Bilkent University

CS 319 Term Project

Analysis Report

Section 3

Team Name: JOKERS Team
Project Name: IQ Puzzler Pro
Project Group Members:

- Burak Erdem Varol
- Ravan Aliyev
- Subhan Ibrahimli
- Ismayil Mammadov
- Mahmud Sami Aydin
- Cihan Erkan

Table of Contents

1. Introduction

IQ Puzzler Pro is a board puzzle game, produced by Smart Games. The game is consist of 12 different pieces, game booklet and game board. Players can play the game with three different modes such as flat 2D, 3D and diagonal 2D. Game booklet provides 40 challenges with increasing difficulties for each game mode. According to Smart Games, IQ Puzzler Pro stimulates cognitive skills such as planning, concentration, logic and problem solving. Smart Games markets IQ Puzzler Pro as a game designed for everyone older than 6(Smart Games, nd).

Aim of this project is creating IQ Puzzler Pro game in digital environment. For this project, our number one priority is conserving unique gameplay of the game. Because of that, main goal of the game will be same as original game, placing pieces to correct places and filling every empty spot in the board. However, because of the minimalistic and simple design of the game, after a few hours of gameplay, the game starts to feel repetitive and boring. Therefore, our second priority is adding new features to game in order to keep players attracted to game, even after many hours of gameplay.

In order to achieve our second goal, we decided to add two main features to game, Level Creator and Multiplayer Mode. First of all, since the original game is designed as a physical game, it is impossible to change pieces. However, for our project, it is very easy to add new pieces and create new levels. Level Creator allows players to create their own custom designed pieces and levels in order to make the game less repetitive. Secondly, we are going to add a Multiplayer Mode to the game. We are going to design Multiplayer mode as a competitive turn based strategy game. Considering most widely known turn based strategy games such as chess and Civilization (invented in 6th century (Chesshere.com, nd) and released in 1991 (Wikipedia, nd)), we believe competitiveness is a key for the success of a game.

2. Overview

This section contains overview of GamePlay, Board, GameModes, LevelCreator, Multiplayer Mode, Settings.

2.1 Gameplay

IQ Puzzler Pro is a both 2 and 3-dimensional puzzle game. The game is originally designed to play on its board but we are going to create the game in a digital environment. There are two main elements of the game: Game board and pieces. Game board is consisting of spots that pieces can fit in. There are also 12 different pieces with different shapes. Aim of the player is placing those pieces in game board without leaving a single spot empty for 2-dimensional case. For 3-dimensional case aim of the player is building a pyramid. Moreover, user will be able to rotate and flip those pieces by clicking on them in order to place those pieces to board. In the beginning the game places some of the pieces to board and ask user to fill rest of the board.

2.2 Board

There are 3 different boards that the game can be played. 2 of them are 2 dimensional and one of them is 3 dimensional. General mechanics of both 2-dimensional boards are similar, only difference is the relative place of spots. However, 3-dimensional case is quite different. User uses a 5x5 square to build the pyramid.

2.3 Game Modes

There will be 2 game modes. Story mode and Custom Game mode. For story mode, user can play one of the classical challenges of IQ Puzzler Pro. For Custom Game mode, players can play one of their own custom designed levels or they can request the game to create a new game by using the original pieces of IQ Puzzler Pro.

2.4 Custom Level Creator

Users can create new levels and pieces with any of the game modes (2-D and 3-D). To do that user will be provided an empty board of their choice, and he/she will be able to fill it with his/her custom-made pieces. Those pieces could have any size and shape and it will be completely up to user. However, game will treat those piece as classic pieces, so during the gameplay, user can rotate and place those pieces to board.

2.5 Multiplayer Mode

2 Player Mode will allow players to compete against each other. 2 Player Mode will be a turn based strategy game based on the original IQ Puzzler Pro game. However, game mechanics will be slightly different. Since every valid IQ Puzzler Pro challenge has exactly one solution, there is exactly one correct place for every piece and Multiplayer mode will be based on this fact. During their turns players will try to figure out which piece belongs to which place and they will get a single chance to try their guesses. If player makes correct guess, he/she will gain points, otherwise he/she will lose some. Moreover, if player makes the correct choice, he/she will get another chance to place another piece, otherwise other player's turn will begin. At the end of the game player with more score will be declared as winner.

However, since we are trying to make this mode as a strategy game, not a luck game, our point scoring system will severely punish players who desires to find the correct place by trial-and-error method and it will reward, or punish more mercily, players who actually thinks before they play. We are going to achieve this by calculating scores by taking into account parameters such as time spent, number of trials with certain pieces, ratio of size of piece and number of remaining empty spaces etc.

2.6 Settings

Player should be able to adjust volume of the in-game music and change language of the game. Moreover, player should be able to change keyboard buttons according to his/her desires.

3. Functional Requirements

This section contains information of functional requirements of IQ Puzzler Pro Project.

3.1 Single Player

Players must be able to play classical IQ Puzzler Pro levels and custom made levels themselves. All game modes, 2D and 3D, must be playable. Players should be able to rotate, flip and move all non-fixed pieces. User should be notified when game ends and a new achievement earned.

3.2 Multiplayer Mode

For 2-Player Mode, players should be notified when a new player's turn begin. After one player finishes his/her turn, game should check correctness of move and calculate score. After game finishes, winner should be declared.

3.3 Level Creator

Players and content creators should be able to create new levels for all game modes. All created levels must follow conventions of original IQ Puzzler Pro, i.e. there must be only one solution and pieces should be composed of adjacent parts. Otherwise, user must be warned and level should not be saved.

3.4 Settings

Player should be able to adjust volume of the in-game music and change language of the game. Moreover, player should be able to change keyboard buttons according to his/her desires.

3.5 Additional Functional Requirements

New Functional requirements introduced in iteration 2 of IQ Puzzler Pro Project.

3.5.1 Random Level Generator

Player can generate random level by using level creator extansion.

4. Non-functional Requirements

This section lists non-functional requirement of the game and contains their breif explanation

4.1 Usability

95% of users should be able to adapt the game within 10 minutes. Colors displayed in the color blind mode, must be distinguishable by color blind users. Anyone who can use a computer should be able to play the game. The game should have language support for English, Turkish, Azerbaijan and Russian.

4.2 Supportability

The game should be platform independent, i.e. it should be available for users of MacOs, Windows and Linux systems in which JRE 8 is supported. Contact information of developers should be provided to users, in order to allow developers to fix newly discovered bugs.

4.3 Game Performance

The game should require less available space than 500mb, and it should have a response time less than 100ms.

4.4 Extendibility

At the end of the project, the project-related documents should be open-source and they will be provided to other developers, so they will be allowed to make changes as desired style and recreate their own game.

5. System Models

This section explains behaviour and structure of the game by using use case, sequence, activity, state and class diagrams.

5.1 Use Case Diagram

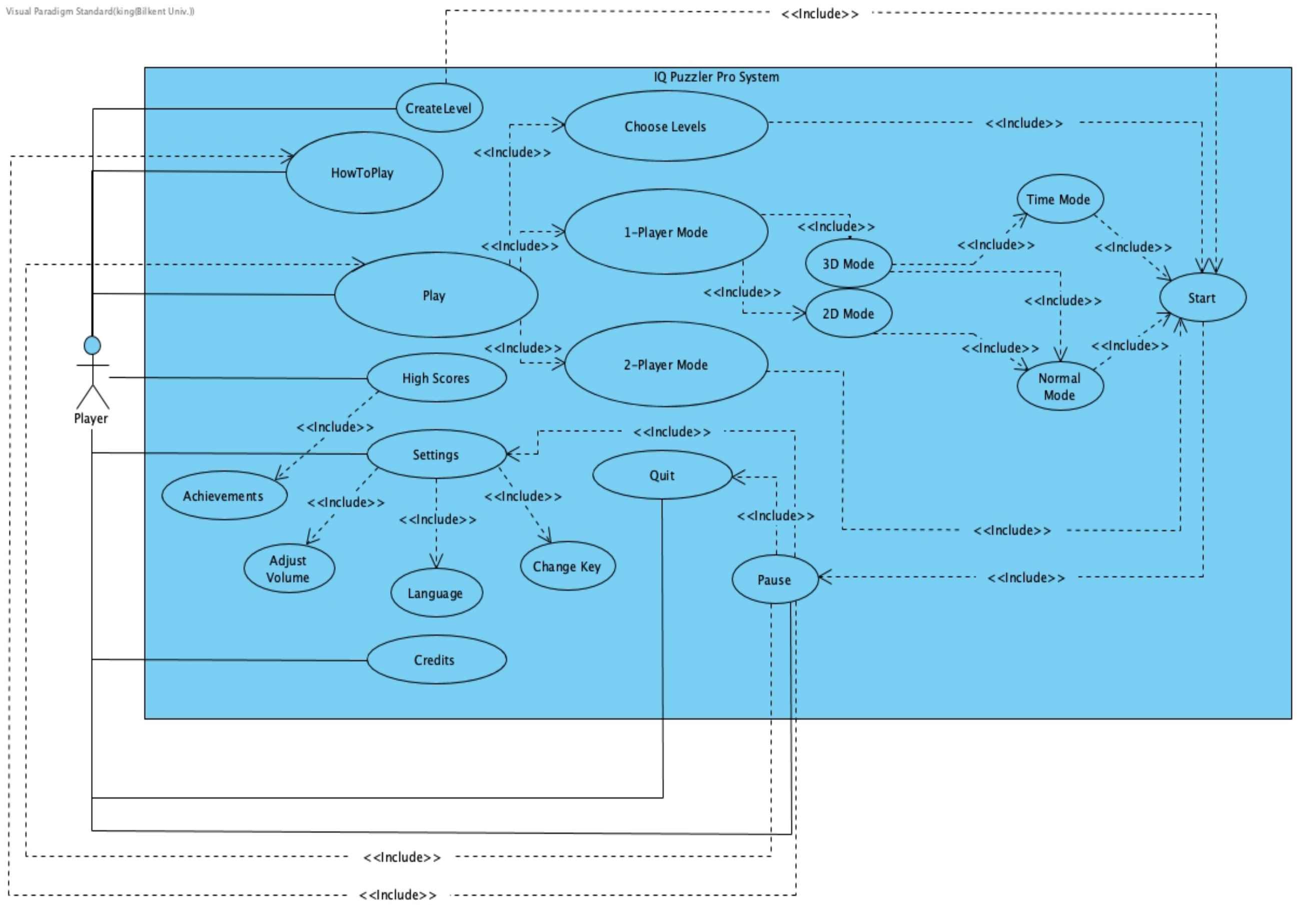


Figure 1 : Use Case Diagram

Use Case diagram of the game.

5.1.1 Play Single Player 2D Time Mode

Use Case: Play Single Player 2D Time Mode

Primary Actor: Player

Stakeholders and Interests:

- Player selects play
- Player selects Single Player Mode
- Player selects 2D Mode
- Player selects Time Mode
- System creates the game table
- System starts the game.

Pre-conditions:

- Player must be in the main menu.

Post-conditions:

Entry-conditions:

- Player should choose the Single Player Mode
- Player should choose the 2D Mode
- Player should choose the Time Mode

Exit conditions:

- Player selects “Quit” button from the Pause Menu.
- Player successfully finished the game by completing level in a given amount of time.

Success Scenario Event Flow:

1. Player chooses “Single Player Mode”.
2. Player chooses "2D Mode".
3. Player chooses "Time Mode".
4. System starts game and timer.
5. Player selects the piece.

6. Player rotates or flips the piece.
7. Player drags the piece on the game board.
8. Player drops the piece on the board.
9. System decides the piece location correct or incorrect.
10. System updates the view of game board.
11. Player tries to fill all spaces on the game board.
12. Player fills the game board with given pieces in a given amount of time.
13. Player wins the game.
14. System finishes the level and initializes the next level.

Alternative Event Flows:

1. If Player wants to return main menu:
    a. Player pauses the game by pressing Pause Button from game screen.
    b. System displays Pause Menu
    c. Player selects "Quit"
    d. System asks confirmation
    e. Player selects "Yes".
    f. System exits the game.
    g. System show the Main Menu.

5.1.2 Play Single Player 2D Normal Mode

Use Case: Play Single Player 2D Normal Mode

Primary Actor: Player

Stakeholders and Interests:

- Player selects play
- Player selects Single Player Mode
- Player selects 2D Mode
- Player selects Normal Mode
- System creates the game table
- System starts the game.

Pre-conditions:

- Player must be in the main menu.

Post-conditions:

Entry-conditions:

- Player should choose the Single Player Mode
- Player should choose the 2D Mode
- Player should choose the Normal Mode

Exit conditions:

- Player selects “Quit” button from the Pause Menu.
- Player successfully finished the game by completing level.

Success Scenario Event Flow:

1. Player chooses “Single Player Mode”.
2. Player chooses "2D Mode".

3. Player chooses "Normal Mode".
4. System starts game
5. Player selects the piece.
6. Player rotates or flips the piece.
7. Player drags the piece on the game board.
8. Player drops the piece on the board.
9. System decides the piece location correct or incorrect.
10. System updates the view of game board.
11. Player tries to fill all spaces on the game board.
12. Player fills the game board with given pieces.
13. Player wins the game.
14. System finishes the level and initializes the next level.

Alternative Event Flows:

1. If Player wants to return main menu:

    a) Player pauses the game by pressing Pause Button from game screen.
    b) System displays Pause Menu
    c) Player selects "Quit"
    d) System asks confirmation
    e) Player selects "Yes".
    f) System exits the game.
    g) System show the Main Menu.

