Hanoi University of Science and Technology
Cyber Security **K67**

Class 149324 - IT3100E - Object-Oriented Programming

Lecturer: Prof. Nguyen Thi Thu Trang

Mini-Project Report:

Mandarin Square Capturing Game

Member list:
Kim Duy Minh– 20225578
Trần Đăng Huy – 20225575
Nguyễn Tuấn Minh – 20225579

Sunday, 9th June, 2024

Contents

1. Assignment of members

General contribution:

1. Kim Duy Minh: 45%
2. Trần Đăng Huy: 45%
3. Nguyễn Tuấn Minh: 10%

Member/Task	Problem Modeling	Model	View	Controller	Other tasks
Kim Duy Minh	Brain-storming for the problem	- OVuong -userSetting	- gameScene.fxml	- guideSceneController - gameSceneController - musicController	- Write report - Fix slide - Interface music
Tran Dang Huy	Brain-storming for the problem	- OVuong - ez -flashbang -umad	- settingScene.fxml	- LogicGame - gameSceneController - musicController	- Fix report - Fix slide
Nguyen Tuan Minh	Draw diagrams	- ticking - zedd	- mainScene.fxml - guideScene.fxml	- mainSceneController	- Picture design - Demo - Design slide

2. Mini-project description

2.1.

2.2. About the game

This game is played by two individuals on a board that features 10 squares arranged in two parallel rows, with a semicircular space at each end. At the beginning, there are 5 little stones in each square and a larger stone in each semicircle. The small stones are worth 1 point each, while the large stones are worth 5 points each. The starting player is selected at random.

Each contestant controls 5 squares on their side of the board. Players initiate their move by lifting all stones from any of their squares and dispersing them one by one in a chosen direction, either clockwise or anticlockwise. They must deposit a stone into each subsequent square or semicircle until they run out of stones. It's not allowed to begin a turn from a semicircle.

The square adjacent to where the last stone is placed is termed the "terminal" square. If the "terminal" contains stones, the player may use them to continue dispersing in the same trajectory. The dispersal concludes when the "terminal" is devoid of stones.

If the "terminal" is empty and is succeeded by a square containing stones, the player captures all stones from that square. If another empty square follows this, and then a square filled with stones, the player can keep capturing in this manner.

Players must maintain their chosen direction throughout their turn without switching.

Scoring is based on the total points from the captured stones, with small stones contributing 1 point and large stones 5 points each.

The game concludes under two conditions:

1. Both semicircles are emptied of stones.
2. A player has no stones left in any of their squares when it's their turn.

2.3. Project requirements

On the main screen, the game should have the following buttons:

