CS 319 Term Project

Analysis Report

Murat Tüver, 21602388

Sera Fırıncıoğlu, 21401803

Berk Yıldız, 21502040

Pegah Soltani, 21500559

Can Savcı, 21300803

Supervisors: Gülden Olgun, Eray Tüzün

Project Group Name: Cerca-Trova / 3C

Project short-name: Katamino

Contents

1. Introduction

The game that we decided to implement and design as a group is called Katamino which is also known as Pentominoes. This game consists of a polygon in a plane which is made of 5 identical squares. Excluding the rotated and reflected versions of each shape, there are 12 different pentominoes. Considering the reflections and rotations as distinct, there are 63 fixed pentominoes in total which provides the game multiple combinations of levels and difficulties. The variety of levels (up to 500 different levels), is one of the many reasons that lead the game to win multiple world wide awards.

Pentominoes were officially introduced and described in the book Polyominoes: Puzzles, Patterns, Problems, and Packings by an American professor named Solomon W.Golomb in 1953. Later on in 1965, pentominoes were introduced to the general public by Martin Gardner in his October 1965 Mathematical Games column in Scientific American.

Nowadays, Katamino is one of the popular puzzle and strategy games among people of all ages. The challenging levels and variety of them not only is a proper entertainment tool for grown ups that are interested in board games, it also serves as a fun, yet educating tool for parents and teachers to help children familiarize with 2D and 3D shapes. This game is also used in primary schools as smooth introduction for space geometry.

In order to make the game more interesting and competitive for the players, we added some features to the game. The players will be able to change the main theme of the game whenever they want to. Also, they will gain coins every time they pass a level. With the gained coins, one can buy single square shapes in order to fill the single gaps on the board in case the level gets more complicated and only one square spot is left empty. Along with the features mentioned above, we provided the players with the option of multiplayer mode in which the board will be divided in half and each player will be given different pieces of the puzzle. The one who finishes the puzzle first, will be the winner and be placed on the leaderboard part of the game.

2. Overview

2.1 Game Board

The game board consists of 12x5 grid which contains a movable divider to specify each level. Levels are numbered on the long side of the board. Totally 12 pentominoes can be placed on the game board.

2.2 Pentomino

A pentomino is a block which is made of 5 equal sized squares connected edge to edge. There are totally 12 pentominoes in the game and each of the pentominoes are in different color and each of them is numbered from 1 to 12 as an ID. Pentomino is the item that player places on the game board appropriately to complete the game.

2.3 Small Pieces

The game includes small pieces called red rectangles and red squares which are not pentominoes. Red squares sizes are 1x1 and red rectangles consist of 2 adjacent 1x1 square. These pieces are being used for filling the small spaces on the puzzle if there is a need. Player can buy these pieces by the coins he/she earned in single player. Some of them will be given to players as default in the multiplayer game.

2.4 Coins

Players are able to earn coins according to their game progress. They earn coins each time they solve a puzzle and complete a level in single player and they earn coins each time they beat their opponent in the multiplayer game.

2.5 Levels

The game consists of 10 different levels. There is a optional tutorial level called ‘Level 0’ which shows the basics of the game to new players. In Level 0, the player will be asked to solve a simple puzzle on 5x5 grid with given 5 pentominoes by following the given instructions. The main game consists of 9

different levels from level 4 to level 12. Each level consists of 10 different puzzles. A player is able to move higher level, if he/she completes 6 of the puzzles in the same level.

2.6 Player Items

A 12x5 grid game board and different numbers of pentominoes will be given to the player according to the his/her level.

Level 4 is played in 5x5 grid space with given 5 different pentominoes. Level 5 is played in 6x5 grid space with given 6 different pentominoes. Level 6 is played in 7x5 grid space with given 7 different pentominoes. Level 7 is played in 8x5 grid space with given 7 different pentominoes. Level 8 is played in 8x5 grid space with given 8 different pentominoes. Level 9 is played in 9x5 grid space with given 9 different pentominoes. Level 10 is played in 10x5 grid space with given 10 different pentominoes. Level 11 is played in 11x5 grid space with given 11 different pentominoes. Level 12 is played in 12x5 grid space with given 12 different pentominoes.

2.7 Game Modes

The game consists of 2 game modes: "Classic Mode" and "Arcade Mode". Both of the modes contain 12 levels. Game modes are available just for single player version.

2.7.1 Classic Mode

In the Classic Mode, player does not have any time limitations and is not able to gain score by solving the puzzles which means there is not a scoring system in the Classic Mode. The aim of the player who plays in Classical Mode is just completing the game by reaching level 12 and solving the all puzzles in level 12. Player can leave the game whenever he/she wants and re-start the game from the level he/she left.

2.7.2 Arcade Mode

In the Arcade Mode, player has time limitations which are specified differently in each level. The given time to the player increases while

he/she moves on the higher levels. Also player gains scores according to remaining time when he/she finishes the puzzle in the limits of his/her level. More remaining time and higher levels bring more score for the player. If the player can not complete the puzzle in given time, game ends and player can restart the game from the level he/she lost, however score deduction is applied for each time up.

2.7.3 Tetris Mode

In the tetris mode, 4 pentominoes will be given to the player in each turn. Player will place these pentominoes onto available spots in the gameboard. Differently from the other modes, gameboard will be a 12x12 grid for the tetris mode. When he/she fills the all squares on a row or column, that row or column dissappears. New 4 pentominoes will be provided to player whenever there is not a remaining pentomino for player to place on board. When there is no available space on the board to place any given pentomino, game ends.

2.8 Leaderboard

There is a leaderboard screen for the players who play in Arcade Mode. Player is able to see his/her scores in the leaderboard. Scores are listed by the names that players enters at the end of each Arcade Mode game.

2.9 Settings

Settings allows players to set sound preferences and choose a different theme to play. There are three different themes that player can play with. Different themes apply to the game by providing different background patterns on game board and pentominoes**.**

2.10 Game Play

Katamino is a 2D single player computer game which is adapted from the board game called ‘Katamino’. Game can be played by one or two players.

2.10.1 Single Player

The aim of the player is placing the given pentominoes into appropriate tiles on the game board in a specific level until there is no free space remaining on the grid. As the player progresses in the levels, workspace of the player on the game board and number of given pentominoes increase too. There are ten levels in the main game from Level 4 to Level 12 and also one additional level for the tutorial which is Level 0.

2.10.2 Multiplayer (2 Players)

The all pentominoes will be given to the players except pentomino 1 and pentomino 12. One of the 2 players chooses 1 maroon rectangle, 1 maroon square and 2 red squares from the small pieces. The other player chooses 1 maroon rectangle and 3 red squares from the small pieces. Then both players choose one pentomino at a time in turn until each has 5 pentominoes. Position the slider on the board between the numbers 6 and 7. Each player must fill the rectangle on his/her side of the board as quickly as possible with all of his/her pieces. To be able to play in multiplayer version, a player has to have an account and login to the game. A player is able to have a friend list, block list and score statistics about previous games within his/her account. Also players can send and receive messages through their accounts.

3. Functional Requirements

- Players can start a new game directly or they can prefer the tutorial option in the case of not having enough knowledge about the game.

- There are two play options which are single player and multiplayer.

- There are two modes which are classic and arcade for the Katamino in single player option.

- In the classic mode, after completing the certain sublevels, players can pass to the new levels of Katamino.

- In the arcade mode, players should complete the levels in a limited amount of time in order to pass to the new levels.

- If players finish the game before the given time is up, they will gain a score and see their scores on the leaderboard at the end of game.

- Player will earn coins according to scores that he/she gained. These coins will be used to by small pieces for using in multiplayer option when needed.

- In single player option, there will be also a tetris mode which is a combination of Katamino specialities with Tetris game.

- Players can change the theme of Katamino whenever they desire.

- Players can turn the sound and music the game on or off.

- In multiplayer option players will have their own accounts with unique usernames.

- In multiplayer, the player who finishes the puzzle first will be the winner and also have his/her name on the leaderboard of the multiplayer section.

- In multiplayer, player will earn coins for each puzzle he/she wins against his/her opponent.

- In multiplayer, the board will be spelt in two identical parts, one for each player. The players will also be provided with different pieces of pentominoes to fill their boards.

- Players in multiplayer mode will be able to send messages to their friends and chat with them. They can also modify their friend list by removing, blocking or unblocking their friends.

4. Non-functional Requirements

4.1 Usability

Katamino will include a tutorial section in order to help new players to understand the game concepts. Also, it will provide a sample level called "level 0" where the player only needs to follow couple of instructions to finish the level.

Since Katamino is a game designed for people from all ages, its controls will be rather simple and user friendly. A common used gaming control technic, drag and drop, will be implemented for placing pentominoes to the board. Additionally, there will be keyboard shortcuts for rotating pentominoes, muting the game, showing pause menu to save player from extra mouse clicks.

4.2 Reliability

Unlocked levels and the leaderboard data for the arcade mode will be stored and conserved in the local machine and it will be possible to access them in other gaming sessions. In case of a possible system or power failure, unlocked levels will be immediately recorded to local files when the player passes the level.

4.3 Performance

We expect Katamino to respond player's inputs in less than 5 ms. Game will use 2D models which will help us to reduce the graphical hardware need of the game. So that, frame rate of the game will be at least 30 fps to provide a good user experience to players. In the storage perspective, only files that will be kept local are the settings configuration file and the single player leaderboard file. This means our game won't need more than

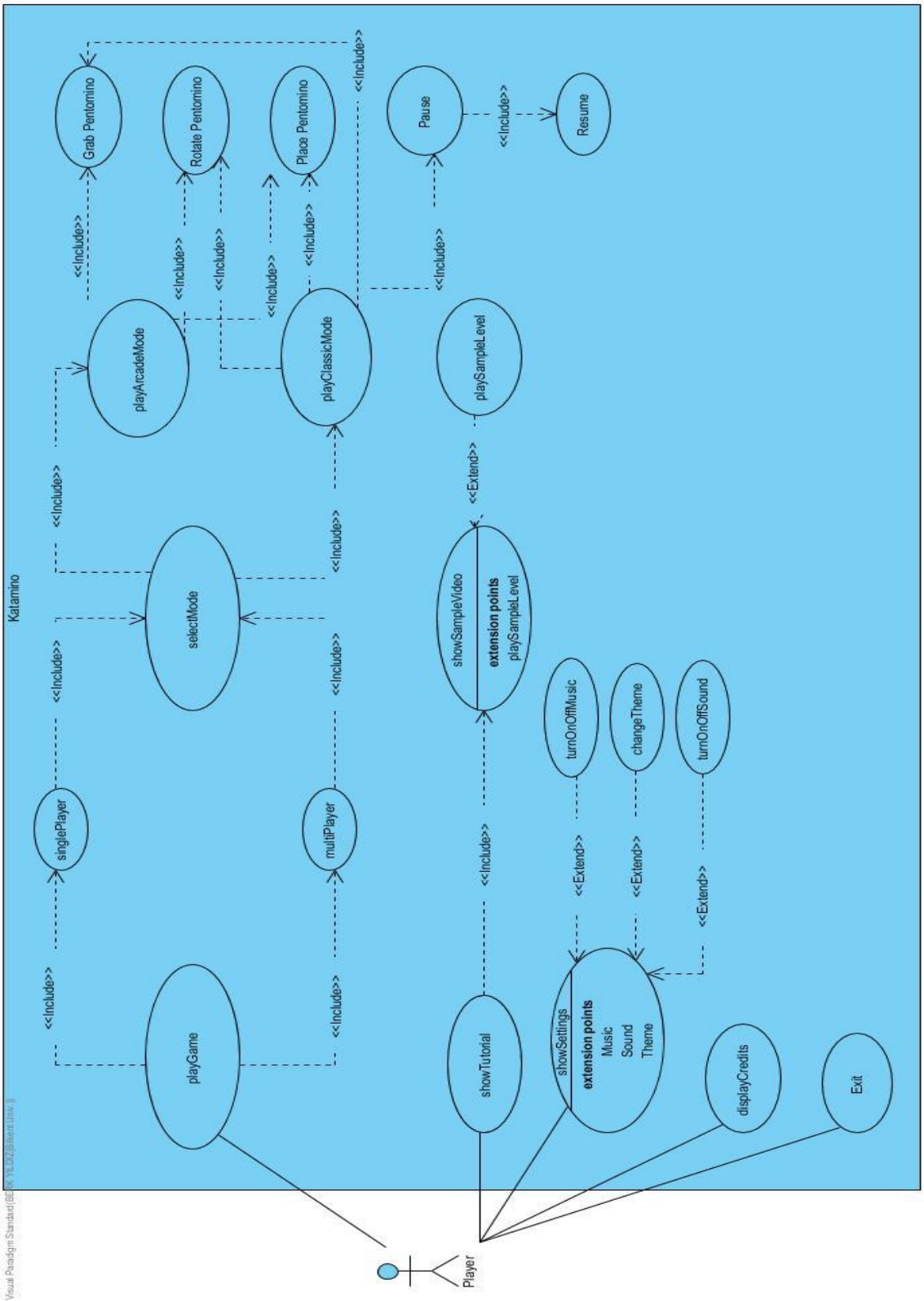
100mb of storage even with the size of the game itself. Memory need of the game won't pass 128mb since our game doesn't have many objects in the screen at once. In the multiplayer gaming mode, user will need a basic internet connection. Our recommended connection speed is 512kbit.

4.4 Supportability

In order to fix future bugs and allow game to be developed for further version, code will be well-commented and files will be well-named. Size of the game will be almost 30 MB since we are planning to add the multiplayer as well as the single player mode. So our game will be downloaded easily from a regular internet connection.

5. System Models

5.1 Use Case Models



Use Case 1

Use- Case: Playing the Game
Primary Actor: Player

Stakeholders and Interests:

- User enters the game by selecting the 'Play Game' option.
- Game initializes with given pentominoes and game board according to level.

Pre-conditions:

- User has to be in the main menu.

Entry Conditions:

- User has to select the number of players (Single Player or Multiplayer).
- User has to select the game mode (Classical Mode or Arcade Mode).

Exit Conditions:

- Player(s) select the 'Quit' on the game screen which returns him/her to main menu.
- In multiplayer, when the challenge finishes.
- In single mode, when the player finishes all the levels.

Success Scenario Event Flow:

1. Player selects 'Single Player'.
2. Game initializes by given pentominoes, game board.
3. Player selects a pentomino and place it into the game board.
4. System updates the pentomino coordinates, available slots of game board and game board view.
5. System passes into following puzzle.
6. Player completes the puzzle by placing all given pentominoes into appropriate coordinates on the game board.

Alternative Event Flow: If player wants to return a previous level that he/she already succeed while solving a puzzle.

1. Player selects 'Quit Game' on the game screen and returns to main menu.
2. Player selects 'Load Game' on the main menu.
3. Player selects the level that he/she wants to return.
4. Game initializes with the given pentominoes and game board.

Use Case 2

Use Case: Entering to Tutorial
Primary Actor: Player

Stakeholders and Interests:

- Player selects 'Tutorial'.
- Level 0 initializes with the given pentominoes and game board.

Pre-conditions:

- Player must be in the main menu.

Entry-conditions:

- Player selects 'Tutorial' option from the main menu.

Exit conditions:

- Player selects 'Continue' button if he/she wants to continue to main game.
- Player selects 'Quit' button if he/she wants to leave tutorial.

