Cs 319-Object-Oriented Software Engineering

Fall 2018

Analysis Report - Iteration 2

Super Katamino

Group 2K / Section 02

- Mert Özerdem - 21300835
- Sine Mete – 21302158
- Hasan Doğan - 21402109
- Sencer Umut Balkan - 21401911
- Mustafa Azyoksul - 21501426

Instructor - Eray Tüzün

Table Of Contents

1. Introduction

Super Katamino is a puzzle game which project group 2K is inspired by a board game called “Katamino”[1]. The basic gameplay of “Katamino” the board game and “Super Katamino” is similar to each other with some improvement on to the vanilla Katamino the board game, which is the completing given board with available pieces without missing a spot on the table. Board game and Super Katamino shares same type of pieces such as 3x1,1x2 and 2x2 pieces. Pieces are rotatable in order to get more combinations and longer gameplay time with different designs.

We will use JavaFX to implement our project. A mouse or touchpad is a must for use of Super Katamino the game.

2. Overview

2.1 Gameplay

Super Katamino is a 2D puzzle game which is inspired by Katamino the board game. So gameplay of Katamino and super Katamino are similar to each other with some improvements added here and there. The aim of the game is to fill all available space on the template table, which varies from 4x4 to 10x10 square template tables to 12x5 rectangle template table, by using generated pieces that designed for specifically to fit on the template table. The player needs a mouse to drag and drop the pieces on the template table in order play the game. If the player misses the place on the template table or spot is already filled, the game returns the piece its original place. The game finishes after template table is filled without a spot and continues to the next level if it is not the last level of the game. Then his or her score will be displayed if a player can manage to beat higher scores on the scoreboard.

2.2 Pieces

Pieces of the Super Katamino is the core part of the game and they are generated randomly. By algorithm that divides nxn template tables by n different pieces which can vary in size and shape such as L, Z and U shaped pieces. These pieces interact with the mouse and moves with the desire of the player's will.

2.3 Template Table

Template table is another crucial part of the Super Katamino The game. Template tables are generated according to the difficulty of the level. These tables can vary from 4x4 to 10x10 square and 12x5 rectangle table. These tables create win condition for the game.

2.4 Time Counter

Time counter implemented to give a sense of achievement to the player. Time counter counts the completion time of the level and assigns a point according to level's difficulty and time spend on the level.

2.5 Score Keeping

Scores add to each other after each level completion until the game is finished or the player exits from the game. Then these scores are displayed on the scoreboard if the point is higher than the previously scored points.

3. Functional Requirements

3.1 Main Functional Requirements

- Play Game

The main function required of the game is "Play Game" function. This game function runs the game and displays on the screen for user to enjoy it. A player can start this function by just clicking the start function on the main menu. Then this function generates a level and pieces in order to start the game.

The game is designed to be played by drag and drop approach. The player clicks on the piece that he wants to drop on the template table and drags it to the place that he sees fit, then the player can drop the piece to fit the place. If the piece is out of place or dropped on a place that is already occupied it returns to the original place that the player picked up the piece.

The main objective in the game is to fill all the available places on the template table without skipping a place on the table. Players can get more score by completing harder difficulty levels or completing in less time interval. If conditions are satisfied game moves on the next level and keeps the score.

- Help

This function designed to help people to play and learn the game as the name "Help" suggests. This function explains:

1. Game Rules
2. How to interact with the game
3. how scores are kept
4. how to mute sound and music

Players can reach the information they needed to play the game by just clicking the help button on the main menu. Which makes the game more accessible to players of all ages.

- Settings

This is a main menu function that is designed for shut off or reactivate the music and sound of the game to get more optimal gameplay experience for the player. This function sets up:

1. Music
2. Sound

- High Scores

Players can get some sense of achievement by the implementation of this function. A player can reach this board by clicking the “High Scores” function on the main menu. This function displays players with high scores on the UI when clicked. These scores are calculated by formulating difficulty of the level x completion interval of the level which equals to score of the player’s end round score. After the level score is calculated it is added to the previous level’s score until the last level of the game is reached or game is prematurely stopped.

- Credits

From main menu, credits screen is available. With credits, players are able to reach the the identity and contact information of the developers of the game.

- Players can start from the first level. The next level is unlocked when previous level is passed.
- Players should be able to control pieces and give inputs via mouse clicks and drags. Players can also retrain a piece move by clicking on that piece.
- Players are able to pause game via pause button.
- Players navigate through main menu and other screens via back and named buttons.

3.2 Additional Functional Requirements

We will add some new features to vanilla Katamino board game in order to make the project more fun in the long run.

1. Random level generator

We will generate the game table random, instead of using solid pictures for every level.

2. Sound and Music

Since it is a desktop game, sound and music is important.

3. Keeping scores according to time and number of used pieces

We give a degree for multiply the score according to variety of used pieces by player. Player will gain more score when s/he uses more diverse pieces. Keeping scores will also occur according to time passed.

4. Limited Piece Mode and Unlimited Piece Mode

There is two modes in our game. Limited piece mode is played with limited number of pieces; unlimited piece mode is played with unlimited number of pieces. Their score evaluation is different.

4. Non-Functional Requirements

4.1 Main Non-Functional Requirements

- Usability: The game is designed as user-friendly as possible. System provides user-friendly menu interface, the user will be able to reach game instructions and settings easily.
- Ease of Learning: The user may not have knowledge about how the game is played, scores and logic of the game; it is fundamental for the user to obtain information about the game concepts to play the game.
- Portability: Portability is an important issue for any software since it makes software be accessible for wide range of users. Therefore, we determined to implement the system in Java for providing cross-platform portability in any operating system in which JVM exists.
- Reliability: Our game will be bug-free. The system should not crash with unexpected inputs. The system will be tested after implementation.

4.2 Additional Non-Functional Requirements

- Frame rates

The frame rate should be in a reasonable range. The minimum and average frame rate should be 20 and 30 frames per second respectively.

- Response time

The average response time between click and reaction should be less than half seconds. The maximum response time between click and reaction must be two seconds.

- Required resources

The game is going to be able to run with 1024 MB of RAM. Also the game will use less than one gigabyte of hard disk space. Checking the total size of the folder in which the game was installed and checking the physical memory in the Windows Task Manager Performance tab are how we are going to know these requirements are satisfied.

