	Flip up this square
+void unFlip(String imageURL, String backgroundColorName)	Flip down this square

1.6 class GameControllerPane extends VBox

1.6.1 Fields

-Text timer	A text that shows timer
-Thread timerThread	A thread that use to run timer.
-int currentSecond	Current time in seconds
-int currentMinute	Current time in minutes
-String secondText	Second of timer
-String minuteText	Minute of timer
-Button newGameButton	Start timer and game button
-Button pauseButton	Pause game button
-Button resetButton	Reset game button
-Button muteButton	Mute button
-Button menuButton	Back to main menu button
-Boolean isMute	Check mute
-Boolean isPause	Check pause
-static StatusPane status	Status from StatusPane

1.6.2 Constructor

+GameControllerPane()	-Initialize Fields -Initialize timer, new game button, pause button, reset button, mute button and main menu button.

1.6.3 Methods

-void initializeTimer()	Create a timer text box.
-void runTimer()	Run timer when the game has started.
-void updateTime()	Update timer text.
+String calculateSecond()	Calculate time in seconds.
+String calculateMinute()	Calculate time in minutes.
-void initializeNewGameButton()	Create a new game button and set the event handler.
+void newGameHandler()	-Start the game and run the timer. -Disable new game and reset button.
-void initializePauseButton()	Create a pause timer button and set the event handler.
+void pauseGameHandler()	-Pause the timer if the timer was running. -Enable reset button.
+void continueGameHandler()	-Continue the timer if the timer was paused. -Disable reset button.
+void endGameHandler()	Sent an alert, stop the timer when player has completed the game. Then collect data and save it to the leaderboard.
-void initializeResetButton()	Create a reset timer button and set the event handler.

+void resetHandler(ActionEvent arg0)	Reset the whole game and timer.
-void initializeMuteButton()	Create a mute button and set the event handler.
-void muteHandler()	Mute or unmute the game audio. And setIsMute depend on the isMute status.
-void initializeMainmenuButton()	Create a main menu button and set the event handler.
Generate getter/setter of fields	

1.7 interface matchAble

1.7.1 Method

+boolean match(CardSquare other)	Check match to other or not

1.8 class StatusPane extends VBox

1.8.1 Fields

-Text currentScore	Text of current score
-Text currentCardLeft	Text of current cards left
-Text currentBombLeft	Text of current bombs left
-Text currentMoveCount	Text of current moves count
-int score	score
-int cardLeft	cards left
-int bombLeft	bombs left
-int moveCount	moves count

1.8.2 Constructor

+StatusPane()	- Initialize fields - set up the pane

1.8.3 Methods

+void updateStatus()	- update new status
+StatusPane getStatusPaneInstance()	- get StatusPane

1.8.4 Class

#class ChangeColorText implements Runnable	Change color of text depend on value change with thread

2.Package gui.LeaderBoard

2.1 class LeaderBoardPane extends VBox

2.1.1 Fields

-VBox indexBox	Vertical box that shows player index.
-VBox timeBox	Vertical box that shows time taken by player.
-VBox scoreBox	Vertical box that shows player score.
-VBox moveBox	Vertical box that shows player move count.
-Button menuBox	Return to main menu button.
-ArrayList<GameRecord> gameData	ArrayList that stores GameRecord.
-Boolean isEmpty	Status of the data inside score data.

2.1.2 Constructor

+LeaderBoardPane()	-Initialize fields -Initialize leaderboard table and main menu button.

2.1.3 Methods

-initializeGameData()	Load the game data from score data.
-initializeIndexBox()	Create an index box and initialize the index.
-initializeTimeBox()	Create a time box and initialize time data from gameData.
-initializeScoreBox()	Create a score box and initialize score data from gameData.
-initializeMoveBox()	Create a move count box and initialize move count data from gameData.
-initializeMenuBox()	Create a time box and initialize time data from gameData.
Generate getter/setter of fields	

3.Package gui.MainMenu

3.1 class ControlPane extends VBox

3.1.1 Fields

-Button playButton	Button that use to switch to game
-Button leaderboardButton	Button that use to switch to leaderboard
-Button tutorialButton	Button that use to show tutorial

-Button muteButton	Button that use to mute BGM
-Button exitButton	Button that use to exit the game
-Boolean isMute	The game background music is mute.

3.1.2 Constructor

+ControlPane()	-Initialize Fields -Initialize play, leaderboard, tutorial, mute and exit button.