Success Scenario Event Flow

1. Player selects 'Tutorial' option in the main menu.
2. A tutorial video starts which explains the game.
3. Player selects 'Continue' button.
4. Level 0 initializes with given pentominoes, game board and instructions to follow.

5. Player places the given pentominoes in to appropriate spaces on the game board by following the given instructions and completes the Level 0.

6. Player clicks 'Continue' button for starting the main game.

Alternative Event Flows: Player can start the main game anytime he/she wants while spending time in Tutorial screen.

1. Player selects 'Tutorial' option in the main menu.

2. A tutorial video starts which explains the game.

3. Player selects 'Continue' button.

4. Level 0 initializes with given pentominoes, game board and instructions to follow.

5. Player directly selects 'Continue' option before completing the Level 0.

6. Main game starts.

<u>Use Case 3</u>

Use Case: Settings

Primary Actor: Player

Stakeholders and Interests:

- Player clicks 'Sound ON/OFF' button to play or mute the game sounds.
- System opens or closes the game sounds.
- Player clicks 'Music ON/OFF' button to play or mute the background music.
- System plays or mutes the background music.
- Player selects one of the listed theme.
- System updates the game theme.

Pre-conditions:

- Player has to be in main menu.

Post-conditions:

- Availability of the game sounds are updated.
- Availability of background music is updated.
- Theme is updated.

Entry-conditions:

- Player selects ‘Settings’ button in the main menu.

Exit-conditions:

- Player clicks the ‘Return’ icon for going back to main menu.

Success Scenario Event Flow: Pausing the background music.

1. Player selects ‘Settings’ in the main menu.
2. System opens the settings window.
3. Player clicks ‘Music ON/OFF’ button to mute the background music.
4. System stops the music play.
5. Player clicks on the ‘Return’ icon in the settings screen.

Use Case 4

Use Case: Viewing the ‘Leaderboard’

Primary Actor: Player

Stakeholders and Interests:

- Player is able to see the list of high scores within the owners of high scores

Pre-conditions:

- Player has to be in main menu.

Post-conditions:

- Player be informed about the high scores.

Entry-conditions:

- Player selects ‘Leaderboard’ button in the main menu.

Exit-conditions:

- Player clicks the ‘Return’ icon for going back to main menu.

Success Scenario Event Flow: Viewing the leaderboard.

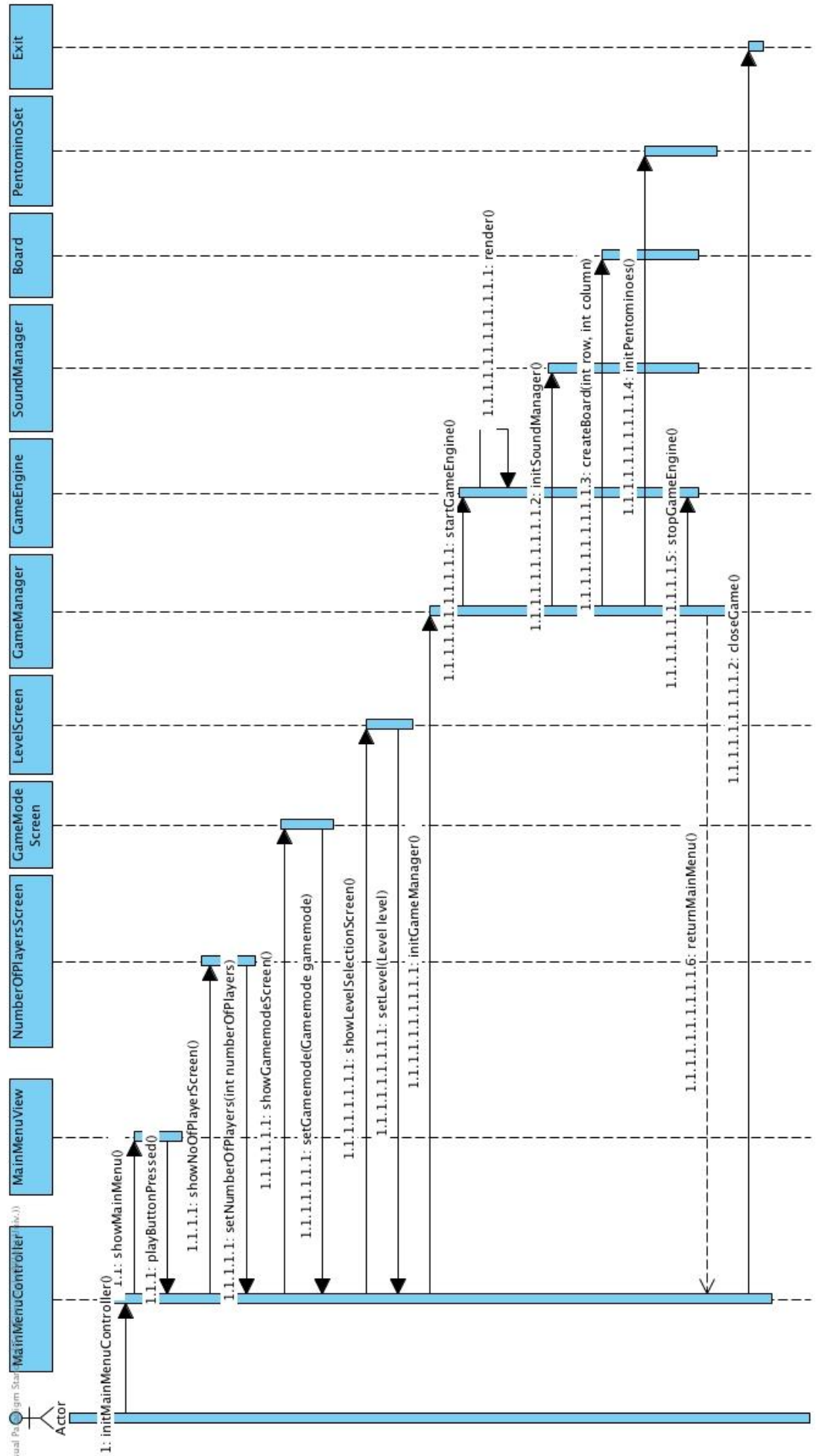
1. Player selects the ‘Leaderboard’ button in the main menu.
2. System displays the leaderboard screen.
3. Player clicks on ‘Return’ icon to go back to the main menu.

5.1. Dynamic Models

5.2.1 Sequence Diagrams

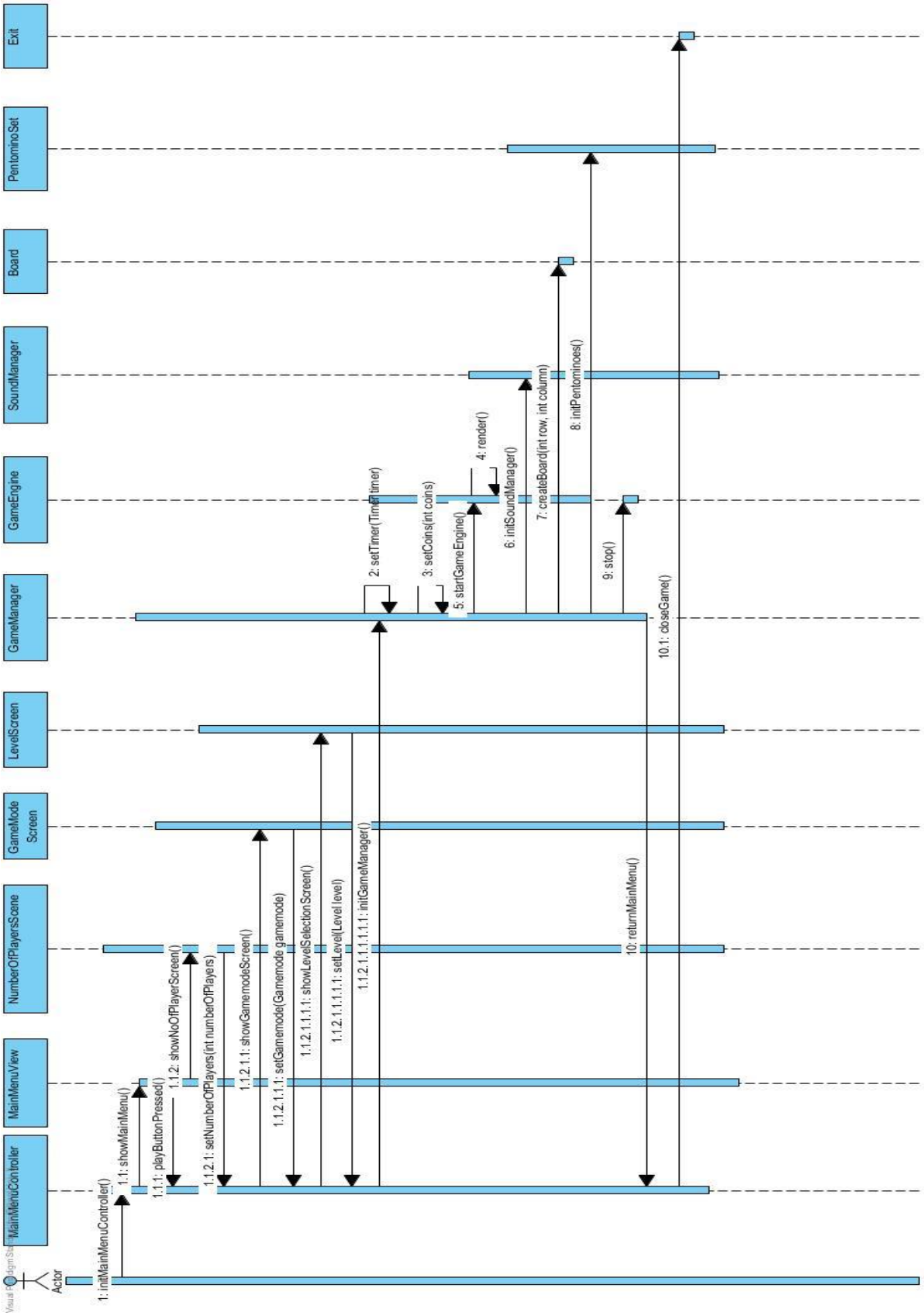
StartGameClassic

Scenario: User starts a Classic Mode and then quits game.
In this scenario, player wants to start a new game under classic gamemode. Player chooses play from the main menu, selects single player then selects the game mode classic and chooses a level. As the player runs the game, a controller called MainMenuController is initialized. Which controls the views associated with the main menu. MainMenuController creates a series of screens in order to learn what type of game user wants to play. In our scenario user goes with the single player and classic gamemode. Then MainMenuController creates a GameManager with given parameters. From now on, GameManager controls the flow of that game state. It creates necessary game objects through their model classes. Example to those game objects are the board and the pentominoes. Additionally, it creates the SoundManager and the GameEngine. SoundManager, plays the music and necessary sound effects when they are needed. GameEngine has the references of the game objects and it is responsible for constantly rendering these objects views in case of a possible change in game objects properties. When the player decides to exit the game, GameManager stops the GameEngine and then calls MainMenuController to show main menu. Afterwards, GameManager is destroyed which results in destroying all the game objects and their managers.


Actor
MainMenuController
MainMenuView
NumberOfPlayersScreen
GameMode Screen
LevelScreen
GameManager
GameEngine
SoundManager
Board
PentominoSet
Exit
1: initMainMenuController()
1.1: showMainMenu()
1.1.1: playButtonPressed()
1.1.1.1: showNoOfPlayerScreen()
1.1.1.1.1: setNumberOfPlayers(int numberOfPlayers)
1.1.1.1.1.1: showGamemodeScreen()
1.1.1.1.1.1.1: setGamemode(Gamemode gamemode)
1.1.1.1.1.1.1.1: showLevelSelectionScreen()
1.1.1.1.1.1.1.1.1: setLevel(Level level)
1.1.1.1.1.1.1.1.1.1: initGameManager()
1.1.1.1.1.1.1.1.1.1.1: startGameEngine()
1.1.1.1.1.1.1.1.1.1.1.1: render()
1.1.1.1.1.1.1.1.1.1.1.2: initSoundManager()
1.1.1.1.1.1.1.1.1.1.1.3: createBoard(int row, int column)
1.1.1.1.1.1.1.1.1.1.1.4: initPentominoes()
1.1.1.1.1.1.1.1.1.1.1.5: stopGameEngine()
1.1.1.1.1.1.1.1.1.1.1.6: returnMainMenu()
1.1.1.1.1.1.1.1.1.1.2: closeGame()

StartGameArcade

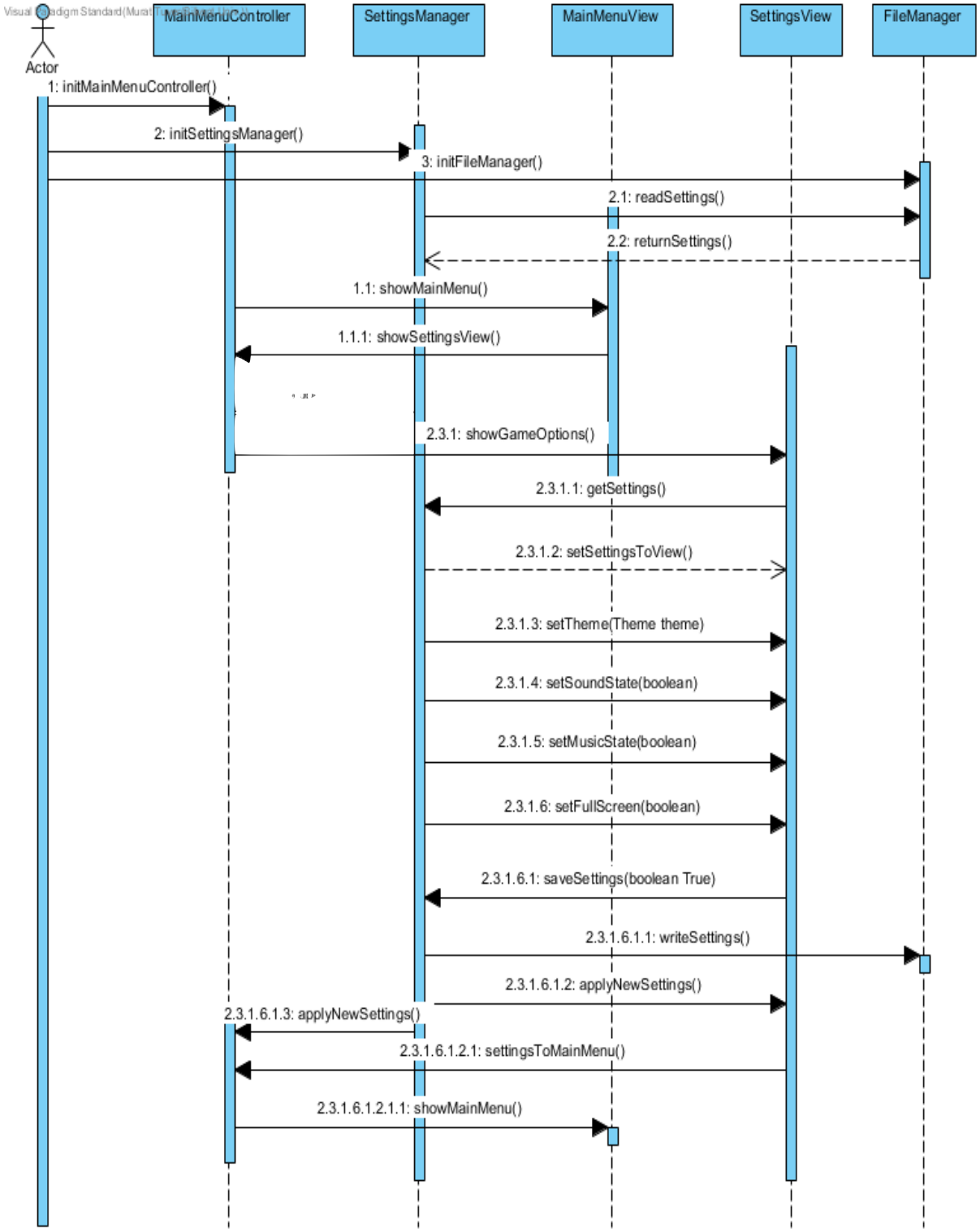
Scenario: User starts a Arcade Mode and then quits game.
In this scenario, player wants to start a new game under arcade game mode. Player chooses play from the main menu, selects single player then selects the game mode arcade and chooses a level. As the player runs the game, a controller called MainMenuController is initialized which controls the views associated with the main menu. MainMenuController creates a series of screens in order to learn what type of game user wants to play. In our scenario user goes with the single player and arcade gamemode. Then MainMenuController creates a GameManager with given parameters. From now on, GameManager controls the flow of that game state. It creates necessary game objects through their model classes. Example to those game objects are the board and the pentominoes. Additionally, it creates the SoundManager and the GameEngine. SoundManager, plays the music and necessary sound effects when they are needed. GameEngine has the references of the game objects and it is responsible for constantly rendering these objects views in case of a possible change in game objects properties. In the Arcade Mode, GameEngine sets the timer and total coins as well. When the player decides to exit the game, GameManager stops the GameEngine and then calls MainMenuController to show main menu. Afterwards, GameManager is destroyed which results in destroying all the game objects and their managers.