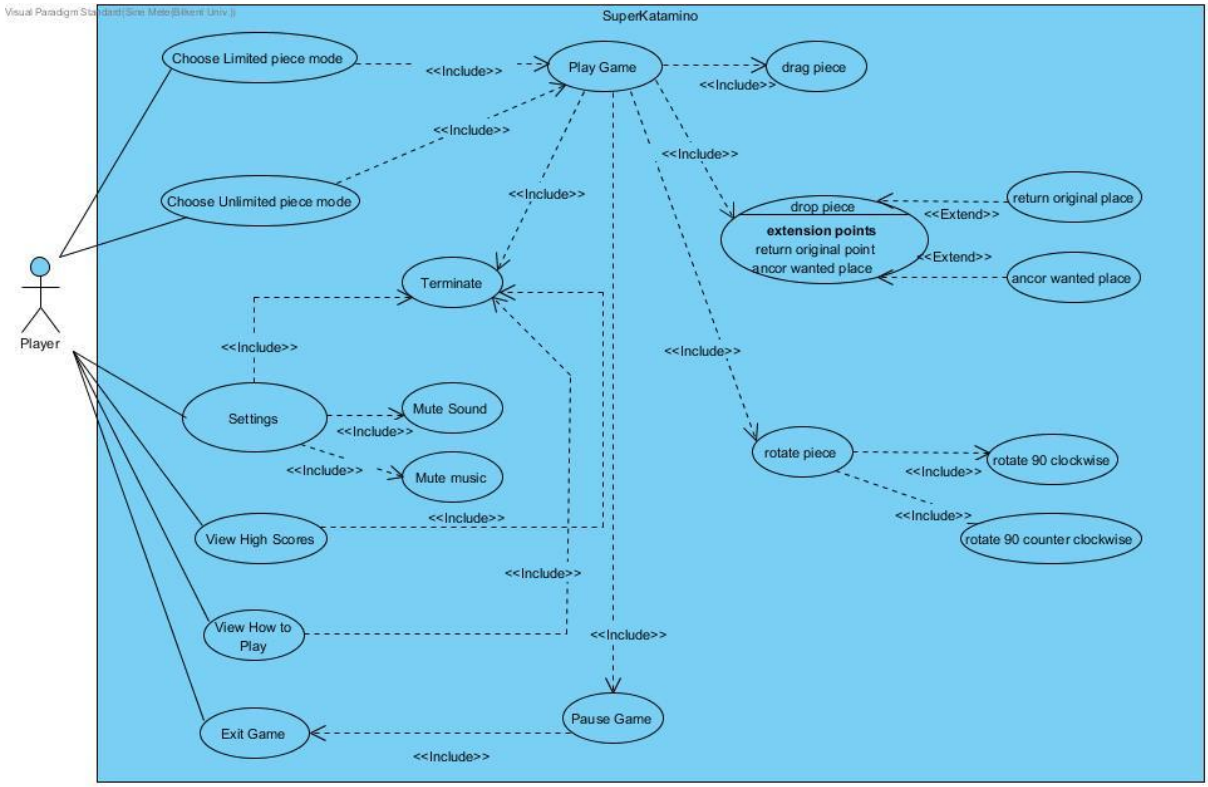
- Platform

The game will be compatible with Windows 7. To play the game, user will be required to use Window 7 or above.

5. System models

5.1 Use case model

In this part of the UML models, we show interactions between the user and the system by using UML use case models with different models that each emphasizes a different aspect of user system interaction.



Use Case 1: Play Game
Primary Actor: Player

StakeHolders and Interest:

1. Player clicks on “Play Game” button to initialize game.
2. System initializes game.

Pre-conditions: Player must be in the main menu

Main flow of events:

1. Player choose Limited Pieces Mode.
2. Program starts game.
3. Player drag pieces to the places s/he considered right.
4. Player fills all the places and solves the puzzle.
5. Program finishes the level and passes next level.

Alternative flow of events:

1. Player starts playing the game.
2. Player clicks red X button and quits.

Entry Condition: Player should choose the game mode.

Exit conditions: Player selects exit button from the pause menu.
Player successfully finished the game by completing all levels
Player terminates the program.

Use Case 2: View HowtoPlay

Primary Actor: Player

StakeHolders and Interest:

1. Showed how to play text.

Entry Condition: Player must be in the main menu.

Exit condition: Player returns to the main menu.

Main Flow of Events:

1. Player chooses to view How to Play screen.
2. The system displays the how to play text about the game.
3. After reading the text, player returns to the main menu.

Use Case 3: View Highscore

Participating actors: Player

Main flow of events:

1.The High Score Screen comes up.
2. System takes high scores from file/HDD and displays on the frame.

Entry condition: Player is in Main Menu.
Player has already opened the game.

Exit condition: Player clicks back button to go back to Main Menu.

Use case 4: ChangeSettings

Participating actors: Player

Entry condition:

1. Player has already opened play game.

2. Player is is in main menu

Exit condition: Player returns to main menu.

Main Flow of Events:

1. Player chooses to view settings menu.
2. The system displays different adjustments on the gameplay.
3. Player chooses settings.
4. Player returns to main menu.

Alternative Flow of Event:

- Player does not change any settings. (go to step 4)

Use Case 5: mute sound
Primary Actor: Player
Main flow of events:
1 Player clicks on “mute sound”
2. System cuts off sound.
Entry Condition:

1. Settings menu must be opened.

Main Flow of Events:

1. Player chooses to view settings menu.
2. The system displays different adjustments on the gameplay.
3. Player chooses settings.
4. Player clicks on mute sound button.
5. Player exits.

Use Case 6: mute music
Primary Actor: Player
Main flow of events:

1. Player clicks on “mute music”
2. System cuts off music.

Entry Condition:

1. Settings menu must be opened.

Main Flow of Events:

1. Player chooses to view settings menu.
2. The system displays different adjustments on the gameplay.
3. Player chooses settings.
4. Player clicks on mute music button.
5. Player exits.

Use Case 7: drag piece
Primary Actor: Player
Stakeholders and Interest:
1. System moves piece with the direction of the mouse.

Entry Conditions:

1. Player must open play game.

2. Player must click on valid piece object.

Main flow of events:

1. Player drags piece in order to fill desired places template table.

Use Case 8: drop piece
Primary Actor: Player
Stakeholders and Interest:

1. Player drops the piece in order to fill available space in template table.

Main flow of events:

1. System returns or anchors the piece.

Entry Condition:

Player must have open play game.

Main flow of events:

1. Player releases right mouse button.
2. Piece attaches to table space

Alternative Flow of Event:

1. Player releases right mouse button.
2. Piece does not match with table space.
3. Piece returns to original space.

Use Case 9: return original place
Primary Actor: Player
Entry Condition:

1. Piece must be dropped on to some place on game UI.

Main flow of events:

1. Player clicks on the dropped piece.
2. System returns the piece to original spot.

Use Case 10: anchor wanted place
Primary Actor: Player
Stakeholders and Interest:

1. Player drops the piece in order to fill available space in template table.
2. System anchors the piece to original spot.
   Entry Condition:
3. Piece must be dropped on to free space on template table.

Use Case 11: rotate 90degree clockwise
Primary Actor: Player
Stakeholders and Interest:

1. System rotates piece 90 degree clockwise according to object specifications.

Entry Condition:

1. A piece must be selected.

Main flow of events:

1. Player clicks on rotate button.
2. Piece is rotated 90 degree clockwise.

Use Case 12: rotate 90degree counter clockwise
Primary Actor: Player
Stakeholders and Interest:
1. System rotates piece 90 degree counter clockwise according to object specifications.

Entry Condition:
1. A piece must be selected.

Main flow of events:
1. Player clicks on rotate button.
2. Piece is rotated 90 degree counter clockwise.

Use Case 13 : Exit
Primary Actor: Player
Stakeholders and Interest:
1. Player clicks on exit button.

Entry Condition:
1. Player clicks on exit button.

Main flow of events:
1. Player clicks exit button.
2. System terminates.