- Start: Start the game
- Instruction: Show guide scene

2.4. Use cases

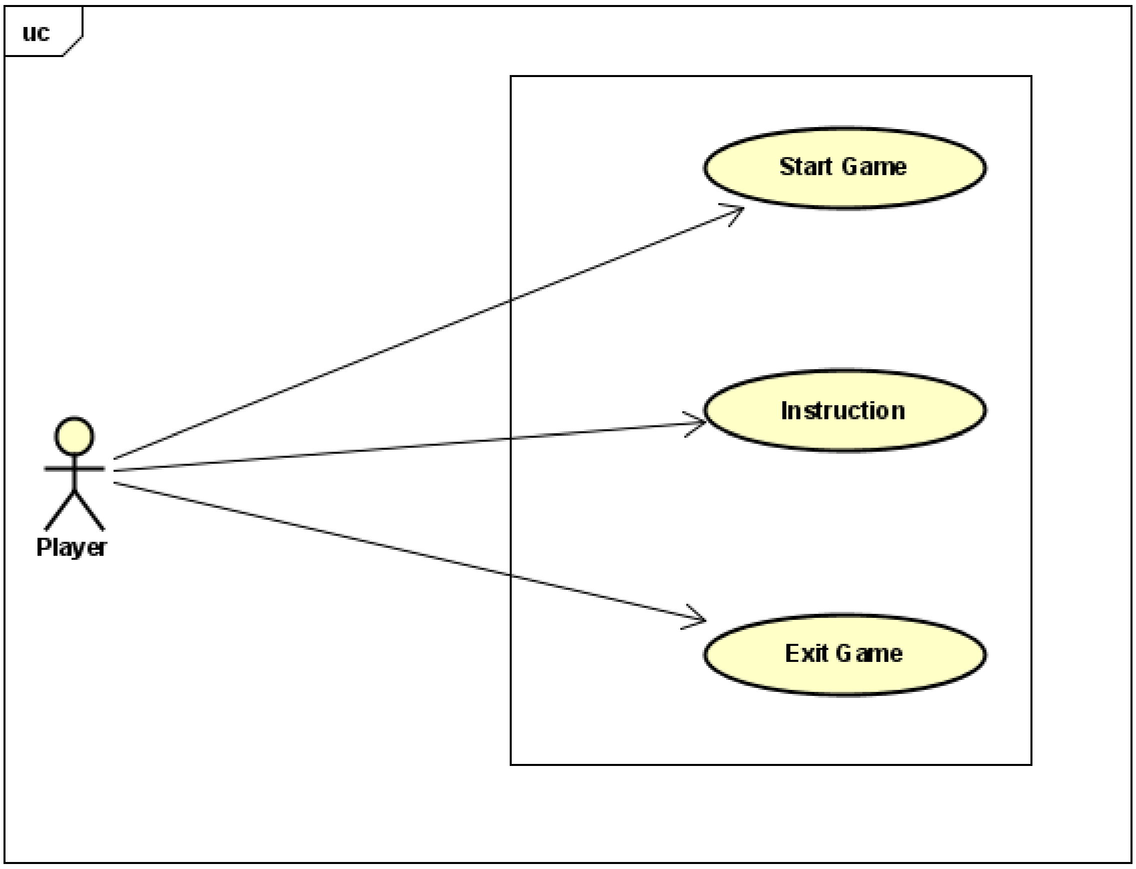


Figure 1: Use case diagram

The user (Player) in this program has two use cases: Start game, and Instructions. Specifically:

- Start game: When players choose to start the game (by clicking the Start button in the Intro Screen), the program displays a playing board, the player then plays the game following the rules until the game is finished. Players engage in the game adhering to the previously outlined rules until its conclusion. If a player opts to leave mid-game, a confirmation prompt appears, and the game terminates based on the player’s choice. Exiting is also an option from the Intro Screen.
- Instructions: When players select the Help option from the Intro Screen, the program will navigate them to the Help Menu. Here, they will find detailed instructions on how to play the game. Additionally, the program will exhibit a visual representation of the board along with the game’s rules for easy understanding.
- Exit Game: In the intro screen or while playing, the player can click the X button to quit the program.

