Bilkent University

Department of Computer Engineering

CS319
Object Oriented Software Engineering
Project Analysis Report

Katamino

Group 2-F

Sena Er	
Zeynep Sonkaya	21501981
Hüseyin Orkun Elmas	21501364
Utku Görkem Ertürk	21502497
Bartu Atabek	21602229

Table of Contents

1. Introduction

Katamino is a strategic and educational board game. It helps kids to understand geometric shapes and develops their visual perception. However, various levels of difficulties allows Katamino to be a game for people of all ages. Katamino different shapes called pentominoes and a 5 x 12 square board with a slider. Purpose of the game is to fill the playing area that is defined by the slider on the gameboard with the given pentominoes. Game gets more difficult as the number of pieces and the playing area increases. It also contains a booklet showing the set of pentominoes for each level[1,2].

A pentomino is a piece in the game that is a combination of 5 squares and there are 12 different pentominoes in the game. Each level of Katamino is played with specific pentominoes that is given in the booklet, the player tries to put all the given pentominoes to the playing area. The area of slider must be either a square or a rectangle. If the user successfully places all of the specified pentominoes without any blanks, filled area is called a "Pento" and the player completes the level. After that, the slider moves one unit and the game starts again with the new difficulty level. The slider starts on the 3rd location and moves up to the 12th location, creates 9 difficulty levels [3].

To implement the game we will use Java platform with the addition of the JavaFX [4] framework because of its superiority on graphics and UI design over standard Java distribution. Our goal is to implement this game by using the object oriented design principles are taught in CS 319.

2. Overview

Katamino 2.0 is a tile based 2D puzzle game which is styled with a retro modern user interface, created for the desktop platform. After the initialization of the application the user is presented with a welcome screen which prompts them with options such as choosing a gameplay mode i.e. single-player, or multiplayer and changing the settings of the game. Players will be able to save their progress and compare their scores with other players.

2.1 Gameplay

Katamino has two gameplay options; single-player and multi-player of which has different gameplay rules and features accordingly.

2.1.1 Single Player

When a player chooses the single-player option for the game, he/she will be presented with two options, classical mode and custom shapes mode. 2D game board, and various pentomino pieces. In each level the goal of the player is to completely fill the board with pentominoes without any spaces. The game board changes while player progresses through the game. After completing each level, player is awarded with a score. Players can rotate and flip the pentomino blocks in order to fit them in the game board. Additionally, they could ask for hints or the solution if the player has enough score, the correct position of a pentomino is indicated on the game board or the solution of the current level is shown to the user.

There are two different modes in the single player gameplay which are Classic (Arcade) and the Custom Shapes mode.

2.1.1.1 Classic (Arcade) Mode

This is the classic katamino, in which player tries to fill 5x3 area at start and in the last level player tries to fill 5x12 area with pentominoes. Before increasing size of the available area by a slider one 5x1 at a time, player has to pass 3 levels. There is a timer to show time past in a level and scores are calculated using time information.

2.1.1.2 Custom Shapes Mode

In custom shapes mode, player can choose desired gameboard from specified list and tries to fill the game board with given pentominoes using flip and rotate options. There is a timer to show time past and scores are calculated using time information.

2.1.2 Multiplayer

In the multiplayer mode of Katamino, Two players will take turns and place pentominoes on the game board. Which is chosen by the players before the game starts. When a player places a pentomino on the board, the opponent will take her/his turn. There will be a time limit for each turn. If a player fails to place any pentomino left to the board or exceeds the time limit, the opponent will be the winner of the game, and the game will end.

2.2 Game Board

The game board will consist of a grid system shaped like a rectangle with a slider, or a custom shape according to the game mode which he/she has chosen.

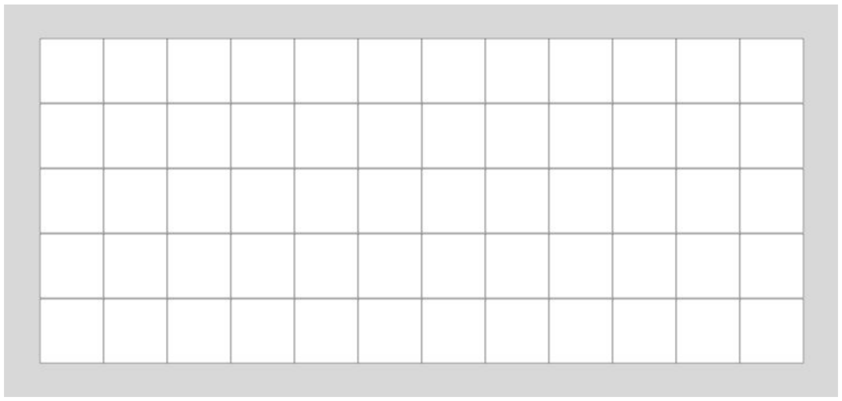


Figure 1: Sample game board for the arcade mode

For the custom shapes mode, the game board will be made out of a square grids resembling objects such as animals, people, square etc. In the board some part of the board can be blocked for the player in order to make the level more complex.

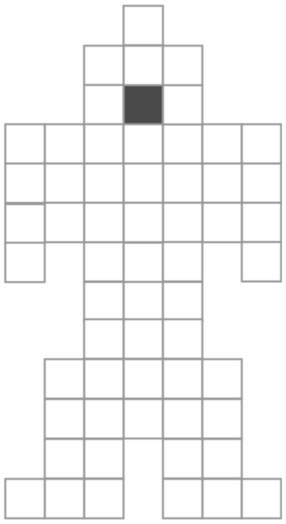


Figure 2: Sample custom game board with the shape of a cyclops.

For the multiplayer game the players will choose a custom shape board.

2.2 Tiles (Pentominoes)

A pentomino is a 2D geometric figure formed by joining five equal squares edge to edge. While rotations and reflections are not considered to be distinct shapes, there are total of 12 different pentominoes. In the case where reflections are considered distinct, there are 18 pentominoes. In addition, when rotations are also considered distinct, there are total of 63 fixed pentominoes. [5]

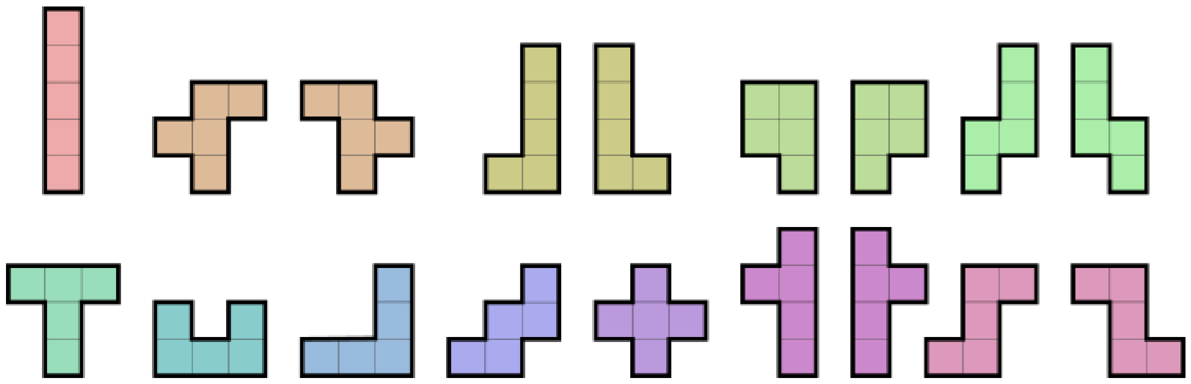


Figure 3: The 12 pentominoes (from left to right I, F, L, P, S, T, U, V, W, X, Y, Z) can form 18 different shapes, with 6 of them being mirrored.

- F, L, S, P, and Y tiles can be oriented in 8 ways: 4 by rotation, and 4 more for the mirror image.
- T, and U tiles can be oriented in 4 ways by rotation. They have an axis of reflection aligned with the gridlines.
- V and W also can be oriented in 4 ways by rotation. They have an axis of reflection symmetry at 45° to the gridlines.
- Z can be oriented in 4 ways: 2 by rotation, and 2 more for the mirror image.
- I can be oriented in 2 ways by rotation. It has two axes of reflection symmetry, both aligned with the gridlines.
- X can be oriented in only one way.

The F, L, S, P, Y, and Z pentominoes are superimposable so that adding their reflections brings the number of one-sided pentominoes to 18 as seen in figure 3. If rotations are also considered distinct, then this results in 63 fixed pentominoes. For example, the eight possible orientations of the L, F, S, P, and Y pentominoes are as follows:

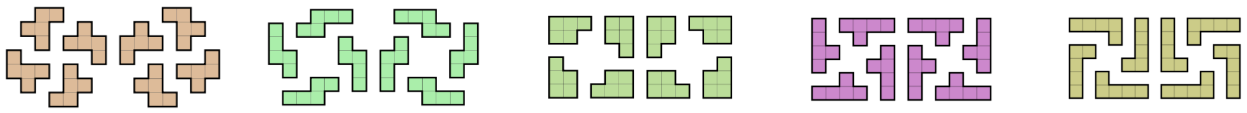


Figure 4: The eight possible orientations of the L, F, S, P, and Y pentominoes.

2.3 Settings

The player will be able to turn the volume down or up.

3. Functional Requirements

3.1 Profile Selection & Creation

Profile Creation: A user must be able to create a new player with a player name that is not an existing player name in the game. The game must create a player and start a fresh game.
Profile Saving: A user must be able to save their progress by clicking a button, and the game must save the current game state, with the corresponding user name.
Profile Selection: A user must be able to choose a player and the game must remember the last game state they saved, and start with that game state.

3.2 Single Player Mode

Level Selection: The player must be able to see which levels they can play, and choose a level from the available levels. If a playable level is chosen, the game must load that level with the. If the selected level is non yet playable, the game must give an error message.

Pentomino Movement: Player must be able to drag and drop pentominoes to a desired location on the screen. The game must approximate the desired location and display the pentomino placed there. Player must be able to rotate the pentominoes 90 degrees clockwise around their center and the game must rotate the pentomino. .If the player is trying to place a pentomino in a way that the pentomino is clashing with other another pentomino or the pentomino has part that will be left outside the game board, the game must warn the player by coloring these parts differently, and not let the player place the pentomino.

Timing & Score: The game must measure how long does the player take to solve a certain level. Using this time information, game will calculate a score.