5.1.3 Play Single Player 3D Time Mode

Use Case: Play Single Player 3D Time Mode

Primary Actor: Player

Stakeholders and Interests:

- Player selects play
- Player selects Single Player Mode
- Player selects 3D Mode
- Player selects Time Mode
- System creates the game table
- System starts the game.

Pre-conditions:

- Player must be in the main menu.

Post-conditions:

Entry-conditions:

- Player should choose the Single Player Mode
- Player should choose the 3D Mode
- Player should choose the Time Mode

Exit conditions:

- Player selects "Quit" button from the Pause Menu.
- Player successfully finished the game by completing level.

Success Scenario Event Flow:

1. Player chooses "Single Player Mode".
2. Player chooses "3D Mode".

3. Player chooses "Time Mode".
4. System starts game
5. Player selects the piece.
6. Player rotates or flips the piece.
7. Player drags the piece on the game board.
8. Player drops the piece on the board.
9. System decides the piece location correct or incorrect.
10. System updates the view of game board.
11. Player fills the given spaces and has to make a pyramid on the game board in a given amount of time.
12. Player wins the game.
13. System finishes the level and initializes the next level.

Alternative Event Flows:

1. If Player wants to return main menu:
    a. Player pauses the game by pressing Pause Button from game screen.
    b. System displays Pause Menu
    c. Player selects "Quit"
    d. System asks confirmation
    e. Player selects "Yes".
    f. System exits the game.
    g. System show the Main Menu.

5.1.4 Play Single Player 3D Normal Mode

Use Case: Play Single Player 3D Normal Mode

Primary Actor: Player

Stakeholders and Interests:

- Player selects play
- Player selects Single Player Mode
- Player selects 3D Mode
- Player selects Normal Mode
- System creates the game table
- System starts the game.

Pre-conditions:

- Player must be in the main menu.

Post-conditions:

Entry-conditions:

- Player should choose the Single Player Mode
- Player should choose the 3D Mode
- Player should choose the Normal Mode

Exit conditions:

- Player selects “Quit” button from the Pause Menu.
- Player successfully finished the game by completing level.

Success Scenario Event Flow:

1. Player chooses “Single Player Mode”.
2. Player chooses "3D Mode".
3. Player chooses "Normal Mode".
4. System starts game

5. Player selects the piece.
6. Player rotates or flips the piece.
7. Player drags the piece on the game board.
8. Player drops the piece on the board.
9. System decides the piece location correct or incorrect.
10. System updates the view of game board.
11. Player fills the given spaces and has to make a pyramid on the game board.
12. Player wins the game.
13. System finishes the level and initializes the next level.

Alternative Event Flows:

1. If Player wants to return main menu:
    a. Player pauses the game by pressing Pause Button from game screen.
    b. System displays Pause Menu
    c. Player selects "Quit"
    d. System asks confirmation
    e. Player selects "Yes".
    f. System exits the game.
    g. System show the Main Menu.

5.1.5 Play Multiplayer Mode

Use Case: Play Multiplayer Mode

Primary Actor: Player

Stakeholders and Interests:

- Player selects play

- Player selects Multiplayer Mode
- System creates the game table
- System starts the game.

Pre-conditions:

- Player must be in the main menu.

Post-conditions:

Entry-conditions:

- Player should choose the Multiplayer Mode

Exit conditions:

- Player selects “Quit” button from the Pause Menu.
- One Player successfully finished the game by completing level.

Success Scenario Event Flow:

1. Player chooses “Multiplayer Mode”.
2. System starts game
3. Player1 selects the piece.
4. Player1 rotates or flips the piece.
5. Player1 drags the piece on the game board.
6. Player1 drops the piece on the board.
7. System decides the piece location correct or incorrect.
8. System updates the view of game board.
9. If Player1 decision corrects in a given amount of time, Player1 will continue.
10. System updates point of Player1.
11. System updates a given amount of time.

12. If Player1 decision incorrects and time exceeded, Player2 will continue.
13. System updates a given amount of time.
14. If game board fills by Player1 or Player2, system decides which player wins the game according to players points.
15. System finished the game.
16. System asks to user rematch or back to the main menu.

Alternative Event Flows:

1. If Player wants to return main menu:
    a. Player pauses the game by pressing Pause Button from game screen.
    b. System displays Pause Menu
    c. Player selects "Quit"
    d. System asks confirmation
    e. Player selects "Yes".
    f. System exits the game.
    g. System show the Main Menu.

5.1.6 Play Choose Levels

Use Case: Play Choose Levels

Primary Actor: Player

Stakeholders and Interests:

- Player selects play
- Player selects Choose Levels
- Player selects a level
- System creates the game table

- System starts the game.

Pre-conditions:

- Player must be in the main menu.

Post-conditions:

Entry-conditions:

- Player should choose the Choose Levels

Exit conditions:

- Player selects “Quit” button from the Pause Menu.

Success Scenario Event Flow:

1. Player chooses “Choose Levels”.
2. System starts game
3. Player selects the piece.
4. Player rotates or flips the piece.
5. Player drags the piece on the game board.
6. Player drops the piece on the board.
7. System decides the piece location correct or incorrect.
8. System updates the view of game board.
9. Player fills the game board with given pieces.
10. System finished the game.
11. System goes back to the main menu.

Alternative Event Flows:

1. If Player wants to return main menu:
   a. Player pauses the game by pressing Pause Button from game screen.

b. System displays Pause Menu

c. Player selects "Quit"

d. System asks confirmation

e. Player selects "Yes".

f. System exits the game.

g. System show the Main Menu.

5.1.7 Change Settings

Use Case: changeSettings

Primary Actor: Player

Stakeholders and Interests:

- Player selects Settings
- System shows the Settings.

Pre-conditions:

- Player must be in the Main menu or Pause menu.

Post-conditions:

Entry-conditions:

- Player should choose the Settings.

Exit conditions:

- Player selects “Back” button from the Settings screen.

Success Scenario Event Flow:

1. Player chooses “Settings”.
2. System shows the Settings.
3. Player adjusts a volume or

4. Player changes language or
5. Player changes key combinations.
6. Player clicks the back button on the Settings screen.
7. System goes back to the main menu.

Alternative Event Flows:

1. If Player wants to return main menu:
    a. Player selects "Back"
    b. System exits Settings screen.
    c. System show the Main Menu.
2. If Player wants to return a game:
    a. Player selects "Back"
    b. System exits Settings screen.
    c. Player returns Pause Menu.
    d. Player selects "Back"
    e. System exits Pause Menu.
    f. Player returns a game.
3. If Player wants to change volume:
    a. Player selects Adjust Volume
    b. Player decides Volume Level.
4. If Player wants to change language:
    a. Player selects Language
    b. Player changes Language
    c. System updates the game language.

5. If Player wants to change key:
    a. Player can selects every different key
    b. Player changes the button selected by Player
    c. System updates the game play keys.

5.1.8 Create Level

Use Case: createLevel

Primary Actor: Player

Stakeholders and Interests:

- Player selects Create Level
- Player selects Flat/Diagonal/3D
- Player selects color for every pieces
- Player selects which piece will be removed on the game board.

Pre-conditions:

- Player must select Create Level in the main menu.
- Player must select Flat/Diagonal/3D.
- Player must select at least one piece remove from the game board.

Post-conditions:

Entry-conditions:

- Player should choose the Create Level.
- Player should choose "Done"

Exit conditions:

- Player should choose the "Back"

Success Scenario Event Flow:

1. Player chooses “Create Level” in the main menu.
2. Player chooses "Done" in the create level screen.
3. System creates and saved level.

Alternative Event Flows:

1. If Player wants to return to main menu:
    a. Player chooses create level in the main menu
    b. Player chooses back in the create level screen.
    c. Player return to the main menu.

5.2 Dynamic Models

5.2.1 Sequence Diagrams

5.2.1.1 Play Game

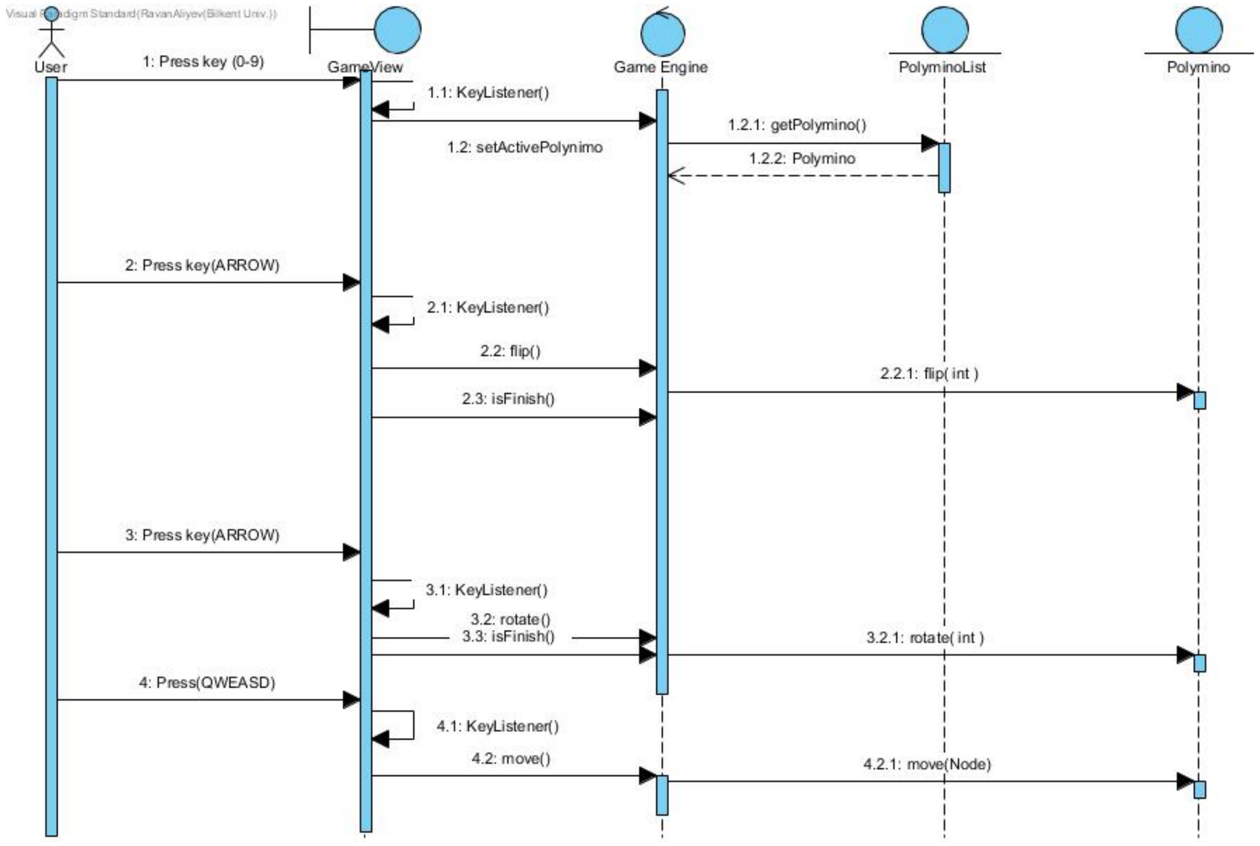


Figure 2 : Start Game Sequence Diagram

When game starts, user can switch on one of the available pieces using tab key to activate them by using setActivePolymino() function and user also allowed to rotate selected piece by using keyboard. After that, user use keyboard's "qasdew" keys the piece to the desired location by using move() function of Game Engine. Then move() function calls move() function of the Game Engine and if move is allowed Game Engine move the piece to desired location with move()

function. Game View draws the game. After that, Game Engine checks if the game is finished and if it’s not, allows user to make another play. In flips and rotate functions same procedures repeates.

5.2.1.2 Level Creates

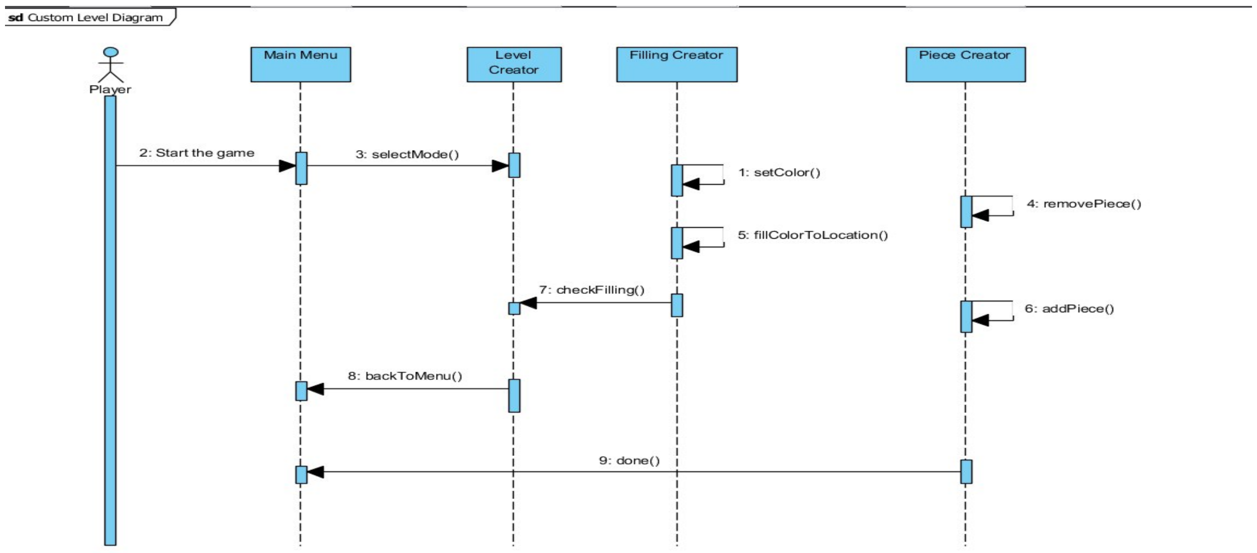


Figure 3 : Custom Level Sequence Diagram

To make a custom level, it first selects the desirable mode by player. After the mode selection, the player is presented with the board interface with empty holes and the colours to fill certain holes by the given buttons.

