Bilkent University

Department of Computer Engineering

CS319
Object Oriented Software Engineering
Project Analysis Report

Katamino 2.0

Group 2-F

Sena Er - 21502112

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 transform the classic board game to the virtual plane and add extended capabilities and features such as custom and larger game boards with more blocks, multiplayer, scoreboards and customization options and many more in order to make it more attractive for the players.

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.

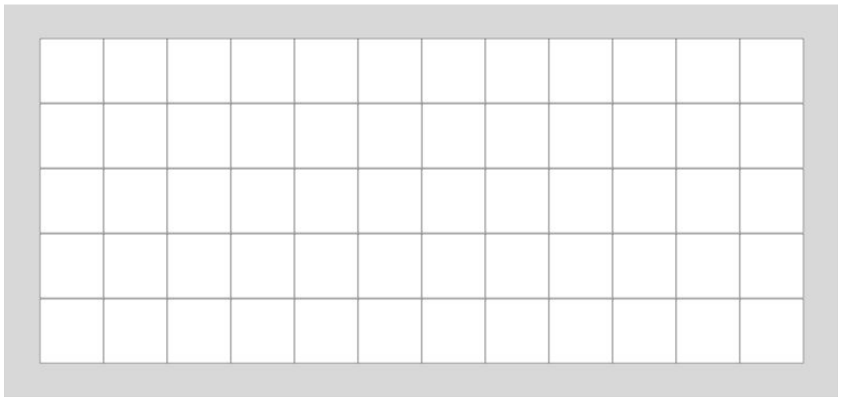


Figure 1: Sample game board for the arcade mode

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.

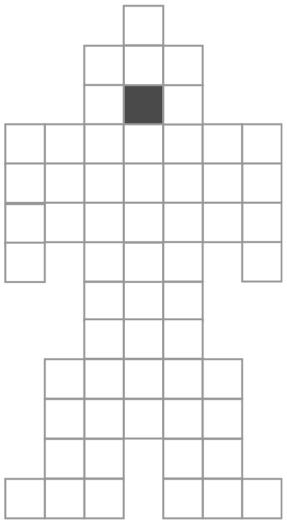


Figure 2: Sample custom game board with the shape of a cyclops.

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]

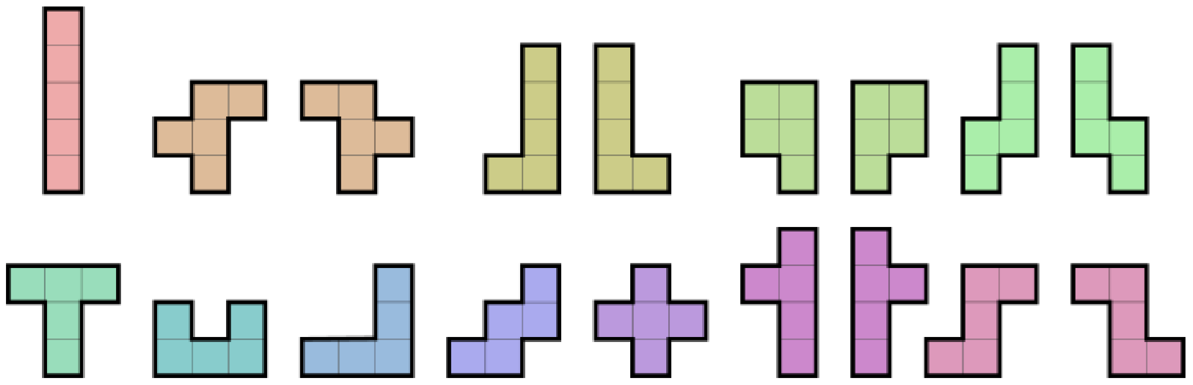


Figure 3: The 12 pentominoes (from left to right I, F, L, P, S, T, U, V, W, X, Y, Z) can form 18 different shapes, with 6 of them being mirrored.

- 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:

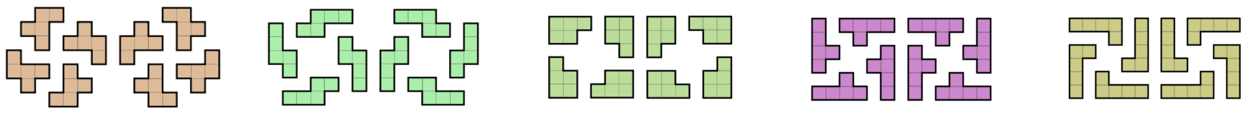


Figure 4: The eight possible orientations of the L, F, S, P, and Y pentominoes.

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.

3.5 Additional requirements

Customizable game board: Game board can be customized in terms of pentominoes and the board itself. This function is available for both singleplayer and multiplayer game modes.
Daily challenges: Game will set daily challenges among randomly selected maps for custom shapes game mode.

4. Non-Functional Requirements

4.1 Performance

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 second.
- Level loading must take less than 0.5 second.
- In both multiplayer and singleplayer game all movements of a pentomino must take less than 0.1 second.
- In multiplayer game and custom single player game, loading a board must take less than 0.5 second.
- When exiting the game in a single game, previous menu must load before 0.5 second.

4.2 Usability

- The interface of the game will be designed such that a new player will be able to learn the control features of the application without consulting other people or user's guide in 10 second after he/she execute the program. For that the controls that player interacts with will be designed intuitively, in order to prevent confusion.
- A new player will be able to play a Katamino game without any guidance in 2 second after he/she start the game.

4.3 Data Integrity

- The game will handle user data accurately, authentically and without corruption. Files that containing game related informations will be checked every time game is launched.
- The user could load the data that is saved by the game, and continue their progress. The user data will be updated 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 other programmers to understand and contribute to it. The documents such as design and analysis reports will be available to help maintenance personnel to revise or enhance the implementation .

5 System Models

5.1 Use-case Model

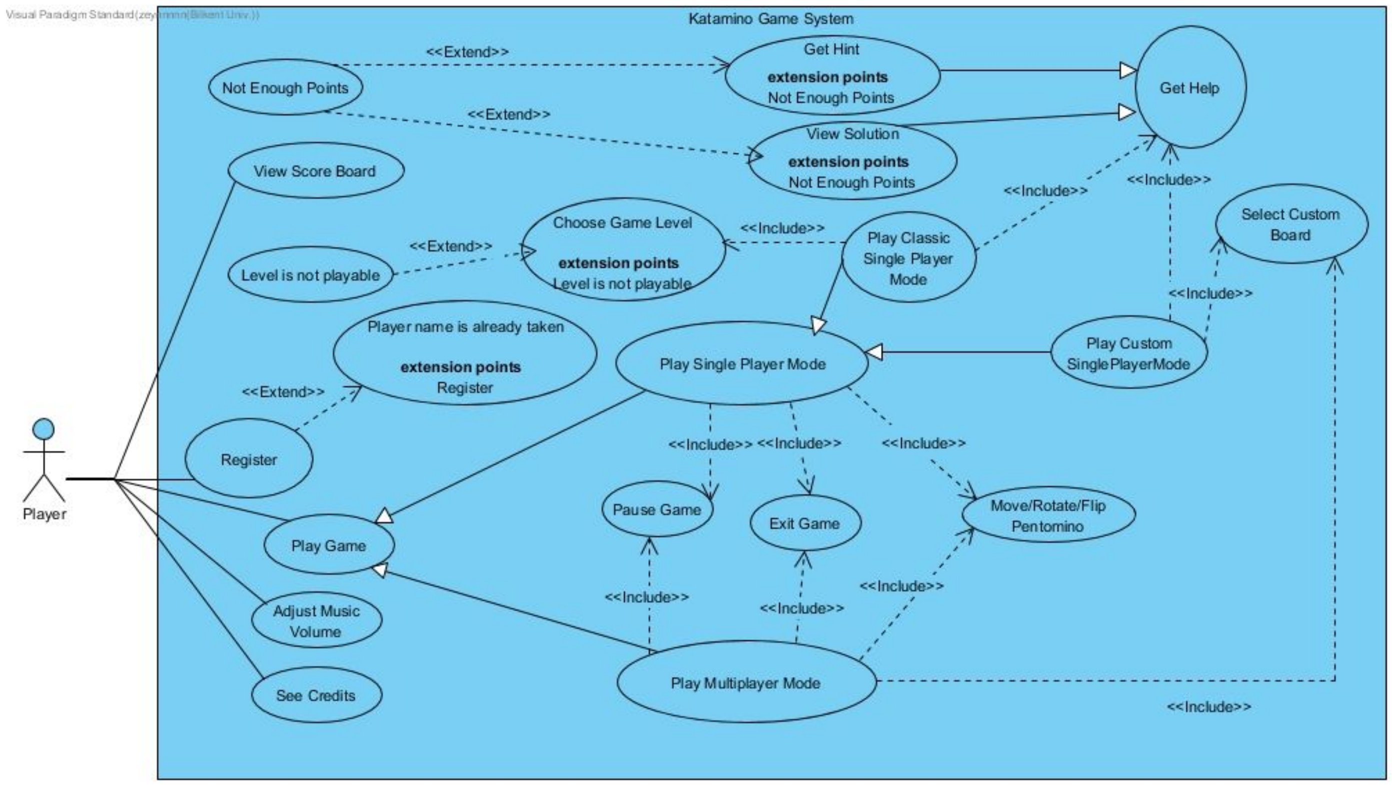


Figure 5: Use Case Diagram

Some use cases in the use case diagram are not represented in text format, since their details can be understood directly from their names and relationships with other use cases.

Use case name	Get Help
Participating actors	Player
Flow of events	1. The Player asks for help during the game via help button. 2. System gives the wanted help.
Entry condition	● Player is playing Single Player game mode,

	● Player has enough points.
Exit condition	● Player recieved help.

Use case name	Get Hint
Participating actors	**Inherited** from GetHelp use case
Flow of events	1. The Player ask for help via ShowHint button. 2. System calculates and deducts points and shows a hint afterwards. Points corresponding for the hint are deducted from the players score.
Entry condition	**Inherited** from GetHelp use case
Exit condition	**Inherited** from GetHelp use case

Use case name	View Solution
Participating actors	**Inherited** from GetHelp use case
Flow of events	1. The Player ask for help via ShowSolution button. 2. System shows the solution. Points corresponding for the solution are deducted from the players score.
Entry condition	**Inherited** from GetHelp use case
Exit condition	**Inherited** from GetHelp use case

Use case name	Not Enough Points
Participating actors	Player
Flow of events	1. Game gives a warning stating the Player does not have enough points.
Entry condition	This use case **extends** Show Solution and Show Hint use cases. It is initiated by the system whenever the player does not have enough points to get help but asks for it.
Exit condition	Player has received and acknowledges an error message stating that Player does not have enough points.