3. Design

3.1. Class diagram and class relationship

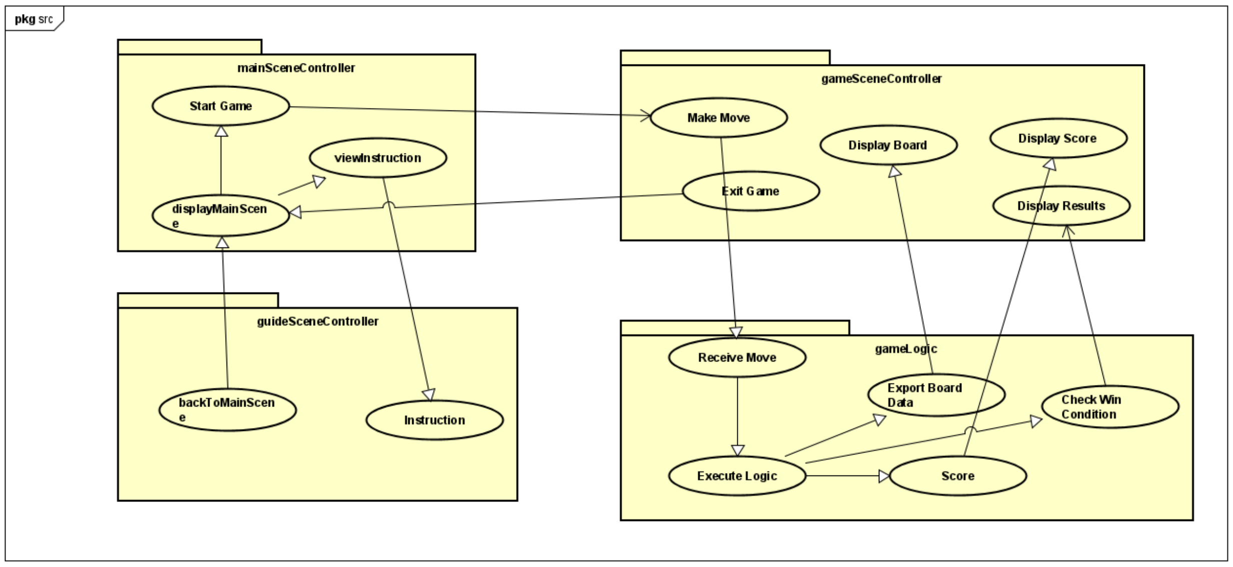
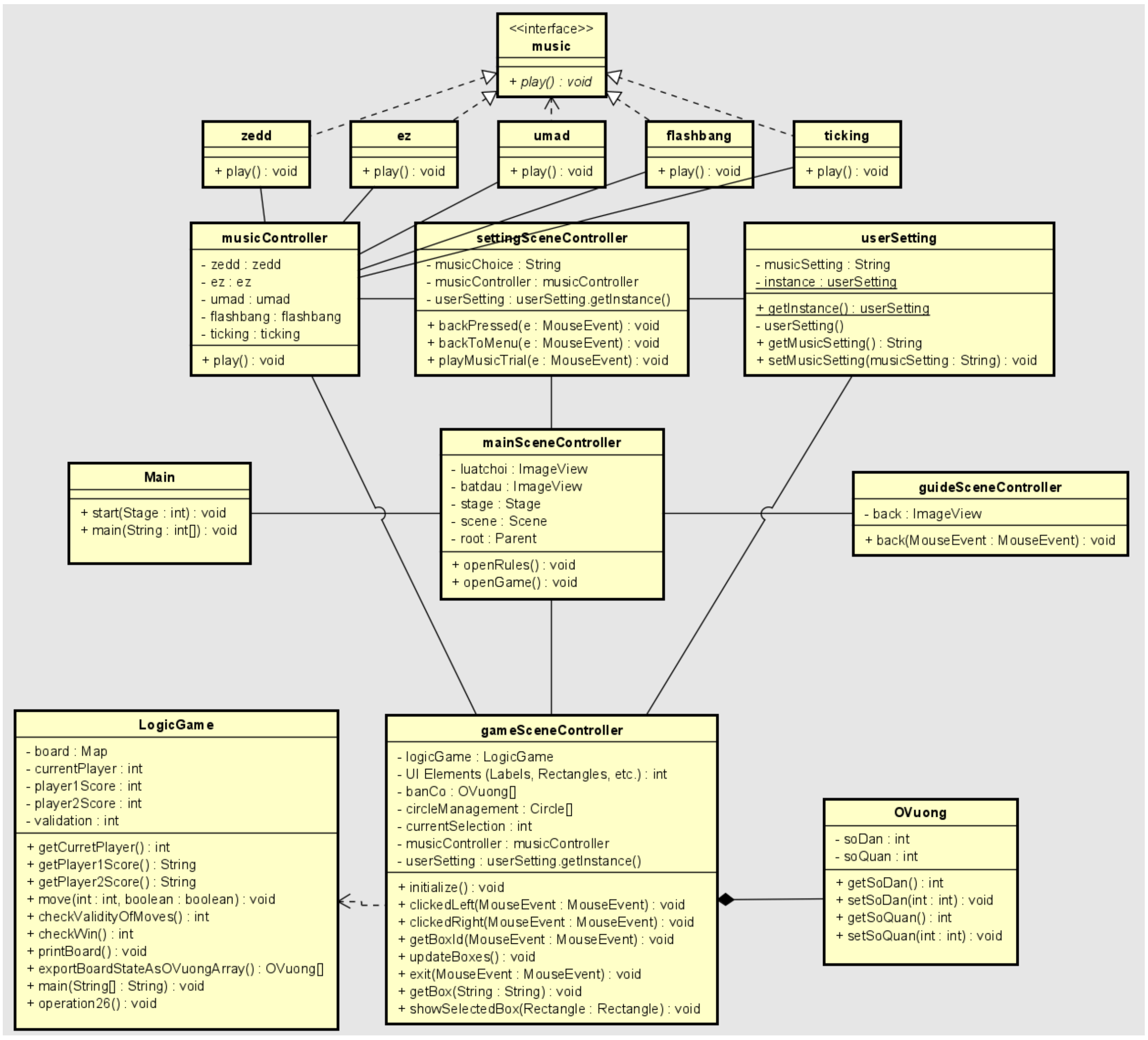