Hints & Solution: A player must be able to ask for hints or a solution in a single player mode game. If the player has enough score, the correct position of a pentomino should be indicated on the game board or the solution of the current level must be shown to the user. If the player can not afford hints or solution an error message should be displayed.

Finishing Level: The game has to check whether the board is completed without any intersections or pieces outside the board. If these conditions are satisfied, than game must increase level, load the new level with new pieces and increase player's score.

Exiting the Game: The player must be able to exit the game at any time and the game must save the current game progress.

3.3 Multiplayer Game Mode

Multiplayer Board: Player must be able to choose a board from the boards that are available. And the game must load that board.

Pentomino Movement: Player must be able to drag and drop pentominoes to a desired location on the screen. The game must approximate the desired location and display the pentomino placed there. Player must be able to rotate the pentominoes 90 degrees clockwise around their center and 180 degrees around x axis (x axis is defined to be the horizontal axis parallel to the board) and the game must rotate the pentomino visually. If the player is trying to place a pentomino in a way that the pentomino is clashing with other another pentomino or the pentomino has part that will be left outside the game board, the game must warn the player by coloring these parts differently, and not let the player place the pentomino.

Turns: After the user places a pentomino on the board, the game must give the turn to the opponent and restart their time.

Timing: The user must be able to set a time limit for each move and the game must measure time for each move of each player.

Winning the Game: The game also has to check if a player can not place a pentomino completely to the board without any intersections. Game also has to check if the active player (The player that has turn to play) is out of time. Also the player must be able to resign at any point during the game. In the above cases, game must declare the opponent as the winner.

3.4 Settings

A player must be able to adjust volume in any screen of the game by reaching settings. And adjust or mute sounds.

4. Non-Functional Requirements

4.1 Efficiency

The game will be implemented with the efficiency constraints in mind. These efficiency constraints are:

- Creating, loading or saving a player must take less than 1 second.
- When the player asks for a hint or a solution; hint, solution or an error message must be displayed in less than 0.5 seconds.
- Level loading must take less than 0.5 seconds.
- In both multiplayer and singleplayer game all movements of a pentomino must take less than 0.1 seconds
- In multiplayer game and custom single player game, loading a board must take less than 0.5 seconds.
- When exiting the game in a single game, previous menu must load before 0.5 seconds

4.2 Usability

The interface of the game will be designed such that a new player will be able to quickly learn and play the game without confusion. The controls that player interacts with will be designed intuitively, in order to prevent confusion.

4.3 Data Integrity

The game will handle user data accurately, authentically and without corruption. The user will be always load the data that is saved by the game, and continue their progress. The user data will be saved after every played level, to minimize lost progress.

4.4 Reliability

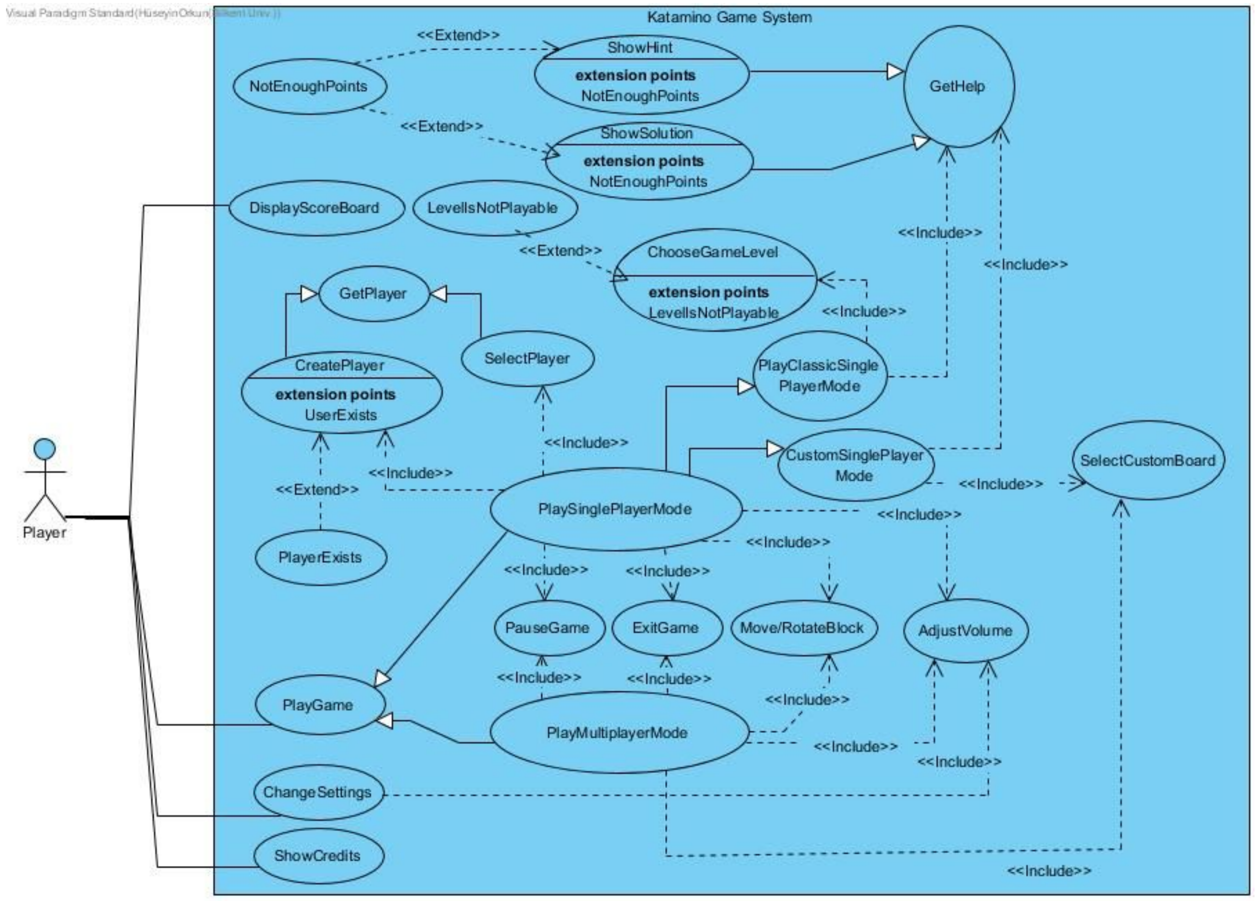
The game will be working correctly, and probability of failure (that is not caused by the environment) of any action in the game will be lower than 0.0001.

4.5 Maintainability

The game will be implemented in a manner that will be easy for others to understand and contribute to it. The documents such as design and analysis reports will be available to help others understand the implementation more easily.

5 System Models

5.1 Use-case Model



Use case name	Show Hint
Participating actors	Player
Flow of events	1. The Player activates show hints feature from his/her playground. 2. Game shows hints or display warning because of insufficient amount of points (NotEnoughPoints).

Entry condition	● The Player is currently in single player mode and playing the game.
Exit condition	● The Player knows where to put building block, OR ● The Player has received error message
Quality requirements	● None

Use case name	Show Solution
Participating actors	Player
Flow of events	1. The Player activates show solution feature from his/her playground. 2. Game shows whole solution or display warning because of insufficient amount of points (NotEnoughPoints).
Entry condition	● The Player is currently in single player mode and playing the game
Exit condition	● The Player knows solution, OR ● The Player has received error message
Quality requirements	● None

Use case name	Display Score Board
Participating actors	Player
Flow of events	1. The Player opens score board from his/her computer. 2. Game shows high score table.
Entry condition	● The Player is currently on the main game screen
Exit condition	● The Player is displayed with high scores
Quality requirements	● None

Use case name	Choose Game Level
Participating actors	Player
Flow of events	1. The Player opens game level screen. 2. The game shows available level screen. 3. The Player chooses desired level. 4. The game checks that the level is available. 5. The Player displays the game screen or gets an error because of level is not available (LevelsNotPlayable).

Entry condition	● The Player is currently in single player mode.
Exit condition	● The Game has been started, OR ● The Player has received error message.
Quality requirements	● None

Use case name	Create Player
Participating Actors	Player
Flow of events	1. The Player enters his/her username. 2. The game checks availability of the username and if username is available it creates the username, if not gives an error (PlayerExists).
Entry condition	● The Player is currently on main game screen.
Exit condition	● The Player’s username is created, OR ● The Player has received error message.
Quality requirements	● None

Use case name	Show Credits
Participating actors	Player
Flow of events	1. The Player requests the game credits. 2. Game displays credits.
Entry condition	● The Player is currently on the main game screen.
Exit condition	● The Player displayed credits.
Quality requirements	● None

Use case name	Play Game
Participating actors	Player

Flow of events	1. The Player activate PlayGame option. 2. The game presents PlaySinglePlayerMode and PlayMultiplayerMode. 3.The Player choose desired option between PlaySinglePlayerMode and PlayMultiplayerMode. 4. The game system starts the specified game. 5. Player plays game interacting with blocks (Move/RotateBlock). Moreover, the Player can use PauseGame to pause the ongoing game, ExitGame to exit the game and AdjustVolume to adjust music volume of the game. 6. The game responses to the Player in terms of specified feature. When Move/RotateBlock is called, the game shows last moved or rotated position on the screen, when PauseGame is called, the game pauses the ongoing game with all current play actions. When ExitGame is called, the game directs user to main game screen. When AdjustVolume is called, the game changes the output sound in terms of adjusted value by the user.
Entry condition	● The Player is currently on main game screen.
Exit condition	● The Player moved blocks, OR ● The Player paused the game, OR ● The Player exits from the game, OR ● The Player adjusts the current volume.
Quality requirements	● None

Use case name	Custom Single Player Mode
Participating actors	Player
Flow of events	1. The Player chooses Custom Single Player Mode. 2. The game shows available Custom Boards (SelectCustomBoard). 3. The Player chooses desired custom board. 4. The Player displays the game.
Entry condition	● The Player is currently in single player mode.
Exit condition	● The Game has been started, OR ● The Player has received error message.
Quality requirements	● None

Use case name	Change Settings
Participating actors	Player

Flow of events	1. The Player requests the settings. 2. The game displays settings. 3. The Player can interact with AdjustVolume and change volume of the game.
Entry condition	● The Player is currently on the main game screen, OR ● The Player is playing the game.
Exit condition	● The Player displayed settings, OR ● The Player adjusted game volume
Quality requirements	● None