5.2. Dynamic models

5.2.1 Scenarios

5.2.1.1

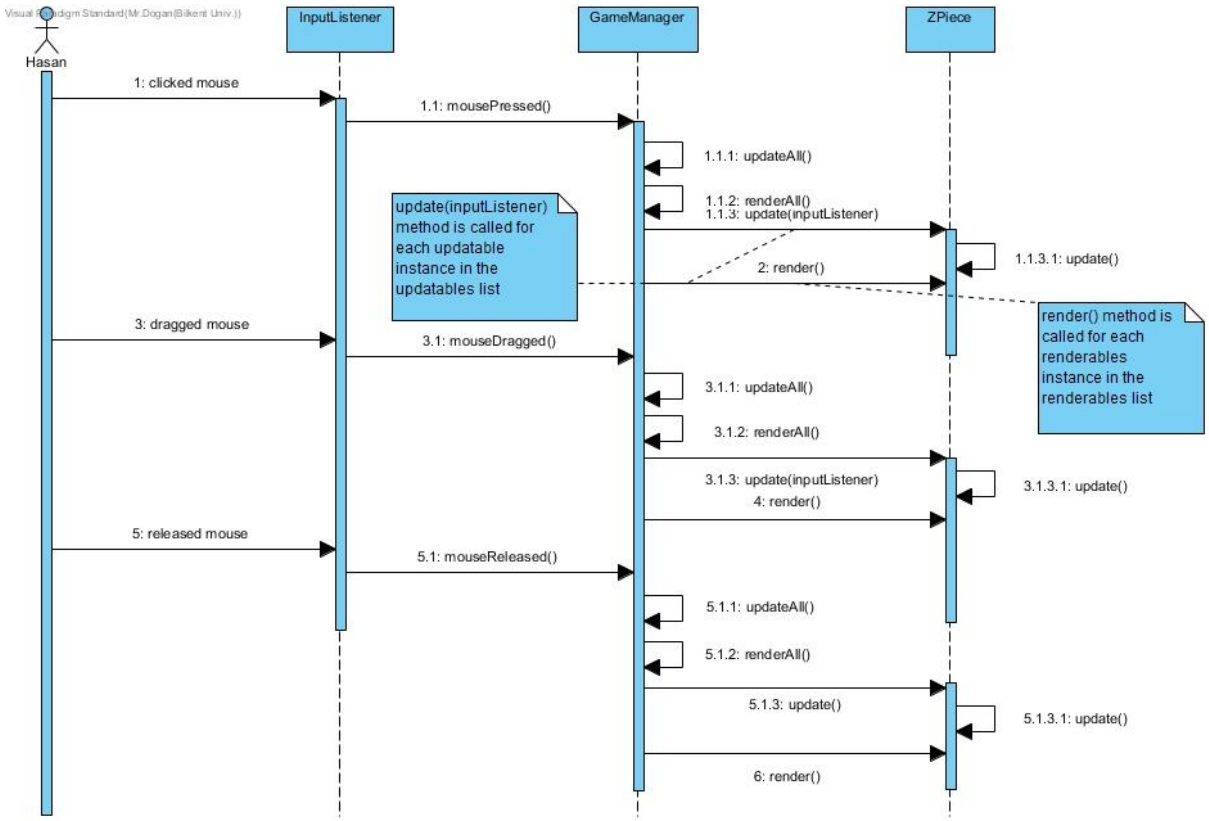
Mehmet, loves puzzle games and one day, he decides to play SuperKatamino. He runs the game and clicks play game. He starts with first level again. He finds the solution in his mind and starts to perform this solution on the game screen. He clicks Z shaped piece and drag it then drop on the table. He thinks his next move on the path of his solution.

5.2.1.2

Bilgenur returned from school to home and opened her computer. She decides to play SuperKatamino. She runs the game and clicks play game. She starts with first level again. She finds the solution in his mind and starts to perform this

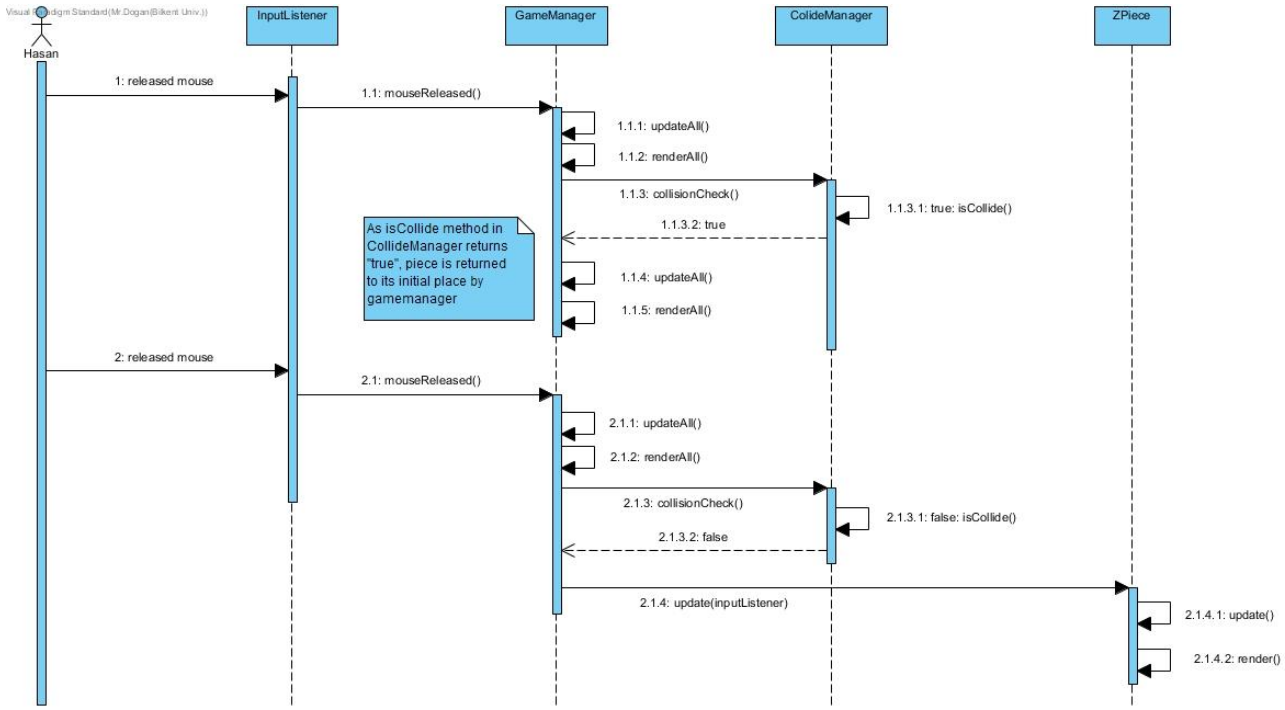
solution on the game screen. She clicks Z shaped piece and drag it then drop on the table. The piece was not approved because it collides with another shape. She stries again.

5.2.2 Sequence Diagrams



We need to indicate that in this sequence diagram, it is accepted that user behaves as expected.

In each flow, user gives an input to InputListener by clicking, dragging or releasing the mouse. InputListener transports it to GameManager by the methods accordingly. (mouseReleased() method when user releases and mouseDragged() method when user drags the mouse etc.) GameManager updates and renders all instances and gives update(inputListener) message to the object which is ZPiece in this situation and Zpiece updates itself. GameManager renders ZPiece last.



The sequence diagram above show what happens when user releases an object while dragging.