Actor
MainMenuController
MainMenuView
NumberOfPlayersScene
GameMode Screen
LevelScreen
GameManager
GameEngine
SoundManager
Board
PentominoSet
Exit
1: initMainMenuController()
1.1: showMainMenu()
1.1.1: playButtonPressed()
1.1.2: showNoOfPlayerScreen()
1.1.2.1: setNumberOfPlayers(int numberOfPlayers)
1.1.2.1.1: showGamemodeScreen()
1.1.2.1.1.1: setGamemode(Gamemode gamemode)
1.1.2.1.1.1.1: showLevelSelectionScreen()
1.1.2.1.1.1.1.1: setLevel(Level level)
1.1.2.1.1.1.1.1.1: initGameManager()
2: setTimer(Timer timer)
3: setCoins(int coins)
4: render()
5: startGameEngine()
6: initSoundManager()
7: createBoard(int row, int column)
8: initPentominoes()
9: stop()
10: returnMainMenu()
10.1: closeGame()

Change Settings

Scenario: User changes settings.

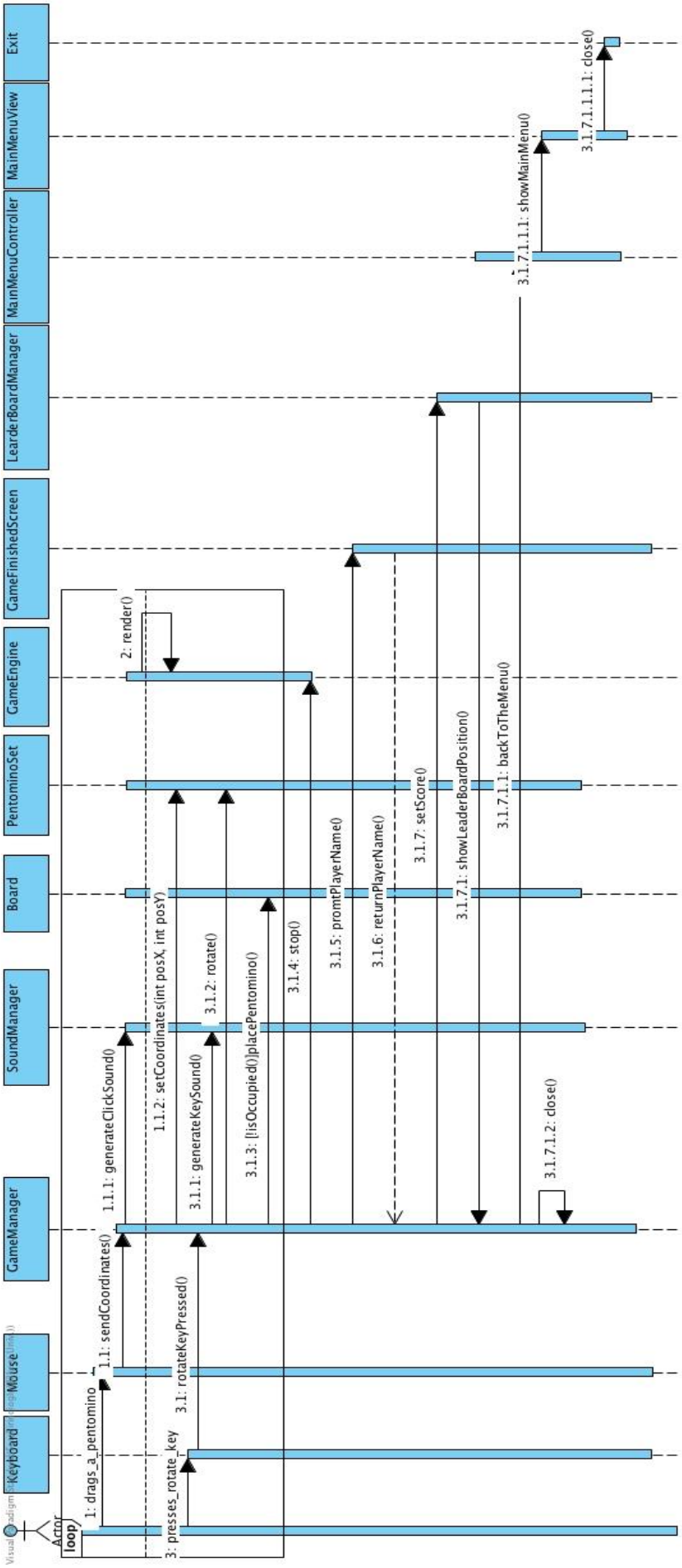
Player wants to change game settings, save them and get back to the main menu. First of all, previous saved settings, default settings if necessary, needs to read from a local file. Therefore, SettingsManager asks FileManager to bring old settings. FileManager returns old values to the SettingsManager in order to apply them to the MainMenuManager. At that point, player can use main menu view to go to the settings section. SettingsView takes settings from SettingsManager to display them to the screen. Player makes changes to the setting by using the SettingView. After player is done with the changes, save settings button is pressed new settings are stored in SettingsManager. Moreover, SettingsManager writes new settings to a local file through FileManager. Then, SettingsManager applies new settings to the SettingsView by this way player gets feedback from the game. Player can now return back to the main menu with desired settings.


Actor
MainMenuController
SettingsManager
MainMenuView
SettingsView
FileManager
1: initMainMenuController()
2: initSettingsManager()
3: initFileManager()
2.1: readSettings()
2.2: returnSettings()
1.1: showMainMenu()
1.1.1: showSettingsView()
2.3.1: showGameOptions()
2.3.1.1: getSettings()
2.3.1.2: setSettingsToView()
2.3.1.3: setTheme(Theme theme)
2.3.1.4: setSoundState(boolean)
2.3.1.5: setMusicState(boolean)
2.3.1.6: setFullScreen(boolean)
2.3.1.6.1: saveSettings(boolean True)
2.3.1.6.1.1: writeSettings()
2.3.1.6.1.2: applyNewSettings()
2.3.1.6.1.3: applyNewSettings()
2.3.1.6.1.2.1: settingsToMainMenu()
2.3.1.6.1.2.1.1: showMainMenu()

User Interactions

Scenario: Player plays a level.

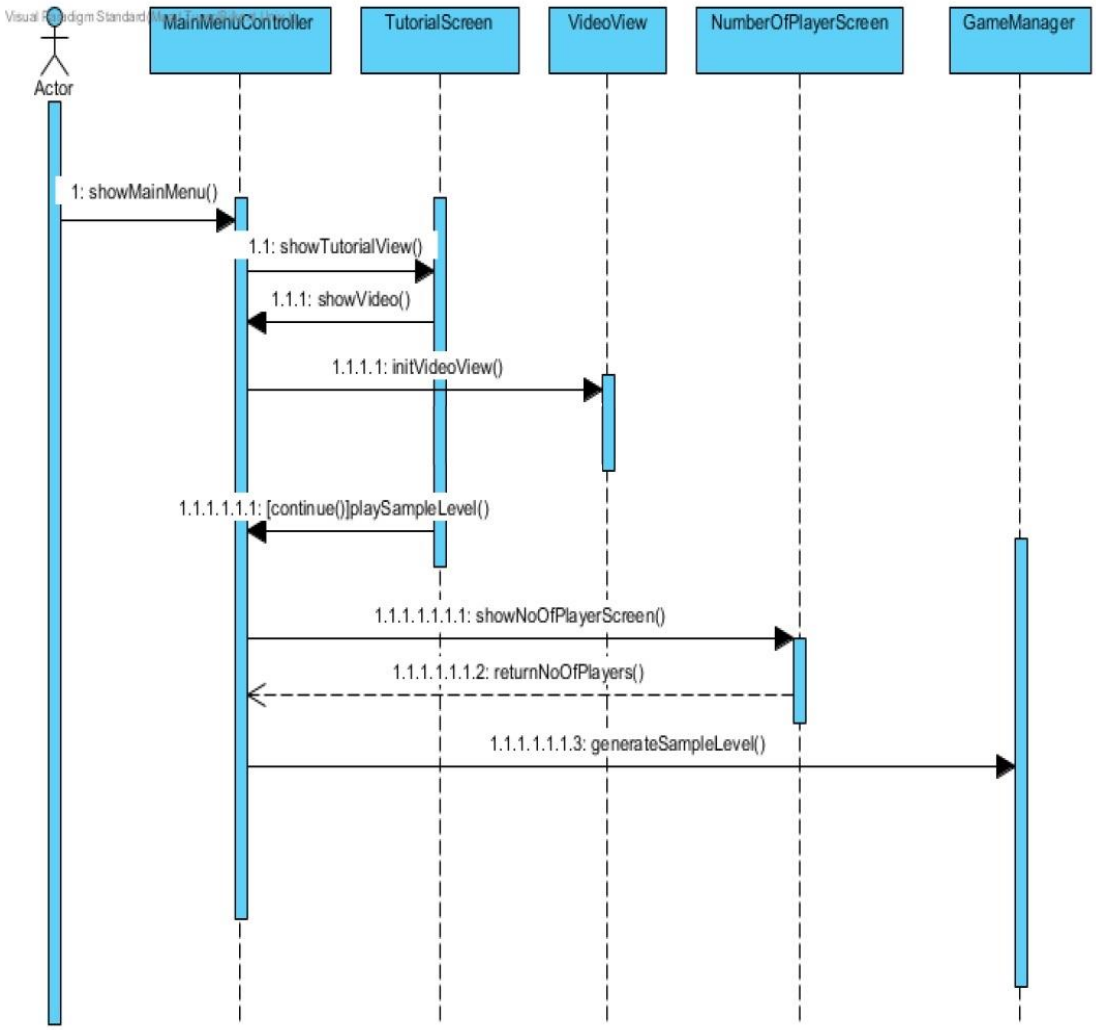
The player wants to select a pentomino, rotate and then place it to the board. Afterwards, board is completed, level is finished and a prompt screen appears to get the player's name to place it in the leaderboard. Player enters his/her name and returns to the main menu. In these events, GameManager listens the mouse and the keyboard. It decides which pentomino is selected and sets its coordinates to the mouse's coordinates, creating a drag and drop mechanism. It also checks whether piece is placed correctly or not. In the meantime, SoundManager generates click and drop sounds while GameEngine is constantly rendering the screen. After making sure game has come to an end, GameManager pops up a GameFinishedScreen in order to get the player's name. LeaderBoardManager takes care of storing the score and shows the rank of the Player. Player is then forwarded to the main menu.


Actor
loop
Keyboard
Mouse
GameManager
SoundManager
Board
PentominoSet
GameEngine
GameFinishedScreen
LearderBoardManager
MainMenuController
MainMenuView
Exit
1: drags_a_pentomino
1.1: sendCoordinates()
1.1.1: generateClickSound()
1.1.2: setCoordinates(int posX, int posY)
2: render()
3: presses_rotate_key
3.1: rotateKeyPressed()
3.1.1: generateKeySound()
3.1.2: rotate()
3.1.3: [!isOccupied()]placePentomino()
3.1.4: stop()
3.1.5: promtPlayerName()
3.1.6: returnPlayerName()
3.1.7: setScore()
3.1.7.1: showLeaderBoardPosition()
3.1.7.1.1: backToTheMenu()
3.1.7.1.2: close()
3.1.7.1.1.1: showMainMenu()
3.1.7.1.1.1.1: close()

Tutorial Section

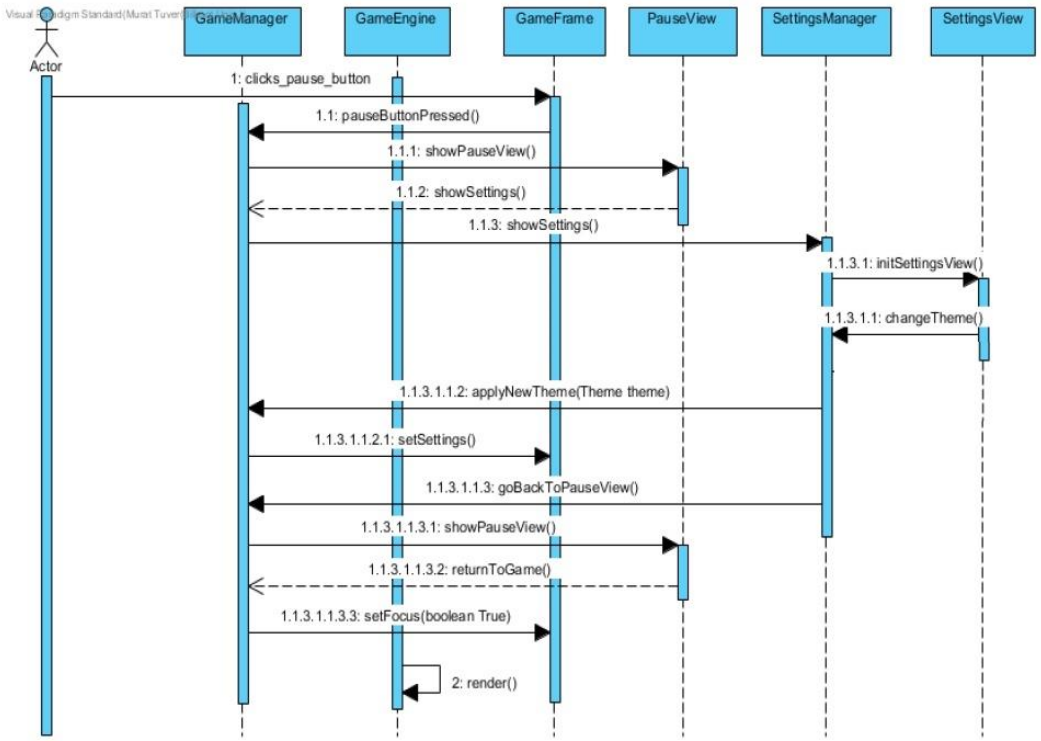
Scenario: User plays tutorial.