Use case name	View Score Board
Participating actors	Player

Flow of events	1. The Player opens score board from their main screen . 2. System access save files and shows high score table of the players where each player is ranked by their score.
Entry condition	The Player is currently on the main game screen.
Exit condition	The Player has viewed the high score board.

Use case name	Select Custom Board
Participating actors	Player
Flow of events	1. The Player opens custom board selection screen. 2. The game shows available boards. 3. The Player chooses desired board. 4. The game loads selected board into the game and game begins .
Entry condition	The Player is currently in custom single player mode.
Exit condition	The Game has been started, OR The Player has received error message.

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 for the player. If so the selected level loads into the game board and game begins .
Entry condition	The Player is currently in classic single player mode.
Exit condition	The Game has been started, OR The Player has received error message.

Use case name	Level is Not Playable
Participating actors	Player
Flow of events	1. Game gives a warning stating the Player can not play the selected level yet.
Entry condition	This use case **extends** Choose Game Level use case. It is initiated by the system when player try to access a level that is higher than his accessibility level.

Exit condition	Player has received and acknowledges an error message stating that Player does not have enough points.

Use case name	Register
Participating Actors	Player
Flow of events	1. Player enters a name to the field asks for a name. 2. System gets the name and checks if the player name already exist or not in the system files. If name does not exist in the file system, game creates a player with the name and shows to the player level selection screen. If name exist, game will give an error (PlayerNameIsAlreadyTaken Use Case).
Entry condition	The Player is currently on level or board selection screen of single player game according to the game mode.
Exit condition	The user has a player account and continues to play OR The user has received error message.
Quality requirements	User register a player account in less than 1 seconds.

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 has displayed credits.

Use case name	Play Classic Single Player Mode
Participating actors	Player

Flow of events	1. The Player chooses classic single-player option from the mode selection menu. 2. The game displays available levels. (**includes** use case ChooseGameLevel) 3. The player chooses an available level. 4. The game loads the game board with corresponding pentominoes and starts the stopwatch. 5. Player places pentominoes on the board. 6. Game is won when the pentominoes are all placed and filled the board. 7. The board is updated with the next level and also the player's score.
Entry condition	The Player is currently in single player mode.
Exit condition	● The Player exits the game
Quality requirements	The Play Classic Single Game can **include** Exit Game, Pause Game, move/rotate/flip blocks, Adjust Volume use cases in any time of the event flow.

Use case name	Play Custom Single Player Mode
Participating actors	Player
Flow of events	1. The Player selects Custom Single Player mode. 2. The game shows available custom boards. (**include** use case SelectCustomBoard) 3. The Player chooses one of the custom boards. 4. The game shows the selected custom board. 5. The Player moves,rotate and drags the the pentominoes onto the board until filling the custom board completely. 6. Game checks if the board is filled if so Player's score is updated and game displays a message.
Entry condition	The Player is currently in Custom Single Player mode.
Exit condition	● The Player exits the game prematurely, OR ● The Player receives the "Congratulations" message.
Quality requirements	The Play Classic Single Game can **include** Exit Game, Pause Game, Move/Rotate/Flip Blocks,Get Help use cases.

Use case name	Play Multiplayer Game
Participating actors	Player
Flow of events	1. The Player chooses PlayMultiplayerMode. 2. The game presents custom game boards. (**include** use case SelectCustomBoard) 3. The Player chooses the desired game board. 4. The game loads and time start for first Player. 5. First Player plays game interacting with blocks and places a pentomino. 6. The game changes turns between the players, stops the time and check if any possible movement of pentomino left. If there is, other player turn begins. 7. Second Player plays game interacting with blocks and replaces a pentomino. 8. 5., 6. And 7. Event repeated. Whenever there was no possible move of pentominoes left, game is over. System declares the winner.
Entry condition	The Player is currently on main game screen.
Exit condition	● The game is over with a winner, OR ● The Player exits the game prematurely,
Quality requirements	The Play Multiplayer Game can **include** Exit Game, Pause Game, move/rotate/flip blocks use cases in any time of the event flow.

Use case name	Pause Game
Participating Actors	Player
Flow of events	1. Player pauses the game. 2. Game pauses time, blurs the screen and shows the pause menu.
Entry condition	The player is playing the game.
Exit condition	Game displays pause menu.

Use case name	Exit Game
Participating Actors	Player
Flow of events	1. Player activates exit game function.. 2. The game loads the previous menu, saves the current board and the pentominoes.

Entry condition	The player is playing the game.
Exit condition	Game displays previous menu.

Use case name	Move/Rotate/Flip Blocks
Participating Actors	Player
Flow of events	1. Player selects a pentomino from the game. 2. The game highlights the pentomino block. 3. Player drags the pentomino over the game board, rotates and flips it by clicking on the corresponding buttons. 4. Game previews the current state of the pentomino block on the screen.
Entry condition	The player is playing the game.
Exit condition	The player dropped (placed) the pentomino block in the game board.

Use case name	Adjust Music Volume
Participating actors	Player
Flow of events	1. The Player requests the settings. 2. The game displays settings menu. 3. The Player can interact with volume slider and adjust the 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 the game volume
Quality requirements	The volume slider must have high accuracy and precision for the music level.

5.4 Dynamic Models

5.4.1 Scenarios & Sequence Diagrams

5.4.1.1 Select Player

User wants to select a player to load his/her progress. The user chooses “Singleplayer” option from main menu, by that user need to select a player. In the next screen, user creates a new player or selects a player from players list.In Sequence Diagram,player creates a new player. 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. System uses the data in Game initialization.

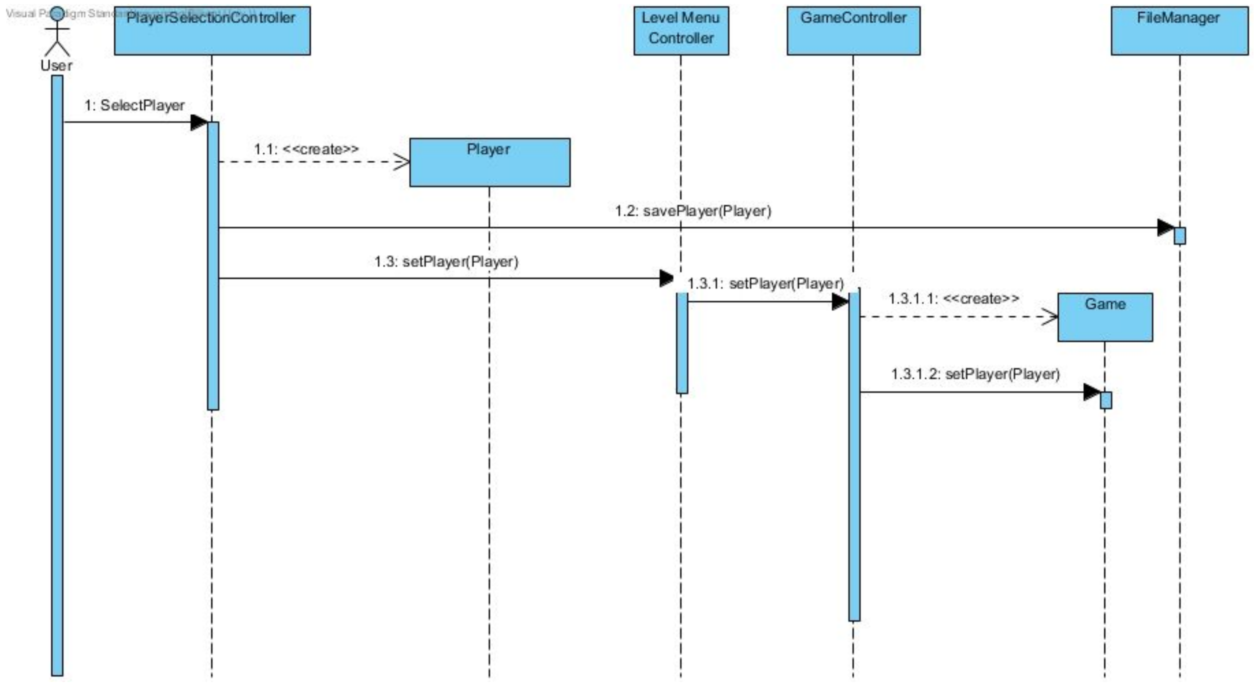


Figure 6: Select Player Sequence Diagram

5.4.1.2 Play Classic Game

After user creates a player and chooses to play classic mode for katamino, player faces an option screen which he/she can choose level to be played. Player chooses a level to play.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.