5.4 Dynamic Models

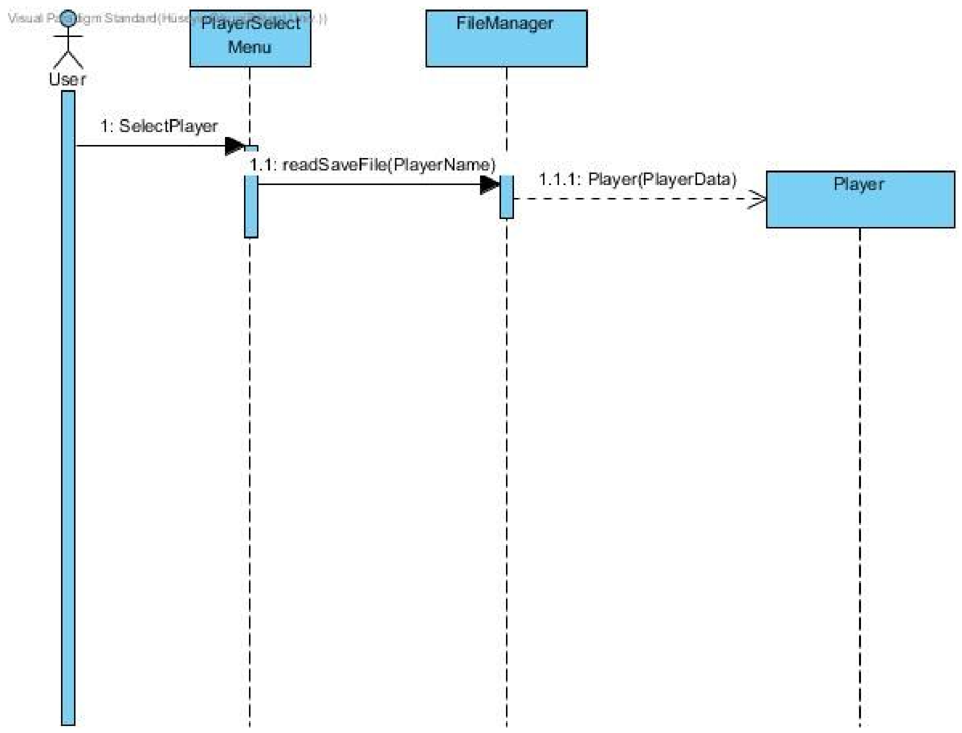
5.4.1 Scenarios & Sequence Diagrams

5.4.1.1 Create Player

After choosing the “Singleplayer” option in the main menu, user can add a new player for the game. In the next screen the user clicks to the plus button in the screen and then write the name of the new player to create a new player. The sequence is intuitive and similar to the Select player sequence given in section 5.4.1.2.

5.4.1.2 Select Player

User wants to select a player to load his/her progress. To select a player, the user chooses “Singleplayer” option from main menu. In the next screen, user selects a player from players list. The game loads player data by reading the save file of the selected user. The game data consists of current levels, score and progress in the game.

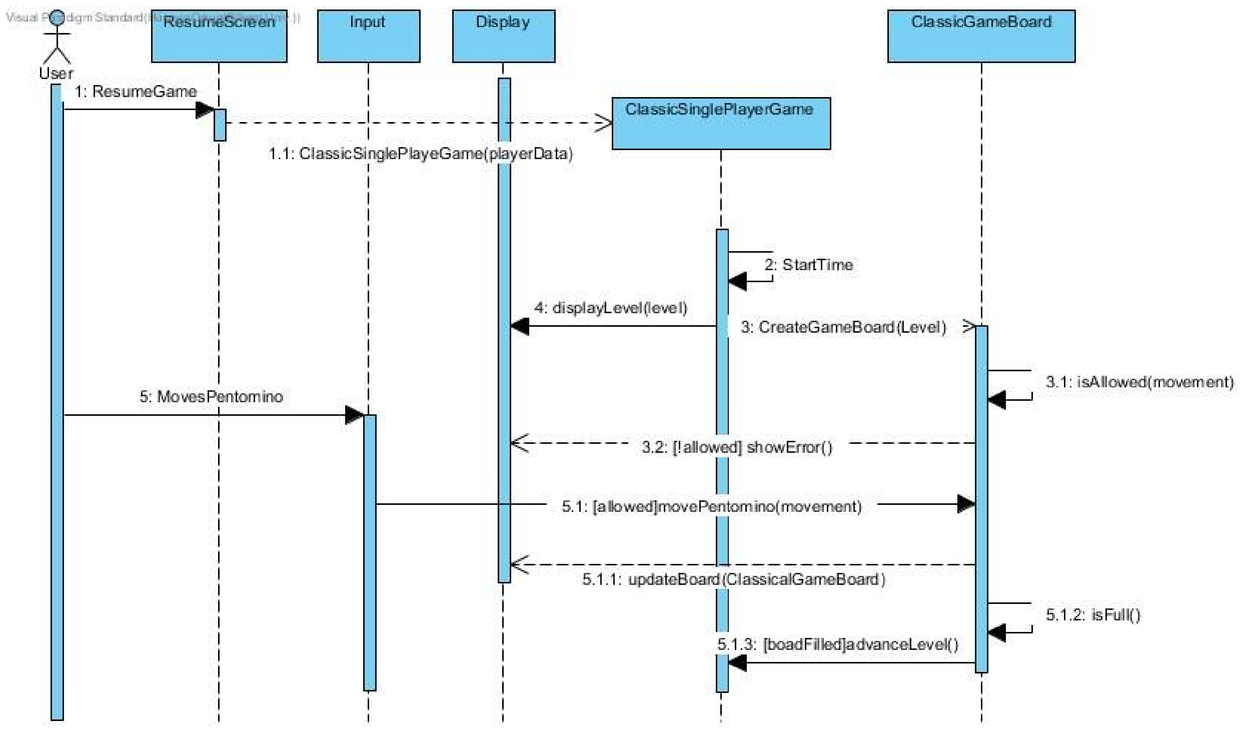


5.4.1.3 Settings

When the user selects settings from the game, pause menu or main menu, the game will display the settings menu. In the settings menu the user can adjust master volume, music volume or can mute the game. The user also can reset his/her progress to start over the game.

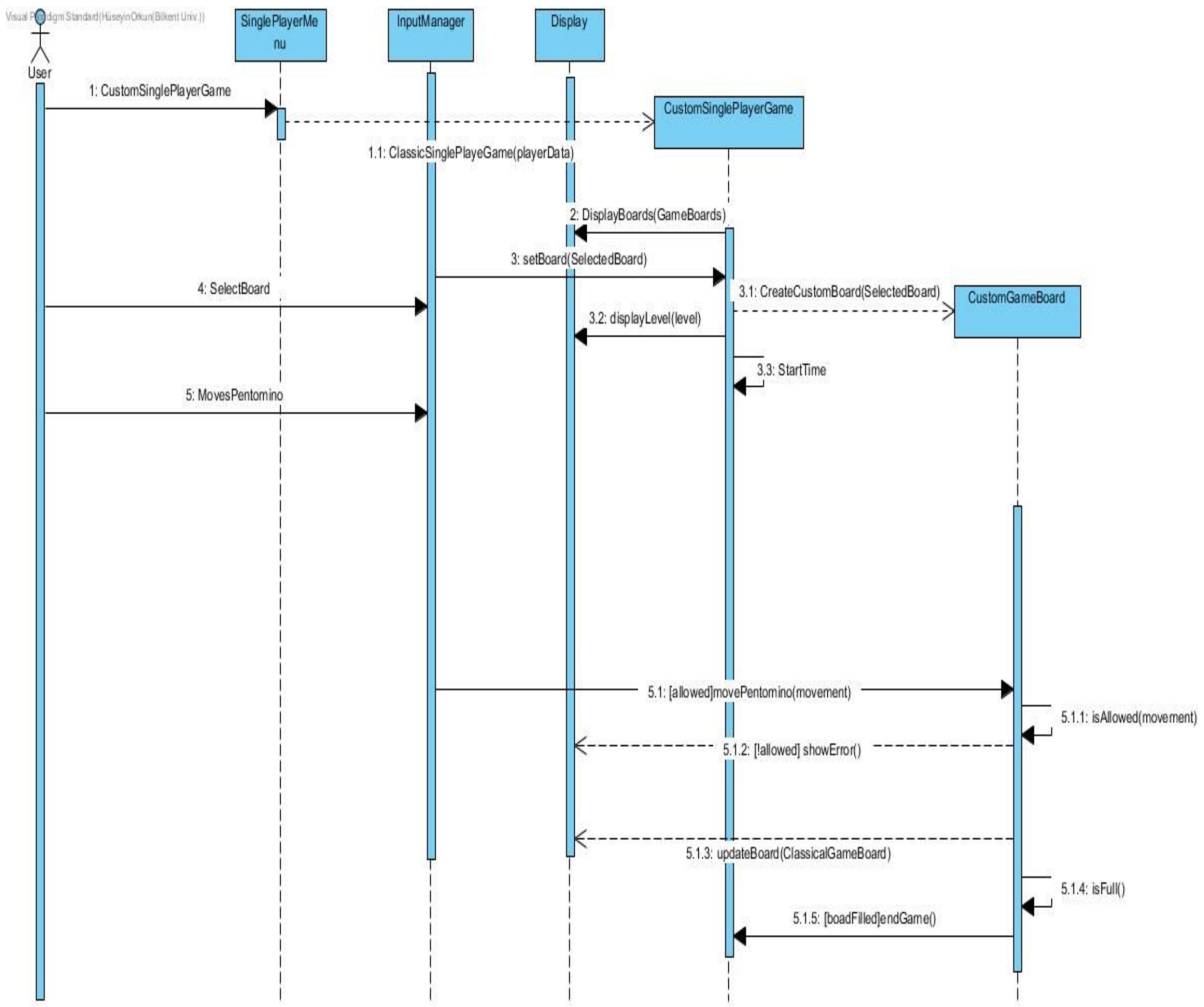
5.4.1.4 Play Classic Single Player Game

After user creates a player and chooses to play classic mode for katamino, player faces an option screen which he/she can choose to resume the earlier game or change level to be played. The player chooses to resume the game. Game loads the game board, the level's corresponding pentominoes pentominoes and starts time. The player moves, rotates and flips the pentomino then places the pentomino to the playing area on the board. Game checks if the movement allowed (clashes and moves that leave a part of pentomino outside of the game board are not allowed). If the move is allowed the display shows the pentomino on the board. When the playing area is full, the level ends and game loads the next level. Corresponding sequence diagram is given below.


