Bilkent University

Department of Computer Engineering

CS 319 Term Project

Icy Tower

Analysis Report

Section 1

Group 1C

Project Group Members:

Cansu Yıldırım

Berke Soysal

Ozan Kerem Devamlı

Supervisor: Eray Tüzün

Contents

1. Introduction

We wanted to design and implement a new and extended version of the well-known action and platform game, Icy Tower. As all of us have been played this game when we were children, we decided to implement this game with passion. Also the structure of the game seemed compatible with the content of our course such that the game can be implemented with a manner of object oriented software engineering.

Our game will have features such that:

- Different levels of difficulty
- Different types of platforms
- Sound and music options
- Character options
- Control button options
- Continuously incrementing game speed
- Bonus and Traps
- In-game pause option
- List of high scores

We will implement the game in Java environment with using the features of the JavaFX API. We choose JavaFX because it provides easy and smooth graphics and UI design properly while it does not make the integration of the objects difficult.

2. Overview

2.1 Gameplay

Icy-Tower is a single player platform game. The main objective of the player is to jump through the bars quickly to reach a higher place of the Icy Tower, and maximizing its point by sliding on the wall and jumping over as much bars as possible at each jump to make combo jump (**Combo Jump:** Player jumps to wall and then jumps to an above bar, but “not the nearest bar”). The game has no end, so there is no “top” of the tower. The bars that player jump will change (usually get harder) as the game continues, there will be different advantages and disadvantages of each platform. There will be three different levels that player can choose at the beginning, easy, medium and hard. The game difficulty will be increased with speed of the game, frequency and length of the upcoming bars. Also, the game difficulty will be increased during the game time. As the game continues, the bars will drop down faster, the frequency of generating bars, and the size of the bars will be decreased.

2.2 Main Character

There will be different type of characters that player can choose from the character settings. The only difference between the characters is their appearance. Character can jump, or move left/right. Character can make combo jump and get more points.

2.3 Map

Map will be generated randomly at each time the player goes above from the game screen. Map consists of bars, bonus and traps. At the beginning of the game the map will

create the bars frequently, but as the game goes on, the map will move to upward faster and the frequency of generating bars will drop.

2.4 Bars

Bars are the objects that player can jump on. There will be different kind of bars (icy, wooden, hot and hardly visible) and each of these has different characteristics. Some of them will be slippery, some of them will drop the game character if he holds there too much, some of them are advantageous for the game character as it is too bouncy etc.

2.5 Additional Items

There will be a single type of bonus and a single type of trap as collectibles. The bonus is a balloon that carries the character up for a while. As a trap, sometimes bombs fall from sky to kill the character. If the character is hit by a bomb, the game will be over. The frequency of creating the balloon and trap changes according to the level. In easy level, there will be more balloons and fewer bombs, vice versa on hard level.

2.6 Score

Game score will be increased as the player will go upward. If the player will jump to the bar with sliding on the wall, s/he passes more than one bar and gets extra points. If a player gets the balloon bonus and moved up by the balloon, he will still get the points for going up.

2.7 Settings

Player can change the volume of the game sounds and music, change the game character, or can change the control buttons from the settings.

2.8 Difficulty

There will be three types of difficulty level in the game, these are: easy, medium and hard. The difficulty of the game is increased by decrementing the frequency of bars, speed up the game screen speed, and shrinking the size of the bars. Player can select game difficulty each time before he starts the game.

3. Functional Requirements

3.1 Play Game

After entering the play game screen, players will be asked about the difficulty level of the game. After choosing the level, the game will start. In each level, player will try to jump on the bars in order to reach as above as possible while getting more points by making combo jumps as mentioned above. The speed of game screen increases by time passes. So the player must accelerate too. Player must be fast enough to not to fall behind the game screen. However, if the player is too fast and intends to exceed the game frame, game screen catches up with the player. In addition, the types of bars will also be changed. With the progression of the player, bars' properties will make the game harder for players. For example, the bars will be shortened. It will be harder to move on these shorter bars. From the level easy to hard, the speed of the game screen changes. Therefore, in the hard level, player experiences faster screen movement even at the beginning of the game. The player can control the character with the left-right and space button (or with A, D, W by changing

the button settings). Moreover, some bonuses have place in the game. Balloon bonus helps the player to move up easily. When the player see the balloon, it is better to get it. Also, there are bombs as traps. If the character encounters with a bomb, it is better to escape to not to be killed.

3.2 Credits

Credits can be accessed by a button on the main menu. The player can see the names of the developers by this screen. In case they want to give a feedback, there will be e-mail addresses of developers on this screen.

3.3 Settings

Settings can be accessed by a button on the main menu. Player can adjust the volume of the music; change the music and the control buttons to play the game from settings screen. Also player can see the different characters and select his/her favourite to play with it from this setting page.