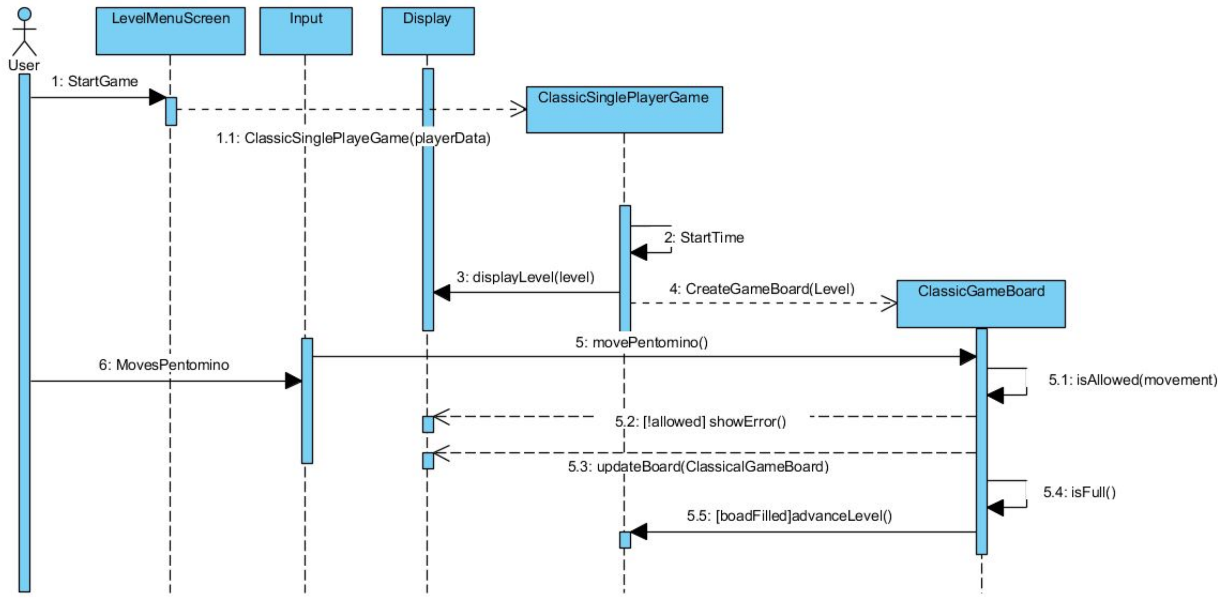


Figure 7: Play Classic Game Sequence Diagram

5.4.1.3 Play Custom Shape 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.

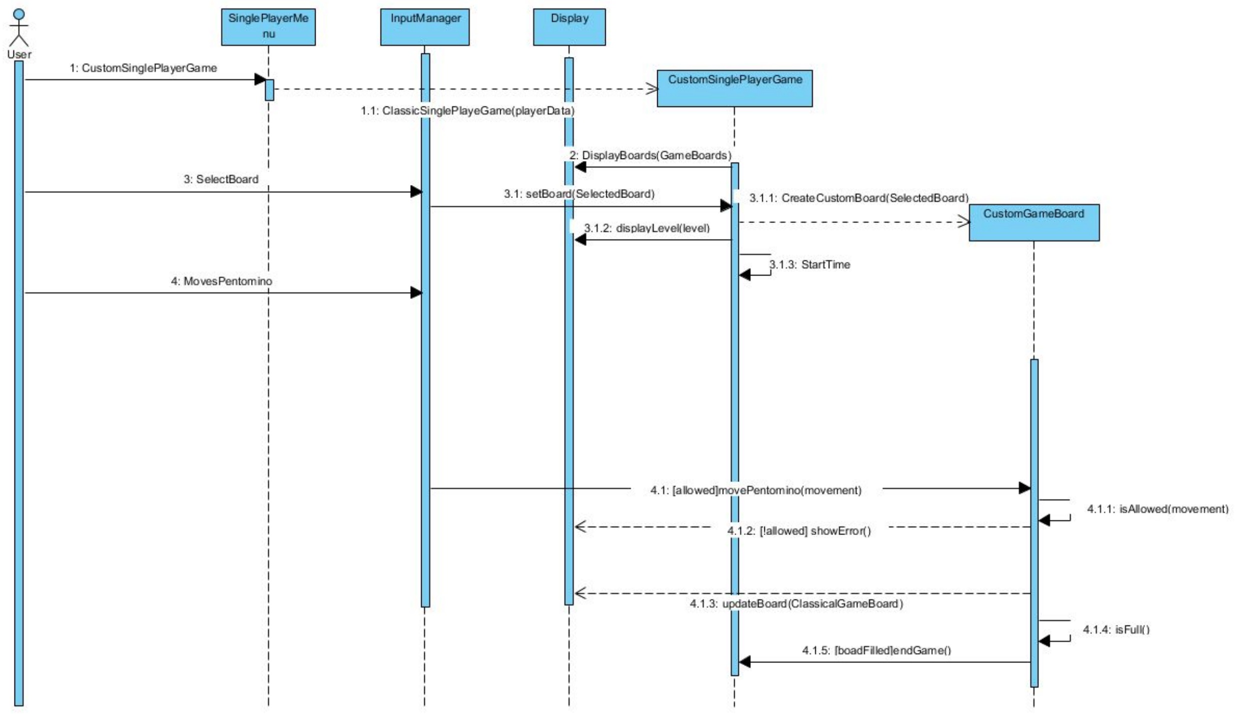

User
SinglePlayerMenu
InputManager
Display
CustomSinglePlayerGame
1: CustomSinglePlayerGame
1.1: ClassicSinglePlayeGame(playerData)
2: DisplayBoards(GameBoards)
3: SelectBoard
3.1: setBoard(SelectedBoard)
3.1.1: CreateCustomBoard(SelectedBoard)
CustomGameBoard
3.1.2: displayLevel(level)
3.1.3: StartTime
4: MovesPentomino
4.1: [allowed]movePentomino(movement)
4.1.1: isAllowed(movement)
4.1.2: [!allowed] showError()
4.1.3: updateBoard(ClassicalGameBoard)
4.1.4: isFull()
4.1.5: [boadFilled]endGame()

Figure 8: Play Custom Shape Game Sequence Diagram

5.4.1.4 Drag and Drop Pentomino

Player is in any game of Katamino and drags a pentomino via a tapping on it and moving. This movement generates a preview and it is given to game controller and game controller create this movement on the display screen by creating a KataminoDragBlock. When player drops the pentomino,game controller uses this to create a pentomino in the location of the first created pentomino. The corresponding sequence diagram is given below.

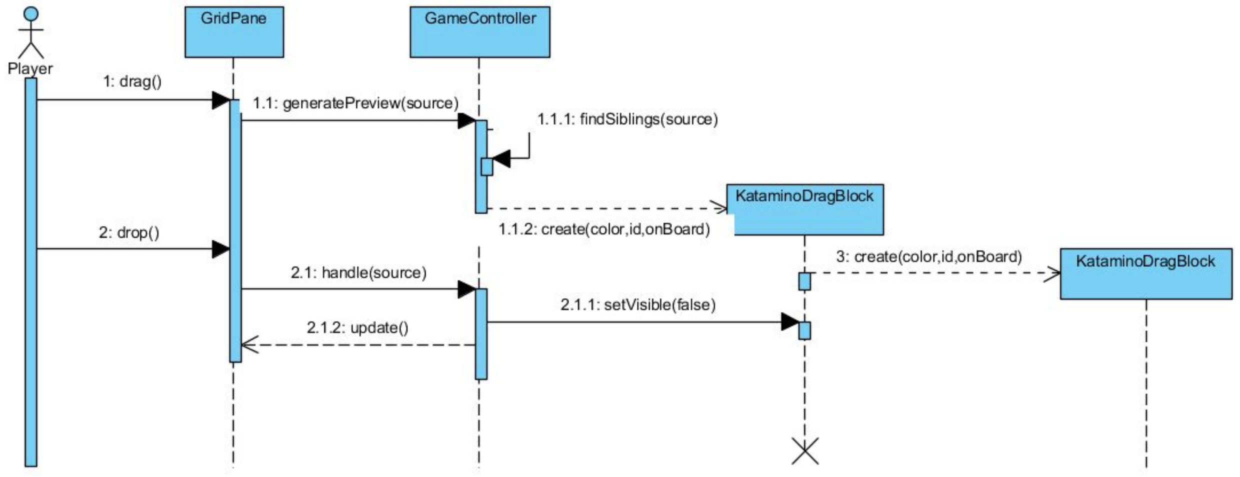


Figure 9: Drag and Drop Pentomino Sequence Diagram

5.4.2 State Diagrams

5.4.2.1 Classic Single Player State Diagram

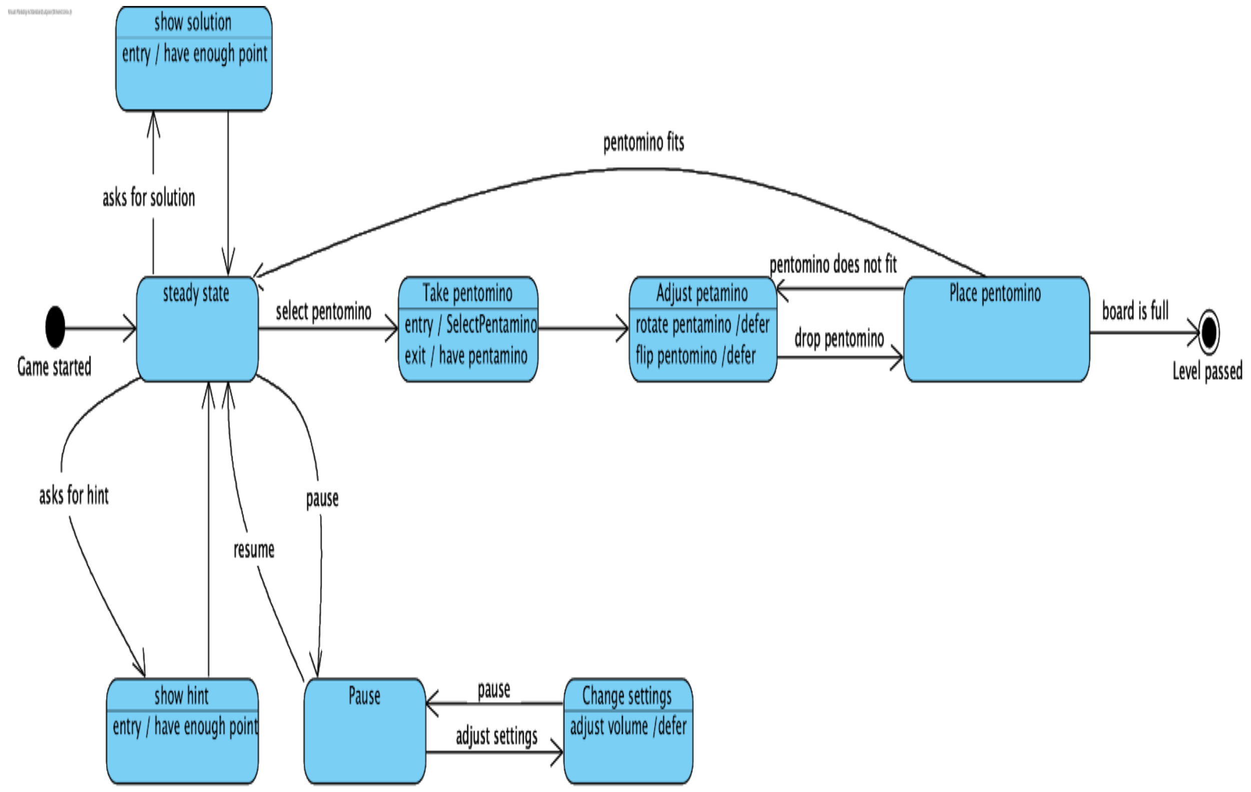


Figure 10: Classic Single Player Game State Diagram

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 flipping. 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. Player have to be on the steady state to pause the game.