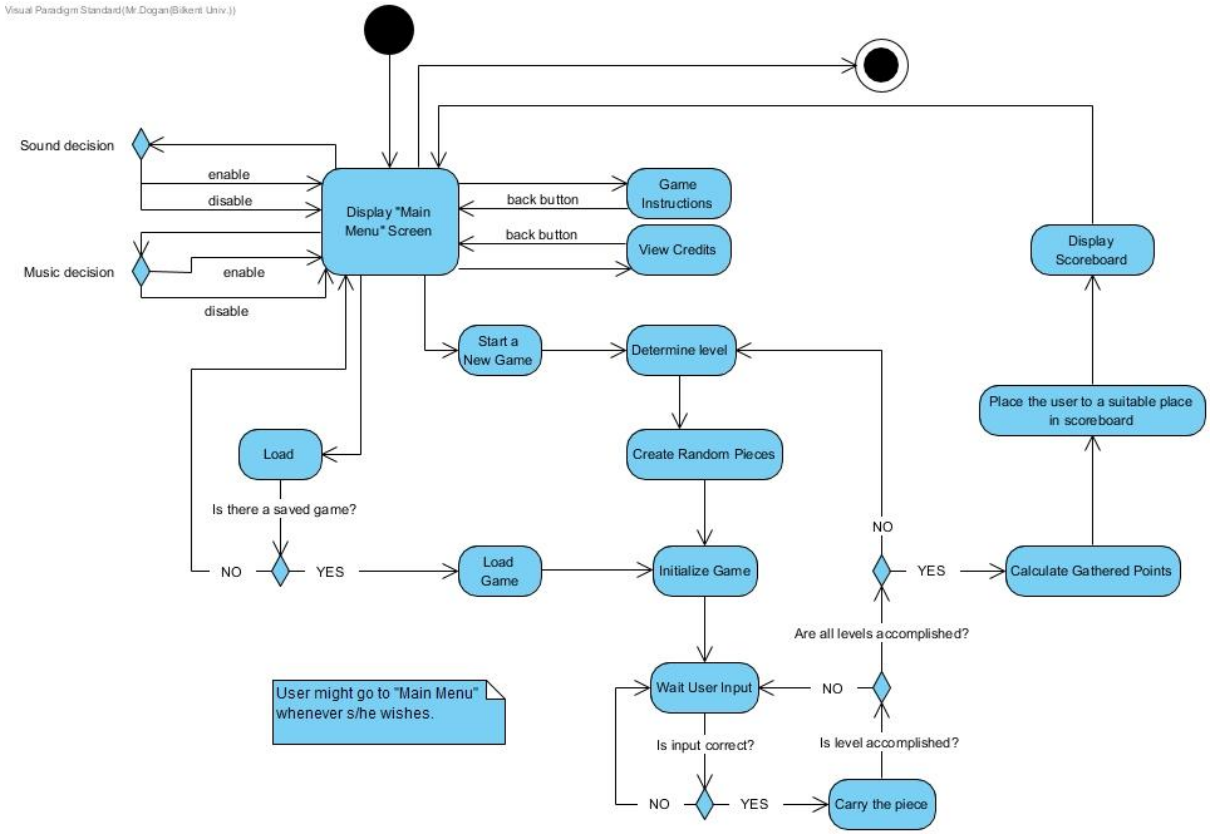
InputListener gets the coordinates and transports to GameManager, then GameManager after updating and rendering all instances leaves CollideManager to check if there is a collision or not. CollisionManager does it by using isCollide() method. When the output is true which means there is a collision, GameManager is notified and it updates and renders all instances and so the object(which is ZPiece in this situation) is returned to its original place. When the output of isCollide() method is false, GameManager again is notified and ZPiece is updated and rendered afterwards.

5.2.3 Activity Diagram

User encounters with main menu when .exe is executed. In main menu there are options to enable and disable in game sound and music. Also player can reach game instructions to learn how to play SuperKatamino. Credits also can be seen by user. If user wishes to start a new game, level is determined, which is 1st level in this case, and pieces that will fit on the board are created randomly and game initialized. If user wishes to load a game which is saved before, system continues from game initialization step and moves forward. After the initialization of the game, system awaits inputs from user and checks if the inputs are correct. If the input is wrong, system keeps waiting till an accurate input is provided. When a given input is correct, system checks if that was the last piece that fulfills the board. If it is not, level is not accomplished and some new inputs are waited from user. If the level

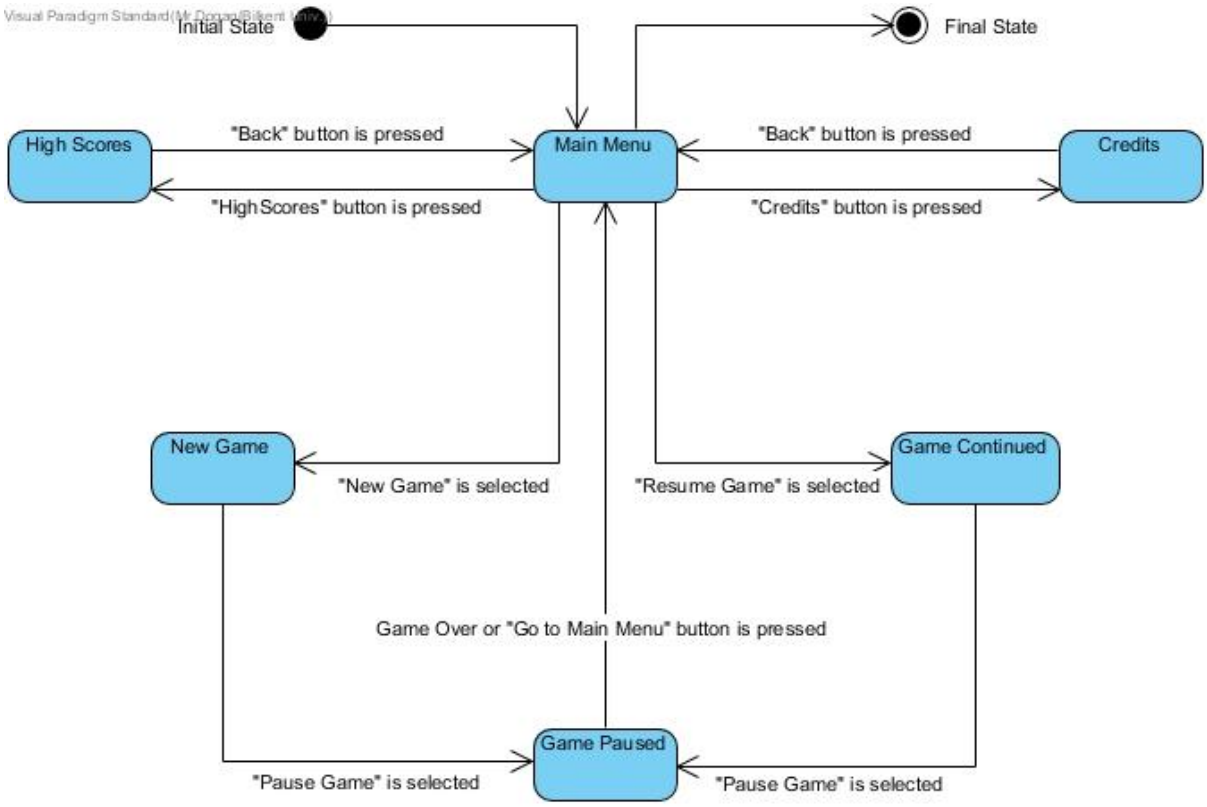
is accomplished and if it is not the last level, system moves on to the next level. If it was already the last level, then system calculates the total point that player gathered and puts in a suitable place on scoreboard. Then player is directed to main menu and is let act according to his/her will. S/he can exit or keep playing starting a new game or a saved one.