After setting colour and filling certain, the player clicks the check button to see if the conditions are satisfied according to the game rules.

Another presented option is to remove filled pieces by remove button. If the player decides to cancel the level creation, there is a 'back to menu' on the bottom of the screen available

5.2.2 Activity Diagram

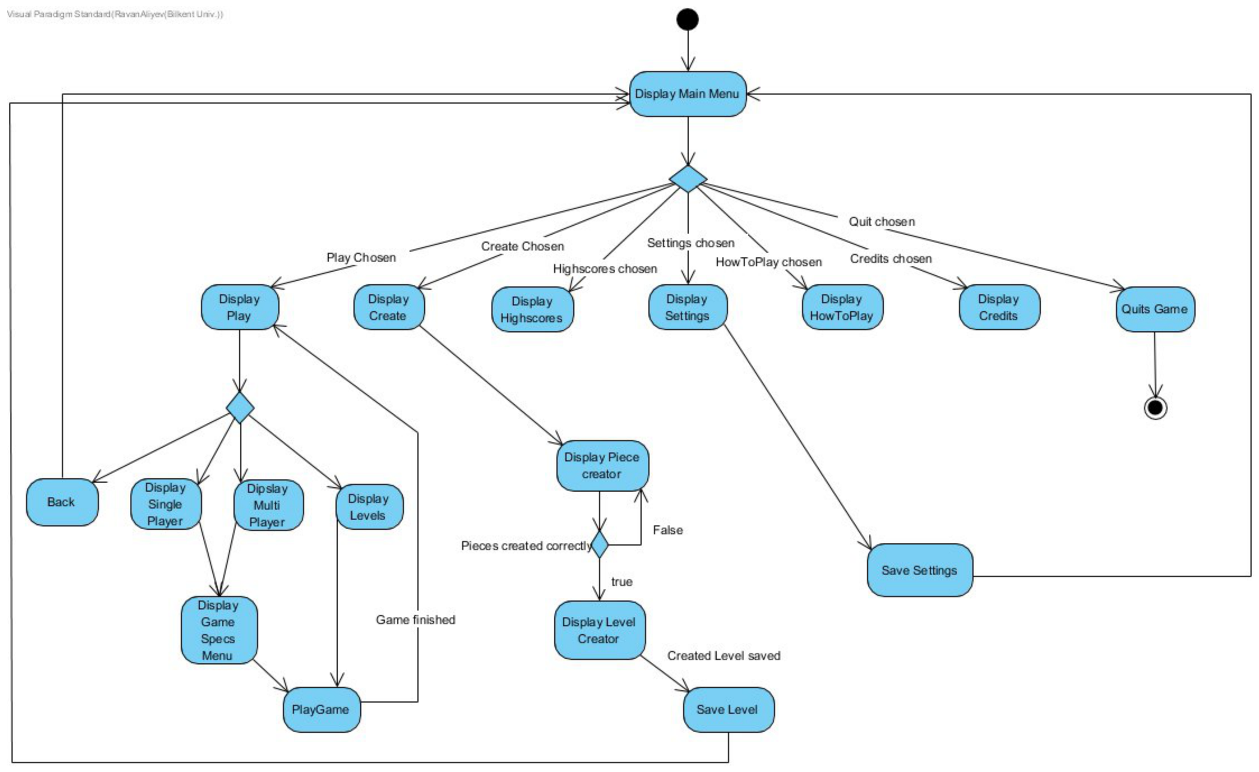


Figure 4 : Activity Diagram

The activity diagram demonstrates the step by step actions and how the system runs the game. While starting the game, it brings the player to choose options in Display Main Menu in which there are 7 actions such as Display Play, Display Create, Display High Scores, Display Settings, Display How to play, Display Credits and Quit Game. There are some sub-activities according to the user's choice in Display Main Menu. In Display Play activity there are 4 sub-activities : Back, Display Single Player, Display Multi Player, Display Levels. There are 2 main activities appeared in order to play the game, either Single or Multi player.

In general, player choose different mode in game, to change the language of the game, to adjust volume and to change key of the keyboard. When the player choose to play, 3 types of game options are given such as choosing available levels, or the player mode in which either 1 or 2 player can play at the same to compete themselves. In single player mode, there are also other modes such as 2D mode or 3D mode which refers to game mode and difficulty level of the game. In addition there is also Time mode option served for the player in the case they want to challenge themselves. There are also other options in Settings part. The player will be able to adjust the volume of the game sound, change the language and alternate the keys for the play.

5.2.3 Multiplayer Game Mode State Diagram

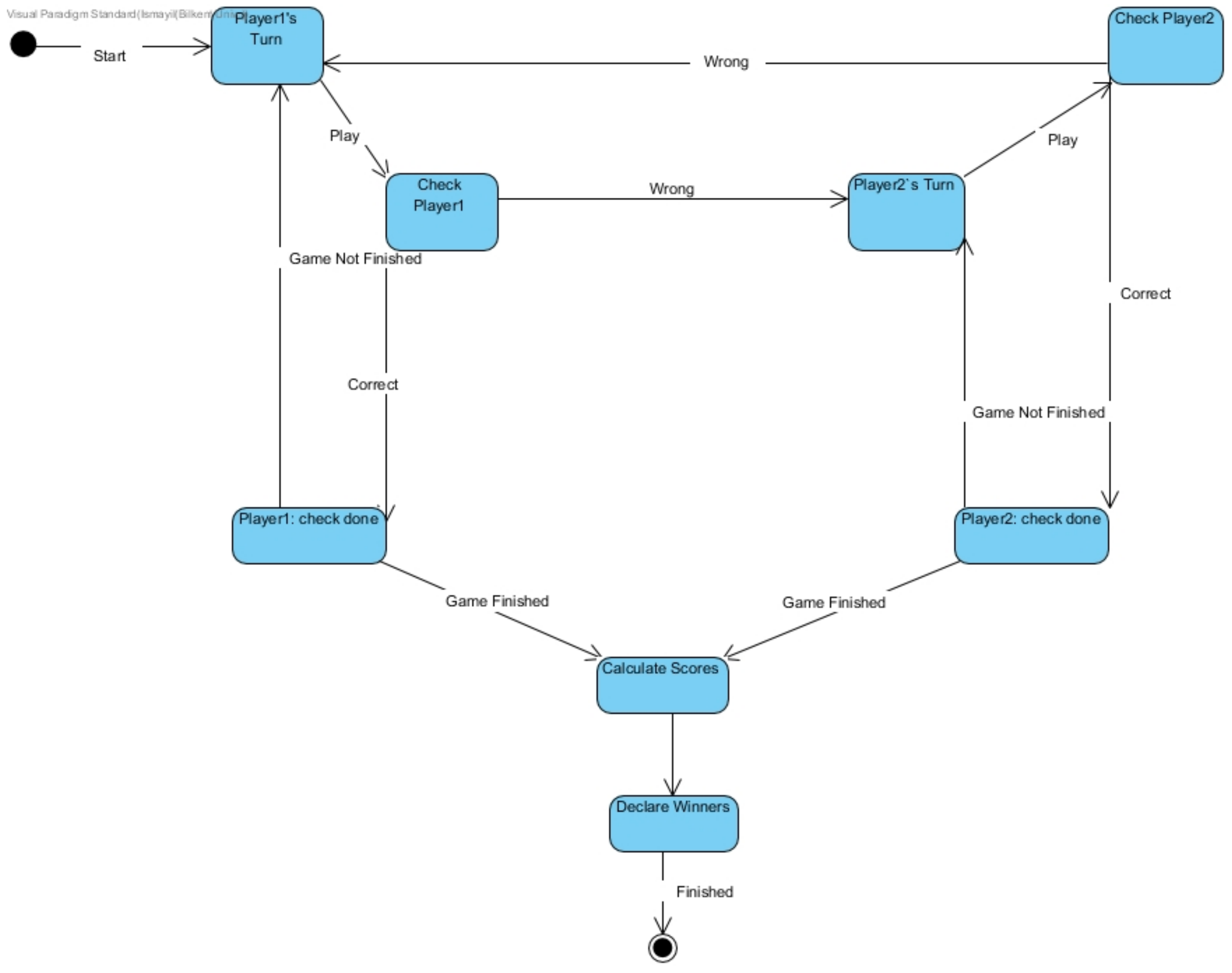


Figure 5 : Multiplayer Game Mode State Diagram

Game starts with Player 1's turn. Player 1 chooses a piece and places it into the board. Then the game checks whether it is correct or not, correctness is predefined by unique solution of the game. If it is correct, it goes state of "Check Done" which checks if the level is completely finished. If the game is finished, calculating state starts and game declares the winner and game ends.

If game is not finished and player does correct move, game provides another turn for the player. Otherwise game state changes to the other players turn. This loop continues until the game board is fully filled.

5.3 Object and Class Model

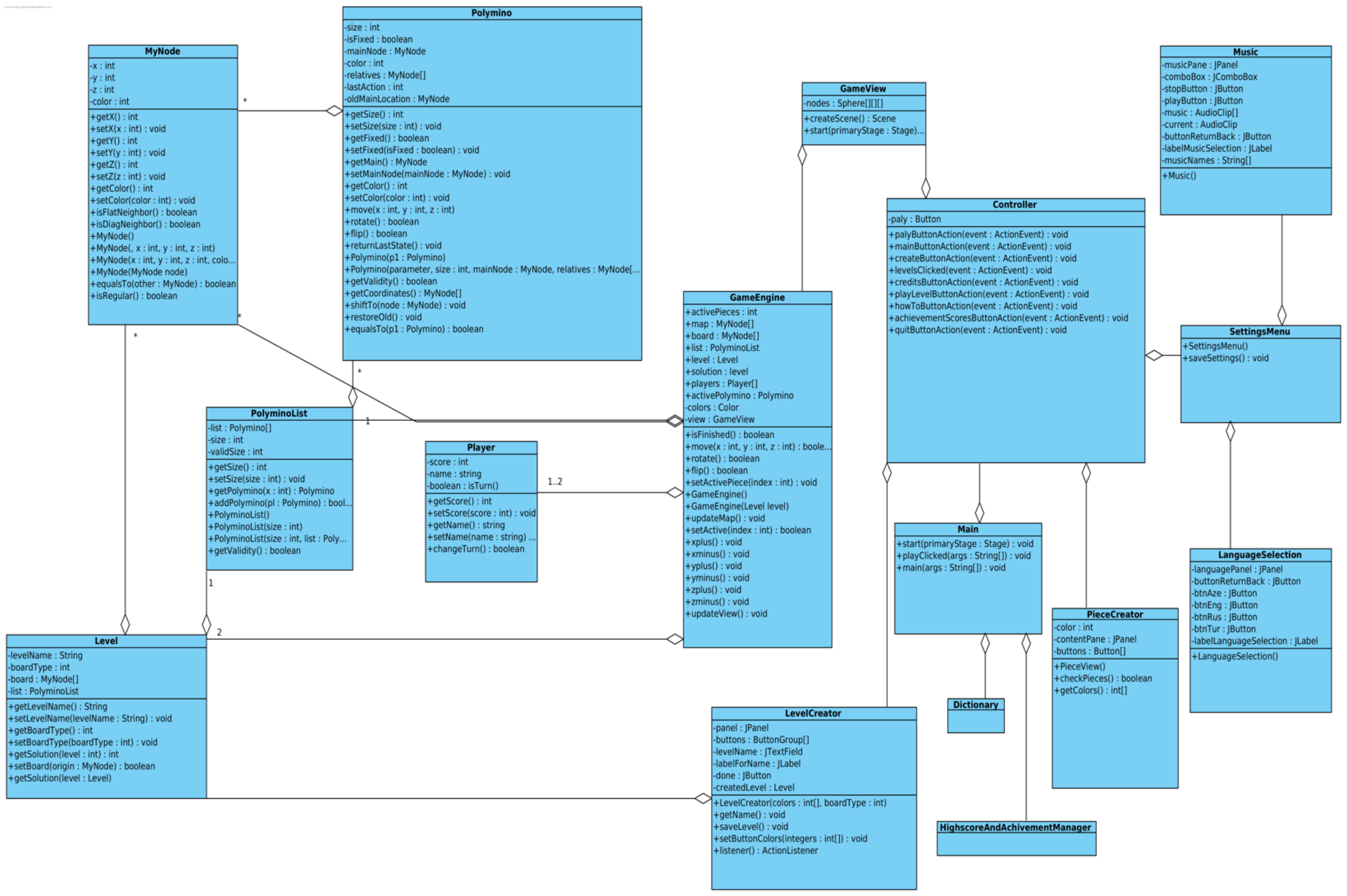


Figure 6 : Object And Class Model

There are 5 package which are named Pieces, Control, View, Level, SettingAndUtilities.