User
ResumeScreen
Input
Display
ClassicGameBoard
ClassicSinglePlayerGame
1: ResumeGame
1.1: ClassicSinglePlayeGame(playerData)
2: StartTime
4: displayLevel(level)
3: CreateGameBoard(Level)
3.1: isAllowed(movement)
5: MovesPentomino
3.2: [!allowed] showError()
5.1: [allowed]movePentomino(movement)
5.1.1: updateBoard(ClassicalGameBoard)
5.1.2: isFull()
5.1.3: [boadFilled]advanceLevel()

5.4.1.5 Play Custom Single Player Game

The user selects "Singleplayer" option in the main menu, then chooses "Custom" in the pop-up. The user then chooses a customized board to play. The game loads the chosen board with the corresponding pentominoes. The player moves, rotates and flips the pentomino then places the pentomino to the playing area on the board. Game checks if the movement allowed (clashes and moves that leave a part of pentomino outside of the game board are not allowed). If the move is allowed the display shows the pentomino on the board. The game is finished when the board is filled with all the given pentominoes. The corresponding sequence diagram is given below.

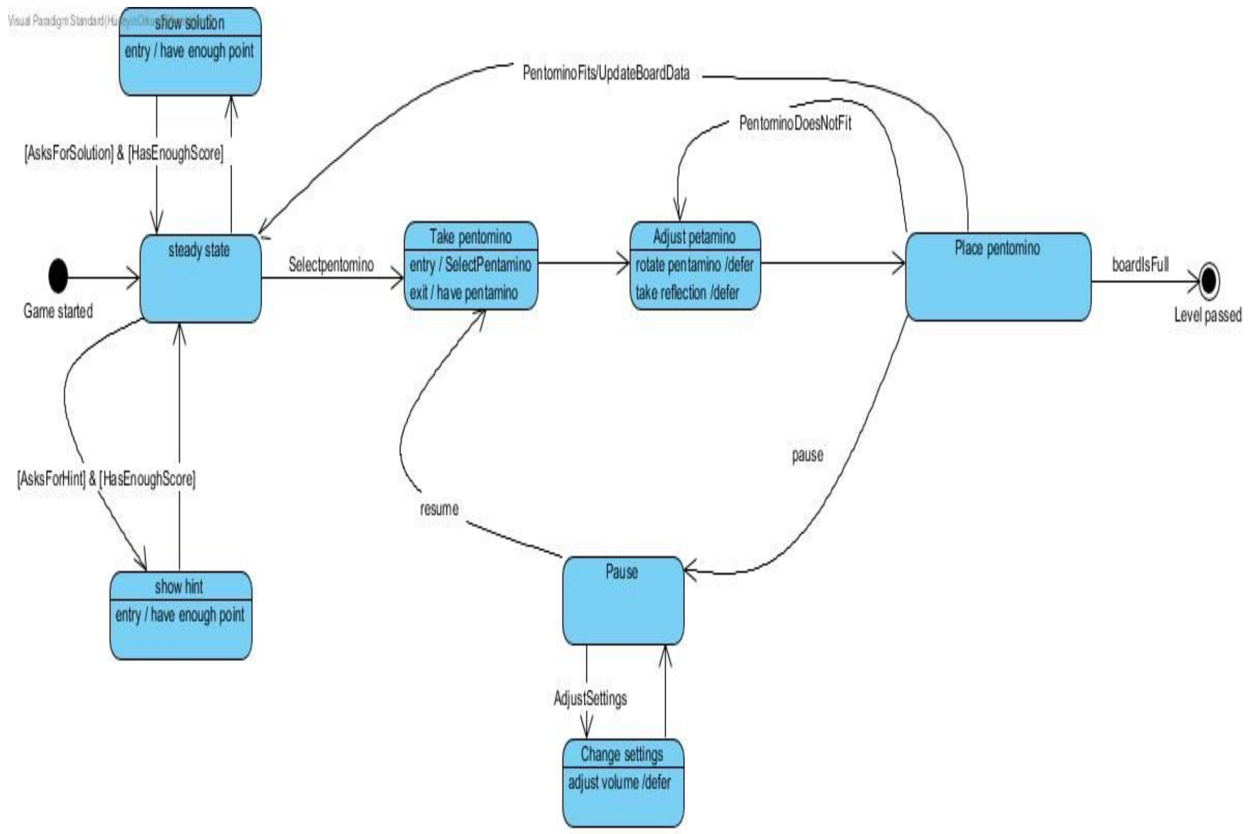

Visual Paradigm Standard(Hüseyin Orkun(Bilkent Univ.))
User
SinglePlayerMenu
InputManager
Display
CustomSinglePlayerGame
CustomGameBoard
1: CustomSinglePlayerGame
1.1: ClassicSinglePlayeGame(playerData)
2: DisplayBoards(GameBoards)
3: setBoard(SelectedBoard)
4: SelectBoard
3.1: CreateCustomBoard(SelectedBoard)
3.2: displayLevel(level)
3.3: StartTime
5: MovesPentomino
5.1: [allowed]movePentomino(movement)
5.1.1: isAllowed(movement)
5.1.2: [!allowed] showError()
5.1.3: updateBoard(ClassicalGameBoard)
5.1.4: isFull()
5.1.5: [boadFilled]endGame()

5.4.1.6 Play Multiplayer Game

The user selects “Multiplayer” option from the main menu. Next screen will present the game boards that are available, the user selects a board in this screen. After board selection, the game starts with an open board, and some pentominoes. The user rotates and places a pentomino to the board, the game checks if the move is allowed. If the move is allowed, the turn passes to the opponent. Turn changes until one player can not put a pentomino onto the board. In which case, that player loses and game ends. The sequence diagram is similar to the Custom Board Single Player Game, therefore not given here.

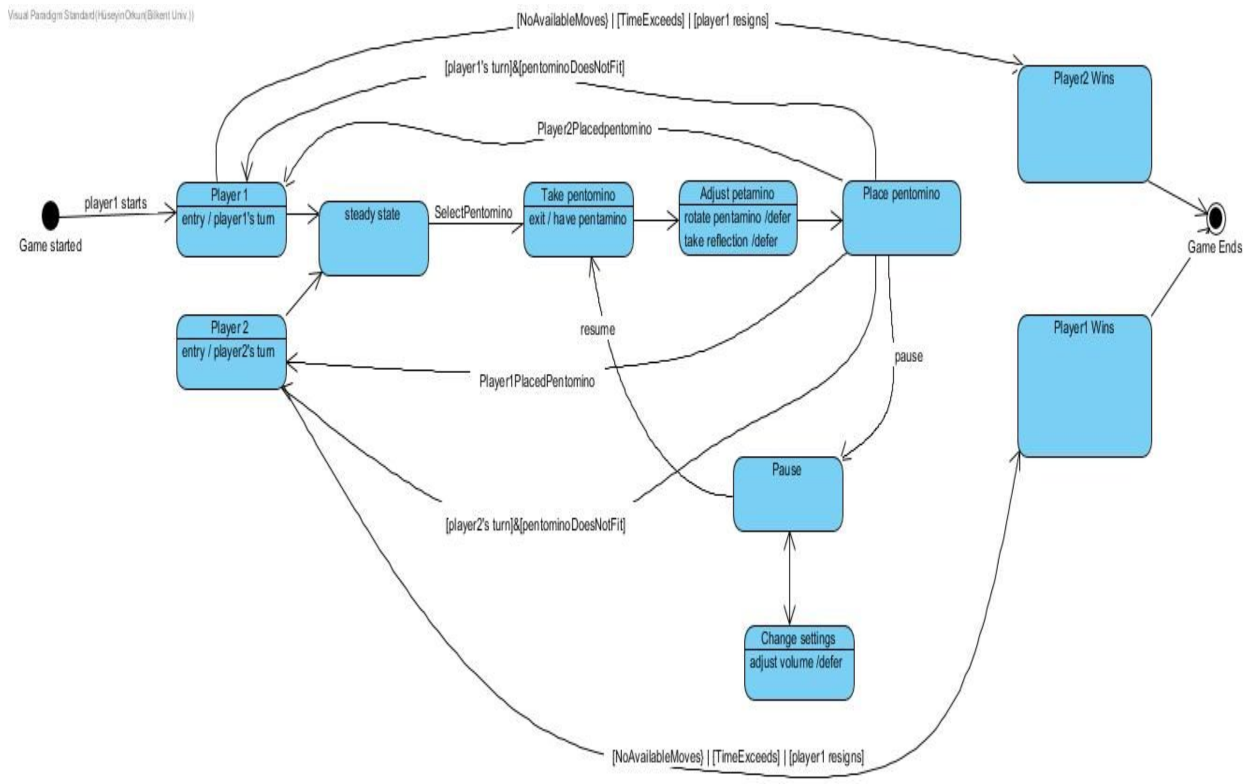
5.4.2 State Diagrams

5.4.2.1 Classic Single Player State Diagram



In classic single player mode. Player starts in a steady state where they ask for hint or solution, if they have sufficient score. Also, player selects a pentomino and adjusts it by moving rotating and taking reflection. Then state transitions into place pentomino state, if the pentomino fits, player can select another pentomino, if pentomino does not fit, player has to adjust and place the pentomino until it fits. When the board is full, level is passed.