In SuperKatamino, exits are from main menu, and user might go main menu whenever s/he wishes.



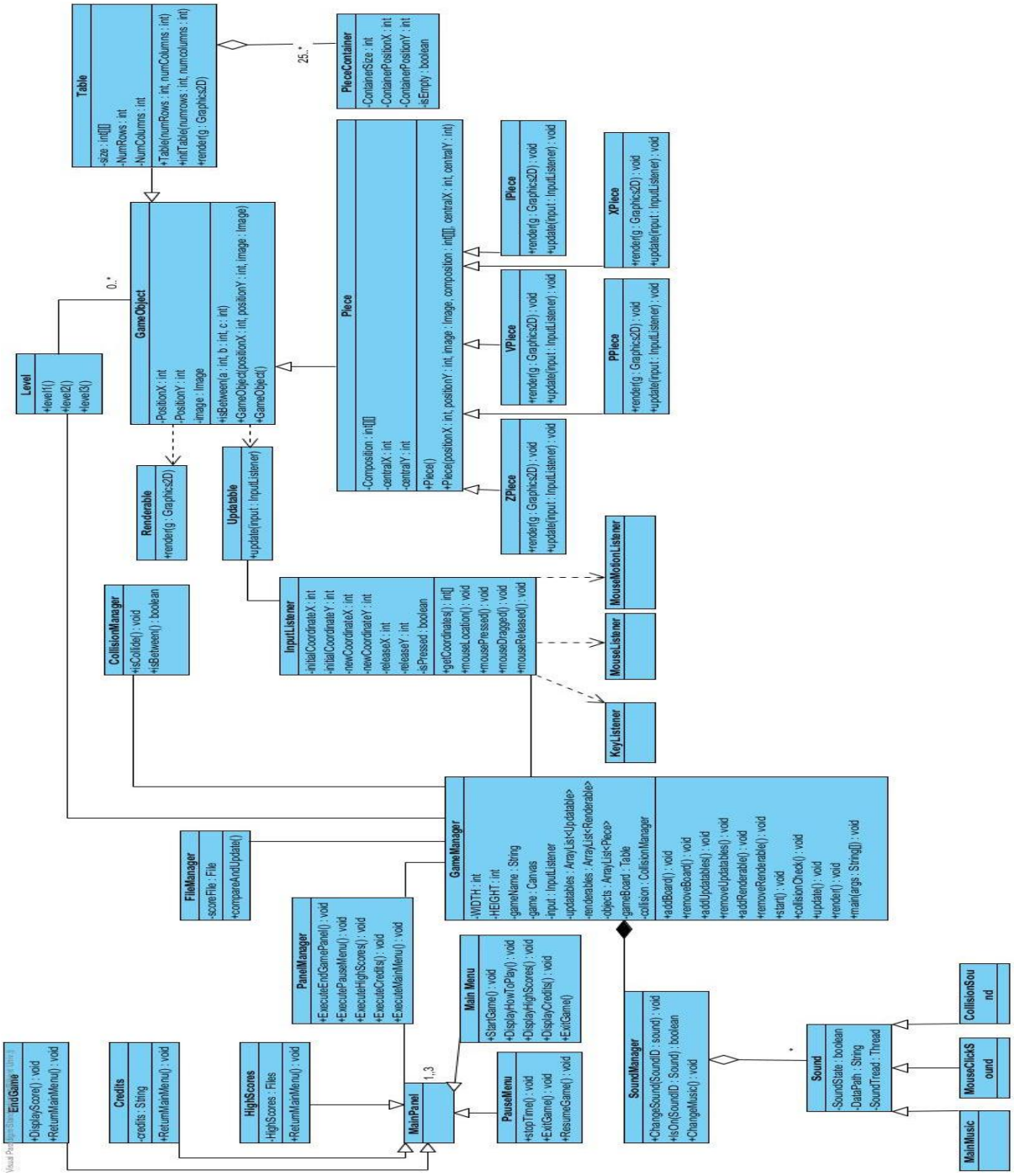
5.2.4 State Diagram

State diagram above displays the state of the game accordingly. Game might be in main menu state, then viewing high scores or credits states are also possible. From main menu state, game may continue to being in a new game state or a continued game state by resuming if game is paused earlier. The actions needed to pass between states are shown written near transitions.