5.3.1 Pieces Package

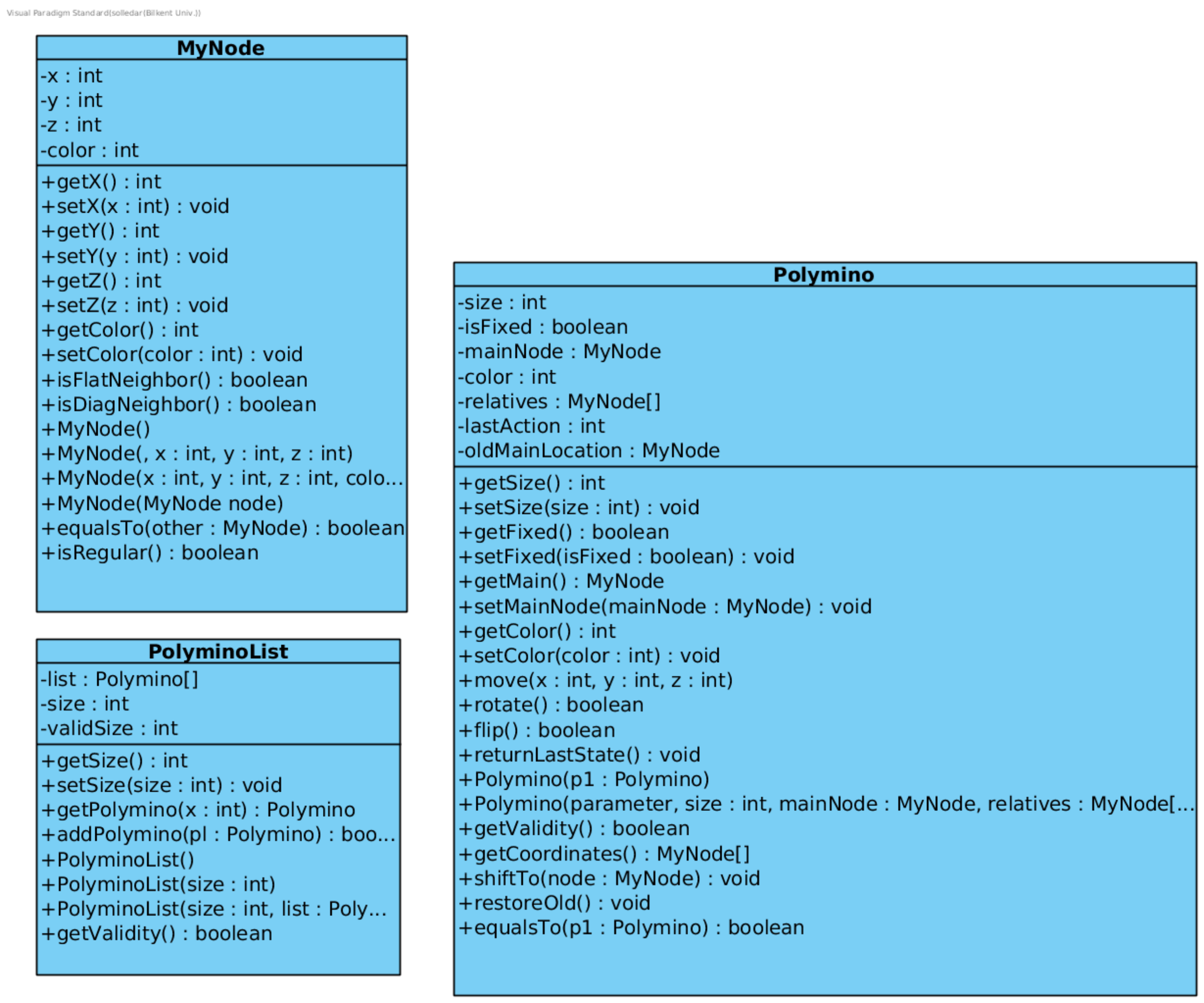

Visual Paradigm Standard(solledar(Bilkent Univ.))
MyNode
-x : int
-y : int
-z : int
-color : int
+getX() : int
+setX(x : int) : void
+getY() : int
+setY(y : int) : void
+getZ() : int
+setZ(z : int) : void
+getColor() : int
+setColor(color : int) : void
+isFlatNeighbor() : boolean
+isDiagNeighbor() : boolean
+MyNode()
+MyNode(, x : int, y : int, z : int)
+MyNode(x : int, y : int, z : int, colo...
+MyNode(MyNode node)
+equalsTo(other : MyNode) : boolean
+isRegular() : boolean
PolyminoList
-list : Polymino[]
-size : int
-validSize : int
+getSize() : int
+setSize(size : int) : void
+getPolymino(x : int) : Polymino
+addPolymino(pl : Polymino) : boo...
+PolyminoList()
+PolyminoList(size : int)
+PolyminoList(size : int, list : Poly...
+getValidity() : boolean
Polymino
-size : int
-isFixed : boolean
-mainNode : MyNode
-color : int
-relatives : MyNode[]
-lastAction : int
-oldMainLocation : MyNode
+getSize() : int
+setSize(size : int) : void
+getFixed() : boolean
+setFixed(isFixed : boolean) : void
+getMain() : MyNode
+setMainNode(mainNode : MyNode) : void
+getColor() : int
+setColor(color : int) : void
+move(x : int, y : int, z : int)
+rotate() : boolean
+flip() : boolean
+returnLastState() : void
+Polymino(p1 : Polymino)
+Polymino(parameter, size : int, mainNode : MyNode, relatives : MyNode[...
+getValidity() : boolean
+getCoordinates() : MyNode[]
+shiftTo(node : MyNode) : void
+restoreOld() : void
+equalsTo(p1 : Polymino) : boolean

Figure 7 : Pieces Package

This package contains physical pieces models and determines their basic behavior.

5.3.1.1 MyNode Class

This class represents atomic piece of the game. It has coordinates, which are x, y, z in the space. This class instances are able to check that another node is neighbour of itself in terms of x-y plane or diagonal plane. It will be used in construction of complex pieces but it can be also used for abstract checks in game especially as coordinate system.

5.3.1.2 Polymino Class

This class is an abstract class for 2D multi-node item. Because of abstractness there is no restriction to create 3D items but in generic usage it will be 2D in the child classes. It can be fixed or mutable.

5.3.1.3 PolyminoList Class

This class provide set of polyminos, which contains 12 pieces for the level creation in original game.

5.3.2 Control Package

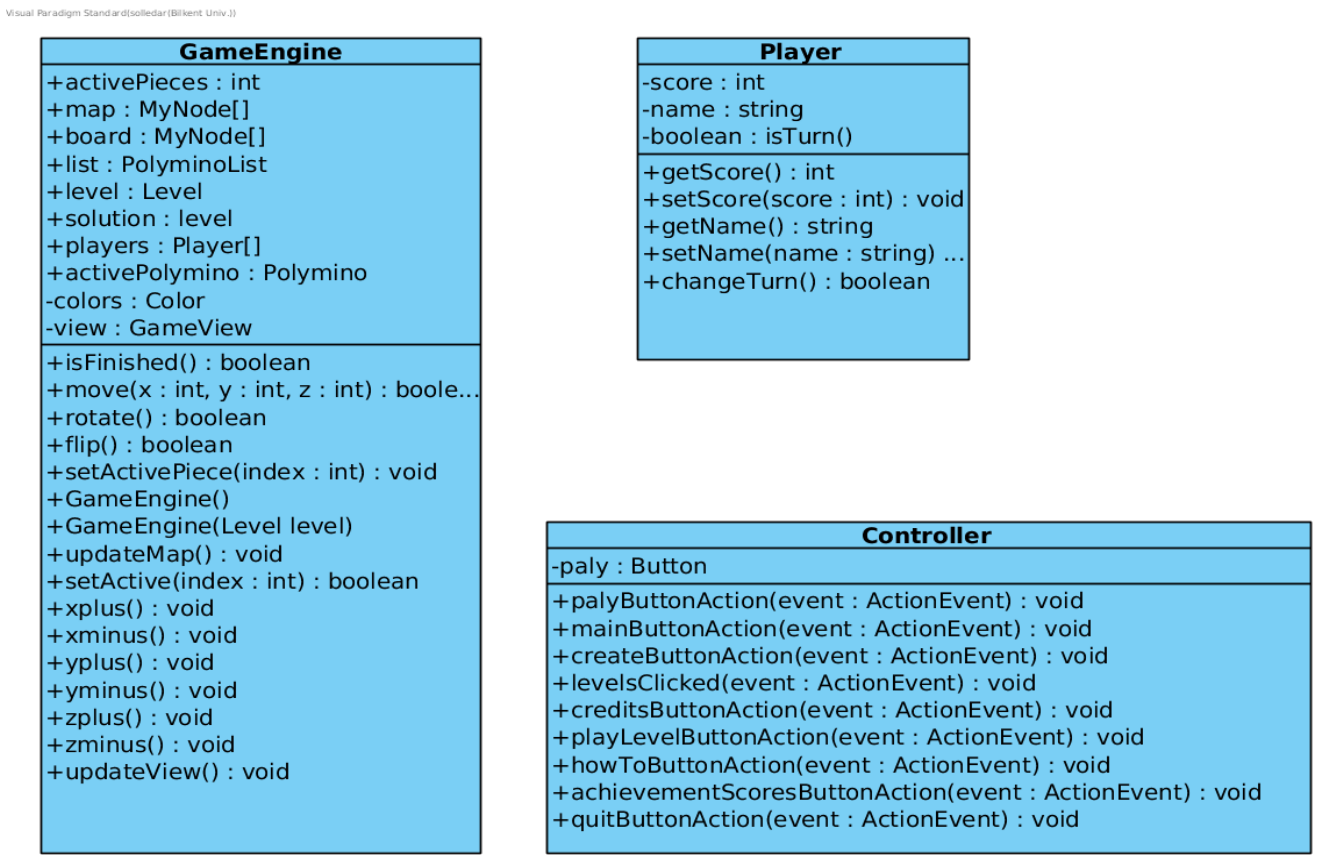

Visual Paradigm Standard(solledar(Bilkent Univ.))
GameEngine
+activePieces : int
+map : MyNode[]
+board : MyNode[]
+list : PolyminoList
+level : Level
+solution : level
+players : Player[]
+activePolymino : Polymino
-colors : Color
-view : GameView
+isFinished() : boolean
+move(x : int, y : int, z : int) : boole...
+rotate() : boolean
+flip() : boolean
+setActivePiece(index : int) : void
+GameEngine()
+GameEngine(Level level)
+updateMap() : void
+setActive(index : int) : boolean
+xplus() : void
+xminus() : void
+yplus() : void
+yminus() : void
+zplus() : void
+zminus() : void
+updateView() : void
Player
-score : int
-name : string
-boolean : isTurn()
+getScore() : int
+setScore(score : int) : void
+getName() : string
+setName(name : string) ...
+changeTurn() : boolean
Controller
-paly : Button
+palyButtonAction(event : ActionEvent) : void
+mainButtonAction(event : ActionEvent) : void
+createButtonAction(event : ActionEvent) : void
+levelsClicked(event : ActionEvent) : void
+creditsButtonAction(event : ActionEvent) : void
+playLevelButtonAction(event : ActionEvent) : void
+howToButtonAction(event : ActionEvent) : void
+achievementScoresButtonAction(event : ActionEvent) : void
+quitButtonAction(event : ActionEvent) : void

Figure 8 : Control Package

This package provides fundamental part of the game. Actually, this package is main part as “playable”.

5.3.2.1 GameEngine Class

This class orients the game package components with other packages. Firstly, it creates player and boards. Secondly it ensures level is loaded. After that point it is responsible for interaction between player and board until level is completed. According to game mode (normal / time).

5.3.2.2 Controller Class

This class controls the user interface and provides actions of menus.

5.3.2.3 Player Class

This class provides calculations for scores both for singleplayer and multiplayer.

5.3.3 Level Package

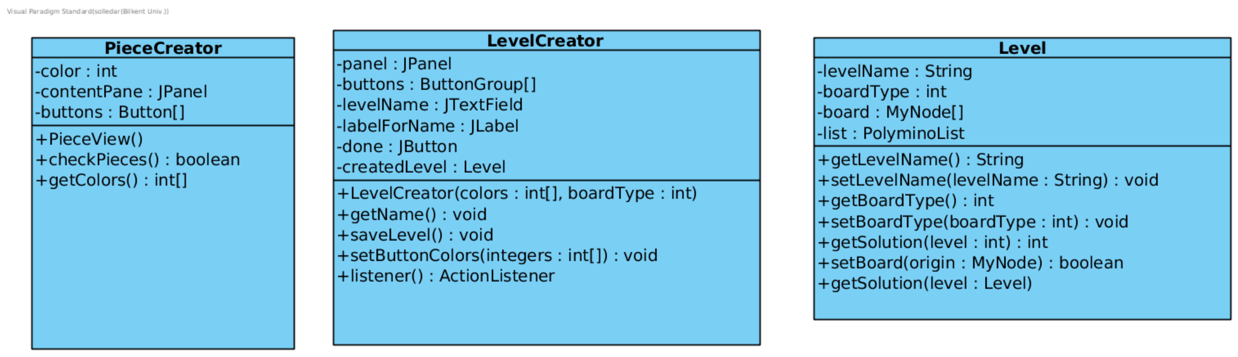


Figure 9 : Level Package

This package comprises classes related with level creating, storing and loading.

5.3.3.1 Level Class

It is modelling a level in terms of goal filled area, piece set and fixed pieces location. It also loads levels into game engine with complete information, which is partially included level class such as filling type and piece set.

5.3.3.2 LevelCreator Class

This class creates custom level by using 2 helping classes. It provides different filling areas and different piece set with original game. Creator can use original sets or add new set. It compares area and pieces to conclude that they are compatible. Creator locates fixed pieces and the class store the level in file. It also manage the user inputs and user interface of the panel.

5.3.3.3 PieceCreator Class

This class is helper class of LevelCreator. It creates new set of pieces for the game and check validity of the pieces. Also It manages user inputs and user interface of the panel of the piece creating part.