When the player wants to play the tutorial, he/she can simply access from the Tutorial tab from the Main Menu. When the player clicks to the Tutorial tab, the TutorialScreen appears. At first, the user watches a video about how to play the game. Then, the program asks the user whether he/she wants to play a sample level or skip it. If the user does not want to continue at the Tutorial section, he/she will be directed to Main Menu screen again. Otherwise, the player will be directed to play a sample level which includes the instructions about the game. The player has to complete the instructions one-by-one successfully. When the user completes all of the instructions, the user will be forwarded to the player number selection screen to start an actual game.


Actor
MainMenuController
TutorialScreen
VideoView
NumberOfPlayerScreen
GameManager
1: showMainMenu()
1.1: showTutorialView()
1.1.1: showVideo()
1.1.1.1: initVideoView()
1.1.1.1.1.1: [continue()]playSampleLevel()
1.1.1.1.1.1.1: showNoOfPlayerScreen()
1.1.1.1.1.1.2: returnNoOfPlayers()
1.1.1.1.1.1.3: generateSampleLevel()

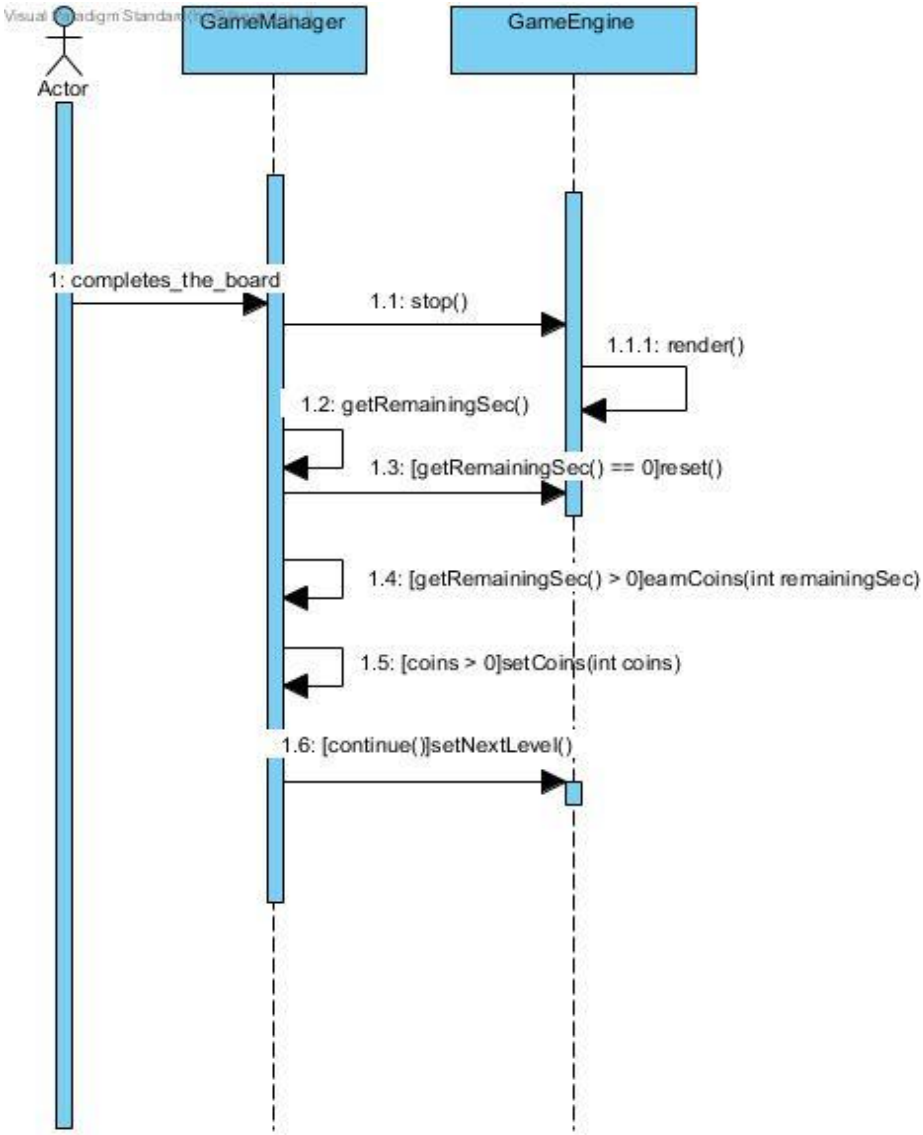
Pause Screen

Scenario: Player pauses the game, goes to the settings screen, changes the theme and goes back to the game. In this scenario, GameFrame recognizes player's click on the pause button which leads GameManager to initialize a PauseView. Player clicks on settings button in the PauseView which creates an additional view called SettingView. It also changes settings of the GameManager. Consecutively, GameManager applies these changes to the GameFrame, allowing player to see desired theme on the next render which will be done by the GameEngine. Finally, player can go back to the PauseView to go back to the game and continue from the state where it was left.

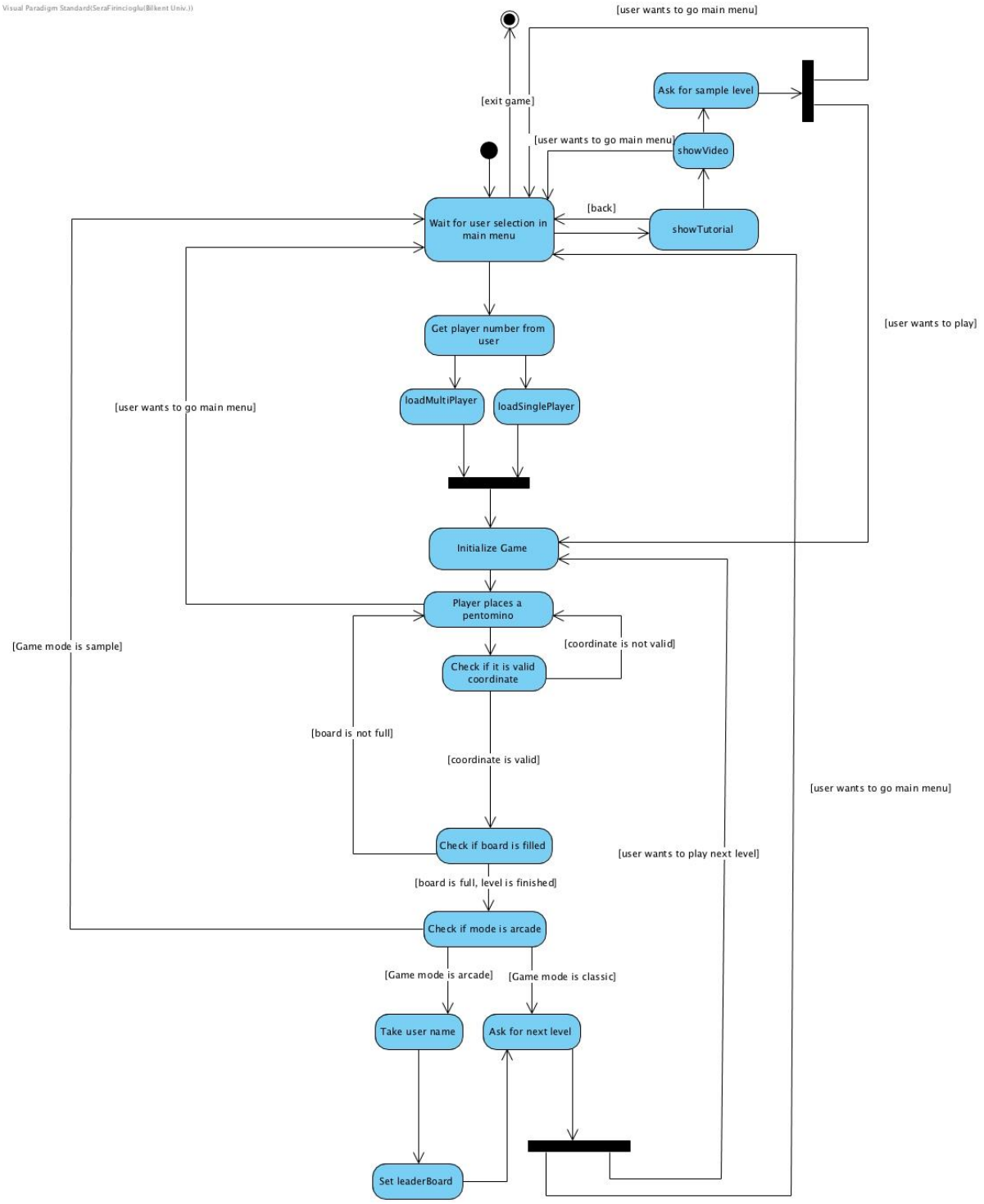


Earn Coins

Scenario: Player finishes the game, goes to the leaderboard screen, earns the coins and goes back to the game. In this scenario, GameManager recognizes that player finished the game. It stops the game and calculates the remaining seconds from the instance of Timer class. If the player completes the puzzle before the timer stops, GameManager calculates the coins that the player earned and sets the new amount of coins. After that, GameManager commands GameEngine to set the next level.



5.2.2 Activity Diagram

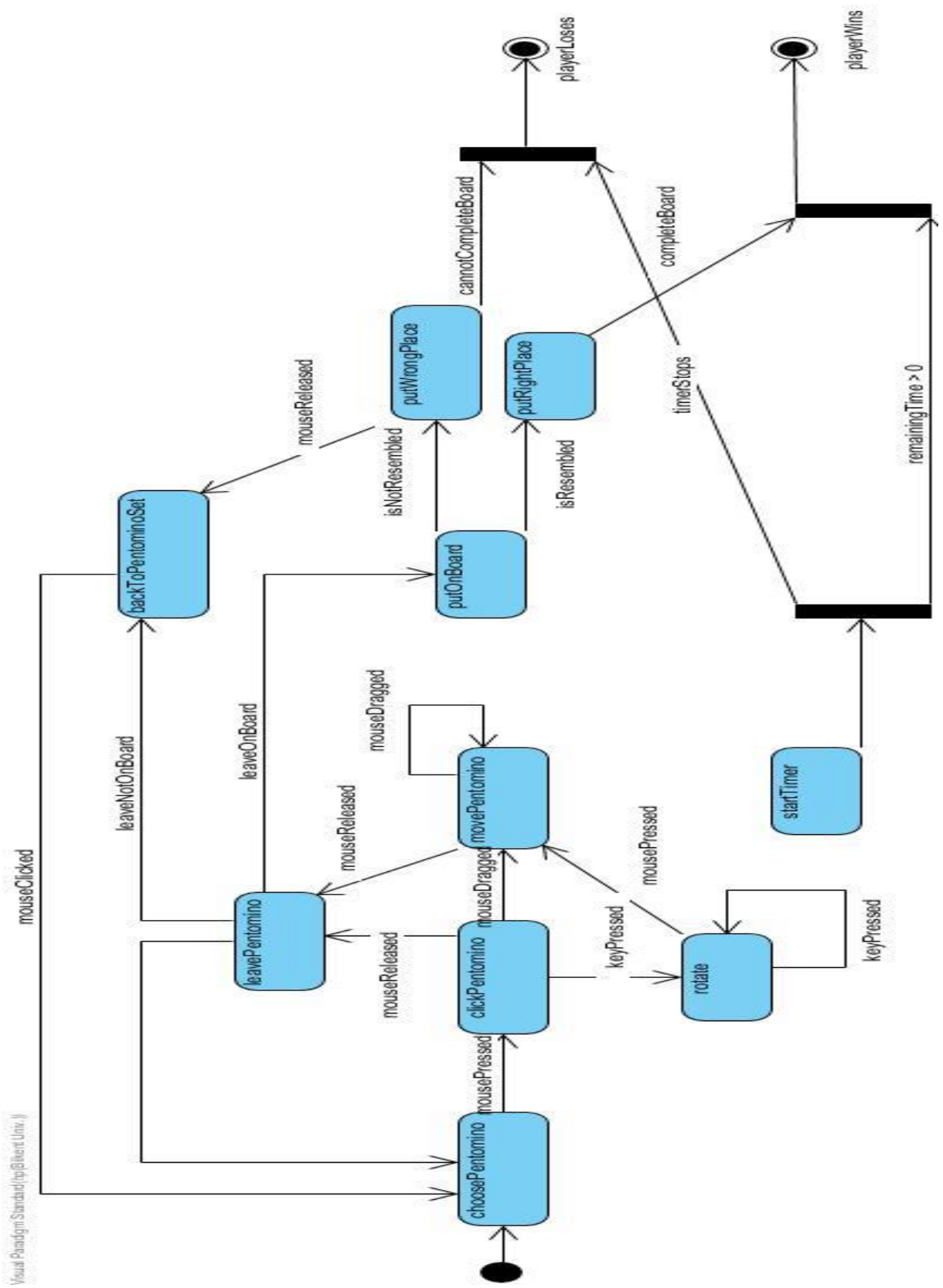

Visual Paradigm Standard(SeraFirinciogluBilkent Univ.))
[user wants to go main menu]
Ask for sample level
[exit game]
[user wants to go main menu]
showVideo
[back]
Wait for user selection in main menu
showTutorial
[user wants to play]
Get player number from user
[user wants to go main menu]
loadMultiPlayer
loadSinglePlayer
Initialize Game
Player places a pentomino
[Game mode is sample]
[coordinate is not valid]
Check if it is valid coordinate
[board is not full]
[coordinate is valid]
[user wants to go main menu]
Check if board is filled
[user wants to play next level]
[board is full, level is finished]
Check if mode is arcade
[Game mode is arcade]
[Game mode is classic]
Take user name
Ask for next level
Set leaderBoard

The program waits the user for its selection in the main menu, the user can exit, can go to the tutorial, can start a new game. When the user presses the exit button, the program exits. When the user goes to the tutorial, an educational video about the game appears and the user watches the video. After the user finishes watching the video, the user can go back to the main menu or if the user wants to learn the game more, he/she can ask for a sample level. The sample level is an instruction based level. It gives the user the instructions to complete the level. When the user finishes the sample level, he/she can go back to the main menu or he/she can dive into the actual game. In the actual game, the program asks the user how many players will play the game. The program can start a single player game or multiplayer game. The program initializes the game according to the input which comes from the user. After the initialization, the user can quit and go back to the main menu as well or the user can start to play the game.