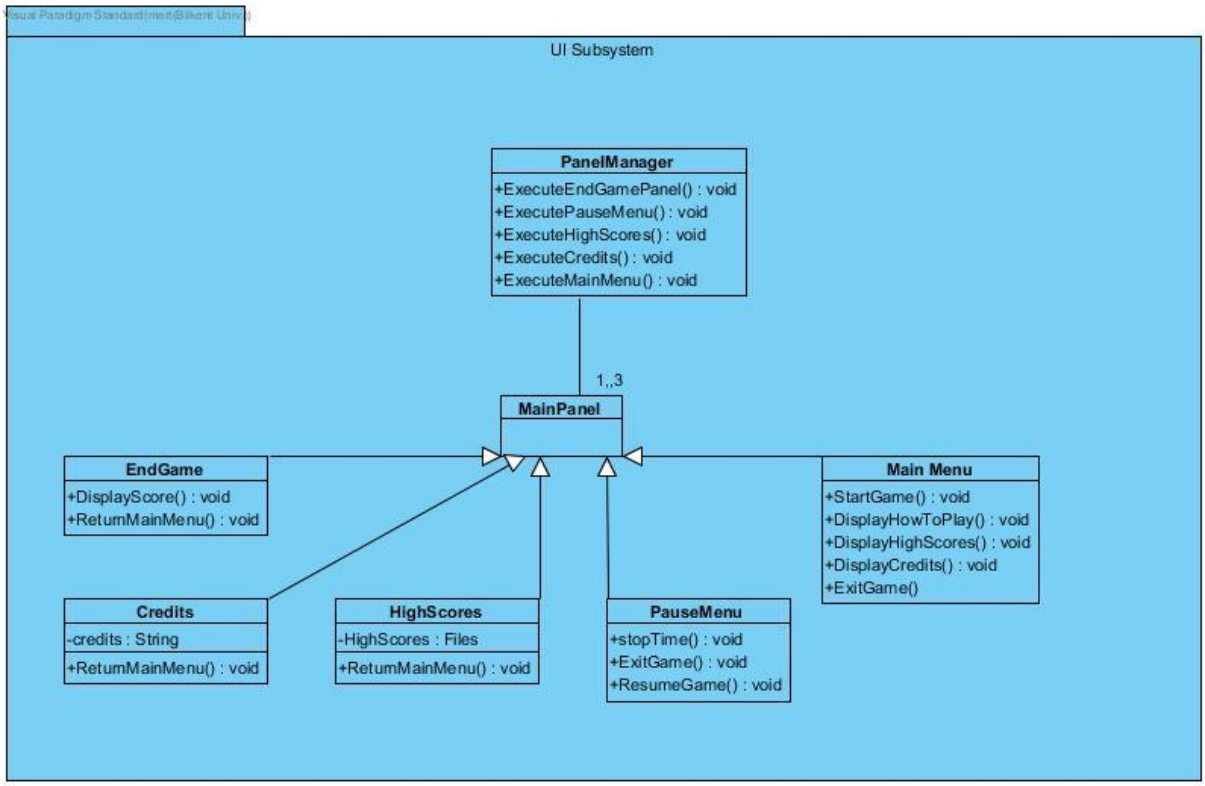
5.3. Object and class model

5.3.1 Class Model



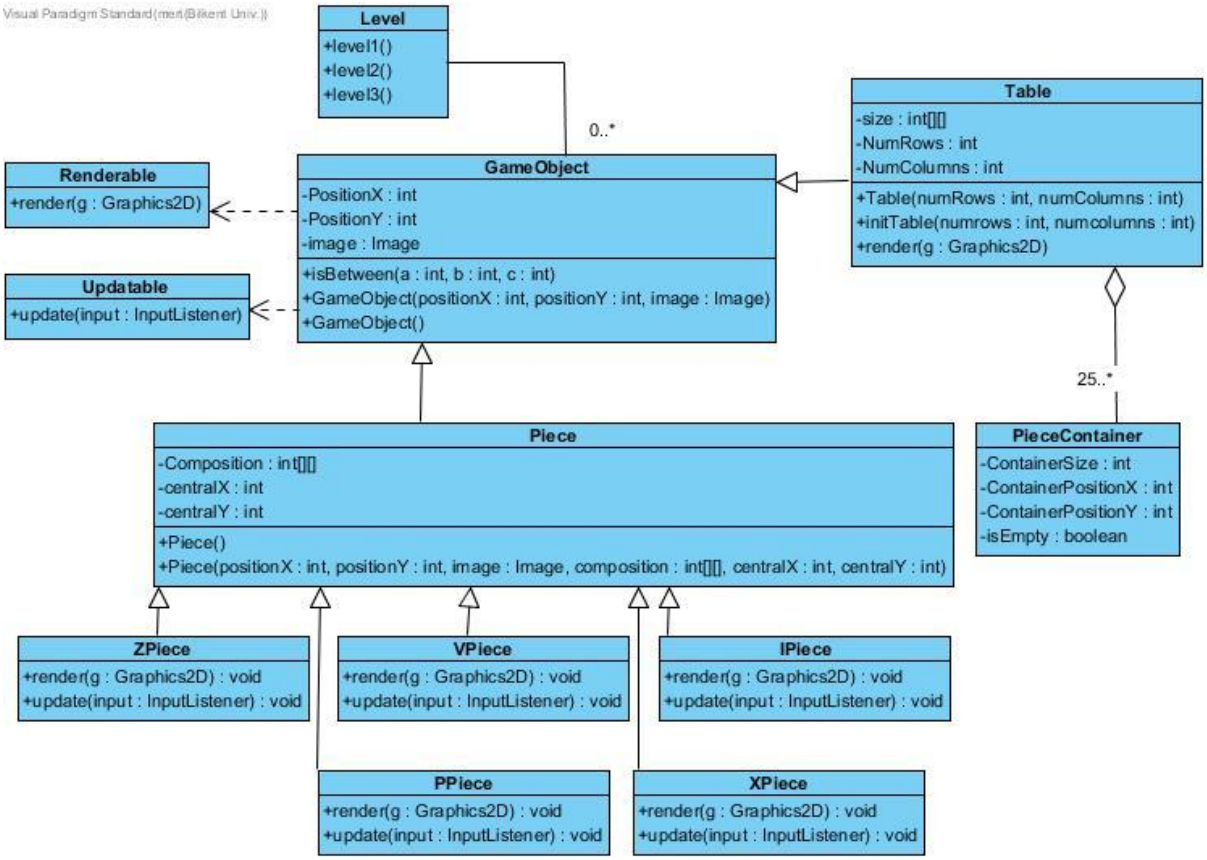
This is the whole object model. This creates and renders levels on the UI.

5.3.2 UI Class Model



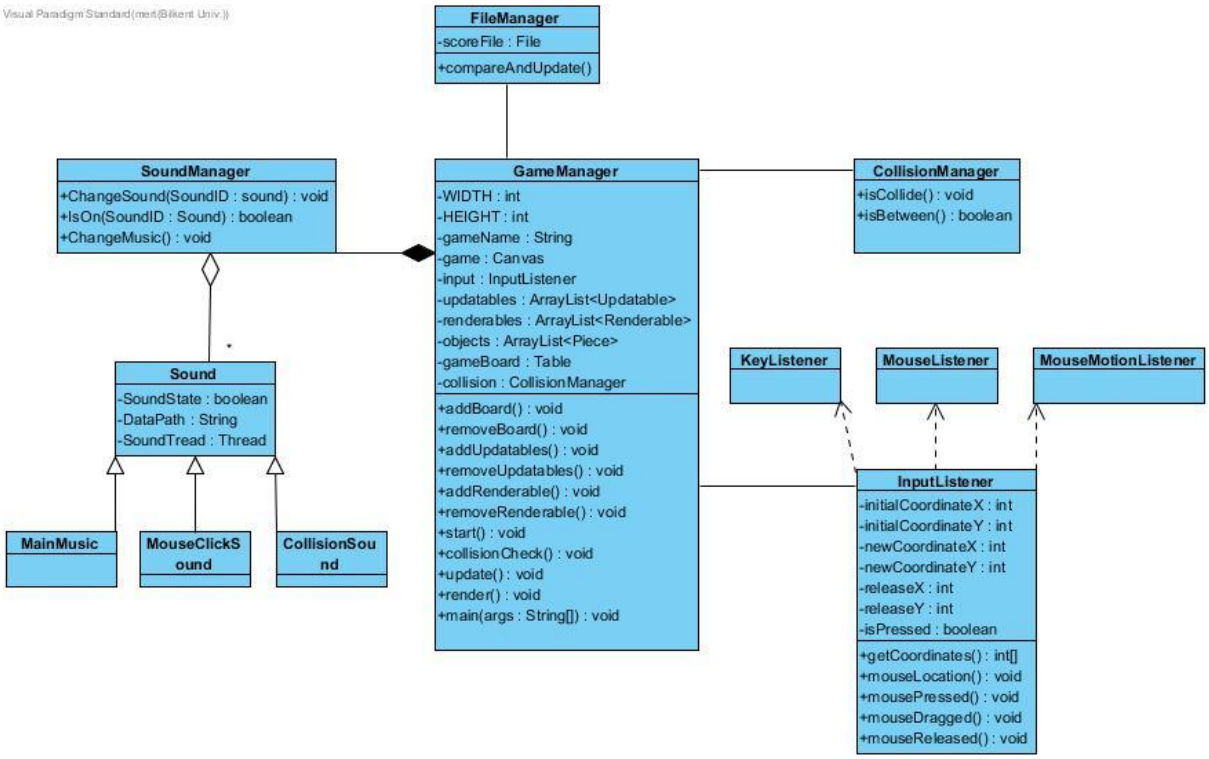
This UI Subsystems Creates canvases to add main JFrame in the GameManager class so main loop can draw the canvas to frame.

5.3.3 Object Class Model



GameObject class is the superclass for the table and pieces due to their common properties like position and image variables. GameObject class implements Updatable and Renderable interfaces to draw themselves on the canvas. These functions are called in the main function of the game in an ArrayList that checks all renderable and updatable Piece objects. The game can have 25 to many PieceContainer that are used in Table object in order to resolve collision of the table and a piece object. Pieces have Composition variable as 2D array to check its shape. They also have Central points kept in order to make it more updatable for future implementations.