5.3.4 View Package

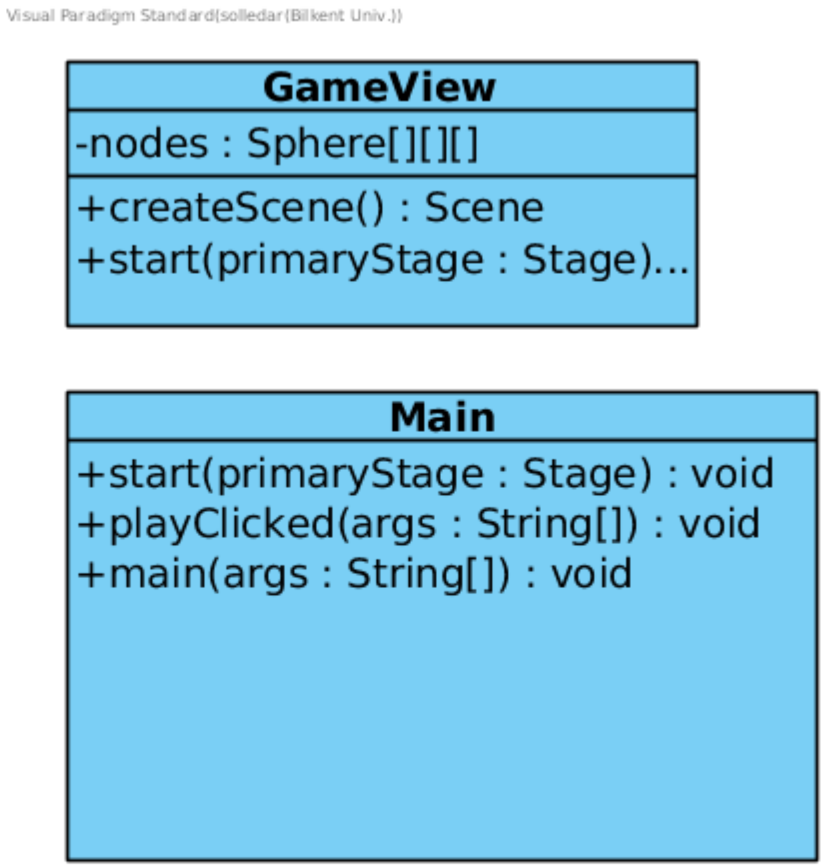

Visual Paradigm Standard(solledar(Bilkent Univ.))
GameView
-nodes : Sphere[][][]
+createScene() : Scene
+start(primaryStage : Stage)...
Main
+start(primaryStage : Stage) : void
+playClicked(args : String[]) : void
+main(args : String[]) : void

Figure 10 : View Package

This package contains menus to navigate the user to use many functionalities especially playing game.

5.3.4.1 GameView Class

It is 3 dimensional representation of the game with spheres.

5.3.4.2 Main Class

This class is starting point of the program. It loads the menus with help of the controller and view menus.

5.3.5 SettingAndUtilities Package

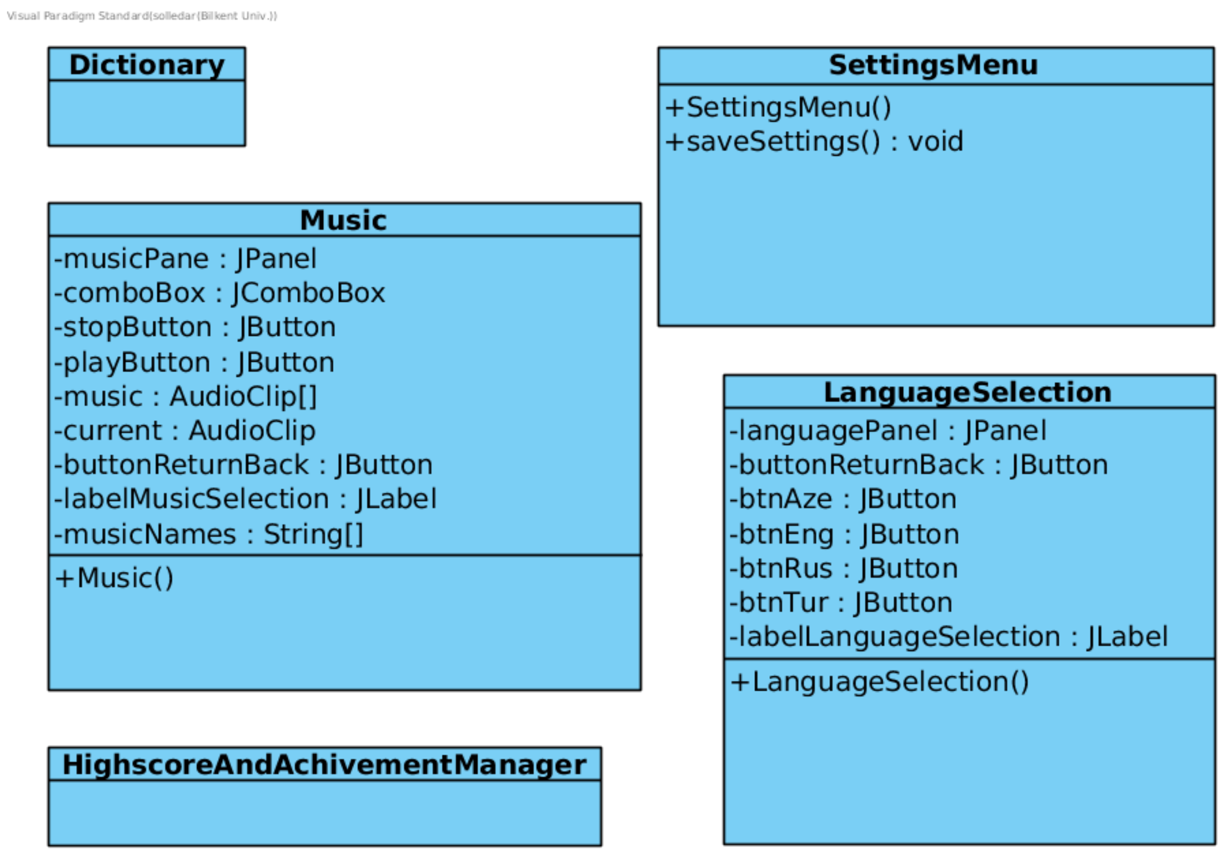


Figure 11 : Settings and Utilities Package

This package contains settings, high score and achievements classes.

5.3.5.1 SettingMenu Class

This class implements desires of user which comes from setting menu.

5.3.5.2 Music Class

This class manages the music and voice settings.

5.3.5.3 HighscoreAndAchievementManager Class

This class updates achievements and high scores according to circumstances of completeness and time after level ends.

5.3.5.4 LanguageSelection Class

This class represents languages for menus in the

5.3.5.5 Dictionary Class

This class holds predefined dictionary for menus.

5.4 Screen Mockups

5.4.1 Main Menu

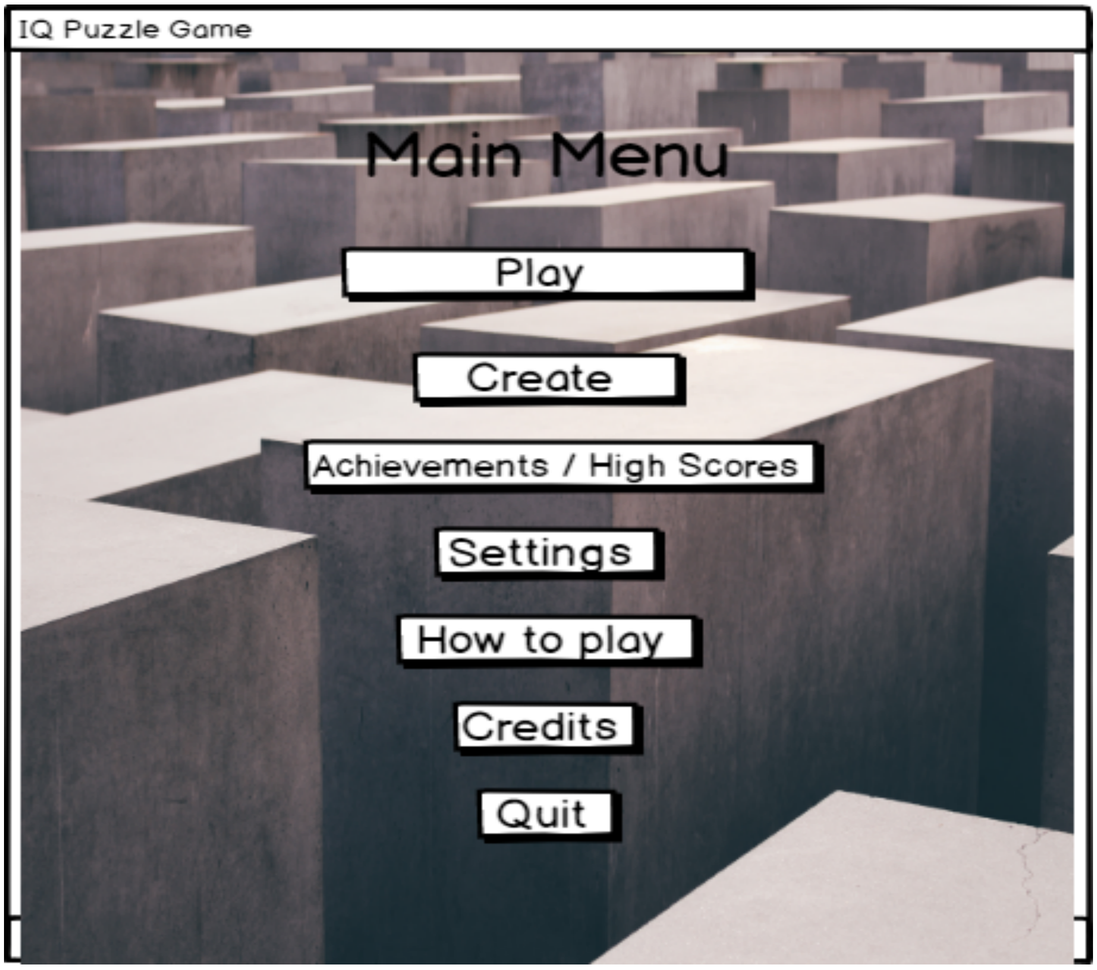


Figure 12 : Main Menu

Main Menu is start page for IQ puzzler game.

5.4.2 Play Menu

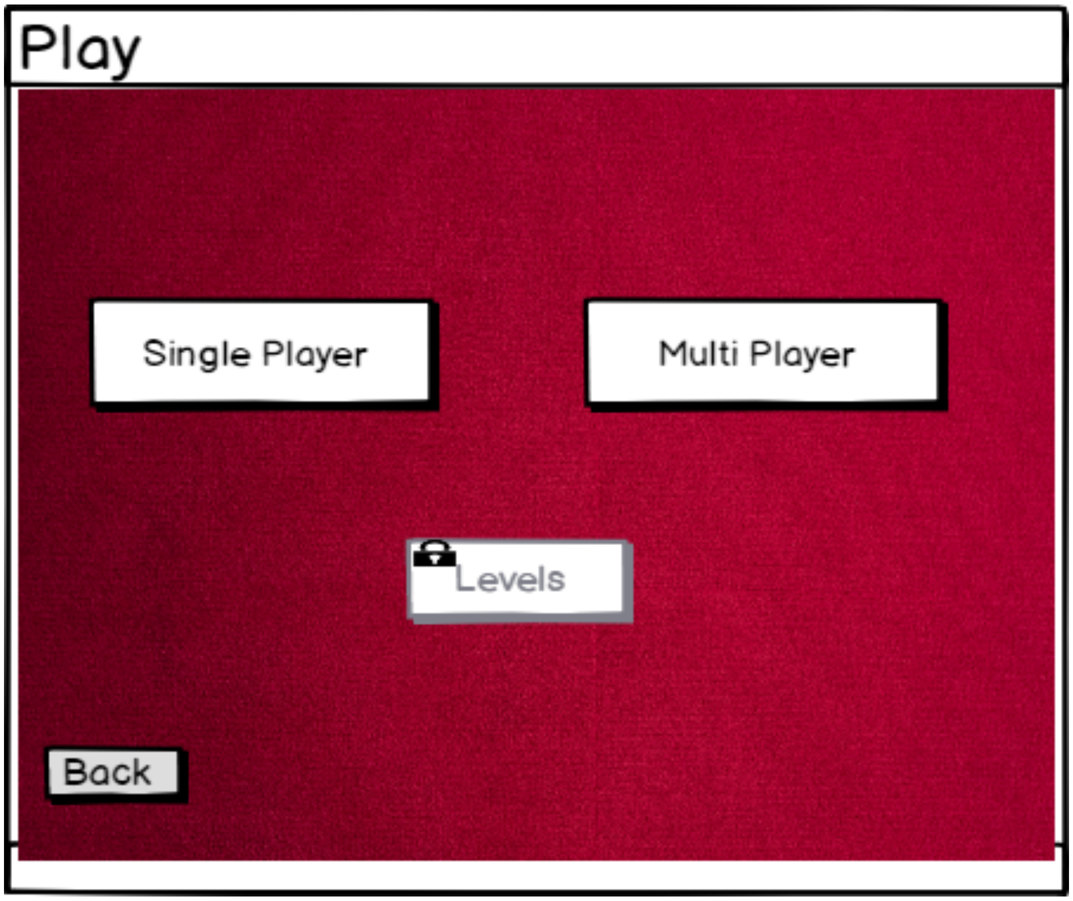


Figure 13 : Play Menu

Player should choose Single Player, Multi Player or Levels, after clicked button. He/she can back to the main menu via Back button.