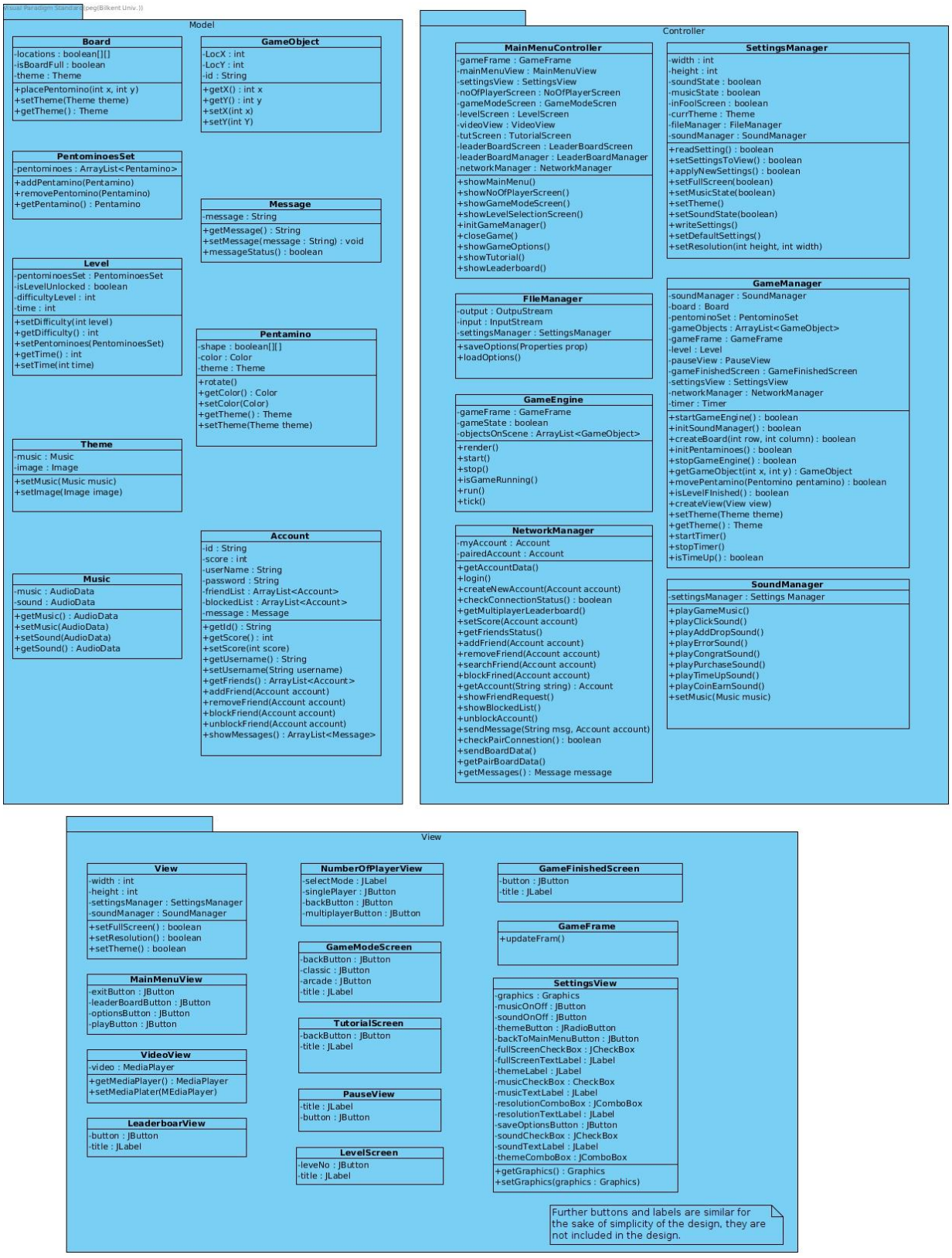
During the game, the user can choose a pentomino and place it. If the position is valid, namely the pentomino is placed on the board, the pentomino stays in there. If the position is not valid, the pentomino goes back to the Pentomino Set. If the position is valid but the blank part of the board does not resemble with the pentomino, the pentomino goes back to the Pentomino Set as well. If the board is completed, the outcomes can vary according to the game modes. If the game mode is Classic, the user can go back to the main menu or the user asks the program to generate the next level. If the user asks the program to generate the next level, the program generates the next level and initializes. If the game mode is Arcade, the program sets the leaderboard and the user asks the program to generate the next level.

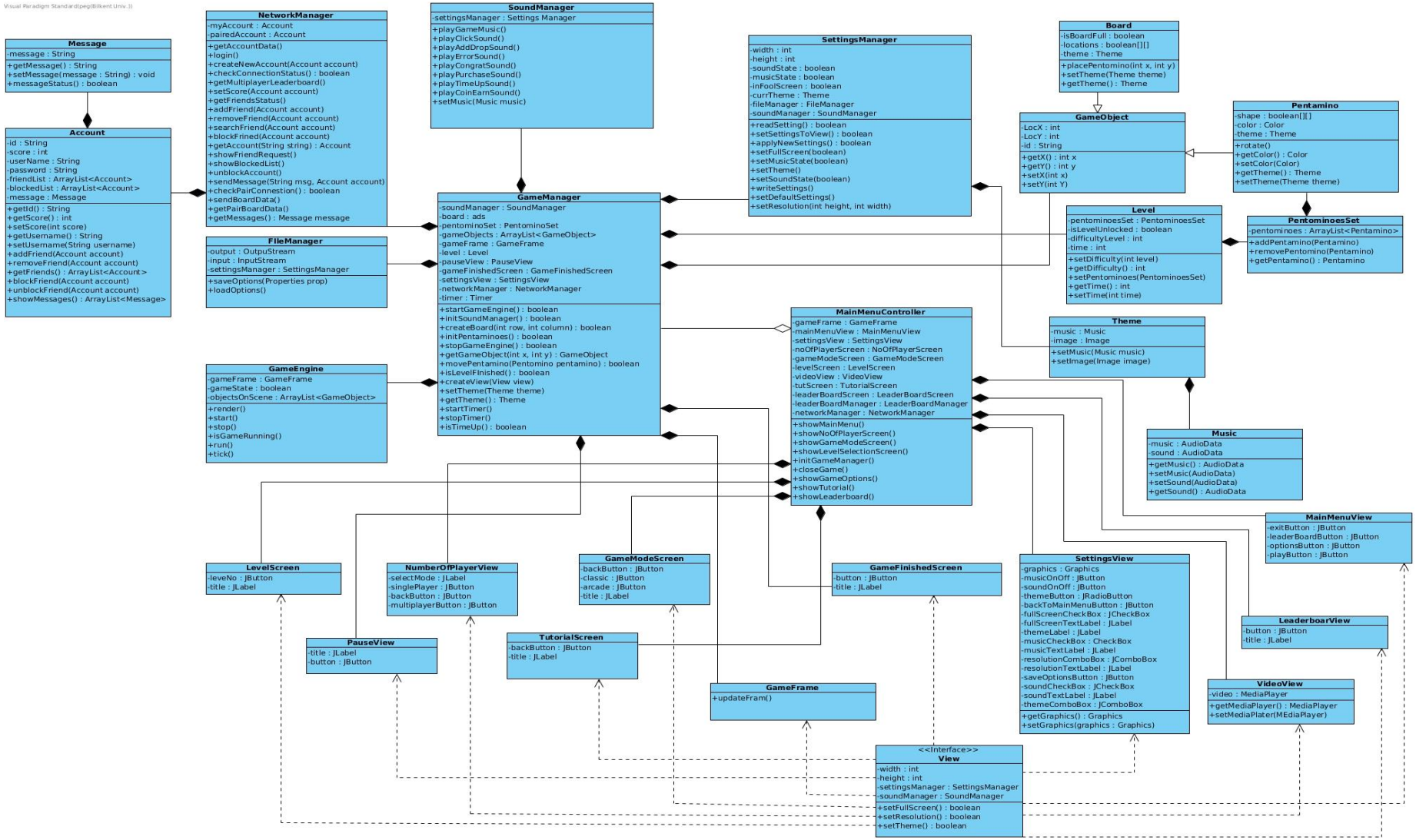
5.2.3 State Diagram

Explanation: The player chooses one of the pentominoes and clicks on them. He/she can leave the mouse button and can choose another pentomino. If the player is determined to use the pentomino, he/she can rotate the pentomino by pressing the specific key or move it wherever he/she wants to move by dragging the mouse and finally he/she can place the pentomino by releasing the mouse. If the player leaves the pentomino on board, he simply places the pentomino on the board. If the pentomino is not placed on the board, it will go back to the Pentomino Set. The player could place the pentomino mistakenly, if the blank part does not resemble with the player's choice of pentomino; therefore, the pentomino will go back to the Pentomino Set as well. Hence, the player has to complete the board by putting every pentomino in place. In the Classic Mode, there is no restriction, the player can win the game by completing the board only. The player will not use in this mode. However, in the Arcade Mode, the game has a restriction called timer, the timer starts counting backwards until the timer stops and shows the value of zero. The player has to complete the board before the timer stops in order to win the game. If the timer stops before completing the puzzle, the player loses the game.

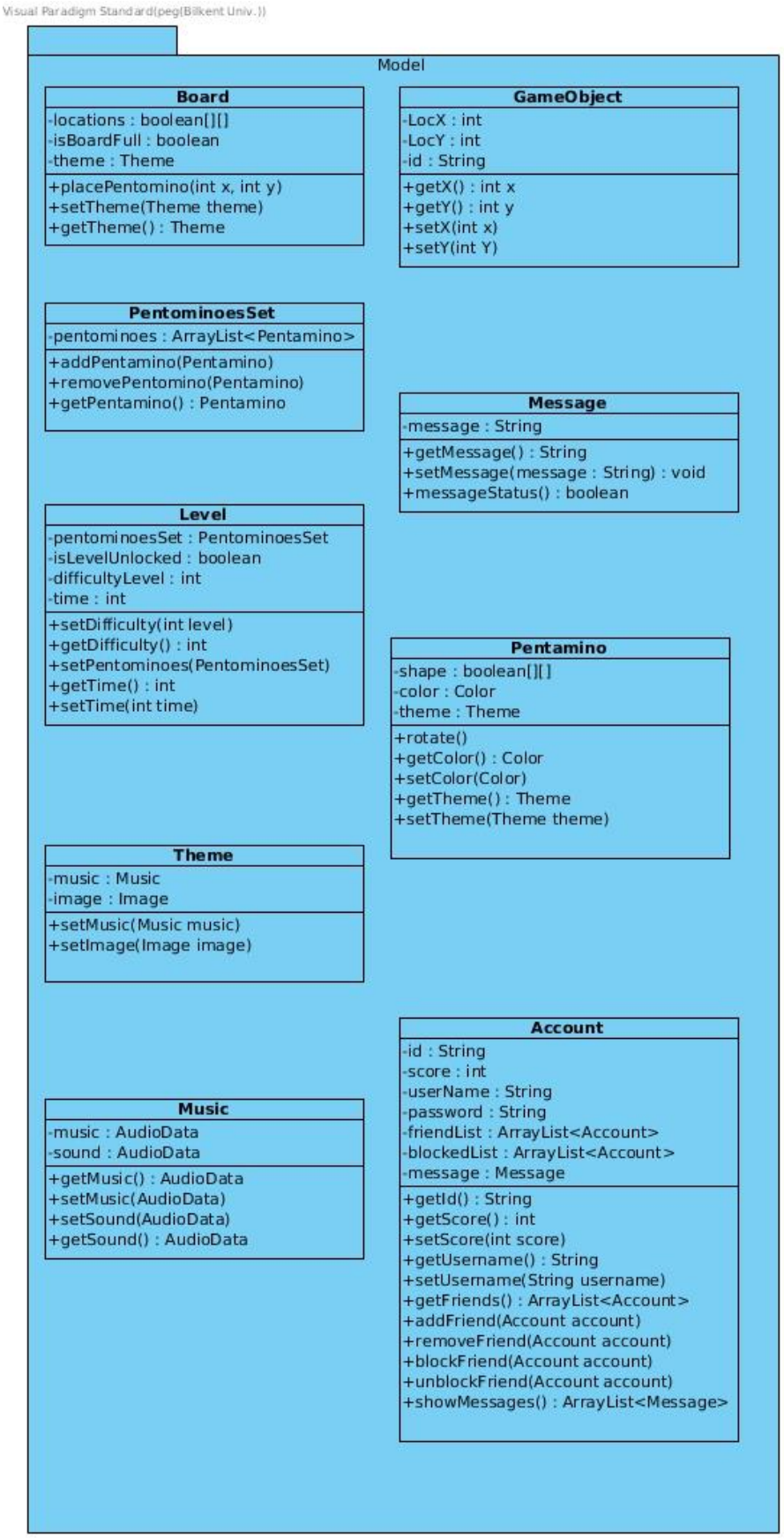

playerLoses
playerWins
cannotCompleteBoard
completeBoard
putWrongPlace
putRightPlace
timerStops
remainingTime > 0
mouseReleased
isNotResembled
isResembled
backToPentominoSet
putOnBoard
mouseClicked
leaveNotOnBoard
leaveOnBoard
mouseDragged
movePentomino
startTimer
mouseReleased
mouseDragged
mousePressed
leavePentomino
clickPentomino
keyPressed
rotate
keyPressed
mouseReleased
mousePressed
choosePentomino