5.4.2.2 Multiplayer State Diagram

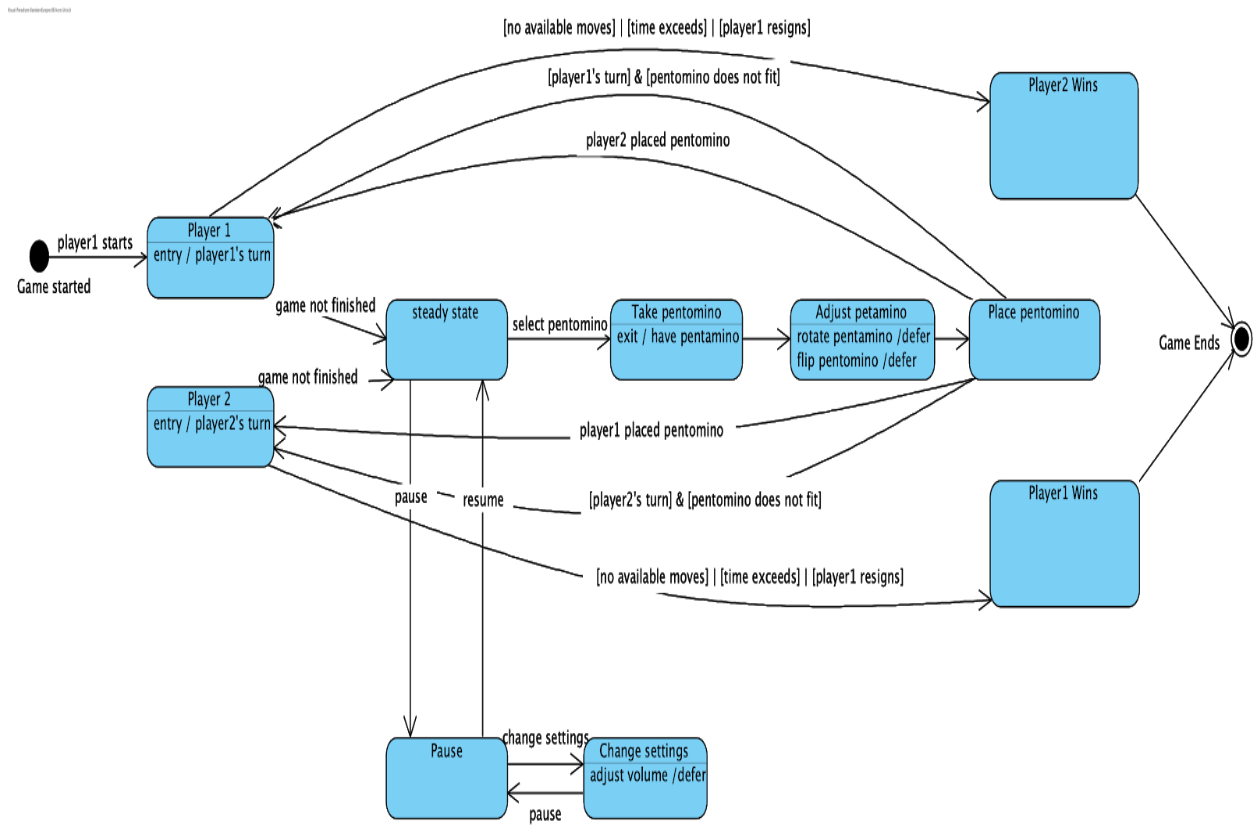


Figure 11: Multiplayer State Diagram

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 on the steady state.

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

5.4.2.3 Single Pentomino State Diagram

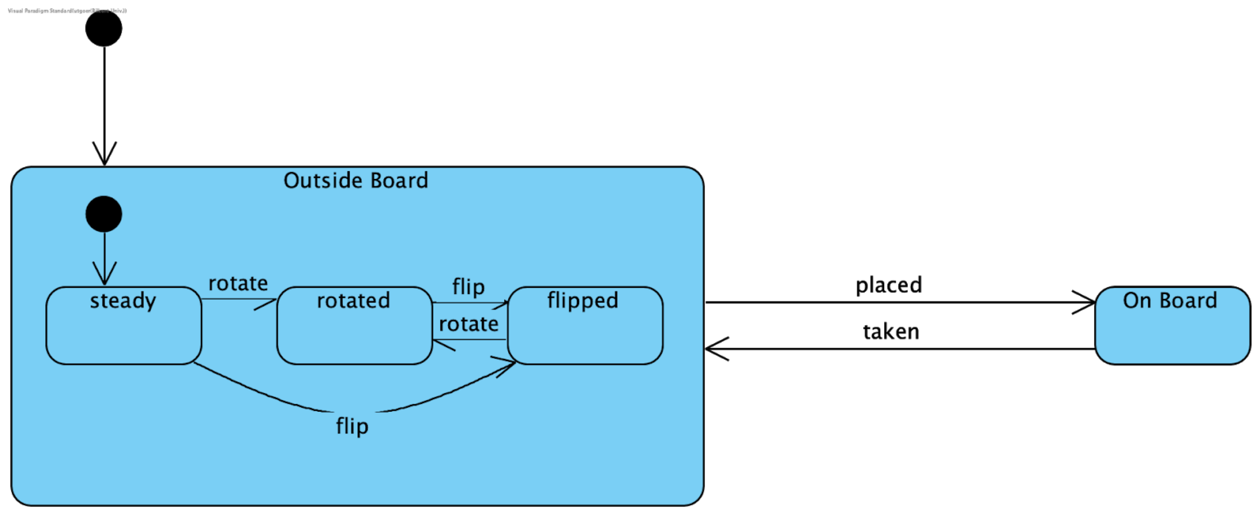


Figure 12: Single Pentomino State Diagram

Pentomino started on the outside of the board, in other words taken. Pentomino can be rotated and flipped in this state. This is shown as sub state in Figure 11. When pentomino placed to the board it considered as on board.

5.4.3 Activity Diagram

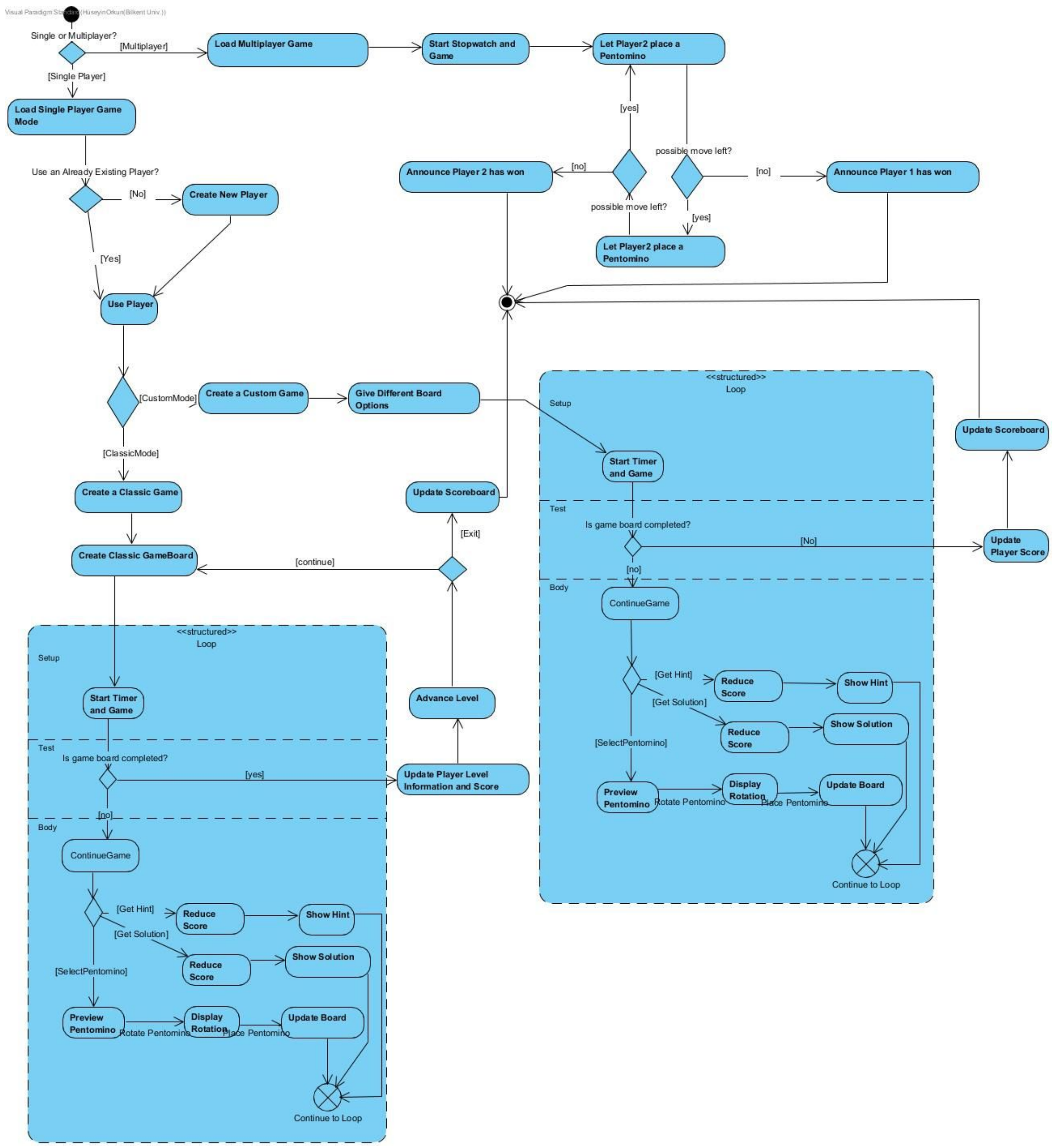


Figure 13: Activity Diagram

The activity diagram shows the options and possible actions of the user in the game. When the player starts the game, they choose between playing single player or multiplayer modes. (Cases 1 and 2)

Case 1 Multiplayer:

In the case that they choose multiplayer, player needs to choose a board to play with. After that, the multiplayer game starts, players take turn playing, and when one player can't place a pentomino to the board opponent wins.