5.4.2.2 Multiplayer State Diagram

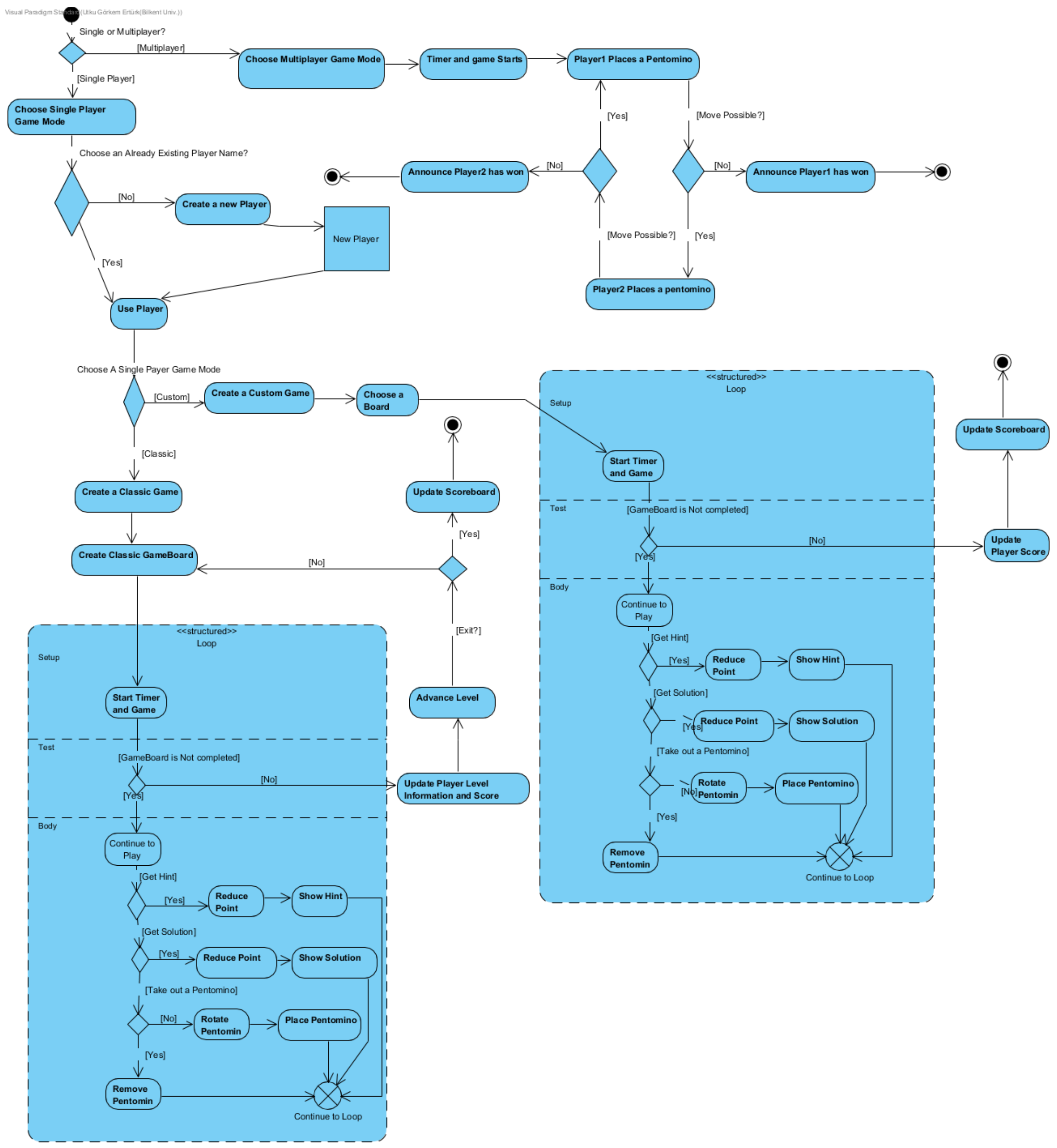


In multiplayer game, the game starts with player 1. Player1 selects and moves a pentomino, when player 1 places a pentomino and if the pentomino fits into the board, player 2 plays. If pentomino does not fit into the board, player 1 has to continue. If one of the players does not have an available move or they exceed the time limit or resign, opponent wins. Players can pause the game.

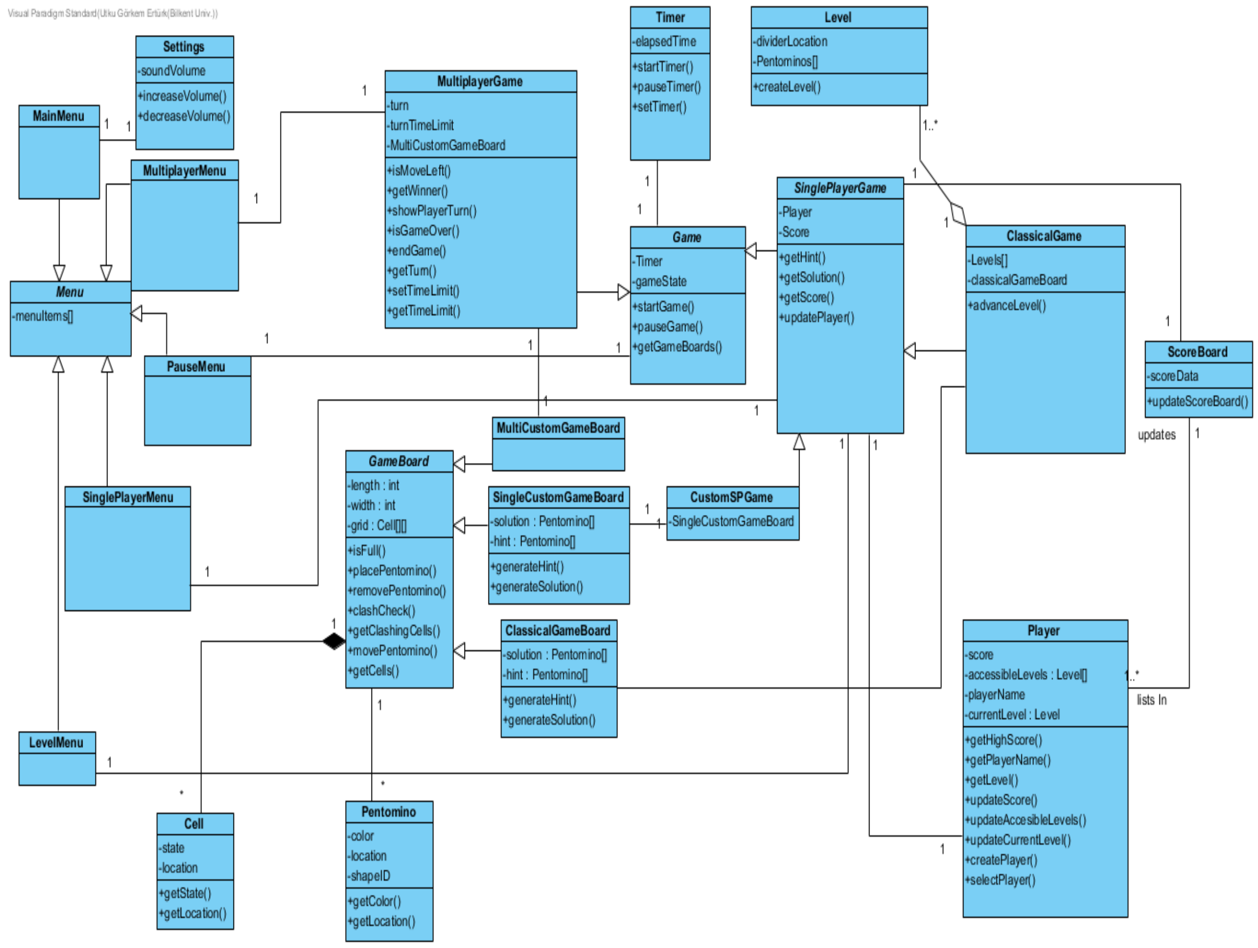
The Custom Single Player Game mode state diagram is very similar to this one therefore will be left out.

Activity Diagram

The activity diagram shows the options and possible actions of the user in the game.



5.6 Object and Class Models



Settings.class
This class controls game volume. *soundVolume* stores current level of sound. *increaseVolume()* increases volume and store the increased volume in *soundVolume* . decreaseVolume() decreases volume and store the decreased volume in *soundVolume*.

GameBoard.class
GameBoard is the main playground of the game and this class is an abstract class that is parent of *MultiCustomGameBoard.class*, *SingleCustomGameBoard.class* and *ClassicalGameBoard.class* to give main capabilities of a basic game board. *length* and width attributes stores the dimensions of the gameboard. *grid* attribute stores the locations of the *Cell*s in 2d dimension. *isFull()* checks if the board full with pentominoes. *placePentomino()* places selected pentominoes to the game board. *removePentomino()* removes placed pentomino on the gameboard. *movePentomino()* make the pentominoes move using JavaFx and input

libraries. *clashCheck()* checks if dropped pentominoes clashes with another pentomino on the gameboard. *getClashingCells()* returns classing cells. *getCells()* returns grid attribute.

MultiCustomGameBoard.class, SingleCustomGameBoard.class and ClassicalGameBoard.class
These all classes specialized version for game board. *SingleCustomGameBoard.class* and *ClassicalGameBoard.class* have solution and hint attributes to store hint and solution. *generateHint()* returns random *Pentomino* objects that lead a solution. *generateSolution()* returns whole *Pentomino* object array that contains a solution. Since in multiplayer game does not depend on the solving of the current puzzle but make your opponent out of move, there is no solution and hint related attributes or methods.

MultiplayerGame.class
This class determines the mechanism of the game. *turn* attribute stores which player's turn it is. *turnTimeLimit* determines the upper limit for a player to play the game session. *MultiCustomGameBoard* attribute contains game board and its controls.*isGameOver()* internally calls MoveLeft() ,which calculates if there is a move left, *getWinner()* ,which determines the winner by looking game status using time and turn, and *endGame()* ,which finishes the game and call menu screen. *showPlayerTurn()* , *getTurn()*, *getTimeLimit()* and *setTimeLimit()* function as their names implies.

SinglePlayerGame.class
This class parent of CustomSPGame and CassicalGame that single player game mechanism occurs. *CustomSpGame.class* and *ClassicalGame.class* differs in terms of level advancement and shape so that there is no need to explain these classes separately. *updatePlayer()* updates player attributes in terms of game status.

Pentomino.class
Pentomino object is building block that placed by player on *GameBoard* object. *color* attribute stores color of the pentomino. *location* attribute stores where pentomino placed on game screen. *shapeID* attribute stores unique key of a pentomino object. Get functions returns specified attributes at their name.

Cell.class
Cell class represents squares of *GameBoard.class***.** *location* attribute stores the location of the cell on the GameBoard object. *state* attribute stores if cell is filled or not by a piece of *Pentomino* object. *getState()* returns *state* attribute and *getLocation()* returns *location* attribute.

Player.class
score keeps total score of the *Player*. *accesibleLevels* stores played and playable levels of the *Player*. *currentLevel* stores last played level. playerName is used for store name to use in high score and game screen. Update methods are called by the *SinglePlayerGame* objects at the end of the game. Get except high score, select and create methods are called by the

SinglePlayerGame objects before game is started.*getHighScore()* called by *ScoreBoard* to display scores.