5.3 Object and Class Model




Visual Paradigm Standard(peg(Bilkent Univ.))
Message
-message : String
+getMessage() : String
+setMessage(message : String) : void
+messageStatus() : boolean
Account
-id : String
-score : int
-userName : String
-password : String
-friendList : ArrayList<Account>
-blockedList : ArrayList<Account>
-message : Message
+getId() : String
+getScore() : int
+setScore(int score)
+getUsername() : String
+setUsername(String username)
+addFriend(Account account)
+removeFriend(Account account)
+getFriends() : ArrayList<Account>
+blockFriend(Account account)
+unblockFriend(Account account)
+showMessages() : ArrayList<Message>
NetworkManager
-myAccount : Account
-pairedAccount : Account
+getAccountData()
+login()
+createNewAccount(Account account)
+checkConnectionStatus() : boolean
+getMultiplayerLeaderboard()
+setScore(Account account)
+getFriendsStatus()
+addFriend(Account account)
+removeFriend(Account account)
+searchFriend(Account account)
+blockFrined(Account account)
+getAccount(String string) : Account
+showFriendRequest()
+showBlockedList()
+unblockAccount()
+sendMessage(String msg, Account account)
+checkPairConnestion() : boolean
+sendBoardData()
+getPairBoardData()
+getMessages() : Message message
FileManager
-output : OutpuStream
-input : InputStream
-settingsManager : SettingsManager
+saveOptions(Properties prop)
+loadOptions()
SoundManager
-settingsManager : Settings Manager
+playGameMusic()
+playClickSound()
+playAddDropSound()
+playErrorSound()
+playCongratSound()
+playPurchaseSound()
+playTimeUpSound()
+playCoinEarnSound()
+setMusic(Music music)
GameManager
-soundManager : SoundManager
-board : ads
-pentominoSet : PentominoSet
-gameObjects : ArrayList<GameObject>
-gameFrame : GameFrame
-level : Level
-pauseView : PauseView
-gameFinishedScreen : GameFinishedScreen
-settingsView : SettingsView
-networkManager : NetworkManager
-timer : Timer
+startGameEngine() : boolean
+initSoundManager() : boolean
+createBoard(int row, int column) : boolean
+initPentaminoes() : boolean
+stopGameEngine() : boolean
+getGameObject(int x, int y) : GameObject
+movePentamino(Pentomino pentamino) : boolean
+isLevelFInished() : boolean
+createView(View view)
+setTheme(Theme theme)
+getTheme() : Theme
+startTimer()
+stopTimer()
+isTimeUp() : boolean
GameEngine
-gameFrame : GameFrame
-gameState : boolean
-objectsOnScene : ArrayList<GameObject>
+render()
+start()
+stop()
+isGameRunning()
+run()
+tick()
SettingsManager
-width : int
-height : int
-soundState : boolean
-musicState : boolean
-inFoolScreen : boolean
-currTheme : Theme
-fileManager : FileManager
-soundManager : SoundManager
+readSetting() : boolean
+setSettingsToView() : boolean
+applyNewSettings() : boolean
+setFullScreen(boolean)
+setMusicState(boolean)
+setTheme()
+setSoundState(boolean)
+writeSettings()
+setDefaultSettings()
+setResolution(int height, int width)
Board
-isBoardFull : boolean
-locations : boolean[][]
-theme : Theme
+placePentomino(int x, int y)
+setTheme(Theme theme)
+getTheme() : Theme
GameObject
-LocX : int
-LocY : int
-id : String
+getX() : int x
+getY() : int y
+setX(int x)
+setY(int Y)
Pentamino
-shape : boolean[][]
-color : Color
-theme : Theme
+rotate()
+getColor() : Color
+setColor(Color)
+getTheme() : Theme
+setTheme(Theme theme)
Level
-pentominoesSet : PentominoesSet
-isLevelUnlocked : boolean
-difficultyLevel : int
-time : int
+setDifficulty(int level)
+getDifficulty() : int
+setPentominoes(PentominoesSet)
+getTime() : int
+setTime(int time)
PentominoesSet
-pentominoes : ArrayList<Pentamino>
+addPentamino(Pentamino)
+removePentomino(Pentamino)
+getPentamino() : Pentamino
MainMenuController
-gameFrame : GameFrame
-mainMenuView : MainMenuView
-settingsView : SettingsView
-noOfPlayerScreen : NoOfPlayerScreen
-gameModeScreen : GameModeScreen
-levelScreen : LevelScreen
-videoView : VideoView
-tutScreen : TutorialScreen
-leaderBoardScreen : LeaderBoardScreen
-leaderBoardManager : LeaderBoardManager
-networkManager : NetworkManager
+showMainMenu()
+showNoOfPlayerScreen()
+showGameModeScreen()
+showLevelSelectionScreen()
+initGameManager()
+closeGame()
+showGameOptions()
+showTutorial()
+showLeaderboard()
Theme
-music : Music
-image : Image
+setMusic(Music music)
+setImage(Image image)
Music
-music : AudioData
-sound : AudioData
+getMusic() : AudioData
+setMusic(AudioData)
+setSound(AudioData)
+getSound() : AudioData
MainMenuView
-exitButton : JButton
-leaderBoardButton : JButton
-optionsButton : JButton
-playButton : JButton
LevelScreen
-leveNo : JButton
-title : JLabel
NumberOfPlayerView
-selectMode : JLabel
-singlePlayer : JButton
-backButton : JButton
-multiplayerButton : JButton
GameModeScreen
-backButton : JButton
-classic : JButton
-arcade : JButton
-title : JLabel
GameFinishedScreen
-button : JButton
-title : JLabel
SettingsView
-graphics : Graphics
-musicOnOff : JButton
-soundOnOff : JButton
-themeButton : JRadioButton
-backToMainMenuButton : JButton
-fullScreenCheckBox : JCheckBox
-fullScreenTextLabel : JLabel
-themeLabel : JLabel
-musicCheckBox : CheckBox
-musicTextLabel : JLabel
-resolutionComboBox : JComboBox
-resolutionTextLabel : JLabel
-saveOptionsButton : JButton
-soundCheckBox : JCheckBox
-soundTextLabel : JLabel
-themeComboBox : JComboBox
+getGraphics() : Graphics
+setGraphics(graphics : Graphics)
LeaderboarView
-button : JButton
-title : JLabel
PauseView
-title : JLabel
-button : JButton
TutorialScreen
-backButton : JButton
-title : JLabel
GameFrame
+updateFram()
VideoView
-video : MediaPlayer
+getMediaPlayer() : MediaPlayer
+setMediaPlater(MEdiaPlayer)
<<Interface>>
View
-width : int
-height : int
-settingsManager : SettingsManager
-soundManager : SoundManager
+setFullScreen() : boolean
+setResolution() : boolean
+setTheme() : boolean

5.3.1 Model Package



The model part of the software helps developer to simulate real-world objects. Basically, we are modelling the objects in the software through creating the classes of the objects. In our game, we want to simulate a Katamino game and its contents such as board, pentominoes etc. In the Model part, we have Board class, GameObject class, PentominoesSet class, Message class, Level class, Pentomino class, Theme class, Account class and Music class.

Board Class: In the class, we are modelling the 2D board with 12 rows x 5 columns. It has locations which consists of 60 unit squares. The board has a theme to seem elegant as well. We can set the theme or we can see which theme is used on the board. The Board class is aware of which unit squares of the board is occupied and whether the board is occupied fully or not.

GameObject Class: The class demonstrates pentomino's number and location. The location of the pentomino can be extracted or the pentomino's coordinates can be set to another coordinates.

PentominoesSet Class: The class has a set of the fundamental pentominoes for solving the given puzzle. For instance, the user needs the specific pentominoes (e.g. number 2 pentomino, number 3 pentomino, number 4 pentomino and number 5 pentomino to complete 4 x 5 board.) to solve the puzzle and the class has a set of pentominoes which consists of number 2, 3, 4, 5 pentominoes. We can add a pentomino to the set, we can remove a pentomino from the set or we can get all of the pentominoes in the set.

Message Class: The message class contains the messages which are coming from the second player to the first player or vice versa.

The class is implemented for the Multiplayer feature. The user can send the message to the second user or the user can receive a message from the second user. We can check the status of sent or received message (e.g. Whether it is sent or not).

Level Class: The class models the level of the game, a instance of the class has a set of pentominoes for the specific level of game, we can set the pentominoes for the specific level. The levels are ordered in terms of increasing difficulty. Therefore, the instance has a difficulty level and we can set the difficulty of the level or we can see how difficult the level can be. The level has a timer in Arcade Mode, that means we can set the timer and we can get what the timer shows. If the user manages to complete the level before the timer stops, the next level is unlocked and it will be more difficult than completed level.

Pentomino Class: The class models the pentominoes, their 2D shape, color and theme. Their theme will resemble with the theme on the board. We can change the color and theme of a pentomino or we can see the theme and the color of the pentomino. The user can rotate the pentomino to put it on the board as the user wants.

Theme Class: In the theme class, a theme has the image and the

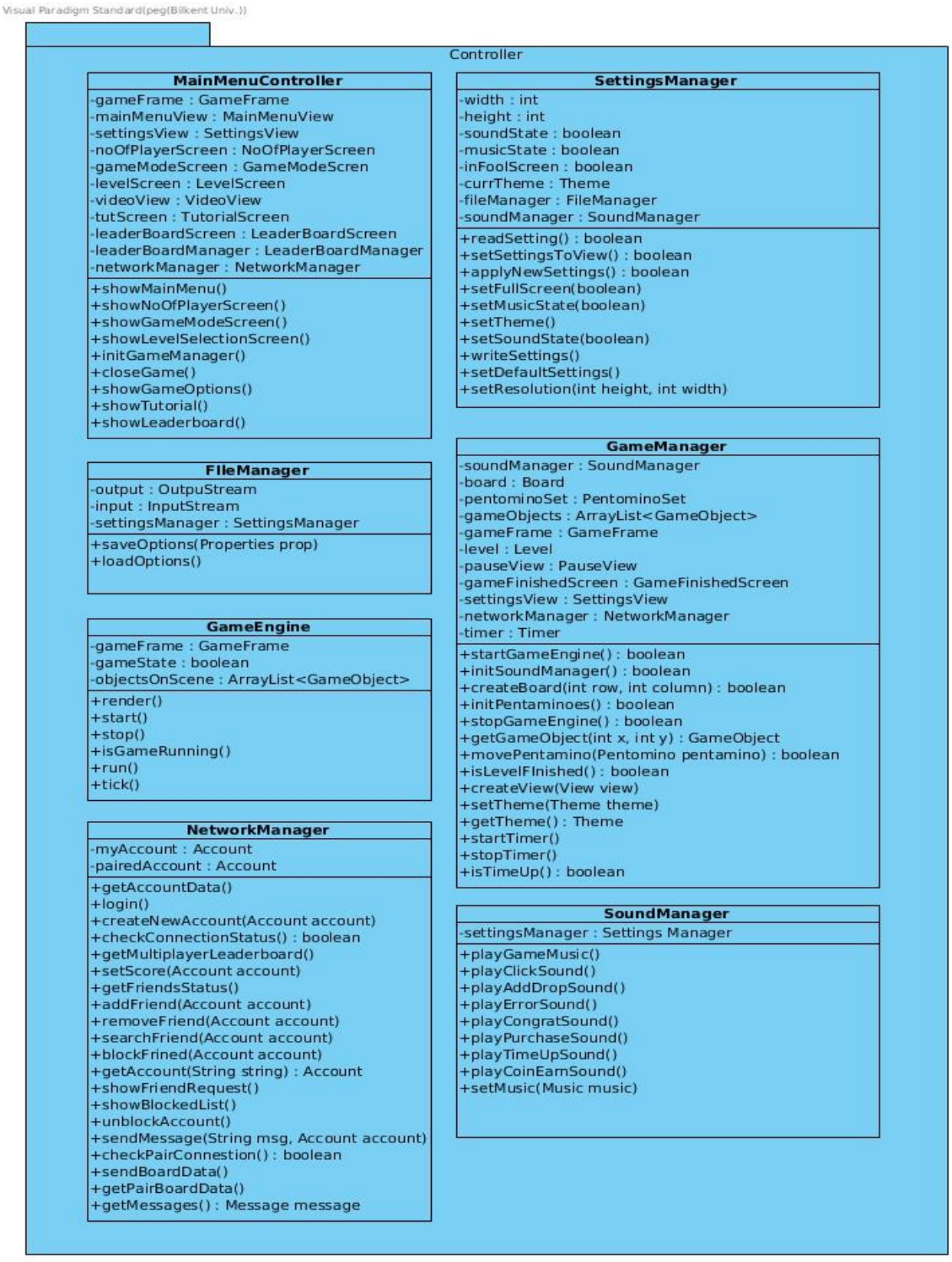
music instance from the Music class which corresponds to the nature of the theme. The image and the music instance can be sent to the SettingsManager so that the SettingsManager can set the image and the music to the appropriate place.

Music Class: The instance from the Music class has been grouped with a music and a sound file which have a common correlation. The music and the sound file are the actual audio files and they can be set to the different music and sound files. They will be sent through the instance of Music class to the SettingsManager.

Account Class: The class simulates the social role of a person. The user will have an account with an user ID. In the account, the account has an username and password so that the user can access to his/her account. When the user signs into his/her account successfully, the progress in the game can be saved through the account and when the user completes a puzzle successfully, the score of the user can be set through his/her account as well. In addition to this,

the user can make a friends list and he/she can block another user as well. The user can chat with his/her friend from the friends list. Namely, the user can see the message which is sent from another user or he/she can send his/her messages to another user. The user can remove another user from the friends list and the user can unblock another user from the blocked list.

5.3.2 Controller Package



Controller package is the package which contain all the decision making mechanisms. For now it consists of 7 classes, each of them is assigned with a different purpose. These classes are in-between the model classes and view class. They call their necessary functions whenever view classes receive an user input.

MainMenuController:

This class is responsible for main menu actions. It instantiates views related to the main menu such as MainMenuView, SettingsView, LearderBoardView, and sets them to the GameFrame. By doing those it allows user to navigate through main menu. Also, one of the most important roles of this class it to instantiate the actual game. In order to achieve this it collects informations by showing couple of views. After that desired game can be created according to user's decisions. It is also possible to see the ranking of the player through generating a leaderboard view through this controller. Moreover, this class closes GameFrame and other controller classes to terminate the whole game.

SettingsManager:

Another part of this package is the SettingsManager. Any changes made through settings are firstly saved to a singleton instance of this class. Then it informs view classes and the SoundManager class about these changes to let them apply. It works with the FileManager class in order to save these setting to a local properties file. SettingsView also gets settings informations from this class to represent it to the user. Example to the settings this class is responsible for are the resolution of the game, whether game is fullscreen or not, desired theme of the game and the states of the sound and music options.

FileManager:

FileManager is the class that regulates necessary output and input streams. It writes game's settings and singleplayer leaderboard to separate local files. Whenever user runs the game, same files are read and set to settings in order to remember users preferences.

SoundManager:

SoundManager plays the music of that level depending on theme of the game. It also generates sounds to give user a feedback. It includes sounds like clicking sound, drag & drop sound, invalid operation sound and time is running short sound and more. It class is important to give user a feedback and make the game more enjoyable by playing music.

NetworkManager:

Network manager is the main communication unit of this game with other players. It communicates with the database to transmit and receive data related to multiplayer part of the game. It is possible to draw paired players board by the data this class fetches. In order to make game lag-free, it will constantly fetch data. It also brings data related to online leaderboard whenever user wishes to shows his/her rank among other players.