Case 2 Single Player:

In the case that player chooses single player mode, player is presented with player select/create menu. User can create a player or choose an existing one from a list of players here. After that, the game present another choice among classic game or custom shapes mode. (Cases 2.1 and 2.2)

Case 2.1 Classic Game Mode:

If user selects the classic game, the game asks them to choose a level, after user selects a level game board loads with corresponding pentominoes. Timer starts an player can move or rotate the pentominoes, ask for help or solution. After player fills the board, the level ends the game updates player's current level and score and loads the next level. If player exits, the game updates the score table.

Case 2.2 Custom Shapes Mode:

If user selects the custom shapes mode, the game asks them to choose a custom board, after user selects a board, the game loads with corresponding board and pentominoes. Timer starts an player can move or rotate the pentominoes, ask for help or solution. After player places all the pentominoes and the board is full, the game updates player score, and the scoreboard.

5.6 Object and Class Models

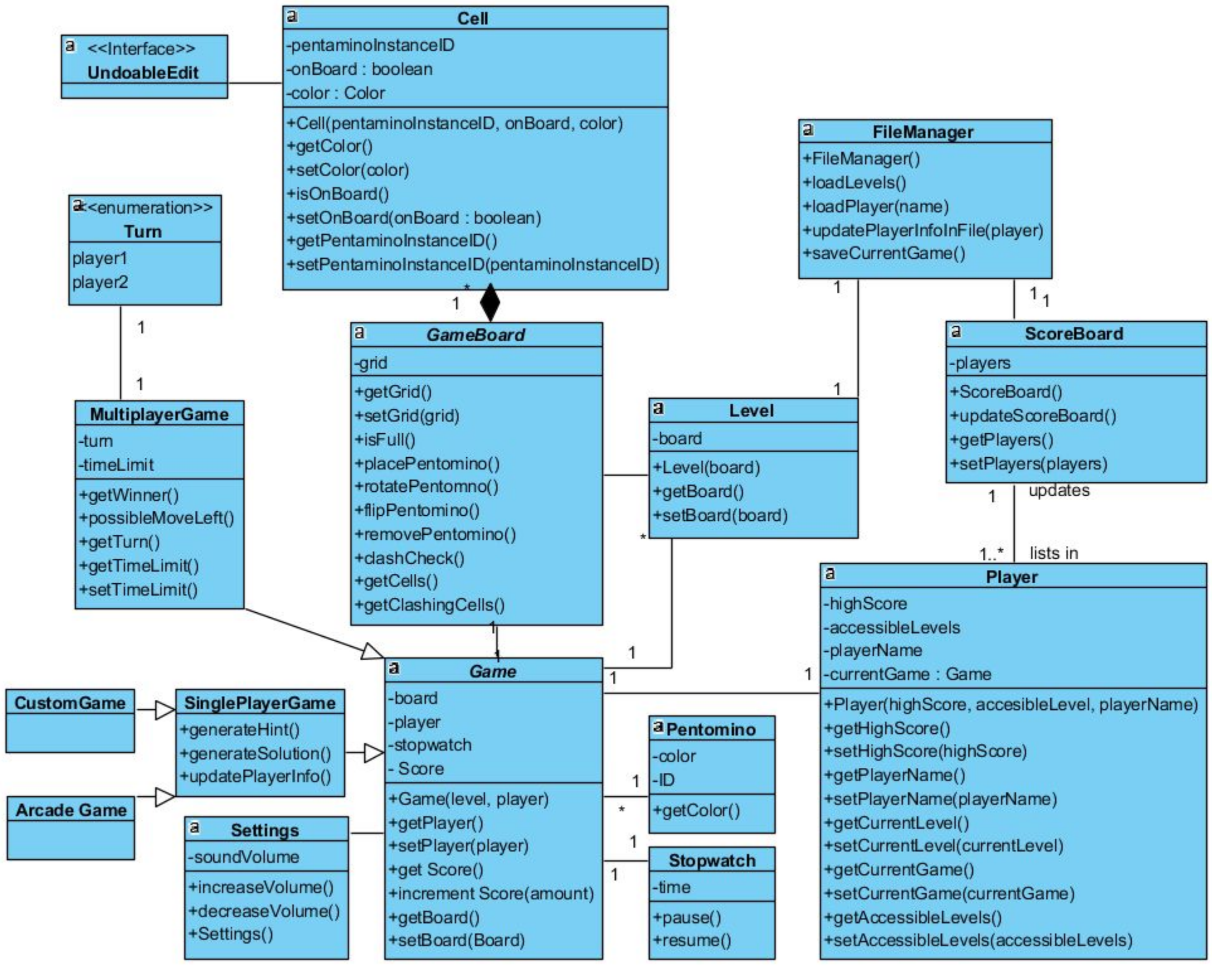


Figure 14: Class Diagram

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, *length* and width attributes store 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. *rotatePentomino()* rotates the pentomino clockwise or counter clockwise and *flipPentomino()* flips the pentomino by its y-axis. c*lashCheck()* checks if the pentomino, that is being attempted to put on the gameboard, clashes with another pentomino on the gameboard. *getClashingCells()* returns classing cells, which will be colored differently to warn the player. *getGrid()* returns grid attribute which gives the information on cell states and locations.

SinglePlayerGame.class
This class is the parent class of *ArcadeGame* and *CustomGame* which support single player mode. *ArcadeGame* is similar to the original real life Katamino box game. The gameboard is a simulation of the original game board. In contrast, *CustomGame* has many different shaped game boards that can be played, for example, a camel shaped game board. The *ArcadeGame* and *CustomGame* subclasses only differentiate in their respective game boards; therefore, the separate explanations are not given. *updatePlayerInfo()* updates the player's attributes after the game has ended. *generateHint()* gives to player a random *Pentomino* object's location that would lead a solution. *generateSolution()* returns whole *Pentomino* object array that contains a solution. Since, the multiplayer game does not depend on completing the gameboard it does not have hints or a solution.

MultiplayerGame.class
In multiplayer game, the objective is to leave the opponent unable to place a pentomino on the board. Therefore, there are significant differences to *SinglePlayerGame.* There is a *turn* attribute which stores whose turn it is. *turnTimeLimit* determines the upper limit for a player to play the game session. If a player exceeds the *turnTimeLimit,* the player automatically loses the game. *isGameOver()* internally calls possibleMoveLeft() which calculates if there is any possible pentomino placement (move) left. If there is no remaining possible move, the winner is announced using the *getWinner()* method. *getWinner* uses possibleMoveLeft() and *getTimeLimit()* methods to determine the winner. *getTurn()*, *getTimeLimit()* and *setTimeLimit()* function as their names implies.

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

Cell.class
Cell class represents the tiles of *GameBoard.class' grid* attribute**.** The *location* attribute stores the location of the cell on the GameBoard object. The *state* attribute stores if cell is filled or not by a piece of *Pentomino* object. *getState()* returns the *state* attribute and *getLocation()* returns *location* attribute Also *pentominoInstanceID* stores the pentomino's ID that occupies the cell. *onBoard* attribute specifies if a cell belongs to the game board. Cell class also includes *color* attribute, which will be used to color cells differently.

Player.class
Player class represents a player, score keeps total score of the *Player*. *accesibleLevels* stores the leves that are playable to player, due to played levels. *currentLevel* stores last played level. playerName is used for storing name to use in high score table and game screen. Update methods are called by the *SinglePlayerGame* objects at the end of the game. Get, except *getHighScore*, select and create methods are called by the *SinglePlayerGame* objects before game is started. Method *getHighScore()* called by *ScoreBoard* to display scores.

Level.class
Level class represents the data of one level, especially the levels board. This class gets its information from FileManager.class, and is used to send board information.

FileManager.class
This class is used for managing saving or loading a player, reading levels and boards from data that is locally stored.

ScoreBoard.class
Scoreboard consists of the players' high scores which is the sum of the scores gained at the each end of the *SinglePlayerGame* either *ArcadeGame* or *CustomGame.*