3.4 How to Play

How to Play screen informs the players about

- The logic of the game

- Control buttons that will be used

- Setting options

- How to record your score

- Bonuses

By reading this, users can play the game without having any trouble.

3.5 High Scores

High Scores screen displays the scores with names of the players in descending order. Therefore, the highest score will be at the top of the list. Every time a player makes a score higher than the previous high score, the new score will be recorded. System asks the player to type a name. With this name, the score takes its place which is the top of the list. So, the player can see the current list whenever he/she wants.

3.6 Exit

Exit screen works to quit the game. After entering exit screen, verification question shows up. If the player verifies the exit command, the system closes.

4. Non-functional Requirements

4.1 Usability

Game will be easy to understand and use for any people from any age and educational level. Even without looking to “How to Play” screen, the player will able to understand the basics and the objectives of the game.

4.2 Maintainability

Game will be written with obeying clean code rules that is common sense for long time Java Developers. Thus, the attributions to the game will be comfortably done by all users of the group. The code will be written understandable and reusable.

4.3 Extendibility

The game will be extendable. Extra features and extensions will be easy to implement. These implementations will be done without making dramatic changes on the core code of the game. We will obey to the Object Oriented Approach to maintain the extendable design.

4.4 Understandability

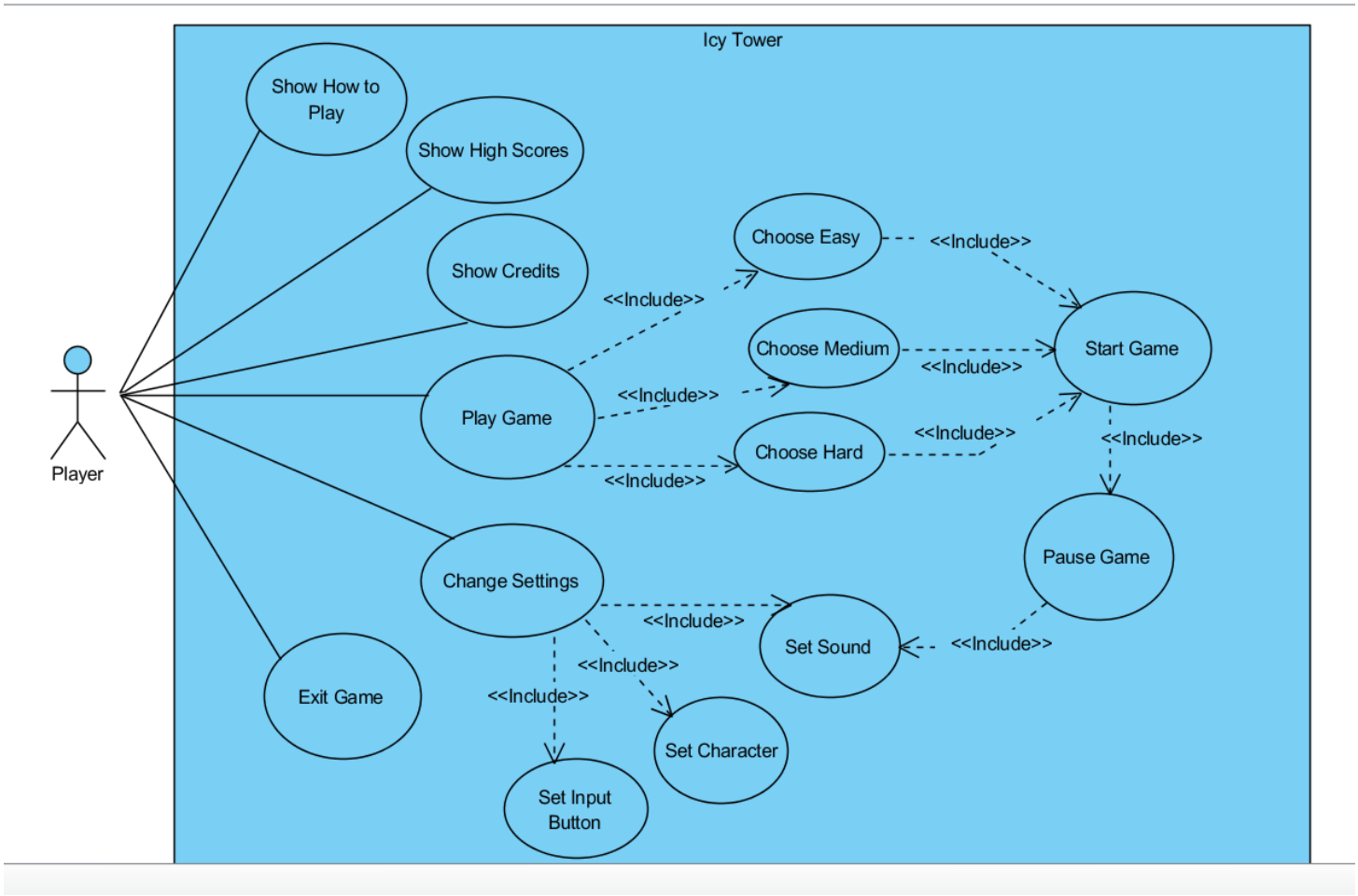
The understandability term is important for us in two aspects, firstly the game will be understandable and to play for customers. Secondly the design of the project and the all code written for it will be understandable and maintainable, also reusable for us and the other programmers around the world, as it is open source software that freely can be extended for all programmers.