5.4.3 Single Player

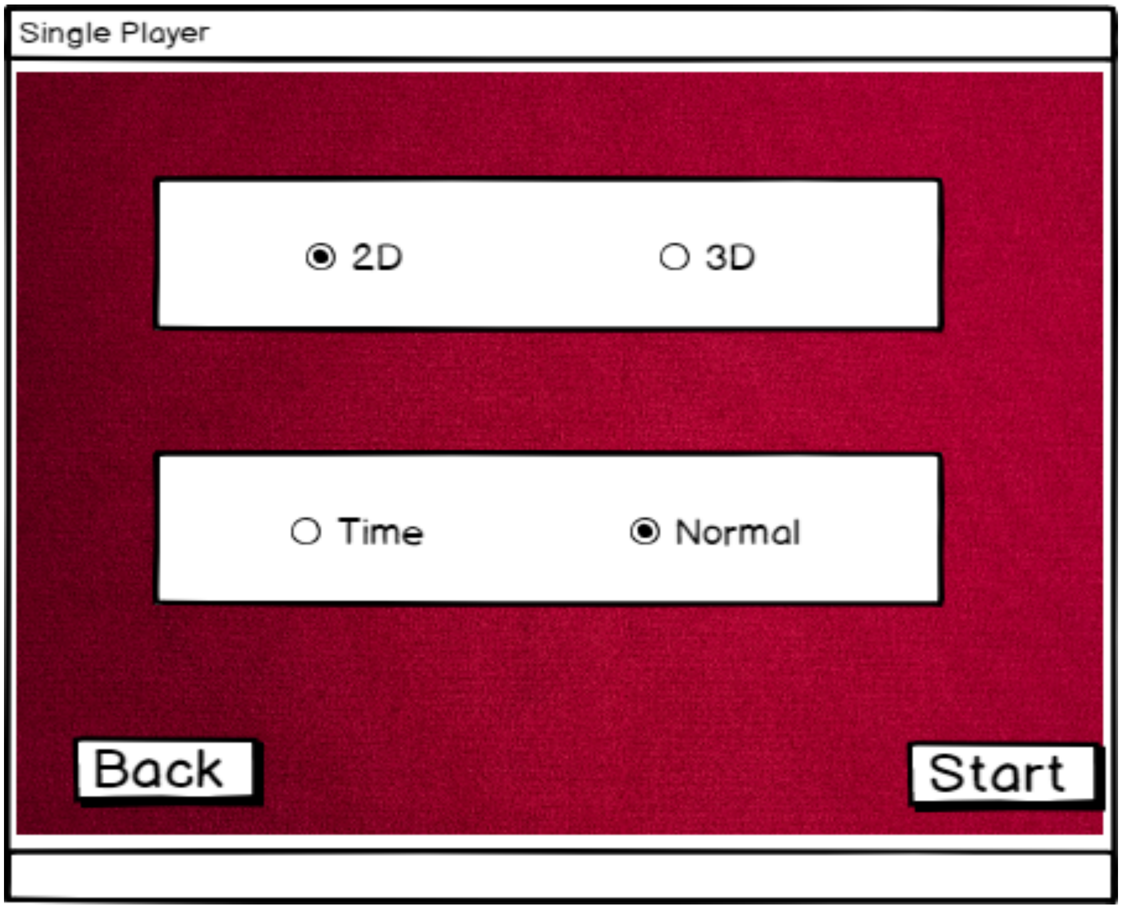


Figure 14 : Single Player Menu

Player can choose 2D or 3D game formats and also limited/limitless time versions for the game.

5.4.4 Levels

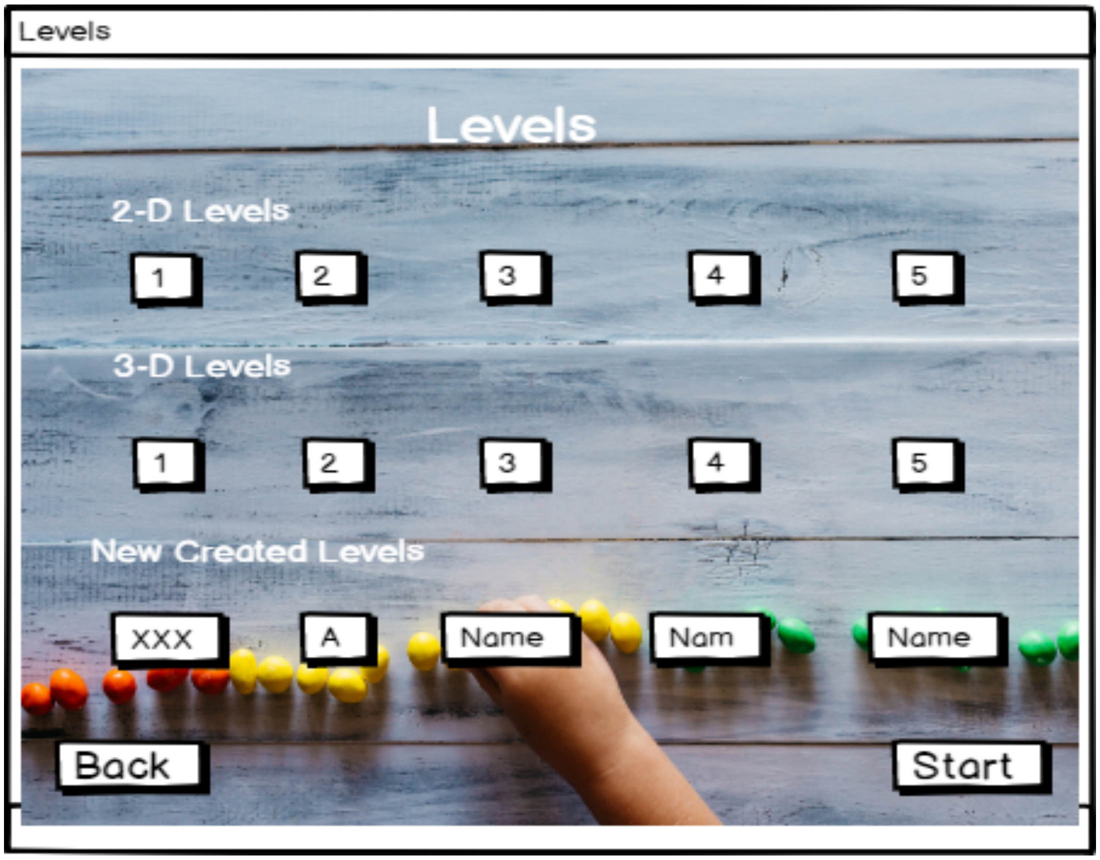
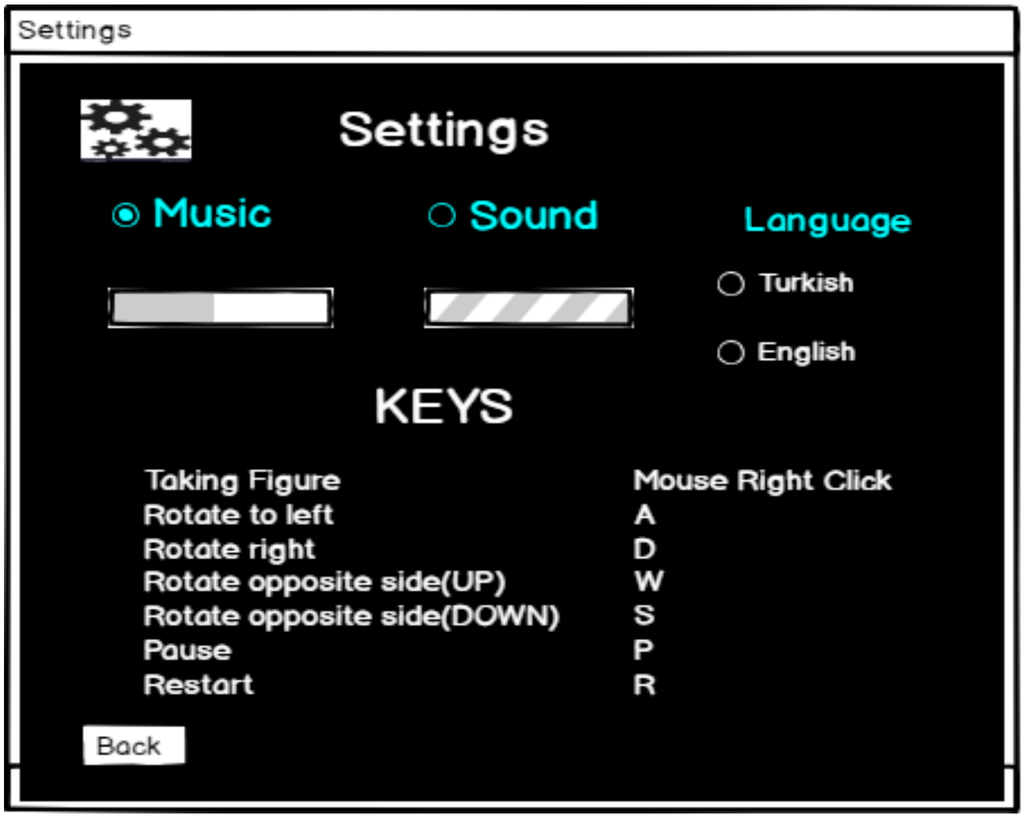


Figure 15 : Levels Menu

Levels part will be paid and user can play all levels in this mode

5.4.5 Settings



[✓ Music] Sound

KEYS

Taking Figure	Mouse Right Click
Rotate to left	A
Rotate right	D
Rotate opposite side(UP)	W
Rotate opposite side(DOWN)	S
Pause	P
Restart	R

Figure 16 : Settings Menu

Settings part player can adjust volume and language. He/she can change keys whatever comfortable for them.

5.4.6 Achievements/High Scores

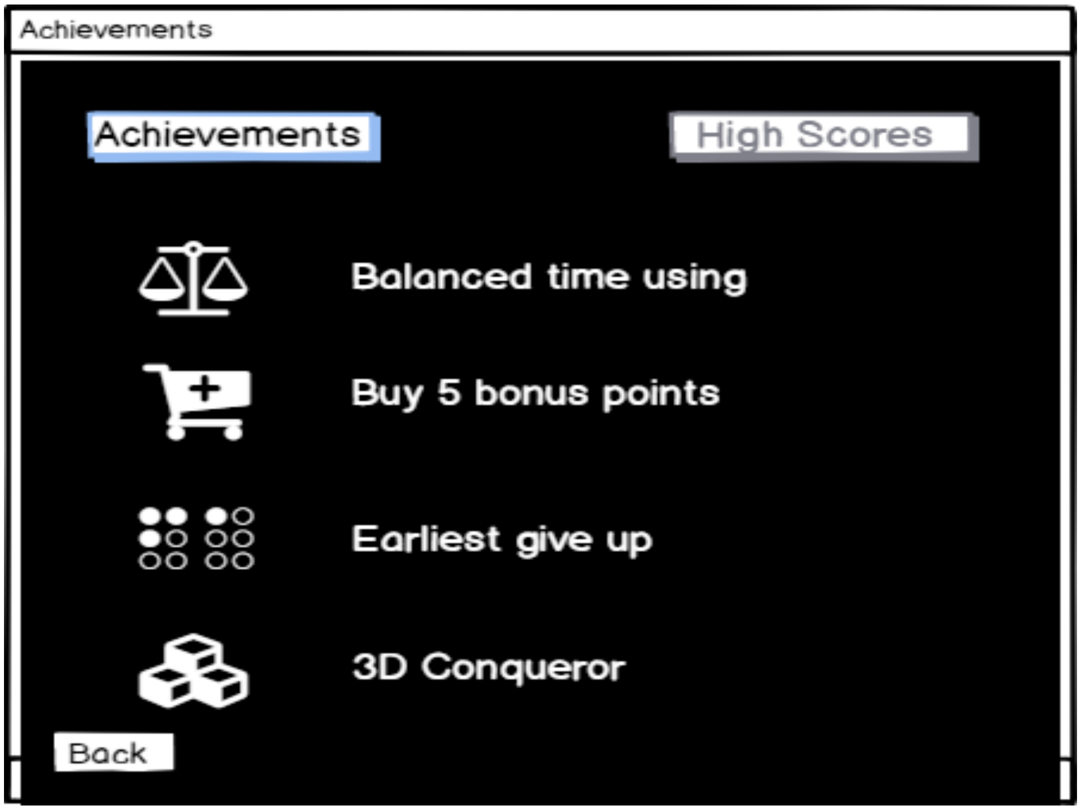


Figure 17 : High Scores / Achievements Menu

People can see the High Scores and Achievements.