5.7 User Interface

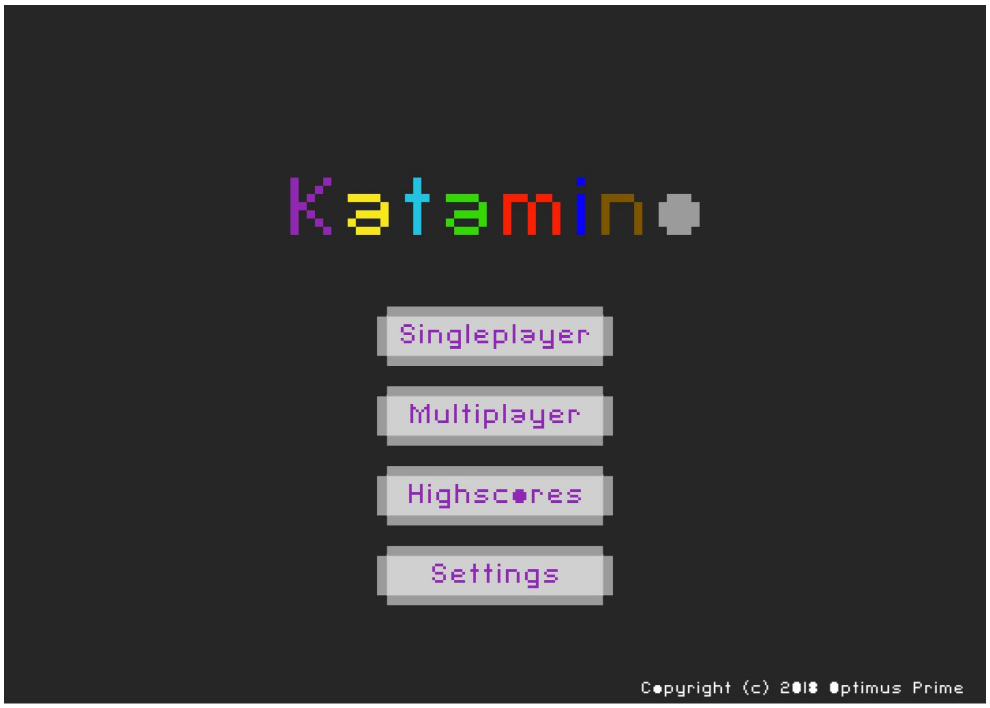


Figure 15: Main menu screen

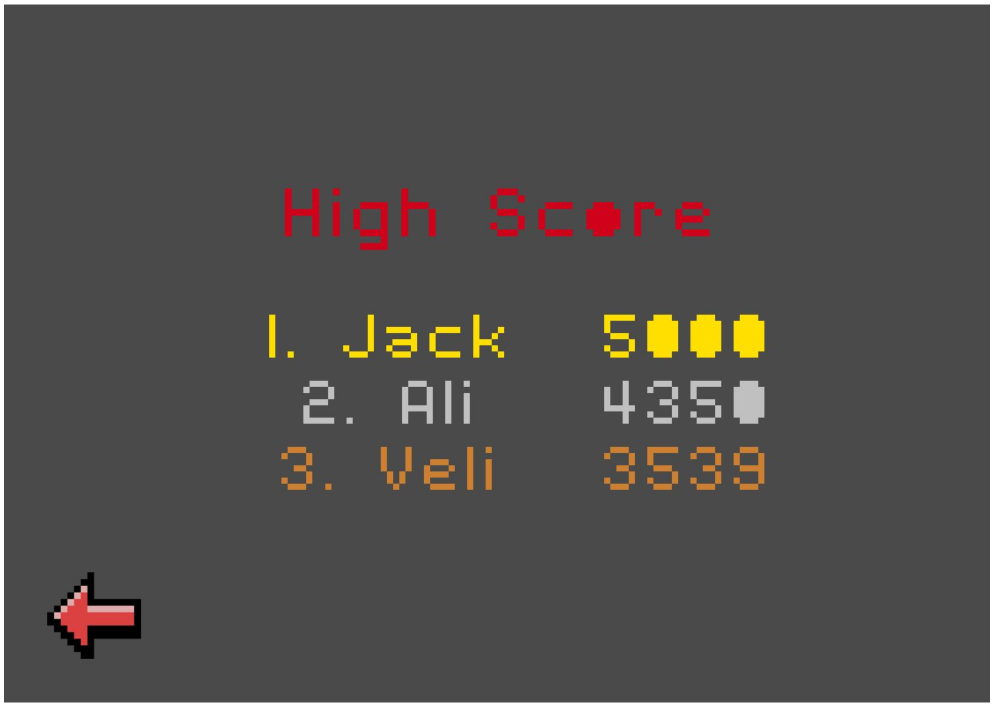


Figure 16: High Score

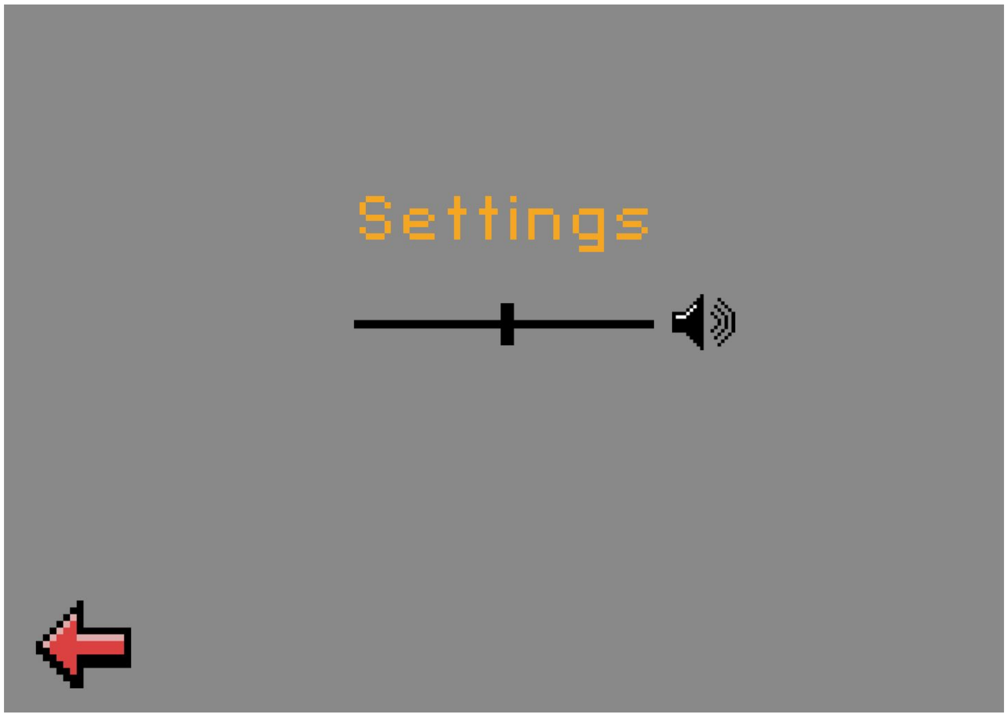


Figure 17: Settings

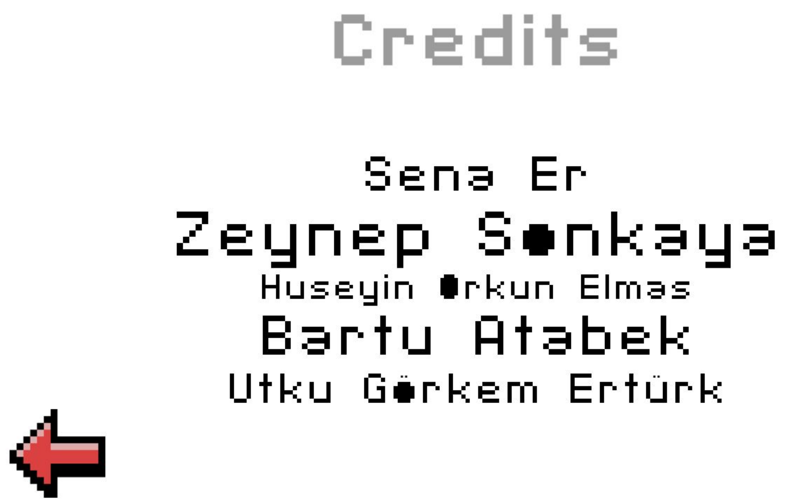


Figure 18: Credits

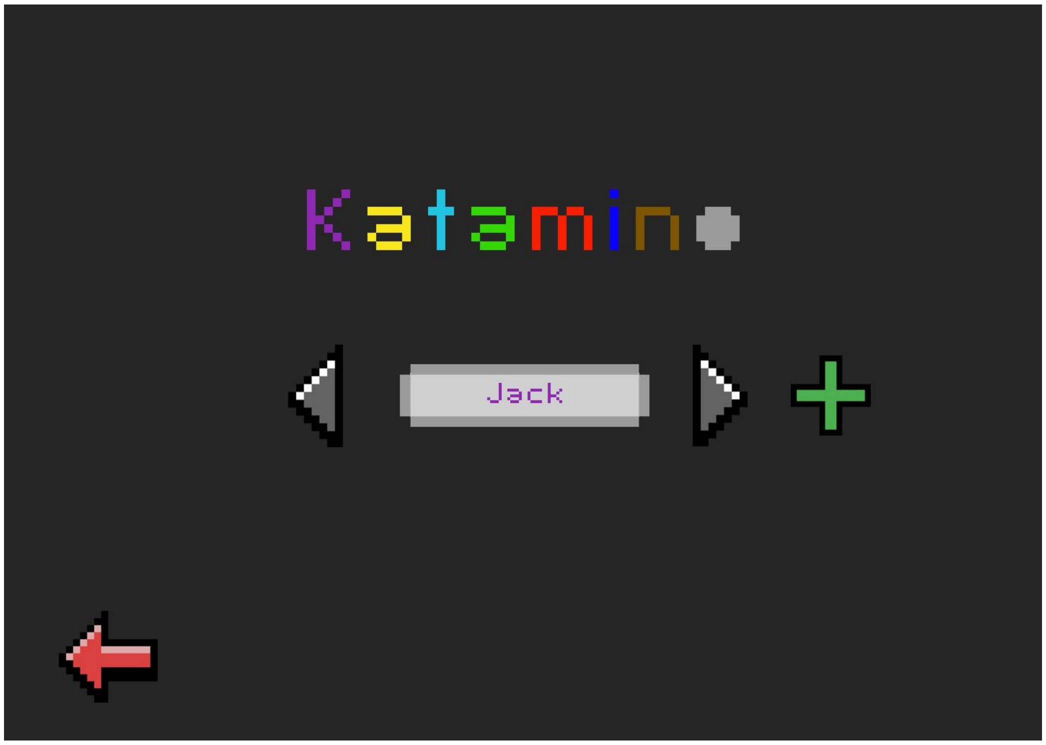
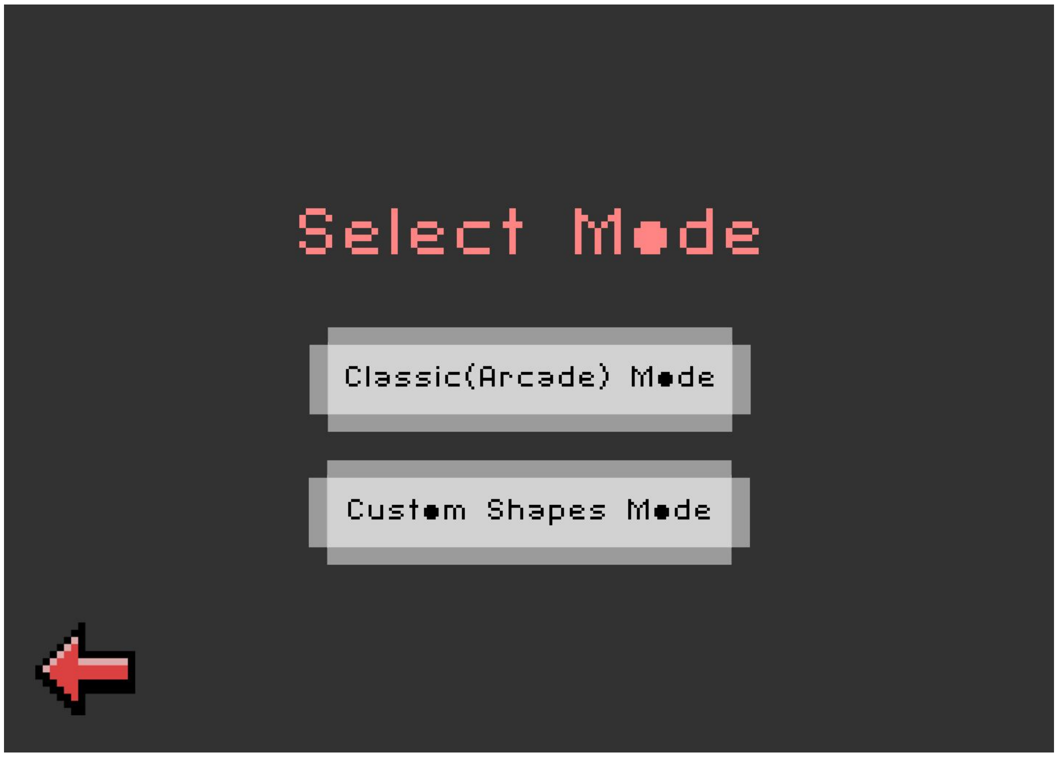
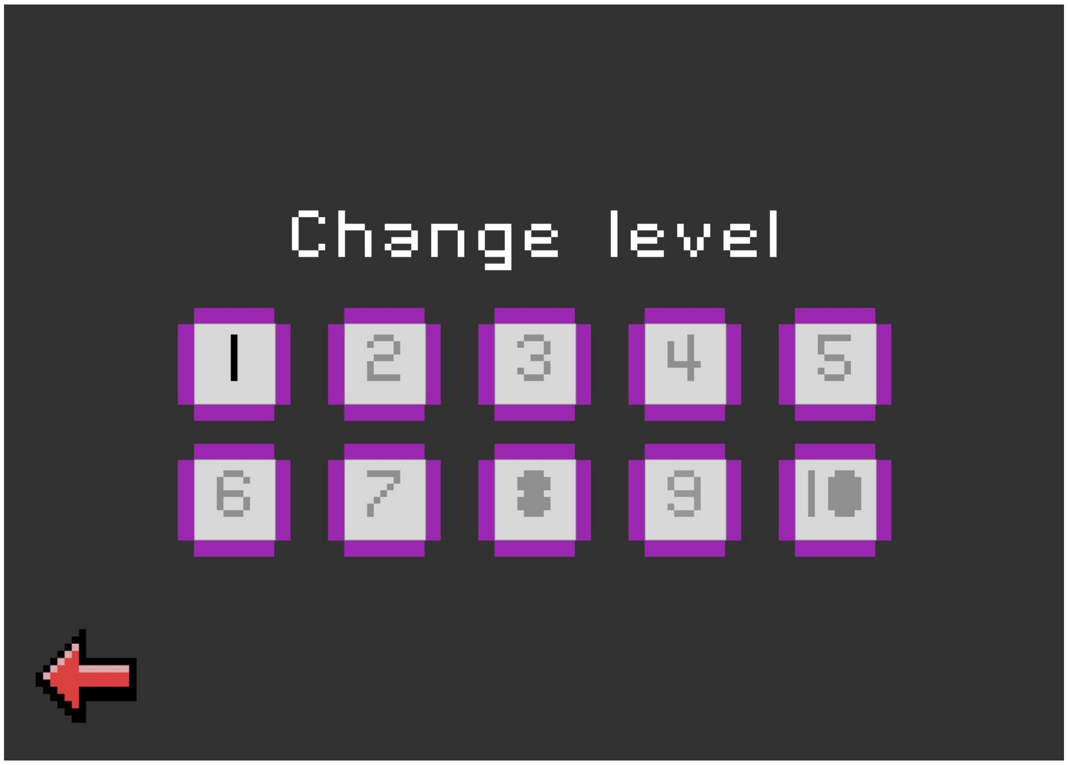