Figure 2: General Class diagram

3.2. Detailed class diagrams:



<<interface>>
music
+ play() : void
zedd
+ play() : void
ez
+ play() : void
umad
+ play() : void
flashbang
+ play() : void
ticking
+ play() : void
musicController
- zedd : zedd
- ez : ez
- umad : umad
- flashbang : flashbang
- ticking : ticking
+ play() : void
settingSceneController
- musicChoice : String
- musicController : musicController
- userSetting : userSetting.getInstance()
+ backPressed(e : MouseEvent) : void
+ backToMenu(e : MouseEvent) : void
+ playMusicTrial(e : MouseEvent) : void
userSetting
- musicSetting : String
- instance : userSetting
+ getInstance() : userSetting
- userSetting()
+ getMusicSetting() : String
+ setMusicSetting(musicSetting : String) : void
Main
+ start(Stage : int) : void
+ main(String : int[]) : void
mainSceneController
- luatchoi : ImageView
- batdau : ImageView
- stage : Stage
- scene : Scene
- root : Parent
+ openRules() : void
+ openGame() : void
guideSceneController
- back : ImageView
+ back(MouseEvent : MouseEvent) : void
LogicGame
- board : Map
- currentPlayer : int
- player1Score : int
- player2Score : int
- validation : int
+ getCurretPlayer() : int
+ getPlayer1Score() : String
+ getPlayer2Score() : String
+ move(int : int, boolean : boolean) : void
+ checkValidityOfMoves() : int
+ checkWin() : int
+ printBoard() : void
+ exportBoardStateAsOVuongArray() : OVuong[]
+ main(String[] : String) : void
+ operation26() : void
gameSceneController
- logicGame : LogicGame
- UI Elements (Labels, Rectangles, etc.) : int
- banCo : OVuong[]
- circleManagement : Circle[]
- currentSelection : int
- musicController : musicController
- userSetting : userSetting.getInstance()
+ initialize() : void
+ clickedLeft(MouseEvent : MouseEvent) : void
+ clickedRight(MouseEvent : MouseEvent) : void
+ getBoxId(MouseEvent : MouseEvent) : void
+ updateBoxes() : void
+ exit(MouseEvent : MouseEvent) : void
+ getBox(String : String) : void
+ showSelectedBox(Rectangle : Rectangle) : void
OVuong
- soDan : int
- soQuan : int
+ getSoDan() : int
+ setSoDan(int : int) : void
+ getSoQuan() : int
+ setSoQuan(int : int) : void