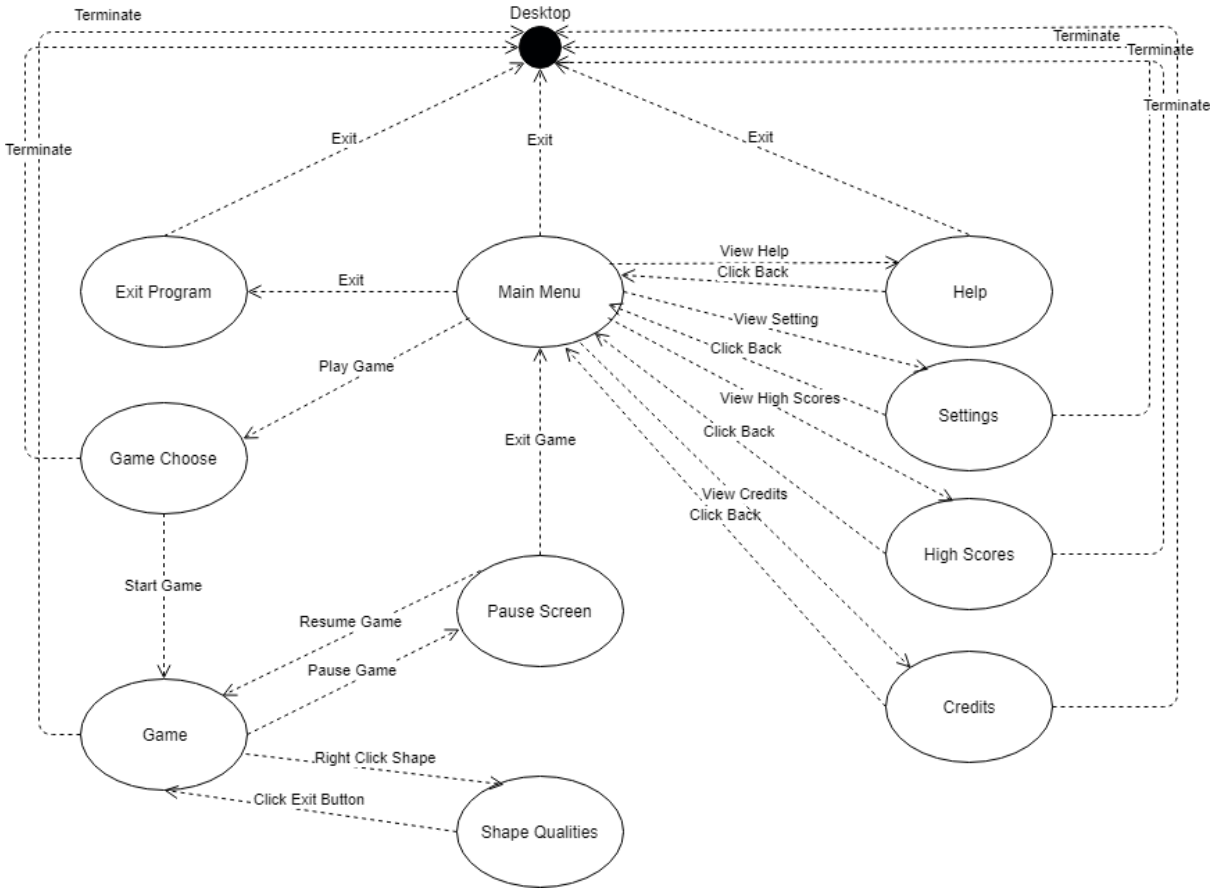
5.3.4 Engine Class Model



This is the core part of the UML class diagram. GameManager has the main function that runs the game in a loop until player gives an exit order to game. The game plays according to players actions that are taken by InputListener class's help. While playing sounds and music in the background with a second thread to main thread. Collisions between the table component and between pieces that are generated checked via CollisionManager class. At the end of the game score of the current game is saves on the file if the current score is higher than at least 10th person on the scoreboard.

5.4. User interface - navigational paths and screen mock-ups

5.4.1 Navigational Path



Navigational path diagram shows the path/relations between the screens. The program starts with Main Menu. Then when clicked play game, game choose screen appears. In this screen, game can be loaded or new game created. Then by clicking start game, game starts. Game screen appears. From game screen, when pause button clicked, Pause menu can be reached. From pause menu screen, user can exit the game or return the game after a while. On game screen, when clicked any shape, shape qualities screen appears in order to rotate shapes. From main menu, when clicked help, user can view help screen. When clicked credits, user can view Credits screen. When clicked settings, user can view Settings screen. When clicked high scores, user can view High Scores screen.

Figure 5.4.2.1: Main Menu

Main Menu is the welcome screen of the game. There are 6 buttons for starting the game, view help, credits, settings, high scores and exit the game. Also there are exit and wide screen buttons.

When clicked the play game button, there is another popup screen to load a game or start a new game.

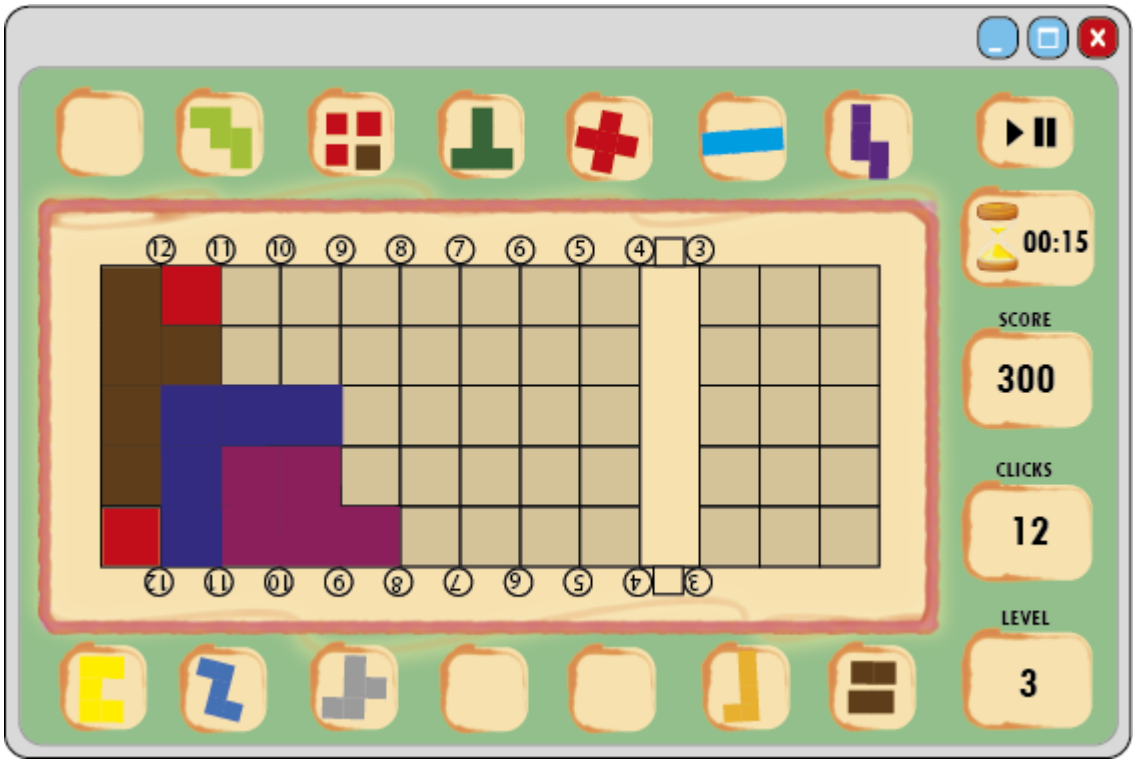


Figure 5.4.2.2: Game Screen

The game screen, shows the user the facilities. It contains table the game is played and shaped to drag. There some information about remained time, level, score and clicks. There is also play/pause button.