Figure 19: Single player user selection or creation



<u>Figure 20:</u> Single player select mode



<u>Figure 21:</u> Single player Arcade mode level change

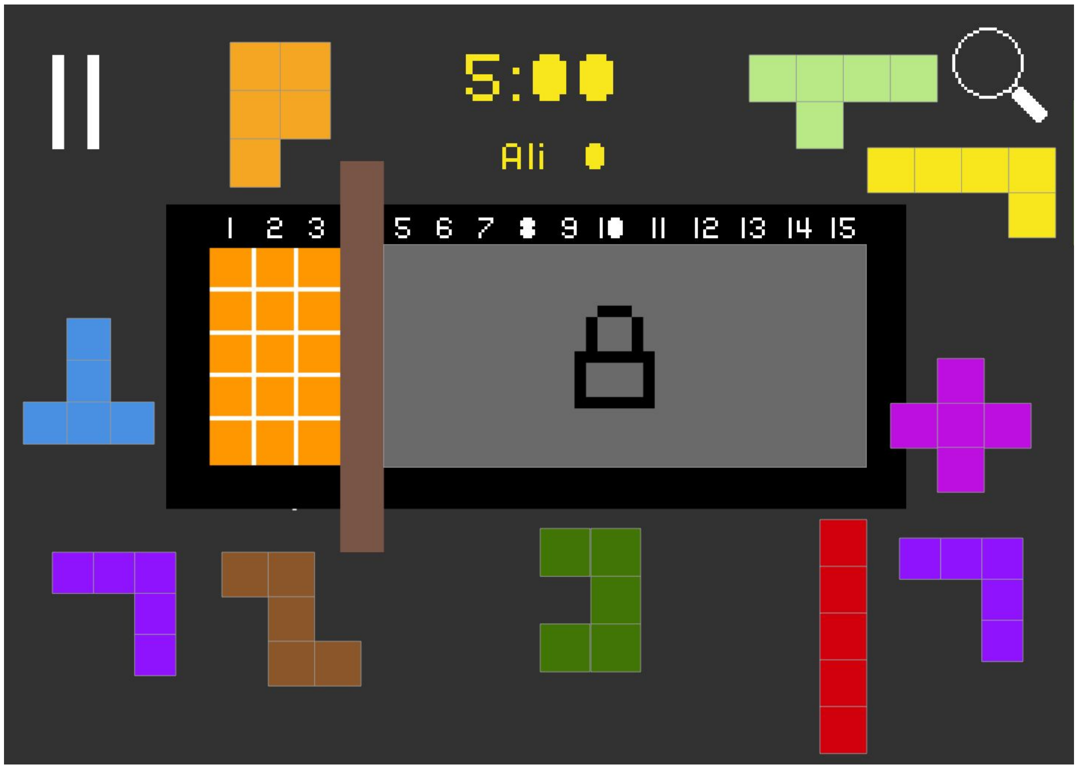


Figure 22: Single player classic(arcade) mode

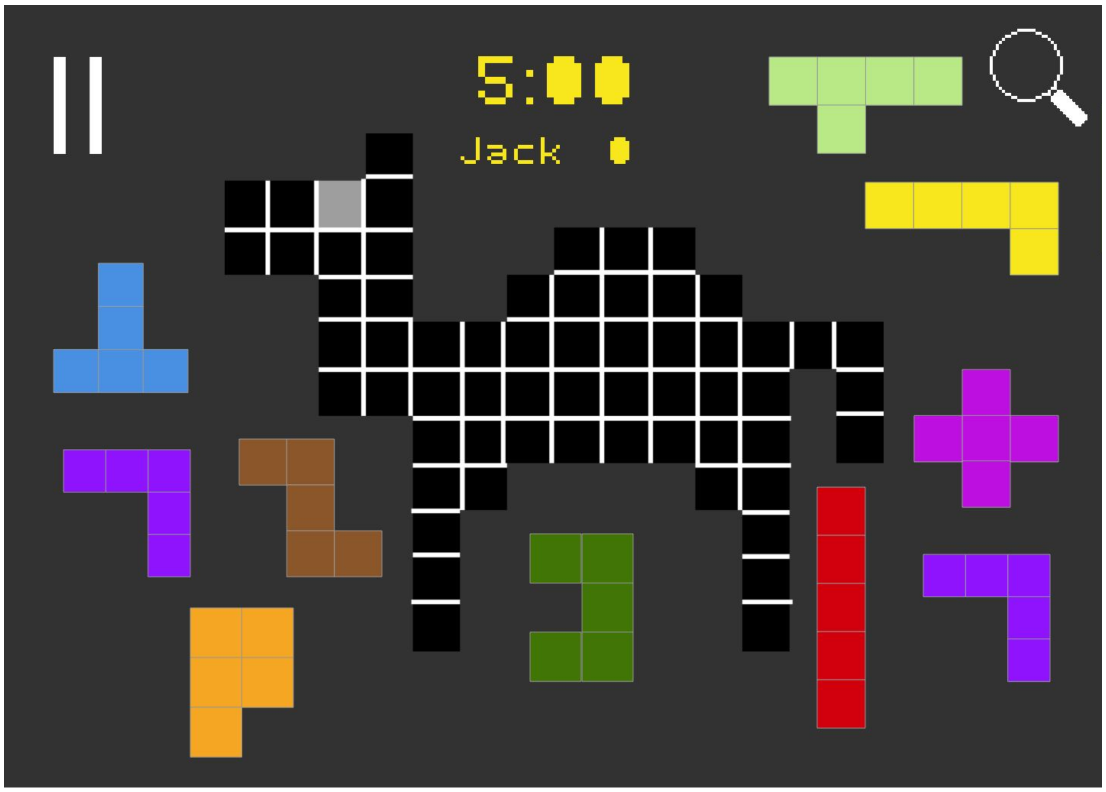


Figure 23: Single player custom shape mode

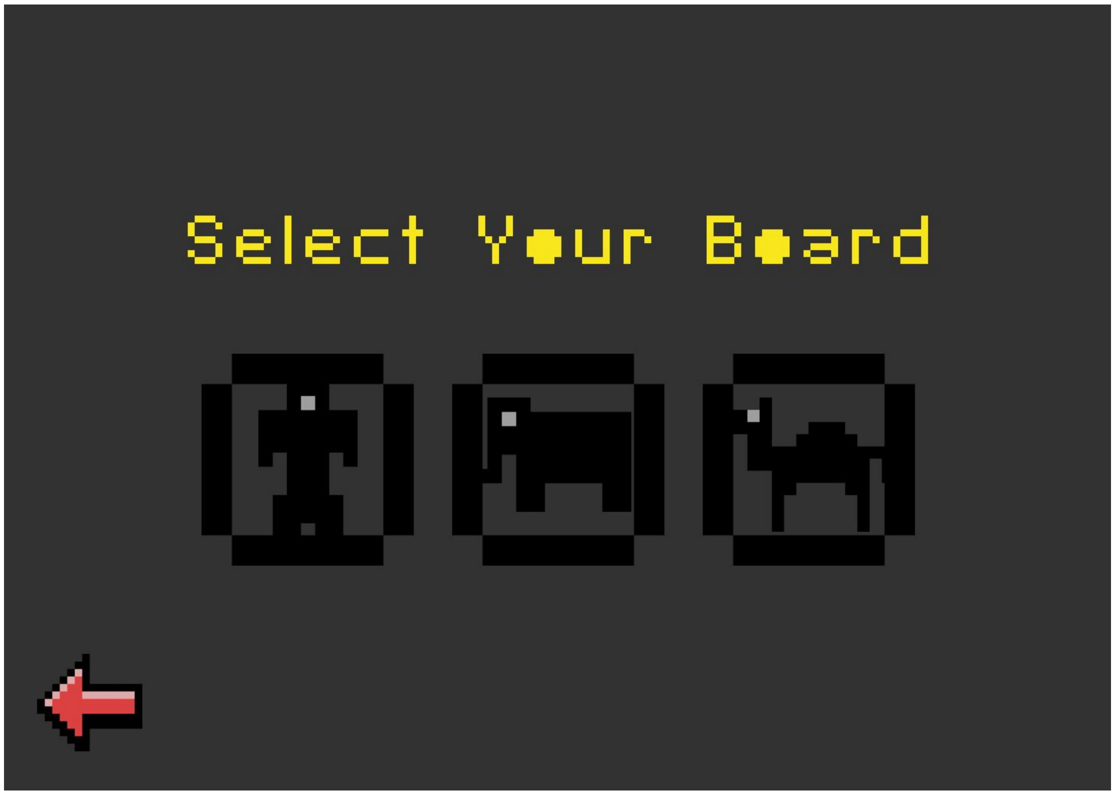


Figure 24: Select board for multiplayer and single player custom shapes mode

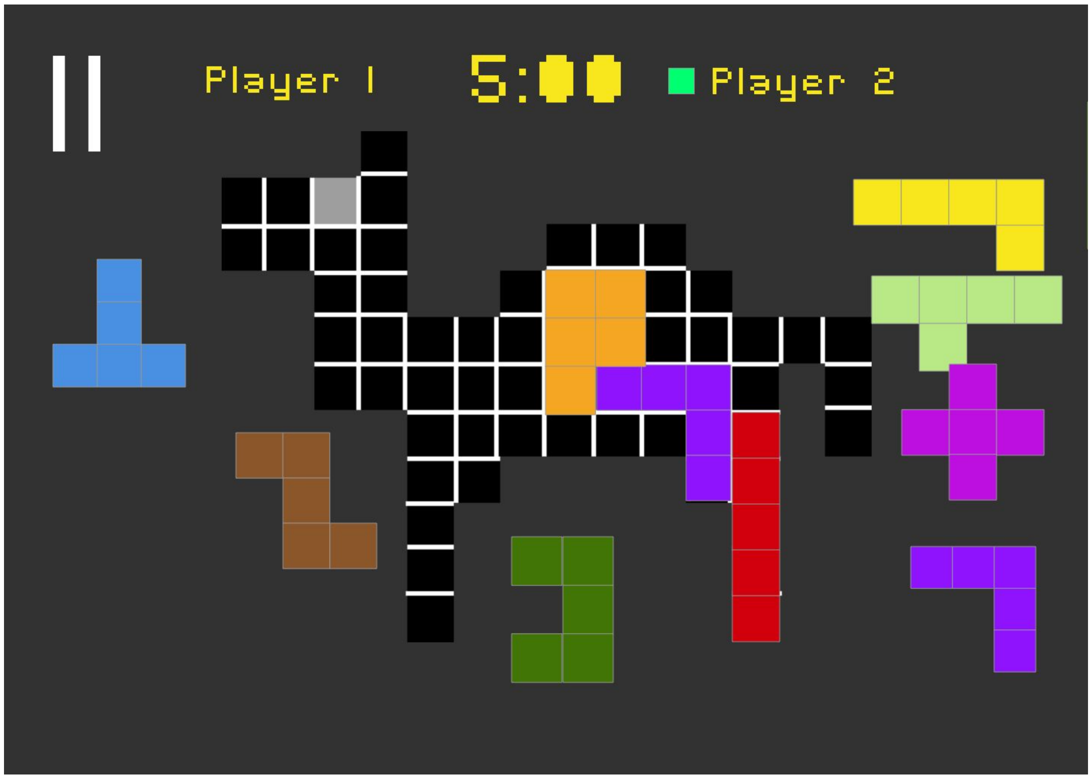


Figure 25: Multiplayer mode

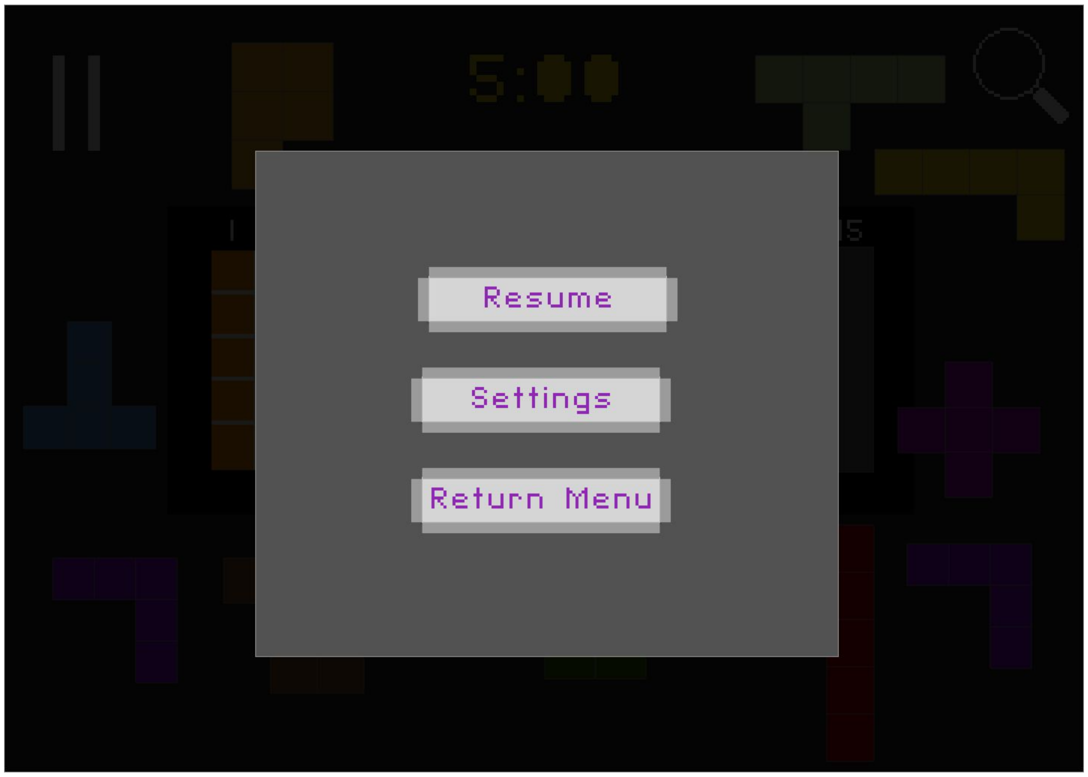


Figure 26: Pause menu

6 Improvement Summary

In the second iteration most importantly we have added new functional requirements which are customizable game boards and daily challenges. Although these features were thought they were not added as requirements however in the second iteration we decided to implement these requirements. Following list explains our improvements more detailed.

- Glossary section is added to the report to prevent confusion on terms that are used frequently.
- Two functional requirements: custom game boards and daily challenges. In the first iteration of the analysis report we added custom shapes as a gameplay mode with level selection. In this iteration we are introducing daily challenges for this mode which will randomly choose three custom shapes from the level library and present them to the player. Additionally we added customizable game boards to both single player and multiplayer so that the players will be able to create their own game boards. Since these additions do not conflict with most our pre-existing diagrams and models we only changed our use case diagram.
- The game mechanics improved in terms of pause.