5.7 User Interface

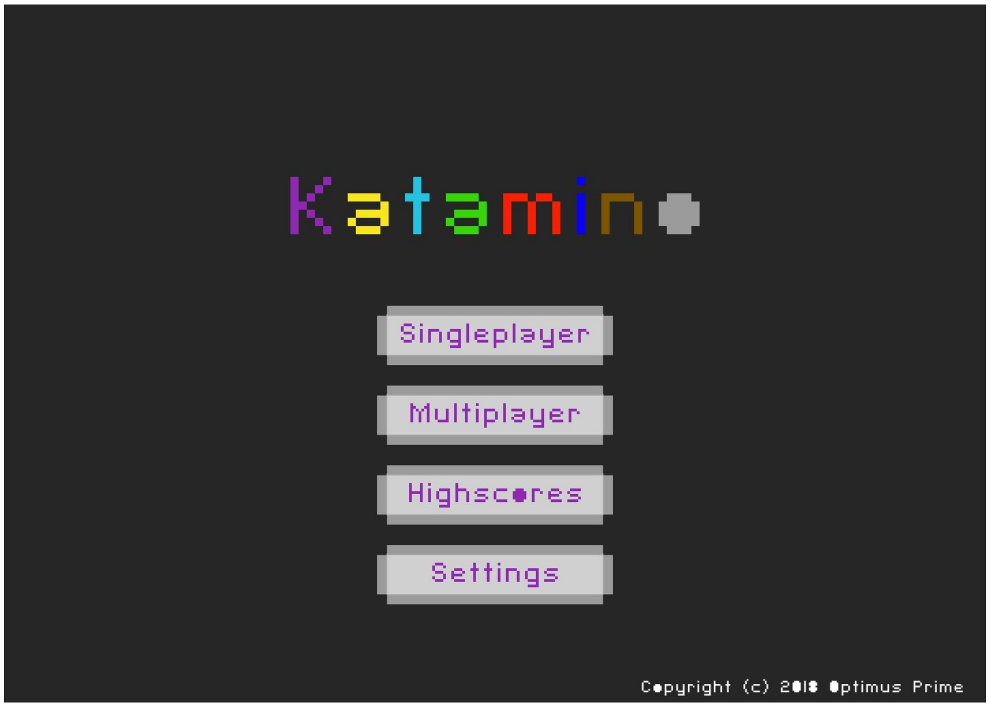


Figure 1: Main menu screen

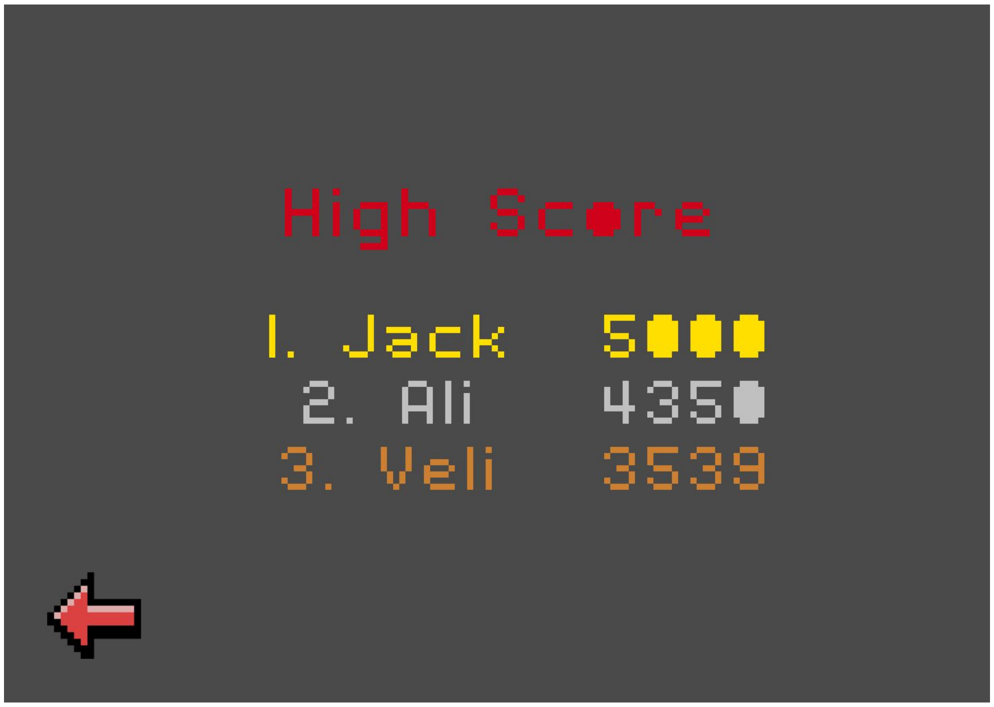


Figure 2: High Score

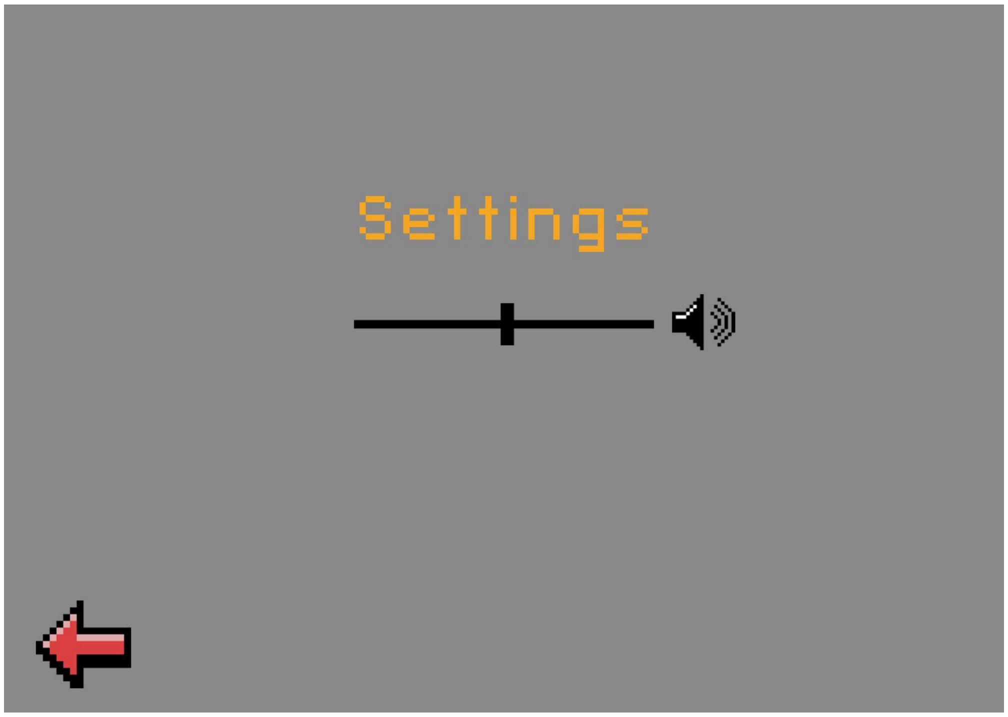


Figure 3: Settings

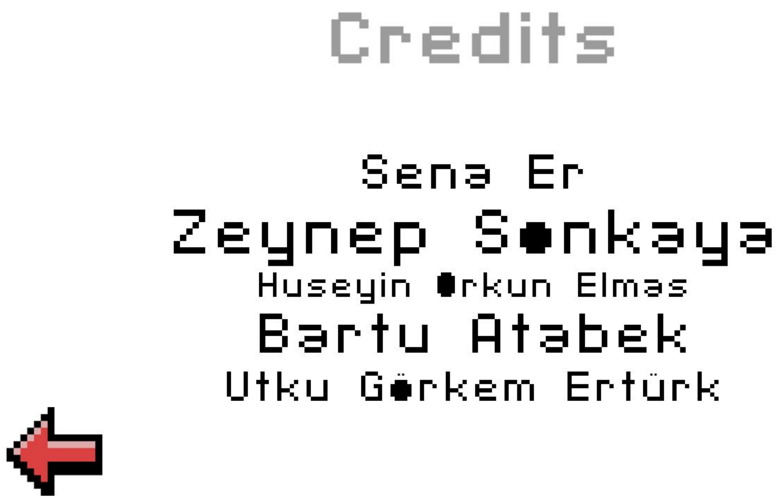


Figure 4: Credits

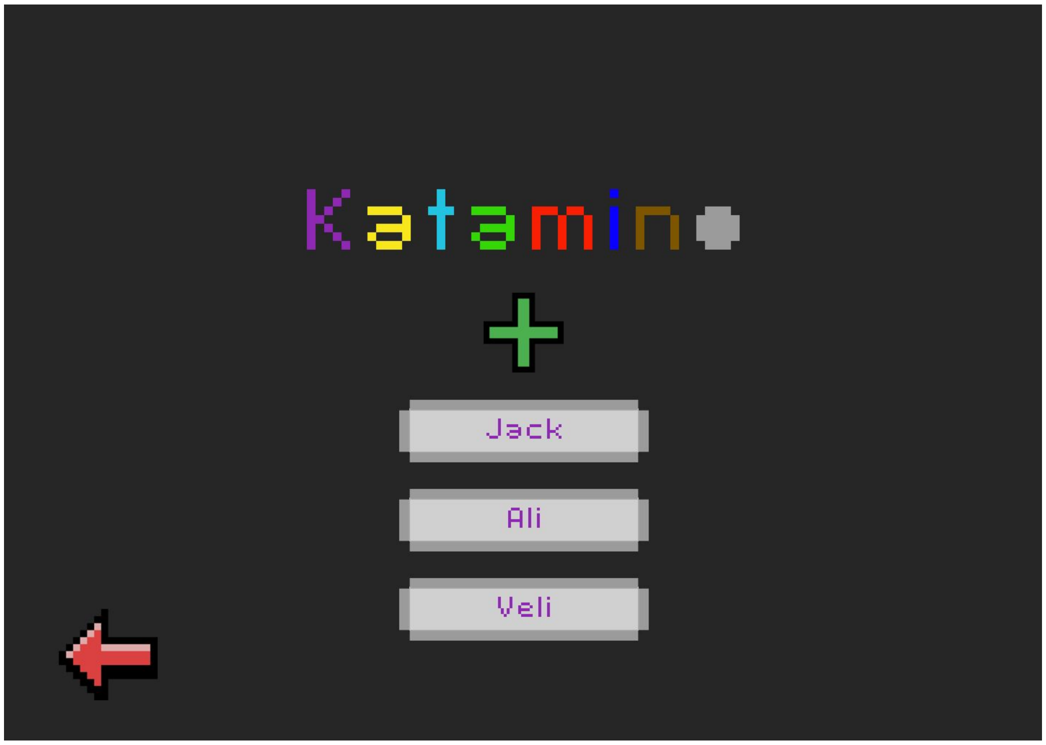


Figure 5: Single player user selection or creation

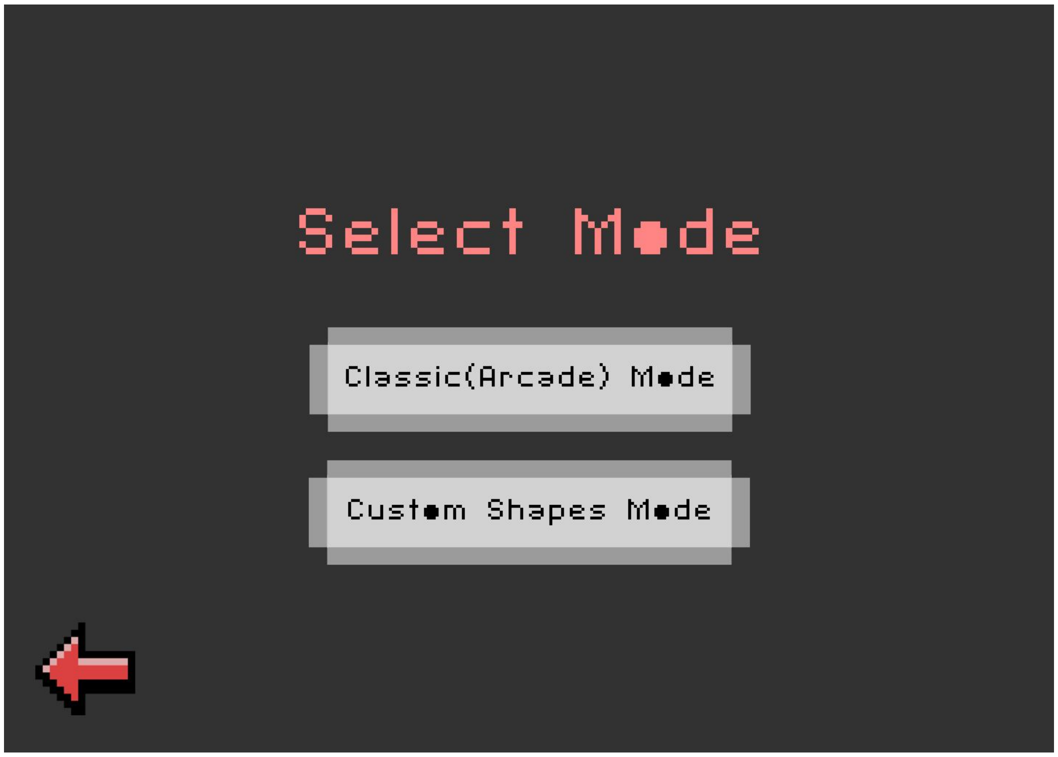


Figure 6: Single player select mode

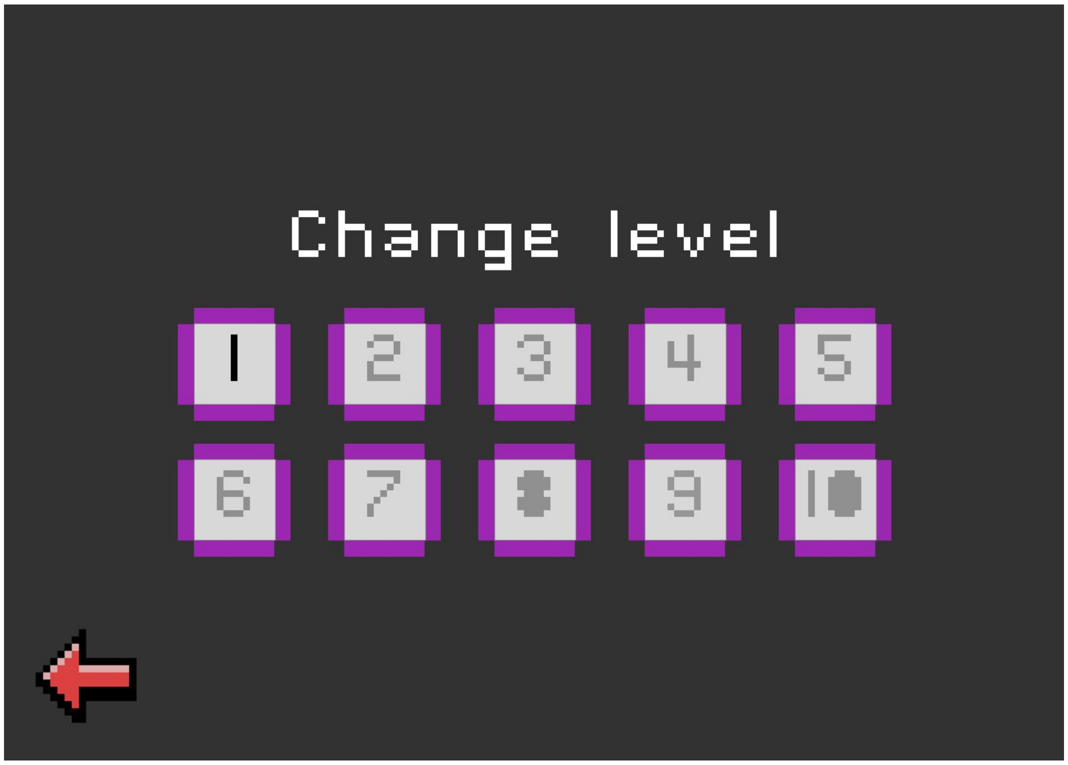


Figure 7: Single player Arcade mode level change

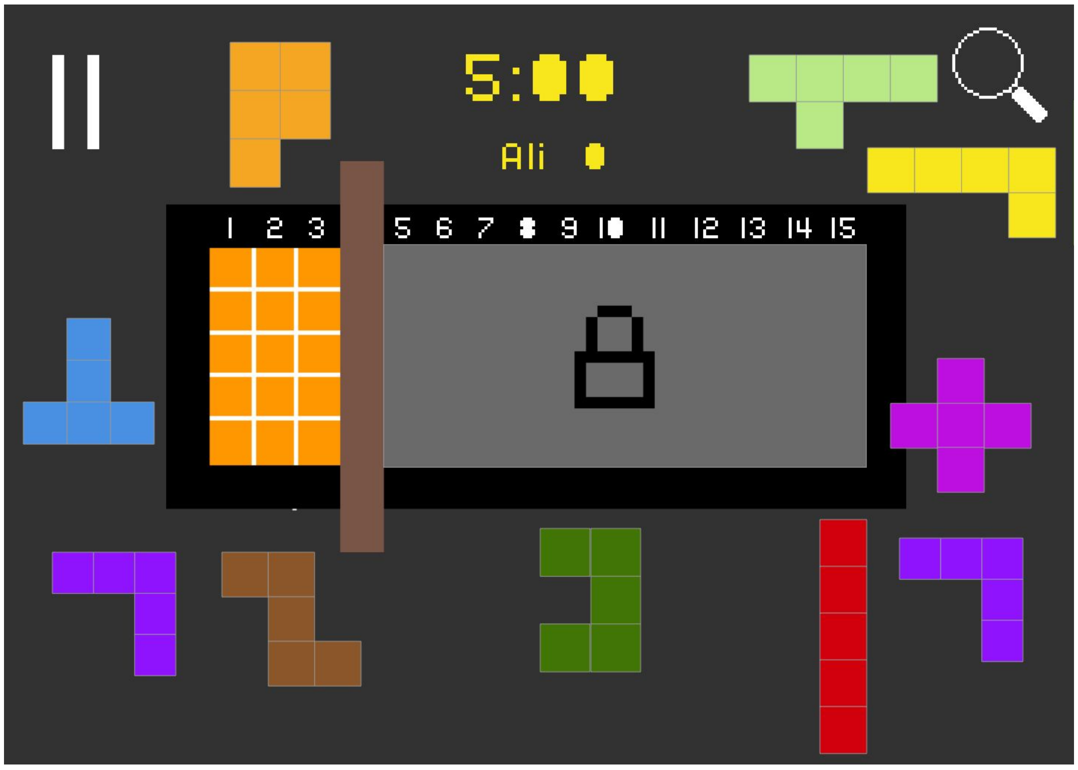