Figure 5.4.2.3: High Scores Screen

This screen shows high scores.

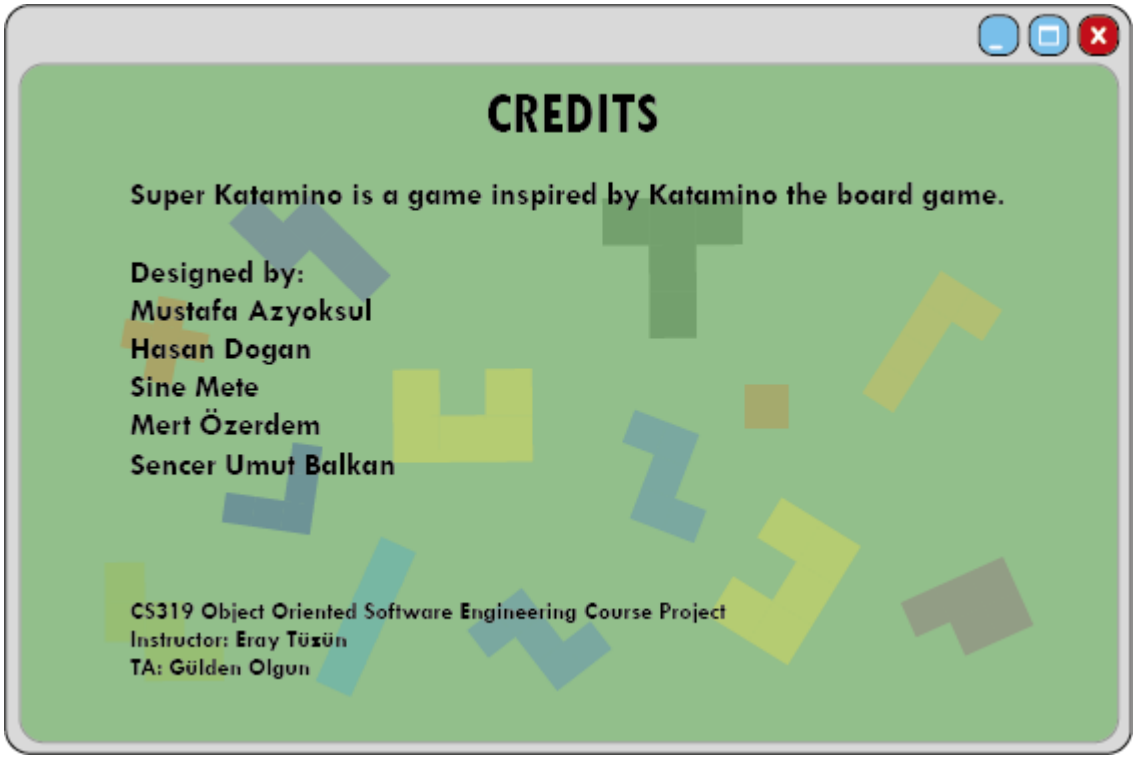


Figure 5.4.2.4: High Scores Screen

This screen shows credits.

Figure 5.4.2.5: Settings Screen

This screen shows settings for sound and music.

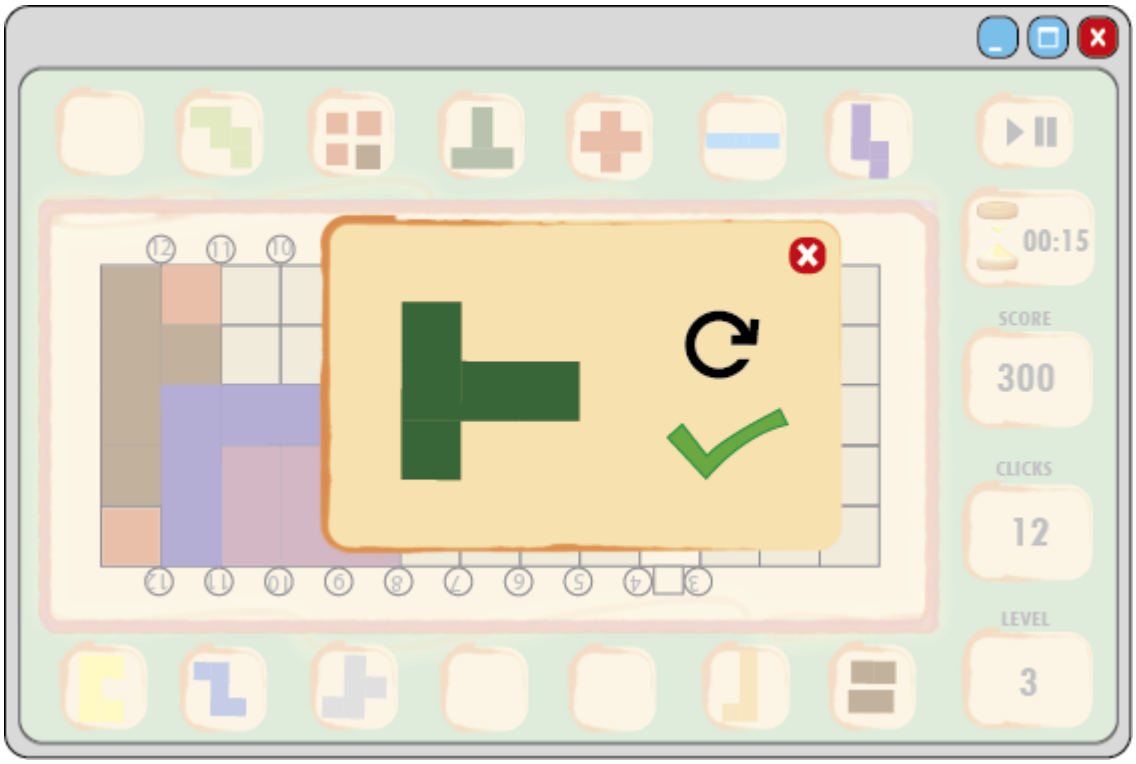


Figure 5.4.2.4: Shape Qualities Screen

When clicked on any shape, this screen appears in order to rotate the shape.

6. Improvement Summary

Throughout this process, we developed our point of view to project and analysis part. We tried to improve our perspective of diagrams and how to fit our project. We had to did major changes in Activity diagram and use case diagram in respect of feedbacks. The diagrams kind of wide and messy, we tried to make them more compact.

Our class diagram changed during implementation phase 1. We realized we need more classes than our first class diagram and we did wrong with some class relations. Our class diagram is more clear and suitable to implementation now.

To conclude, in every step, our opinions of project and diagrams are changing and developing, it shows that the process will continue like this until our project is finished.

7. Glossary & References

[1] https://boardgamegeek.com/boardgame/6931/katamino