Figure 3: Controller package class diagram

Explaination:
a)mainSceneController class:
-openRules(): open guideScene
-openGame(): open gameScene

b)guideSceneController class:
-back(): open mainScene

c)gameSceneController:
-initialize: initialize game default value
-clickedLeft: a method to listen for the Left arrow if it is

clicked or not
-clickedRight: a method to listen for the Right arrow if it is clicked or not
-getBoxId: return the id of the OVuong that is currently selected
-exit: exit to mainScene
-getBox: convert OVuong ID String to fxml id
-showSelectedBox: show a small circle to indicate the current OVuong that is being selected

d)LogicGame class:
-getCurrentPlayer: set the turn for the current player
-getPlayer1Score: return player 1 current score
-getPlayer2Score: return player 2 current score
-move: execute current movement
-checkValidityOfMoves: check if the move is valid and return value for the console to print out
-checkWin: check if any player win the game after any movement execution
-printBoard: print the current board to the console for debugging
-exportBoardStateAsOVuongArray: send the data of current board for the gameScene to display
-main: just a main method to check the logic execution

e)settingSceneController:
-backPressed & backToMenu: back to mainScene
-playMusicTrial: play the music sample for user to choose the knockout sound

f)userSetting
-userSetting: constructor to create userSetting
-getInstance: check if a userSetting is created or not, if not then create new. This method implement singleton design pattern to make sure that only 1 setting is created at a time and make sure all other classes using the same userSetting
-getMusicSetting: retrieve user setting
-setMusicSetting: set the user setting

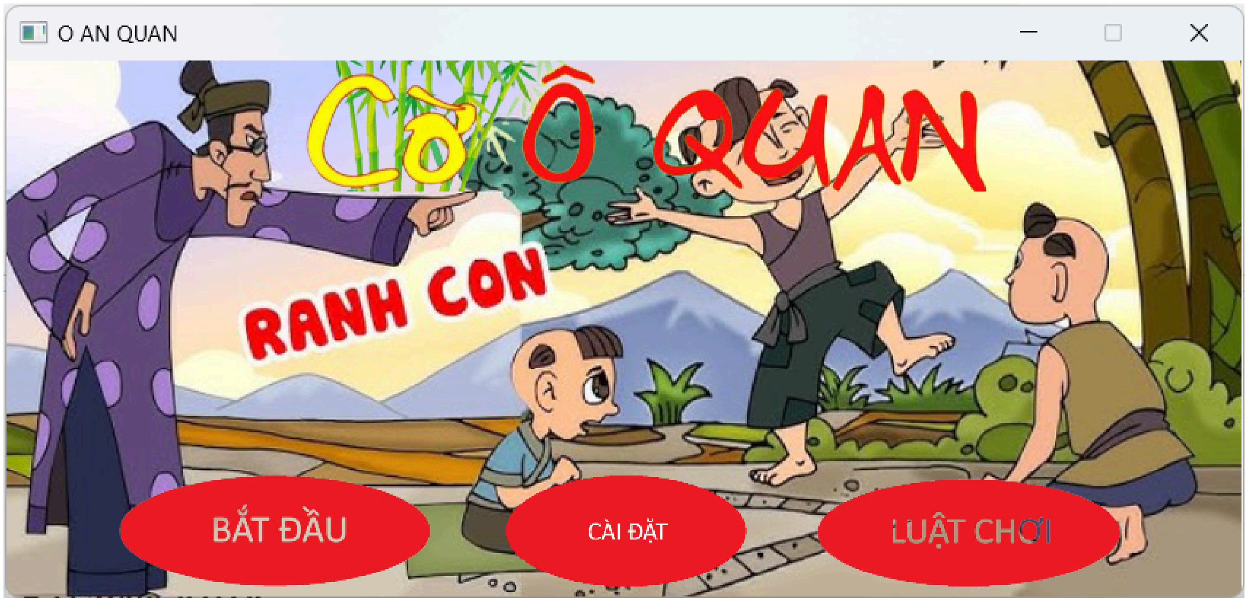
g)musicController:
-play: play the music based on the userSetting argument

h)zedd,ez,flashbang, umad,ticking
These class are just the music with only 1 method implemented from the music Interface