Figure 8: Single player classic(arcade) mode

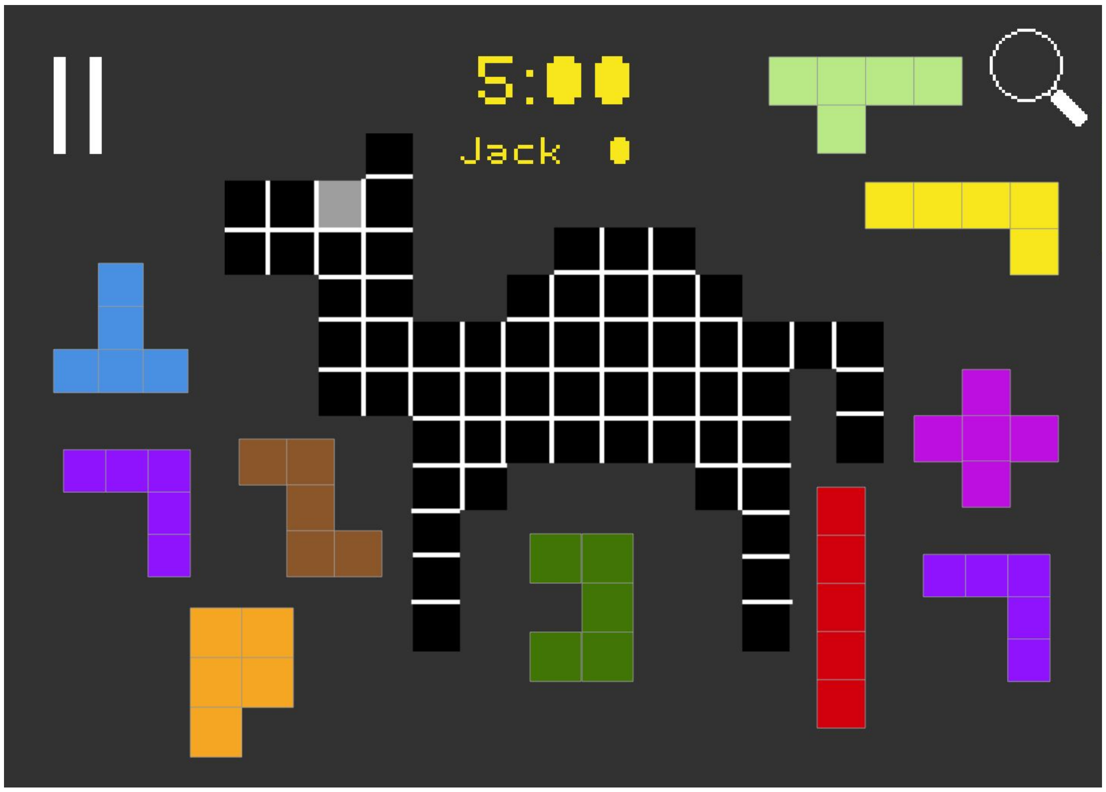


Figure 9: Single player custom shape mode

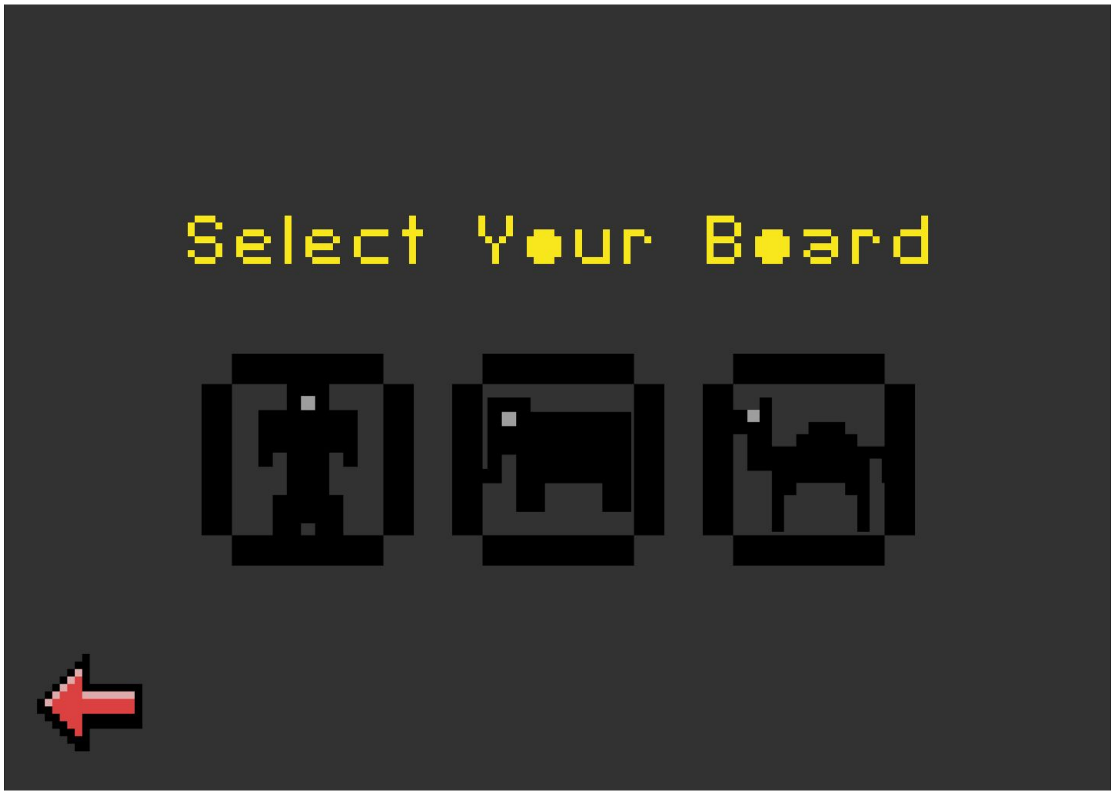


Figure 10: Select board for multiplayer and single player custom shapes mode

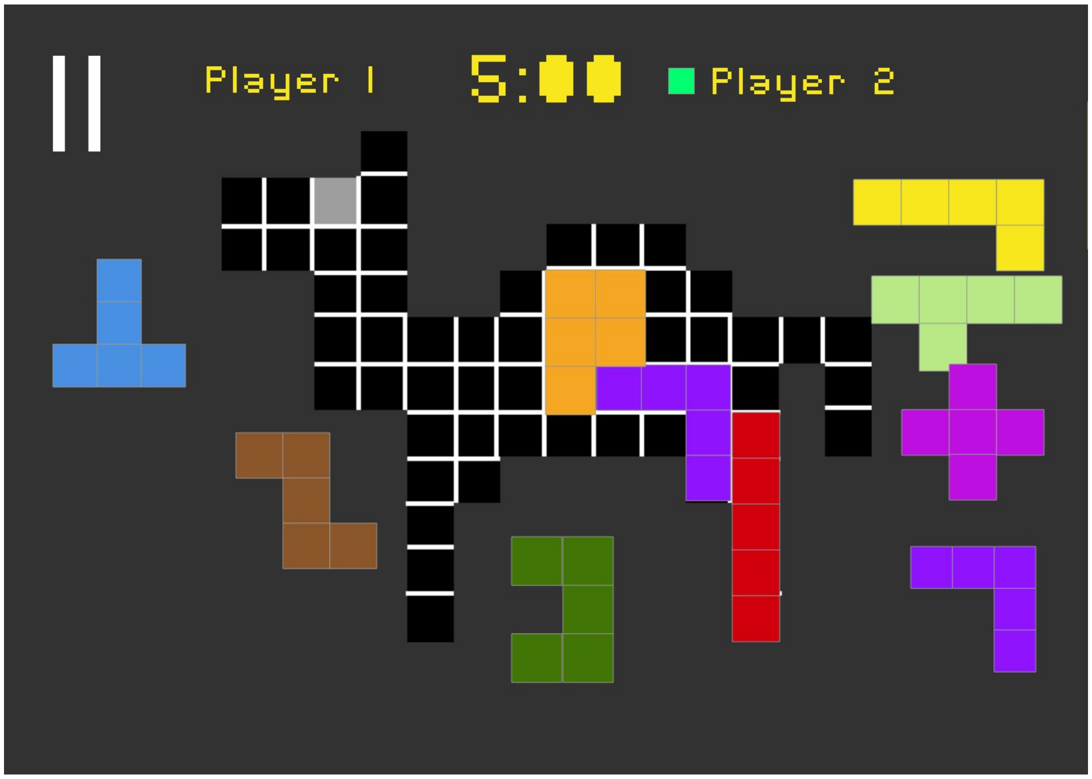


Figure 11: Multiplayer mode

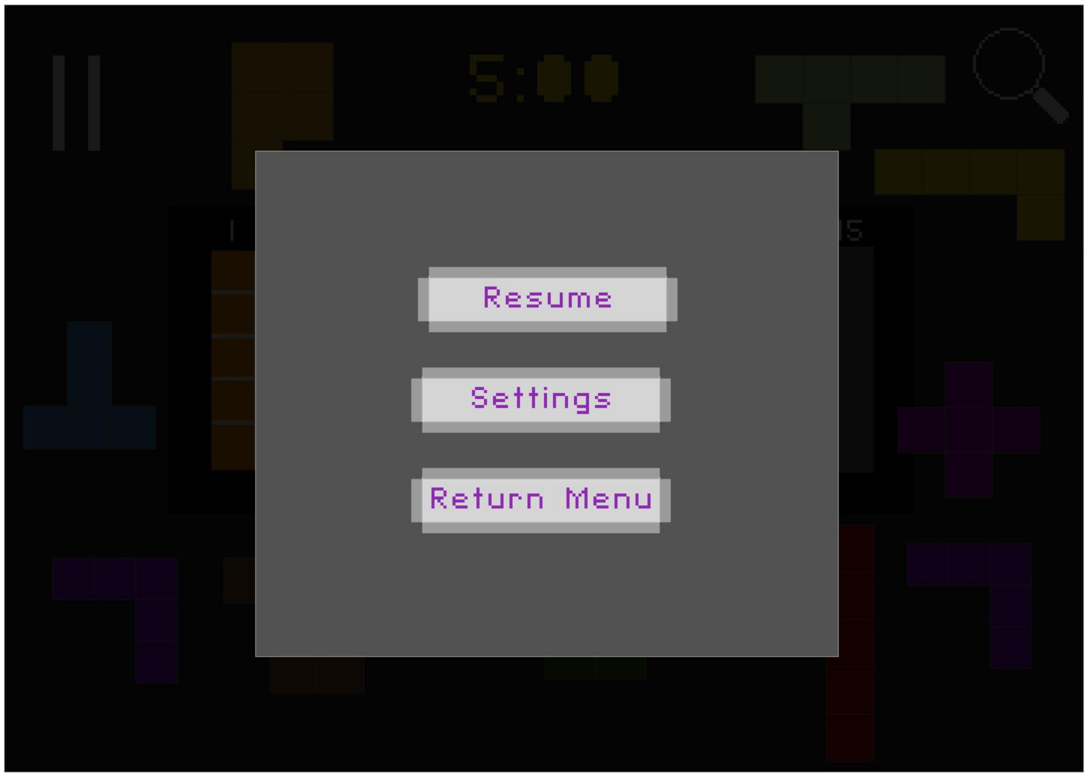


Figure 12: Pause menu

6 Glossary & References

References

[1]"Katamino", *BoardGameGeek*, 2018. [Online]. Available: https://boardgamegeek.com/boardgame/6931/katamino. [Accessed: 20- Oct- 2018].

[2]P. Games, "Katamino", *En.gigamic.com*, 2018. [Online]. Available: http://en.gigamic.com/game/katamino. [Accessed: 20- Oct- 2018].

[3]"TEACHER SHEET", *En.gigamic.com*, 2018. [Online]. Available: http://en.gigamic.com/files/media/fiche_pedagogique/educative-sheet_katamino-english_01-2014.pdf. [Accessed: 20- Oct- 2018].

[4]"Java SE | Oracle Technology Network | Oracle", *Oracle.com*, 2018. [Online]. Available: https://www.oracle.com/technetwork/java/javase/overview/index.html. [Accessed: 20- Oct- 2018].

[5]”Pentomino”, Wikipedia.com, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Pentomino. [Accessed 20- Oct- 2018]