GameEngine:

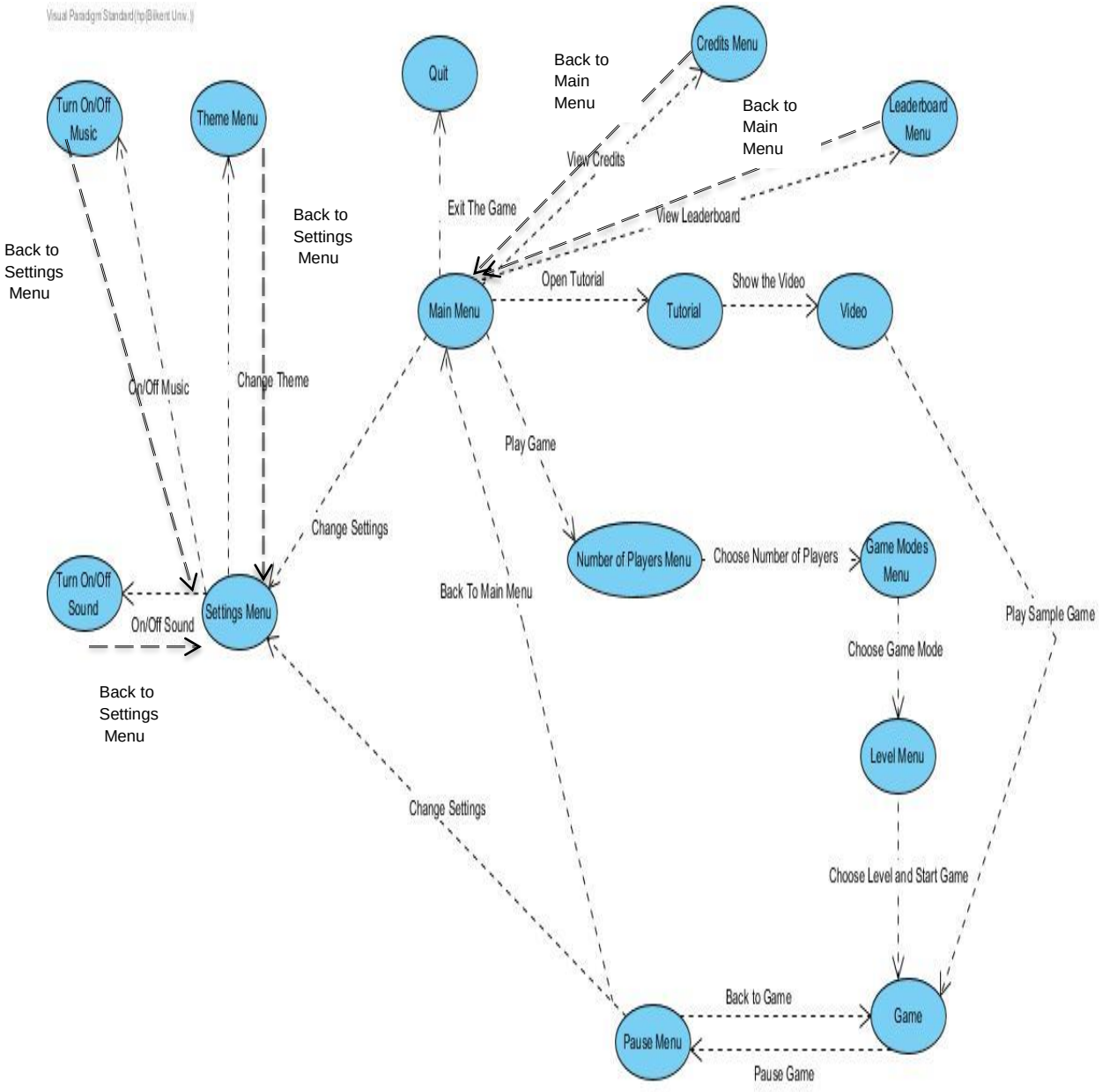
Purpose of this class is to constantly render the GameObjects contained in the GameManager class to show movements such as drop & drop on the GameFrame. For now it uses a single thread approach but depending on the lack of fps, multi-thread approach might be used. It also prints the fps value to the console to help developers to keep track of performance of the game.

GameManager:

GameManager class is might be the most important part of this game. It controls the main flow of the game. It starts and stops the GameEngine which essentially means all of the visual contents of the game. It also checks the game state to determine whether the game is still being played or not. All of the game logic is also calculated and done in this class. Locations of the pentominoes are changed by the help of this. Moreover, inputs given by the user such as the coordinate of the mouse click, name of the button being pressed and other inputs are handled here. View classes like the PauseView is generated from this class when required the certain input are received from the user. Another duty of this class is to forward user to the correct screens when needed or when the game is over.

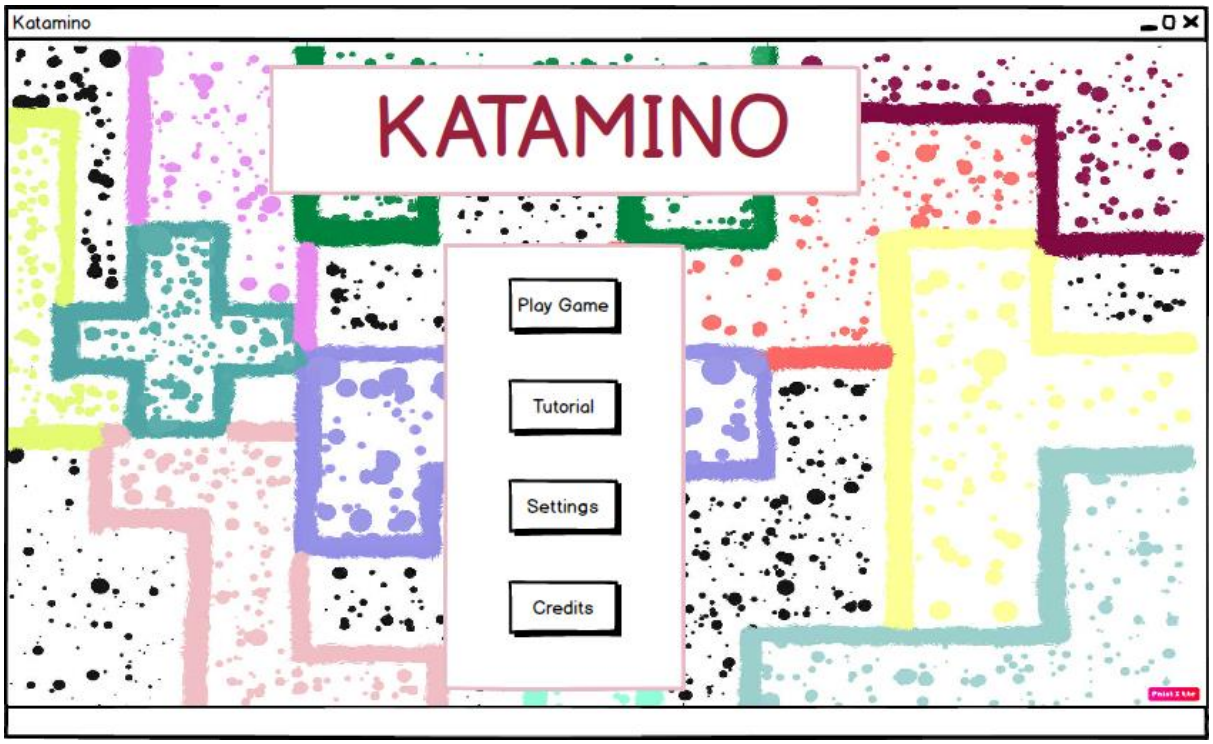
5.4 User Interface

5.4.1 Navigational Path



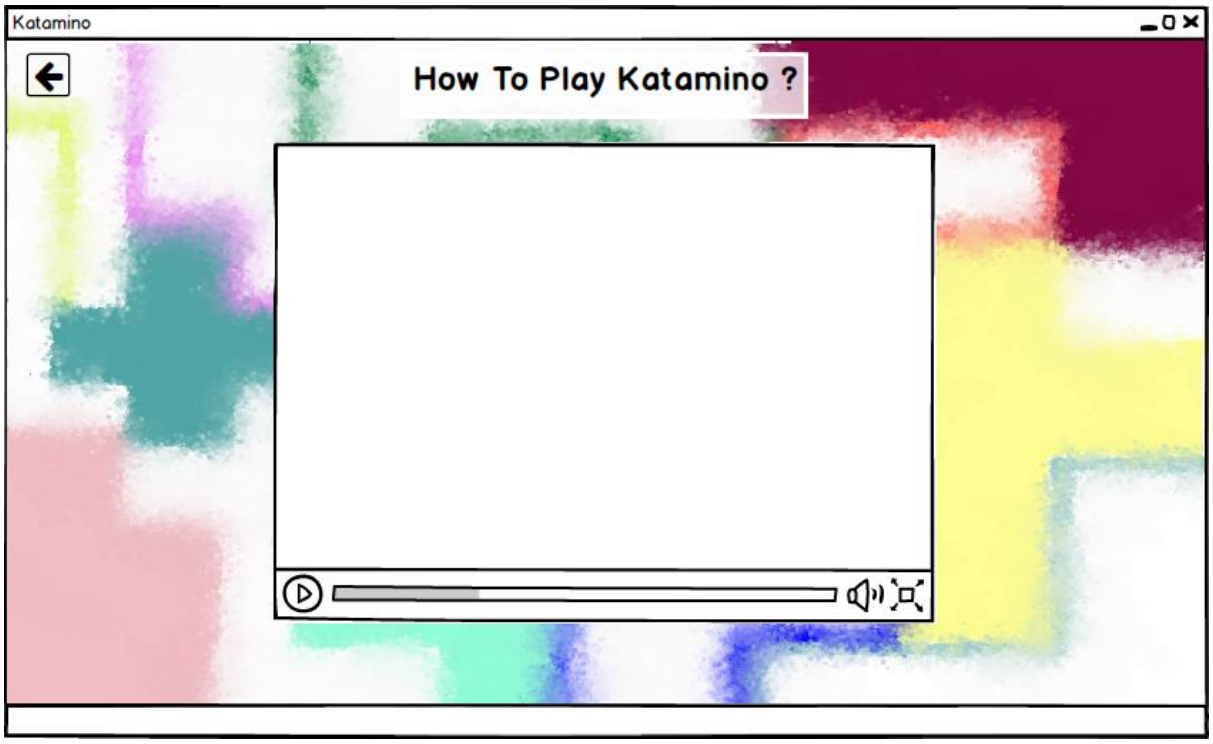
5.4.2 Screen Mock-Ups

5.4.2.1 Main Menu

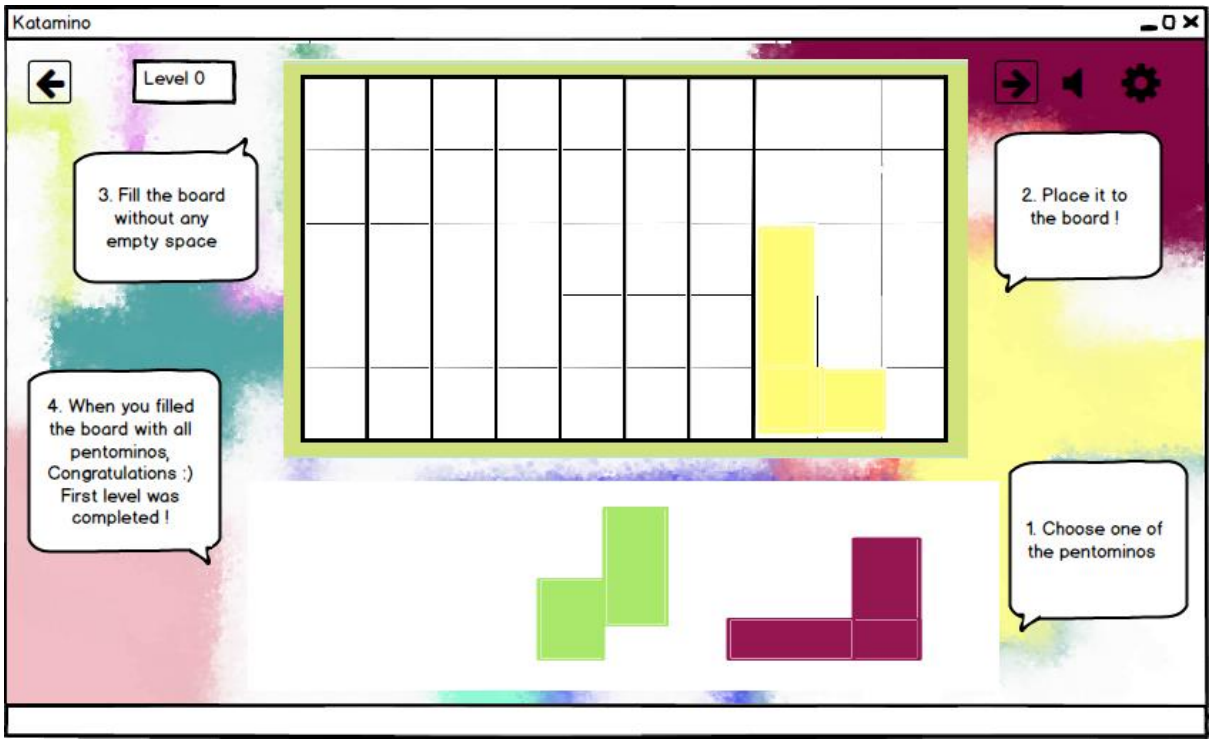


Main menu is the start menu of the game which consists of four buttons and one extra exit button. The four buttons are Play Game, Tutorial, Settings and Credits respectively. If the user chooses Play Game, a new game initializes. If the user chooses Tutorial, the tutorial initializes. If the user chooses Settings, he/she can adjust the settings of the game. Finally, if the user chooses Credits, he/she can press f to pay respect of the creators of the game.

5.4.2.2 Tutorial



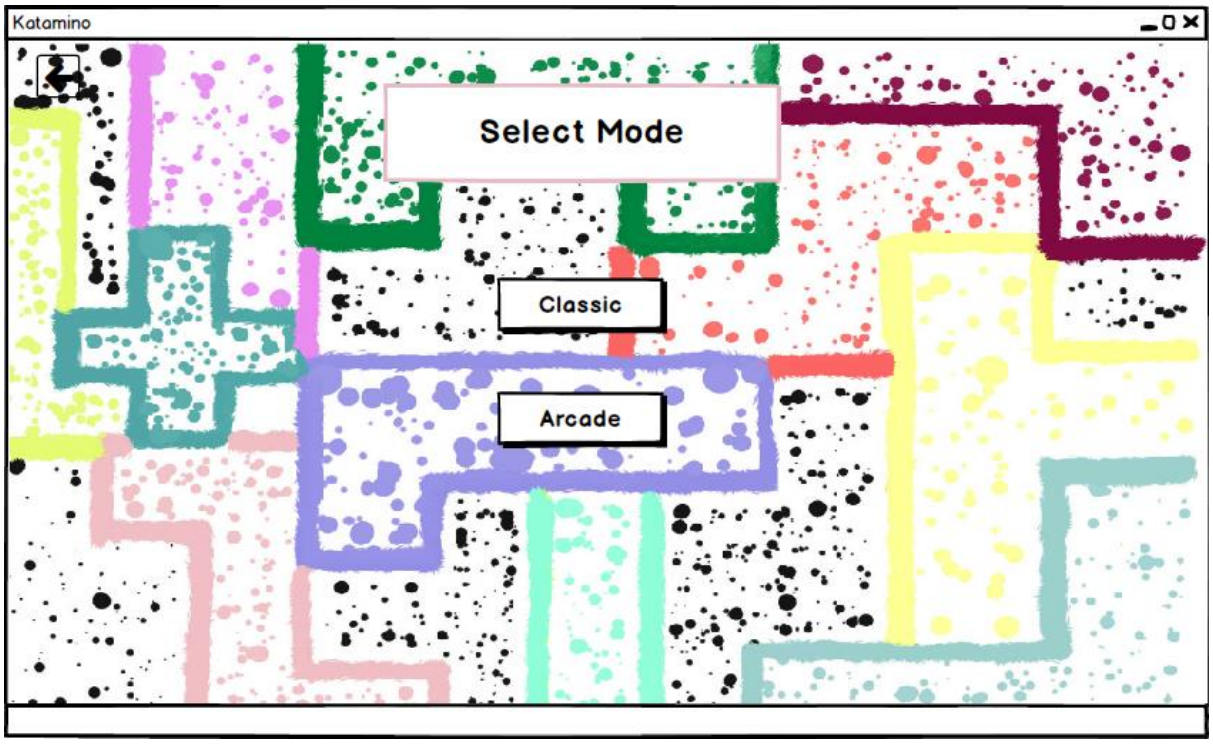
Tutorial screen provides the user with the information about the game mechanics such as rules, controls and shortcuts. At first, the user can access how to play the game by watching the video. After watching the video, the user can go back to the main menu.