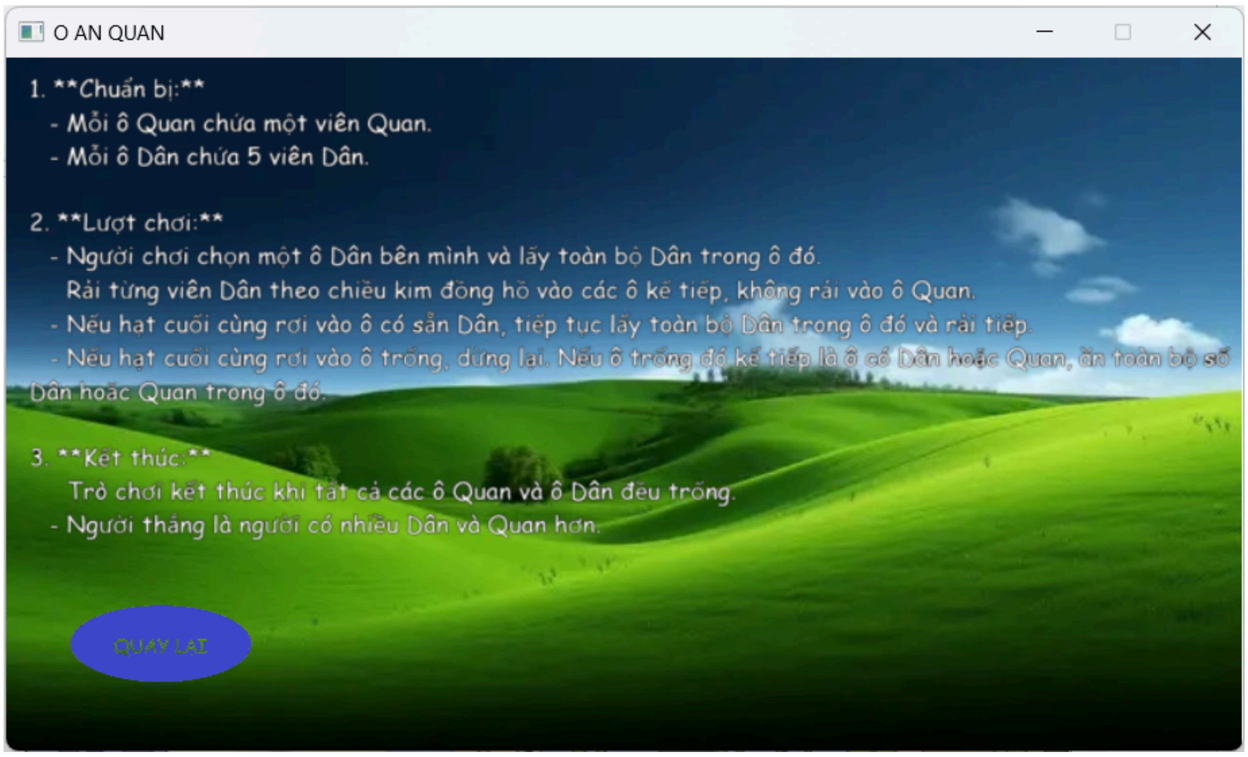
-play: play the current class music

4) Game UI

4.1) Main screen

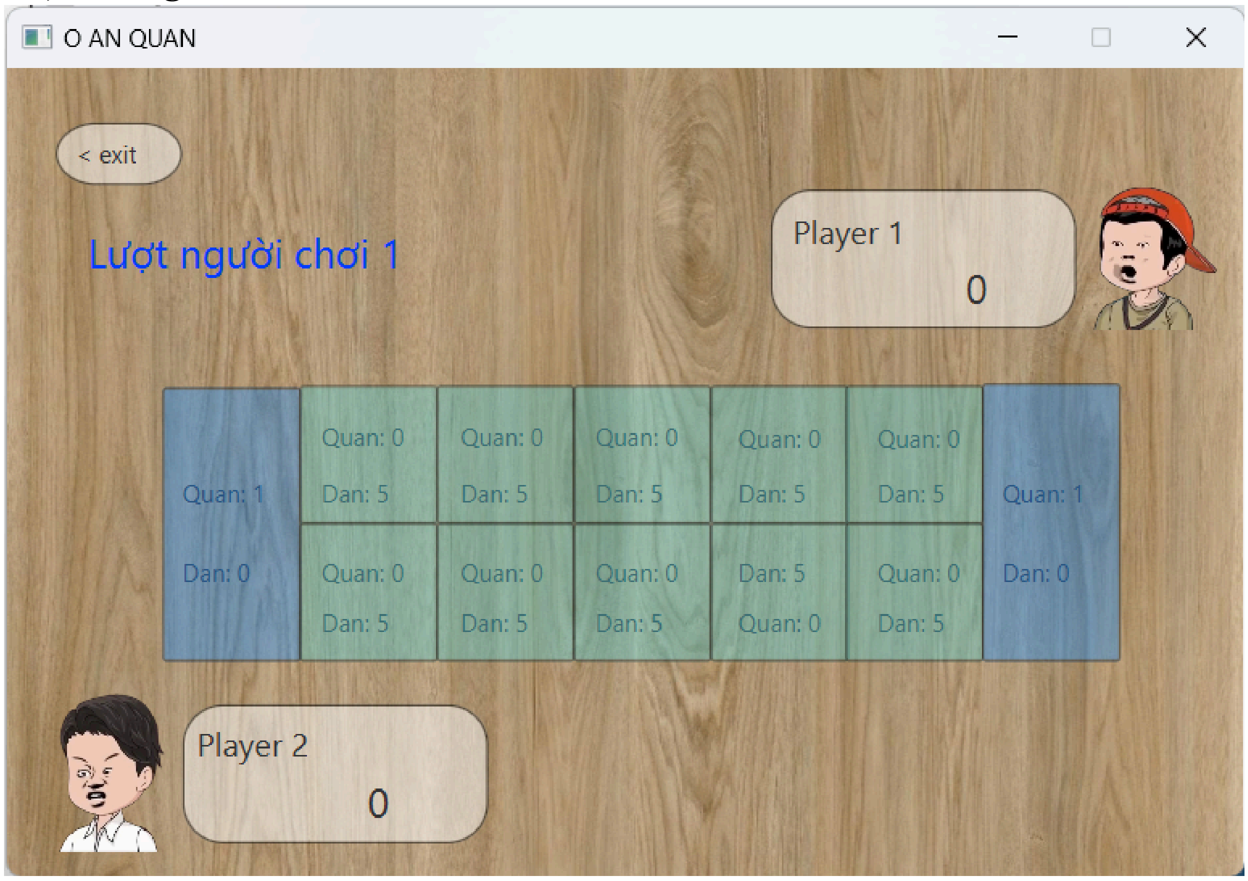


4.2) Instruction screen

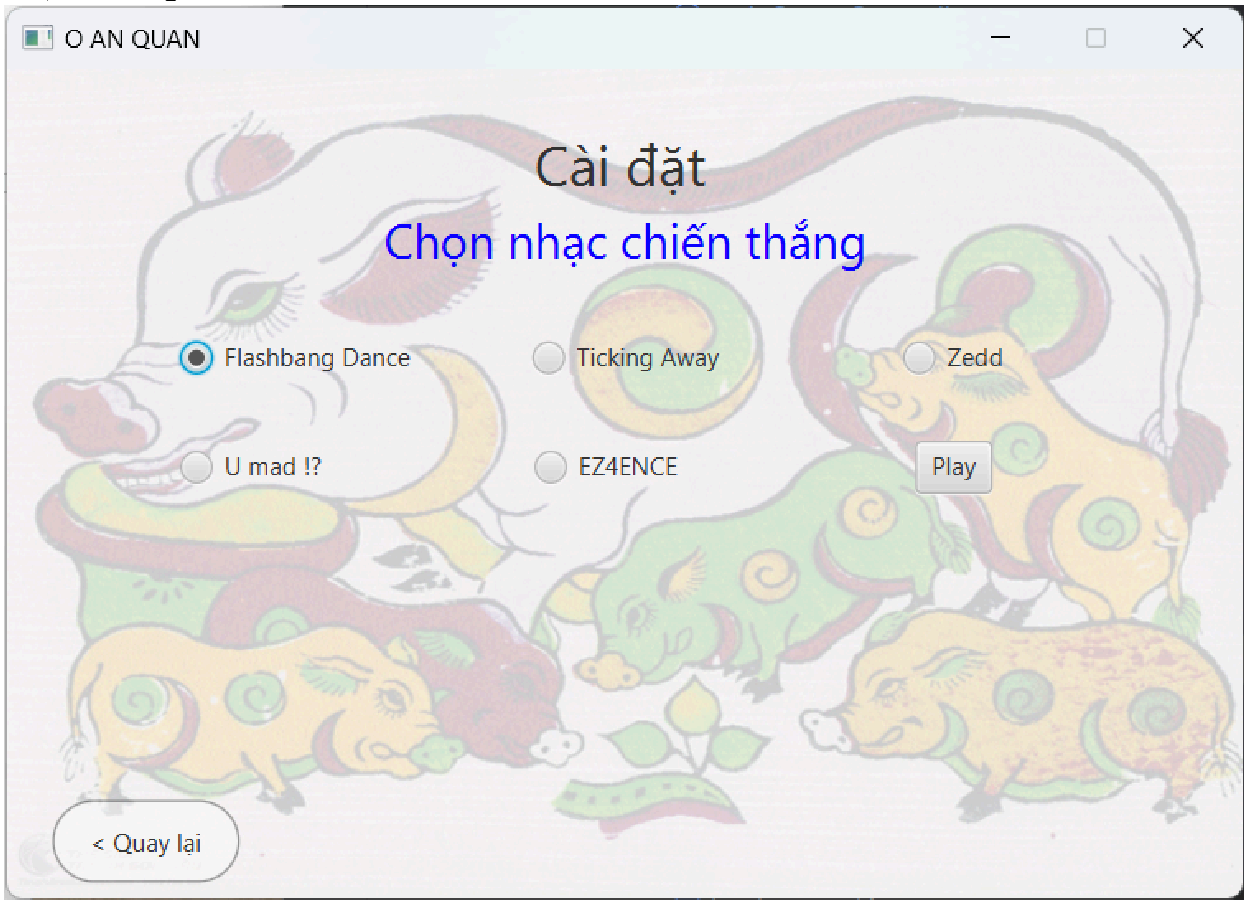


```
1. **Chuẩn bị:**
   - Mỗi ô Quan chứa một viên Quan.
   - Mỗi ô Dân chứa 5 viên Dân.

2. **Lượt chơi:**
   - Người chơi chọn một ô Dân bên mình và lấy toàn bộ Dân trong ô đó.
     Rải từng viên Dân theo chiều kim đồng hồ vào các ô kế tiếp, không rải vào ô Quan.
   - Nếu hạt cuối cùng rơi vào ô có sẵn Dân, tiếp tục lấy toàn bộ Dân trong ô đó và rải tiếp.
   - Nếu hạt cuối cùng rơi vào ô trống, dừng lại. Nếu ô trống đó kế tiếp là ô có Dân hoặc Quan, ăn toàn bộ số
Dân hoặc Quan trong ô đó.

3. **Kết thúc:**
     Trò chơi kết thúc khi tất cả các ô Quan và ô Dân đều trống.
   - Người thắng là người có nhiều Dân và Quan hơn.
```

4.3) Main game screen:



4.4) Setting screen:


O AN QUAN
Cài đặt
Chọn nhạc chiến thắng
[✓ Flashbang Dance]
Ticking Away
Zedd
U mad !?
EZ4ENCE
Play
< Quay lại