3.1.3 Methods

-void initializePlayButton()	Create play button and set event handler.
-void initinitializeLeaderboardButton()	Create a leaderboard button and set the event handler.
-void initinitializeTutorialButton()	Create a tutorial button and set the event handler.
+void tutorialAlert()	Create an alert box that shows a simple tutorial of the game.
-void initinitializeMuteButton()	Create a mute button and set the event handler.
-void muteHandler()	When the game’s main menu music is mute, play the main menu music and set the text to “Mute”. When the game's main menu music is playing, mute the song and set the text to “Unmute”.
-void initinitializeExitButton()	Create an exit button and set the event handler.
Generate getter/setter of fields	

4.Package logic

4.1 class GameLogic

4.1.1 Fileds

-static GameLogic instance	static GameLogic instance
-boolean isGameEnd	Check is game end yet
-boolean isGamePause	Check is game pause
-boolean waitForFlipDown	waitForFlipDown = false
-int bombLeft	bombLeft = 6
-int cardLeft	cardLeft = 30
-int score	score = 0
-int moveCount	Move count
-SquareMark[][] boardMark	Array 2D of enum SquareMark of each square
-SquareType[][] boardType	Array 2D of enum SquareType of each square
-SquareState[][] boardState	Array 2d of enum SquareState of each square
-List<SquareType> boardTypeRef	boardTypeRef = new ArrayList<>() - use to reference for shuffle
-final List<SquareMark> boardMarkRef	boardMarkRef = Arrays.asList(SquareMark.ZERO, SquareMark.ONE, SquareMark.TWO, SquareMark.THREE, SquareMark.FOUR, SquareMark.FIVE, SquareMark.SIX); - use to reference index of enum SquareMark
-CardSquare flipUpCard	Card that now flipped up
-boolean readyToMatch	readyToMatch = false

4.1.2 Constructor

+GameLogic()	Call method newGame()

4.1.3 Methods

+void newGame()	- Initialize Fields - Shuffle square in board
+void pauseGame()	Set isPause to opposite of current
+void updateBombLeft(int updatedBomb)	Update bombs left with updatedBomb value
+void updateCardLeft(int updatedCard)	Update cards left with updatedCard value
+void addScore(int addedScore)	Add score with addedScore value
+void addMove()	Add moves count by 1
+void initializeBoardMark()	Initialize boardMark
+void updateState(int x, int y, SquareState state)	Update current state of game
+void checkGameEnd()	Check is game end yet
+static void initializeNewGame()	Initialize game every time of new game
Generate getter/setter of fields	

4.2 class GameRecord

4.2.1 Fields

-String timeTaken	String that contain time taken by a player.
-String totalScore	String that contain total score count.
-String totalMove	String that contain total move count.

4.2.2 Constructor

GameRecord(String timeTaken, Integer totalScore, Integer totalMove)	Initialize fields

4.2.3 Methods

Generate getter/setter of fields	

4.3 class ScoreData

4.3.1 Field

-static ArrayList<GameRecord> gameScoreData	List of data from play game

4.3.2 Methods

+static void saveData(GameRecord gameRecord)	Save new data
+static ArrayList<GameRecord> loadData()	Return data list

4.4 enum SquareMark

4.4.1 enum

ONE, TWO, THREE, FOUR, FIVE, SIX, ZERO;

4.5 enum SquareState

4.5.1 enum

FLIPUP, FLIPDOWN, CONCEALED, REVEALED, MATCHED, SECURED;

4.6 enum SquareType

4.6.1 enum

CARD1, CARD2, CARD3, CARD4, CARD5, CARD6, CARD7, CARD8, CARD9, CARD10, CARD11, CARD12, CARD13, CARD14, CARD15, MINE, NOTHING;

5.Package main

5.1 class AudioController

5.1.1 Fields

-AudioClip backGroundMusic	Audioclip for background music
-AudioClip inGameSound	Audioclip for in game sound
-AudioClip levelComplete	Audioclip for level completion sound.
-Thread audioThread	A thread that use to play background music.
-Thread gameThread	A thread that use to play in game music.
-Thread completeThread	A thread that use to play level completion sound.

5.1.2 Constructor

+AudioController()	-Initialize audio string path -Set up audio clip for every file.

5.1.3 Methods

+void playBackGroundMusic()	Play main menu background music with thread

+void muteBackGroundMusic()	Stop main menu background music
+void playInGameMusic()	Play in game background music with thread
+void muteInGameMusic()	Stop in game background music
+void playLevelComplete()	Play when game complete
Generate getter/setter of fields	

5.2 class Main extends Application

5.2.1 Field

+static AudioController audioPlayer	Audio controller of main menu

5.2.2 Methods

+void start(Stage primaryStage)	Initialize the main menu scene and show the scene.
+static void main(String[] args)	Main application

5.3 class SceneController

5.3.1 Field

+static GameControllerPane dashboard	Game controller pane in Game scene

5.3.2 Methods

+static void switchToMainMenu(ActionEvent event)	Initialize the main menu scene and show the scene.
+static void switchToGame(ActionEvent event)	Initialize the game scene and show the scene.
+static void	Initialize the leaderboard scene

switchToLeaderboard(ActionEvent event)	and show the scene.