5.4.7 Game Interface

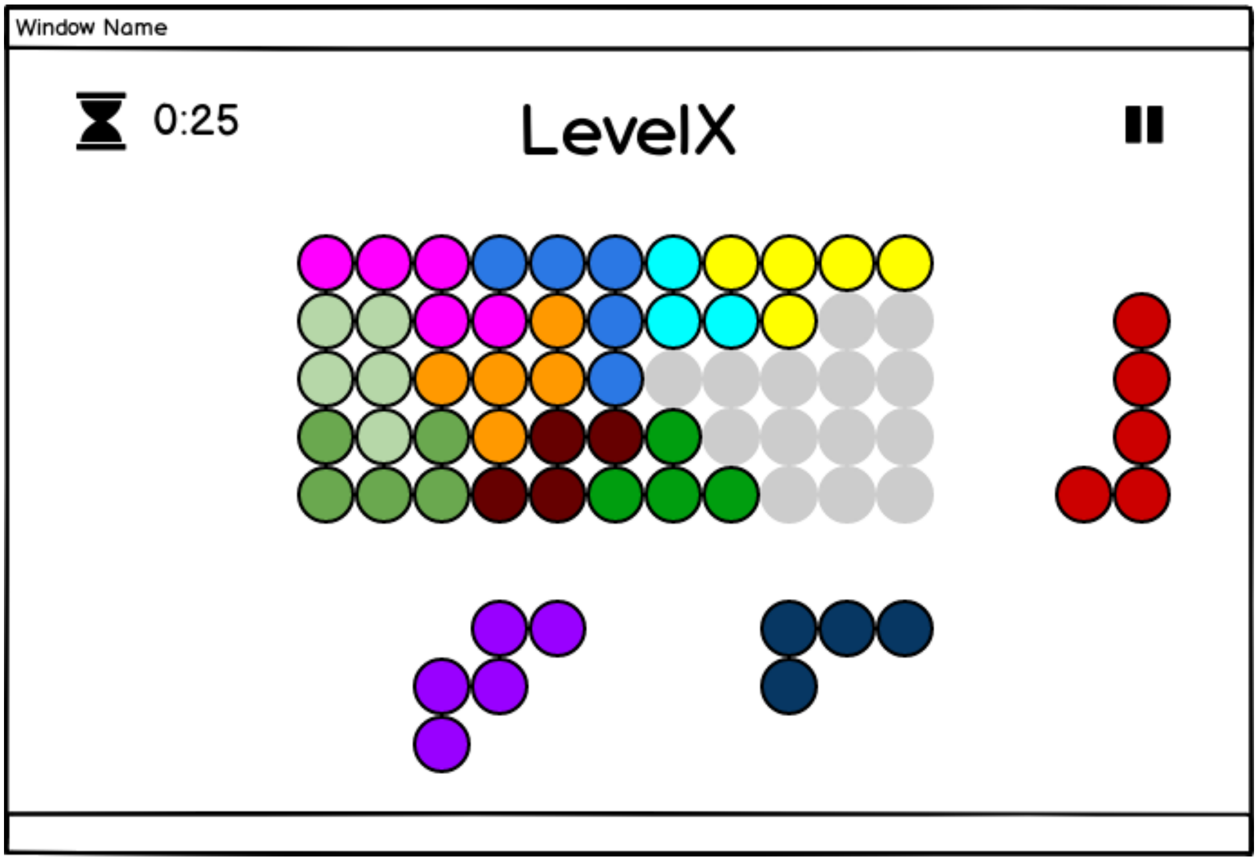


Figure 18 : Game Interface Menu

This is the screen that player plays the game.

5.4.8 Create Level Screen

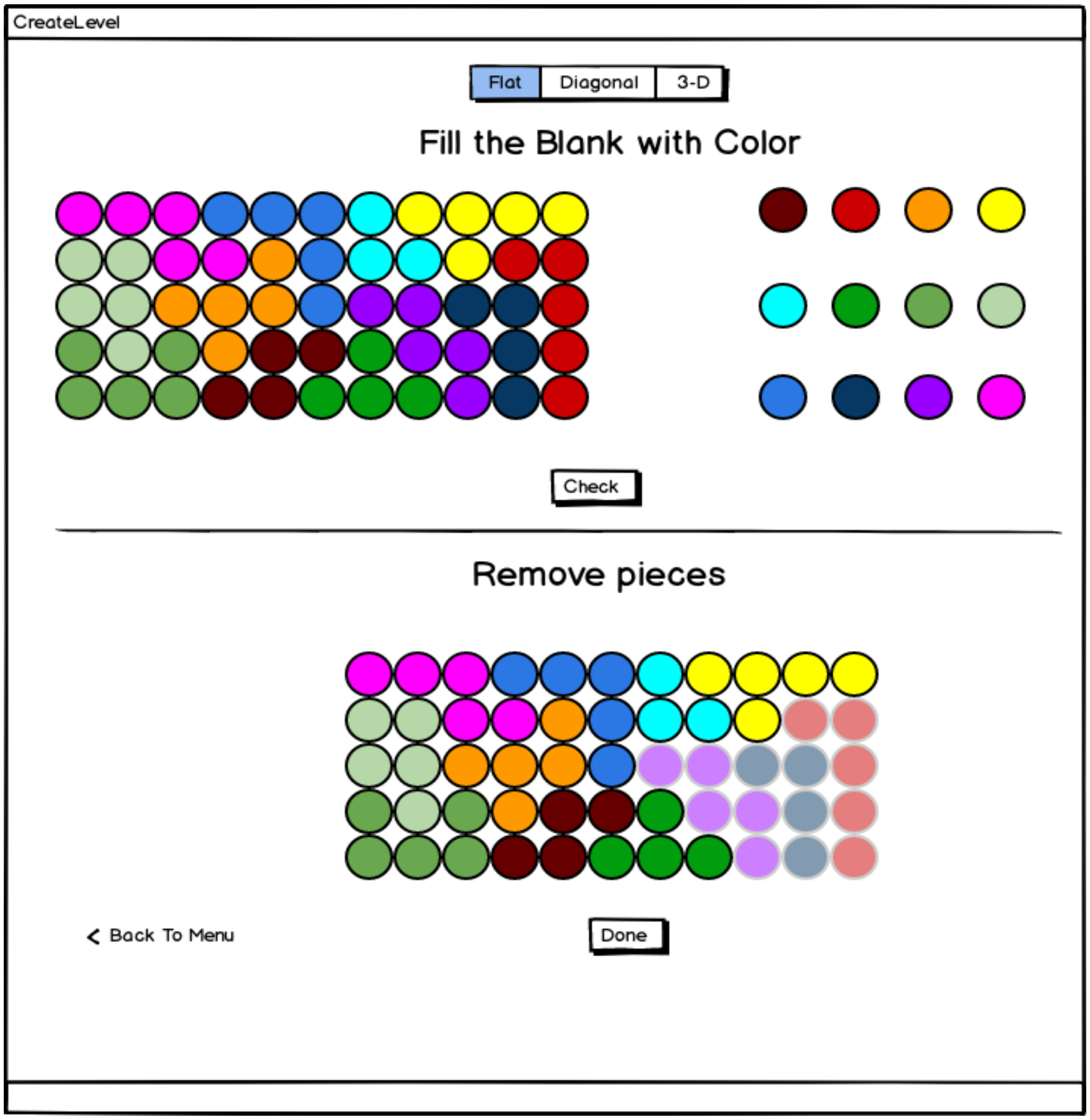


Figure 19 : Create Level Standard Screen

In create level part, player can create his/her own level, save and play this level.

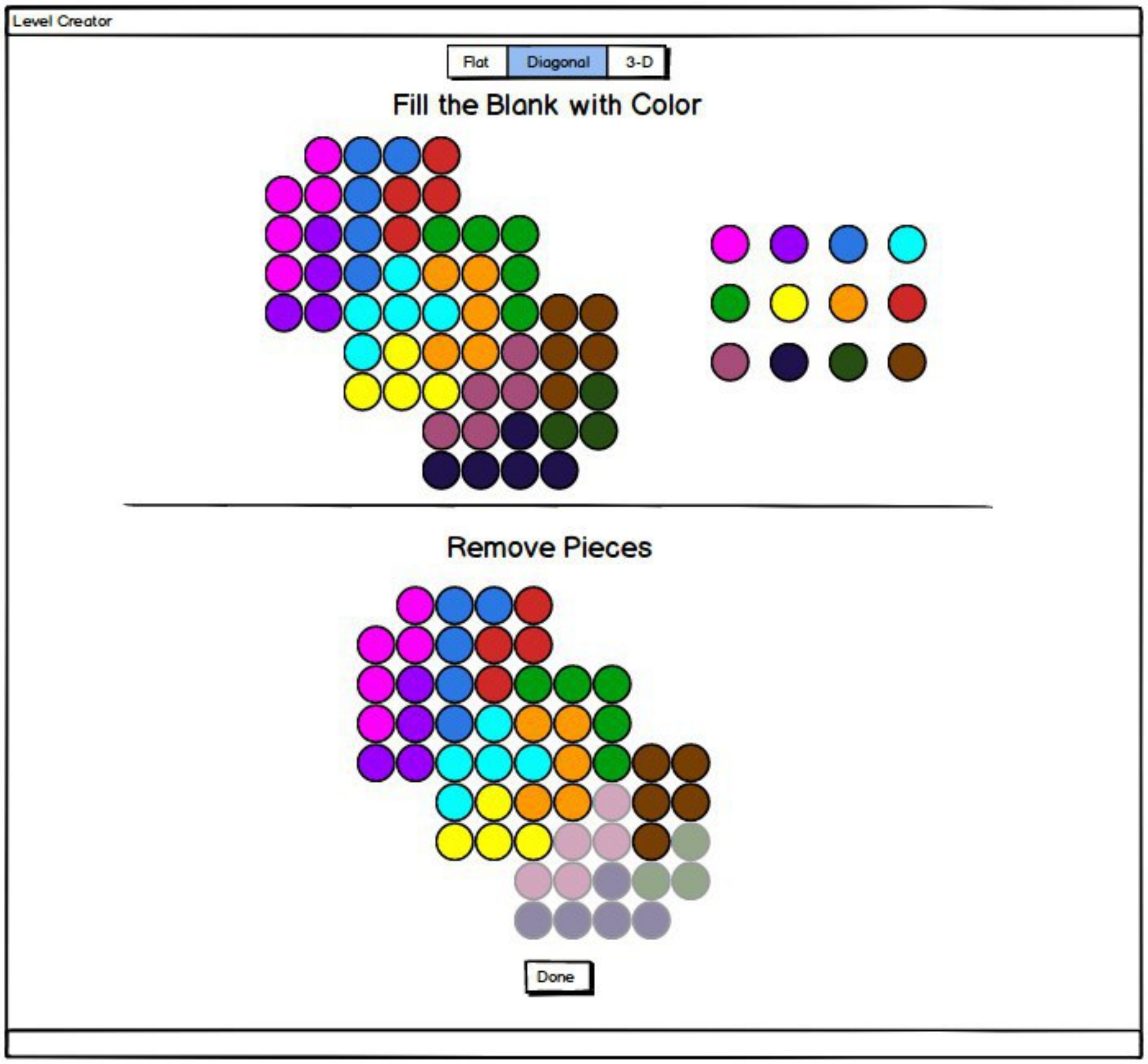


Figure 20 : Create Level Screen with Diagonal

In create level part, player can create his/her own diagonal level, save and play this level.