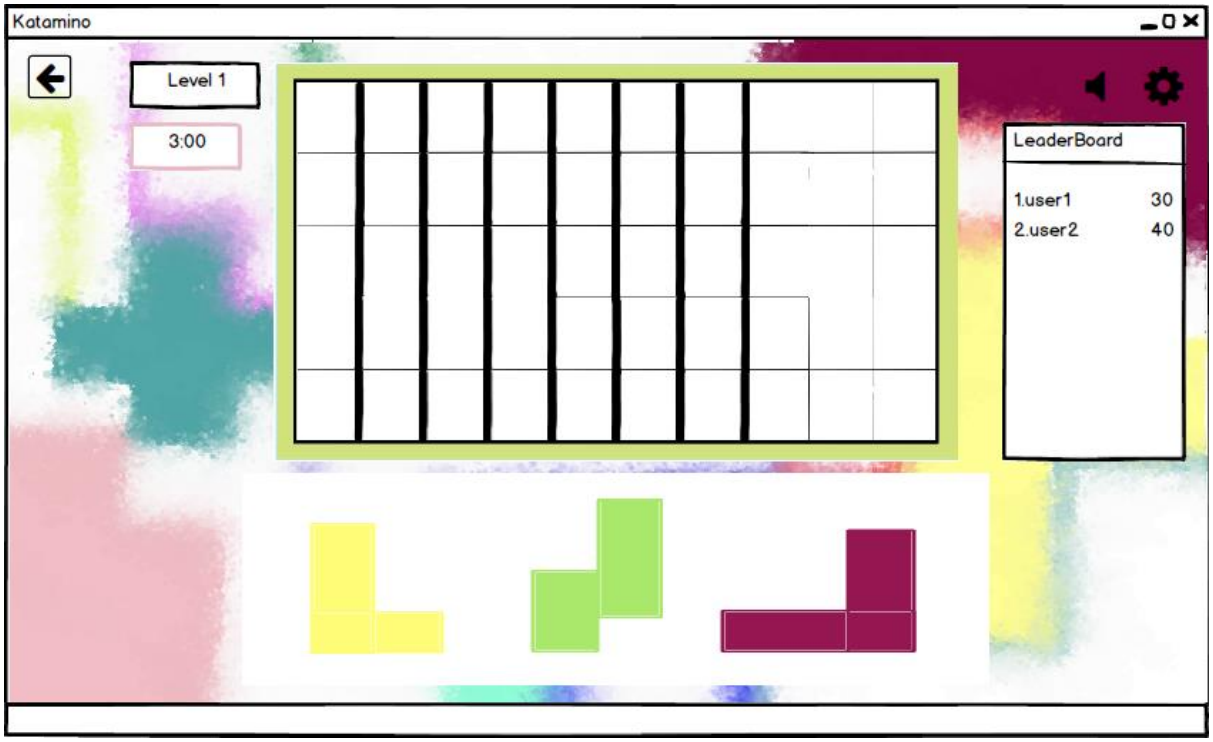
If the user wants to continue, a sample level will be provided. The user can finish the sample level by following the instructions. To do that, the user can choose one of the pentominoes and places it on the board. By following the steps, the user should fill the board without the empty space. When the user filled the board with all pentominoes, the sample level will be completed.

5.4.2.3 New Game

When the user clicks New Game, the program provides the user with the selection of the number of players screen. The user can press Single Player button, if the user wants to play itself. If the user wants to play with his/her friend, he/she can press Multiplayer button.

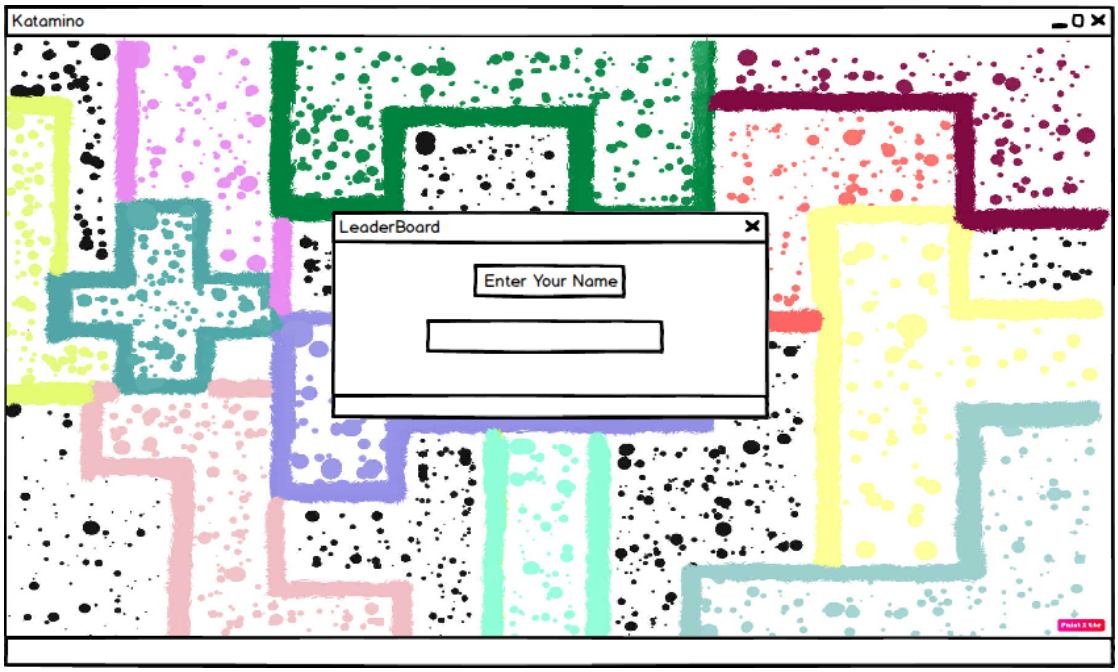


After the user presses Single Player or Multiplayer button, he/she can choose the game modes, which consists of Classic and Arcade mode. If the user chooses the Classic Mode, the classic Katamino game with no competition (eg. timer) appears. The Arcade Mode, the competition mode, appears on the screen when the user presses the Arcade Mode button.



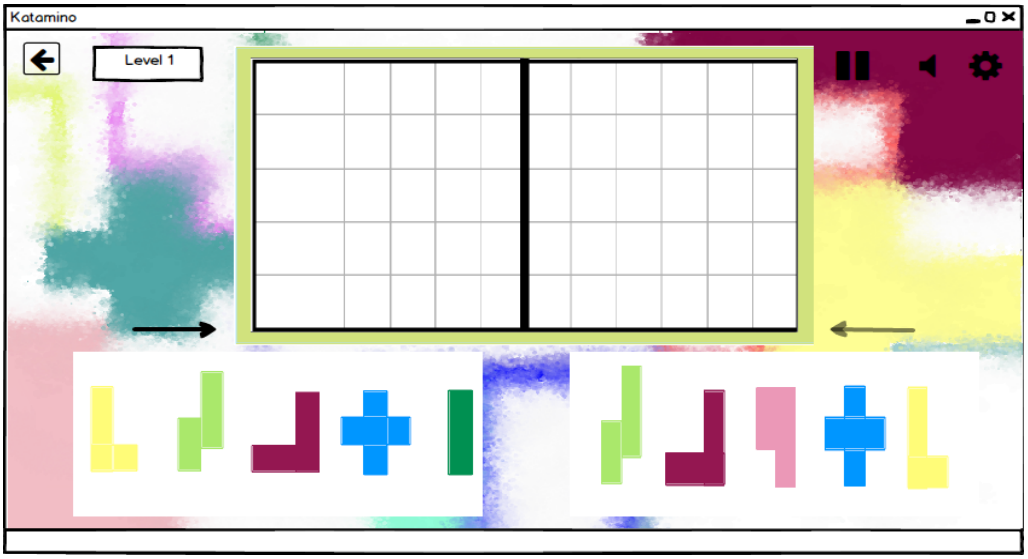
In the Game Screen, the users are going to see the 12 x 5 Board, which consists of 60 squares and some polygons which consists of 5 squares called Pentominoes. The user tries to place the pentominoes to complete the board while the timer continues to decrement itself.

When the user completes the board, the level completion screen appears. The screen informs the user that he/she successfully completes the puzzle.



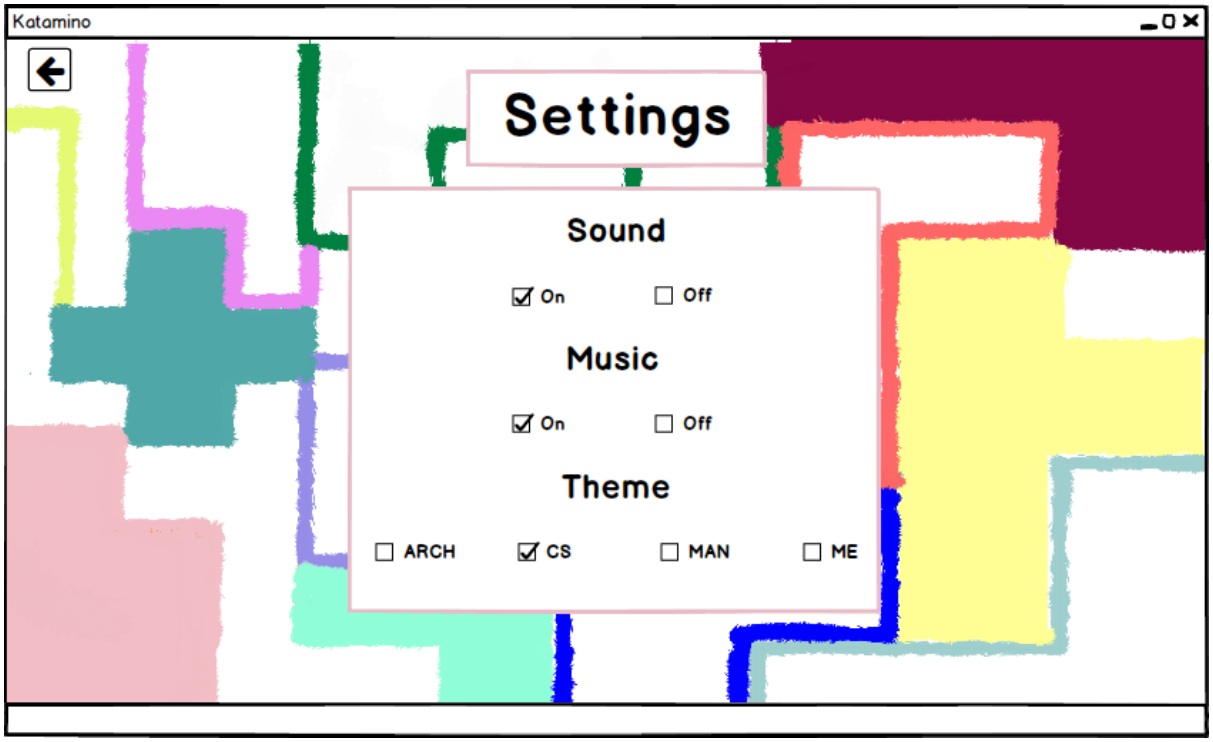
After player finish the level, he/she can enter his/her name in the enter your name screen. Then name will be added to the leaderboard.

5.4.2.4 Multi Player Game



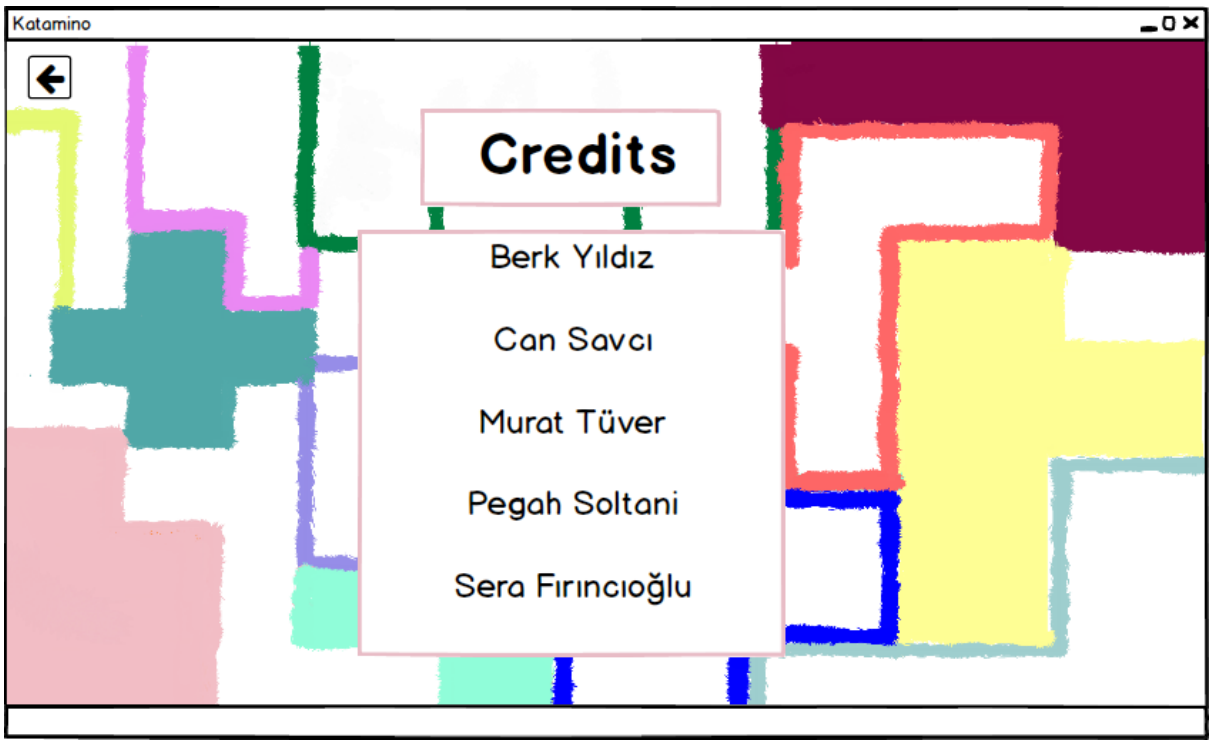
Player can also choose multiplayer mode. There will be arrows in the screen that shows which players turn. There is going to be equal pentominoes and space for both players.

5.4.2.5 Settings



Settings screen provide user to change of sound, settings and theme.

5.4.2.6 Credits



Credits screen shows the contributors of the game to the user.

6. Conclusion

As a summary of the whole project, we decided to implement Katamino which is a board game that is suitable for all ages. The game basically consists of a board, a level divider and pentominoes which are different shapes made of identical squares. The user will be asked to fill the board using the shapes that are provided by the game itself. Along with the fundamental features of the game, we are also planning to develop other features in order to make the game more competitive and interesting for the user. These extra features consist of changing the theme of the game anytime desired by the user, capability of winning scores and purchasing extra shapes, multiplayer mode and a messaging system.

7. References

luckypennyshop, “Katamino Puzzle Game - Fill The Pentominos To Make A Square!,” *YouTube*, 03-Aug-2016. [Online]. Available: https://www.youtube.com/watch?v=v_AWN2O1qeo&t=25scom/watch?v. [Accessed: 20-Oct-2018].

“Homepage,” *Gigamic Games*. [Online]. Available: http://en.gigamic.com/game/katamino-family. [Accessed: 21-Oct-2018].

“Pentomino,” *Wikipedia*, 22-Sep-2018. [Online]. Available: https://en.wikipedia.org/wiki/Pentomino. [Accessed: 21-Oct-2018].