- We updated the use case diagram. Also, the description of the use case diagram was updated. In use case diagram, player for multiplayer game were two separate actors as Player1 and Player2, we decided to make them one single actor called Player. Also earlier get player use case is deleted and replaced with register use case and its new extend use case Player name is already taken is added. Settings Use case is replaced with Adjust Music Volume use case.
- Class Diagram was updated; new classes were added and explained in the explanation part. Also, solution domain classes were removed and other classes were simplified.
- We realized that all board classes (i.e custom game board, multiplayer game board single) can be represented by one class, gameBoard this improved the readability of the report and flexibility of the system.
- Old sequence diagrams were updated and new sequence diagrams (Select Player, Drag and Drop Pentomino) were added.
- Content was updated in Non Functional Requirements for making them more testable.
- Activity diagram was updated; the activities are written from the point of view of the system only.
- Mockups and design is updated to conform a more user friendly experience especially for the player selection GUI.
- State diagrams are edited for Multiplayer and Classic Single Player. Also a new state diagram is added for Single Pentomino.

7 Glossary & References

Glossary

Pentomino: Pentomino is the name of pieces in the game, all pentominoes consist of 5 cells.
Rotation: Rotations are 90 degrees and clockwise or anti-clockwise.
Flip: Flip is a type of transformation where the the selected pentomino gets replaced by its symmetry by x axis.
Clash: Clashes happen when a player tries to place a pentomino on a cell that already is occupied with another pentomino.

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]