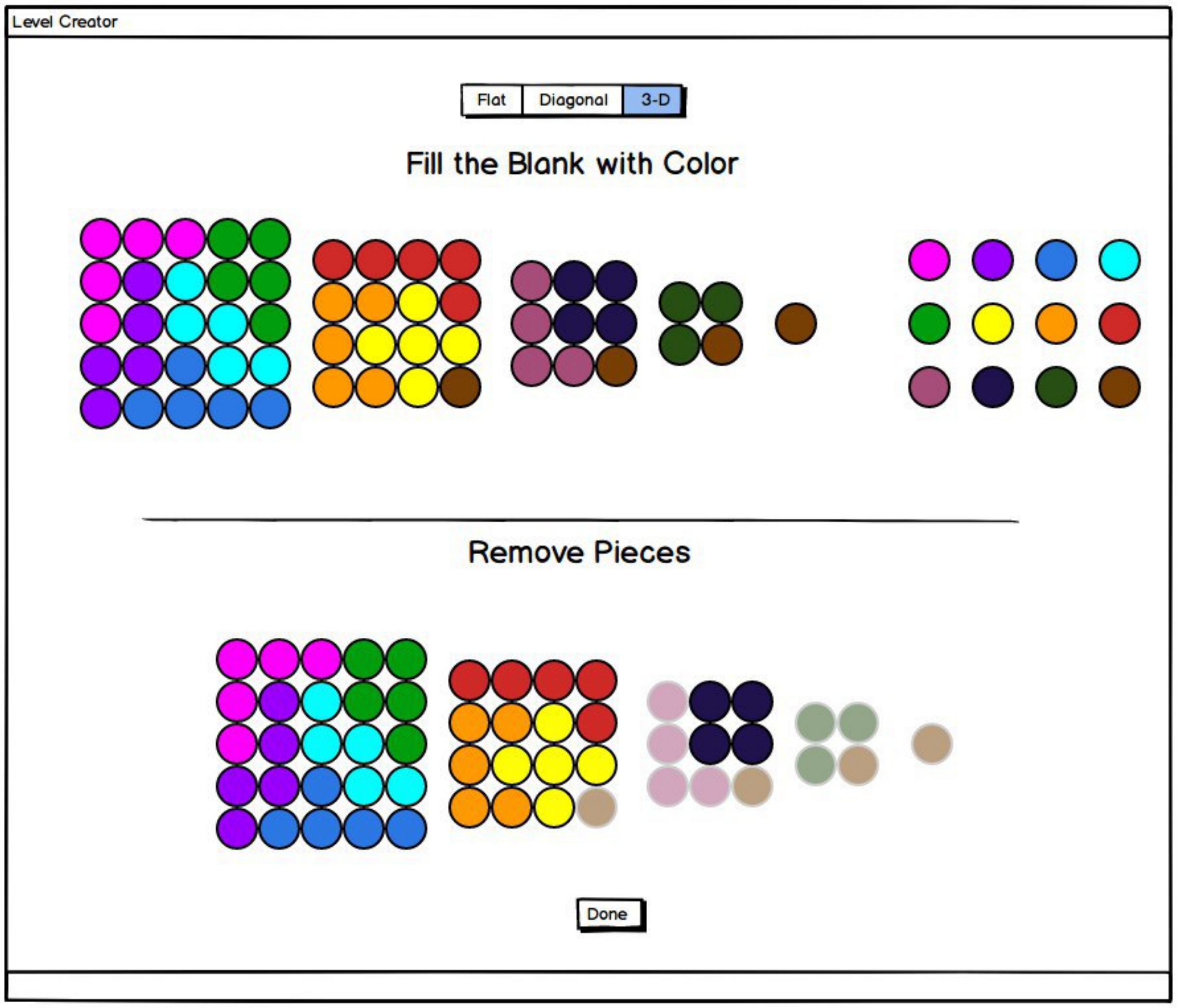
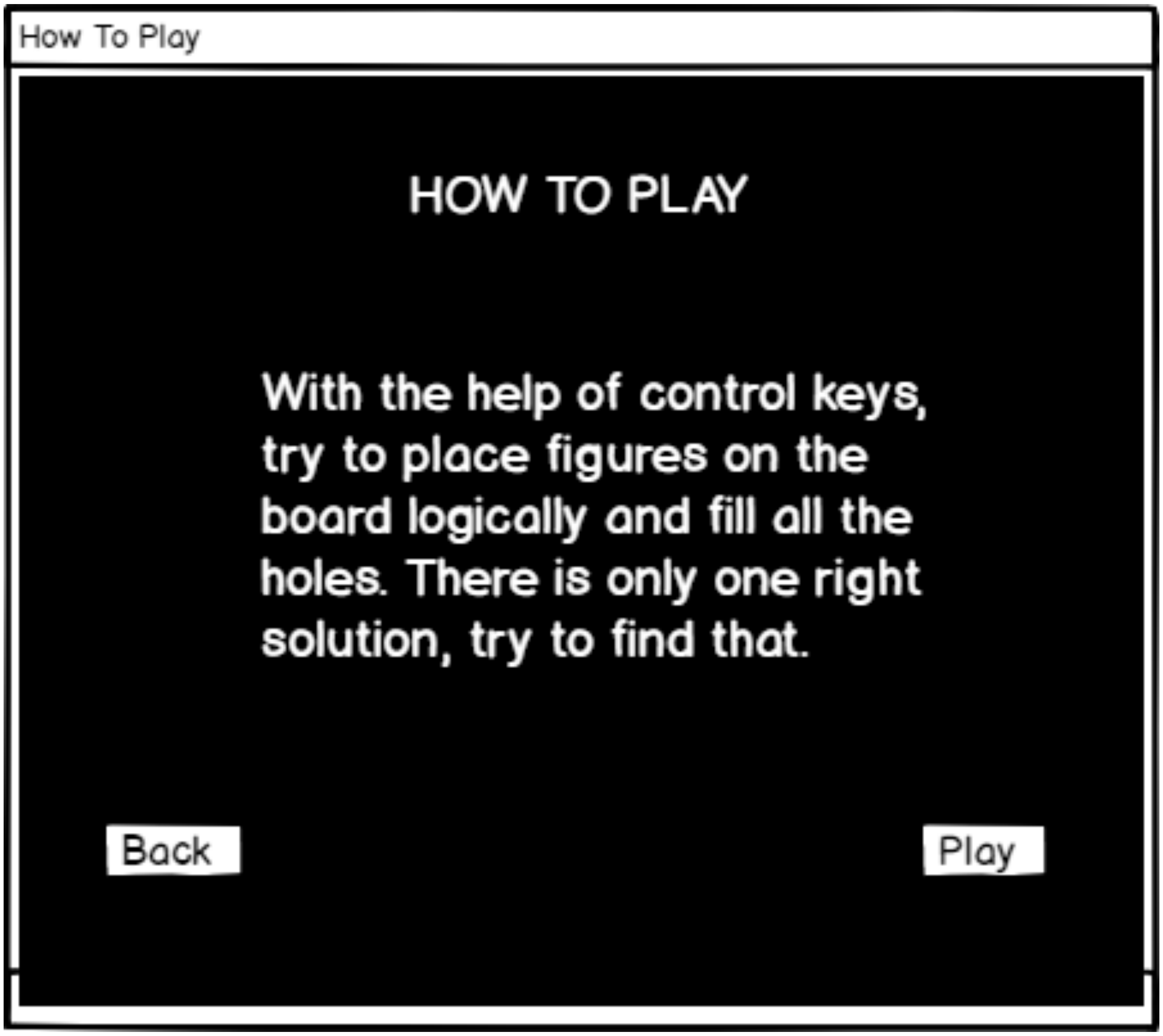


Figure 21 : Create Level Screen with 3D

In create level part, player can create his/her own 3D level, save and play this level.

5.4.9 How To Play Screen



With the help of control keys, try to place figures on the board logically and fill all the holes. There is only one right solution, try to find that.

Figure 22 : How To Play Screen

Player can get hints about how to play this game

5.4.10 Pause Screen

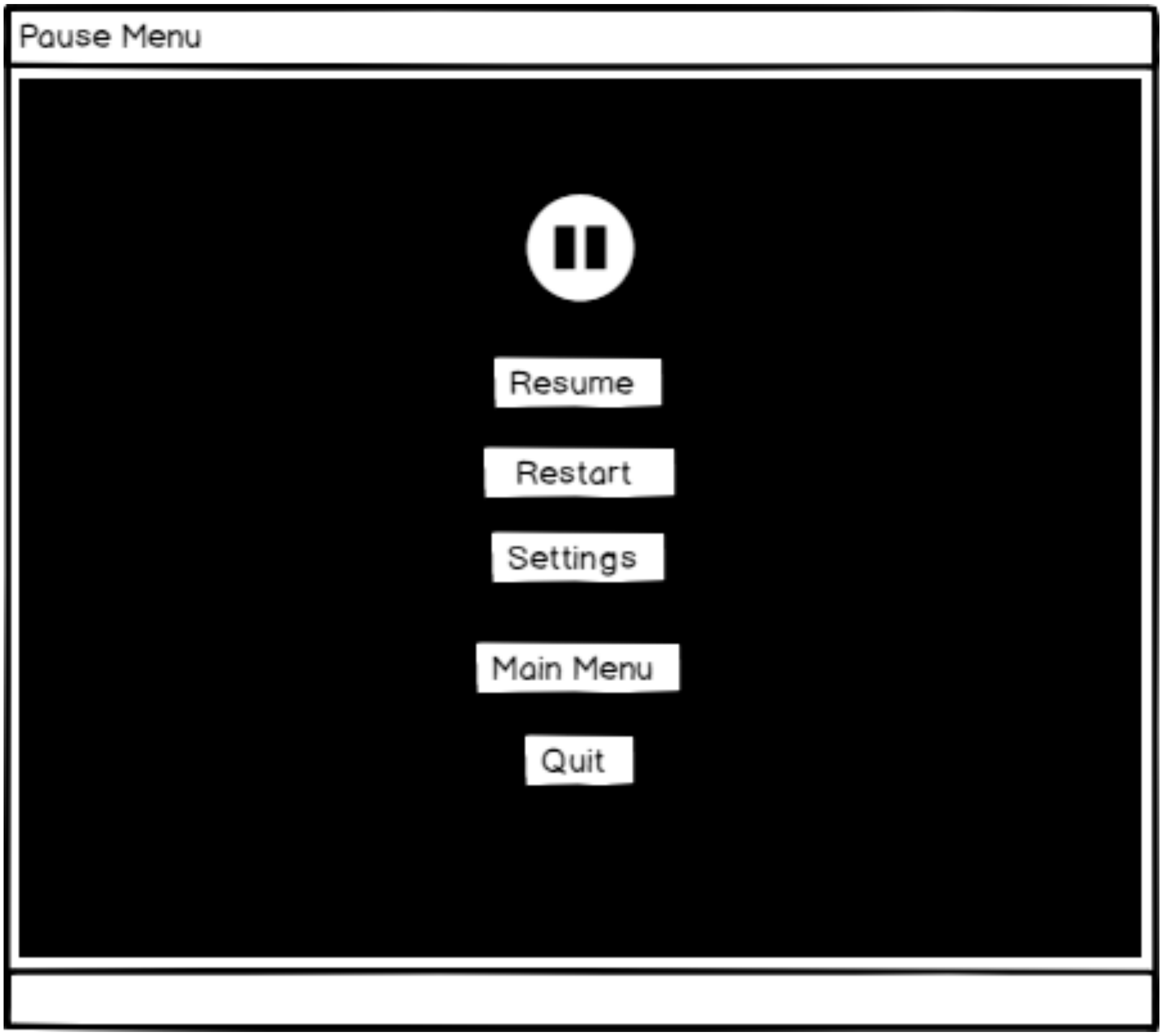


Figure 23 : Pause Screen

Player can stop the game, adjust volume or quit the game via pause game button.

5.4.11 Credits Screen

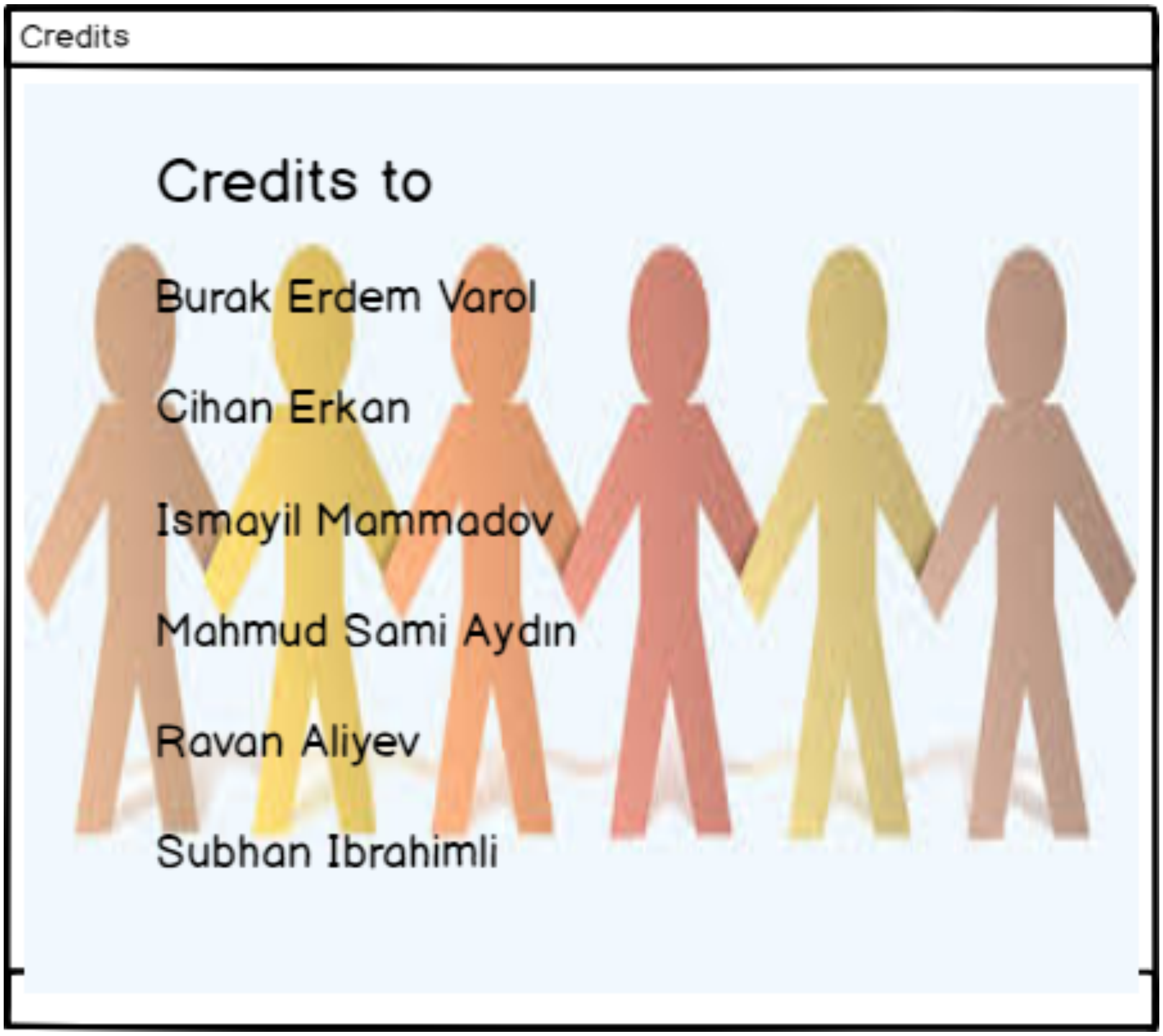


Figure 24 : Credits Screen

Player want to know who creates the game.

6. Improvement Summary

- We completely changed our introduction part as stated in feedback.
- We revised “Overview” part, added a preamble part and more detailed information about 2-Player Mode.
- We revised our functional requirements as asked in feedback, we listed our functional requirements and explained them briefly. We also added a new functional requirement, “Random Level Generator.”
- We completely changed “Nonfunctional Requirements” as asked in feedback.
- We improved our use case diagram. We removed unnecessary use cases and revised <<include>> relations of use cases.
- We updated our class diagram, we removed redundant classes and we decided use less classes for UI.
- We added captions for all figures.
- We removed conclusion part as it was not suitable for an SRS document.

7. Glossary

Flip: rotating thing on XY line

Multiplayer Mode : Two player game mode

Node: One sphere in coordinate system.

Polymino: Plane geometric figure formed by joining one or more equal sphere edge to edge.

Rotate: rotating thing on Z line

8. References

Civilization (series). (2018, November 19). Retrieved from https://en.wikipedia.org/wiki/Civilization_(series)

Chesshere.com. (n.d.). Chess History. Retrieved November 27, 2018, from https://www.chesshere.com/resources/chess_history.php

IQ Puzzler Pro. (n.d.). Retrieved from https://www.smartgames.eu/uk/one-player-games/iq-puzzler-pro