4.5 Reusability

We maintain the object-oriented approach in our project that so that we can reuse the code as we wanted. For instance the map manager we will implement can be used for a completely different game, or the bar class we create for the game can be used for a Super Mario game without much trouble.

5. System Models

5.1 Use Case Models



Use Case: Play Game

Actor: Player

Stakeholders and Interests:

- Player wants to play the game

- System asks the difficulty level to the player and starts the game

Pre-condition: Player must be in the menu

Post-condition: System displays the difficulty levels

Entry-condition: Player selects the “Play Game” button from the menu

Exit-conditions:

• Player dies

Event Flow:

1- Player selects the “Play Game” button from the menu

2- Player selects the medium level of the game

3- Player selects the “Start Game” button

4- Player jumps to the bars and gets points

5- Player jumps to the bars with sliding the wall so that s/he gets more points

6- Player falls down to the space

7- The game is over

Alternative Event Flow:

1- Player’s speed is less than the speed of the game screen

   • Player falls behind the screen move

   • Player becomes out of the game screen

   • Player dies

2- Player wants to pause the game

   • Player presses the button “Pause” on the game screen

   • Game pauses

Use Case: Show How to Play

Actor: Player

Stakeholders and Interests:

- Player wants to learn how to play the icy tower game
- System shows the how to play screen

Pre-condition: Player must be in the menu

Post-condition: -

Entry-condition: Player selects the “How to Play” button from the menu

Exit-condition: Player selects the “Back to Menu” from the how to play screen

Event Flow:

1- Player wants to learn how to play

2- Player selects the “How to Play” button from the menu

3- System displays the how to play screen

Alternative Event Flow:

1- Player wants to go back to the menu

2- Player selects “Back to Menu” button from the how to play screen

3- System displays the menu

Use Case: Show Credits

Actor: Player

Stakeholders and Interests:

- Player wants to see the credits of the icy tower game
- System shows the credits screen

Pre-condition: Player must be in the menu

Post-condition: -

Entry-condition: Player selects the “Credits” button from the menu

Exit-condition: Player selects the “Back to Menu” from the credits screen

Event Flow:

1- Player wants to see the credits of the game

2- Player selects the “Credits” button from the menu

3- System displays the credits screen

Alternative Event Flow:

1- Player wants to go back to the menu

2- Player selects “Back to Menu” button from the credits screen

3- System displays the menu

Use Case: Show High Scores

Actor: Player

Stakeholders and Interests:

- Player wants to see the high scores of the game
- System shows the high scores screen

Pre-condition: Player must be in the menu

Post-condition: -

Entry-condition: Player selects the “High Scores” button from the menu

Exit-condition: Player selects the “Back to Menu” from the high scores screen

Event Flow:

1- Player wants to see the scores of the game

2- Player selects the “High Scores” button from the menu

3- System displays high scores screen

Alternative Event Flow:

1- Player wants to go back to the menu

2- Player selects “Back to Menu” button from the high scores screen

3- System displays the menu

Use Case: Change Settings

Actor: Player

Stakeholders and Interests:

- Player wants to adjust the game settings
- System asks which type of the setting the user wants to adjust
- System changes the chosen setting

Pre-conditions:

- Player must be in the menu
- Player must be in the pause menu for “Sound Setting”

Post-condition: Settings are updated

Entry-conditions:

- Player selects the “Settings” buttons from the main menu
- Player selects the “Sound Setting” button from the pause menu

Exit-condition:

- Player selects the “Back to Menu” button from the settings screen
- Player selects the “Resume Game” button from the pause menu

Event Flow:

1- Player selects the “Settings” button from the main menu

2- Player selects the “Sound Settings” button

   • Player selects the music

   • Player adjusts the volume

   • Player repeats the steps 5 and 6

3- Player selects the “Character Settings” button

   • Player selects the character type

   • Player repeats the steps 5 and 6

4- Player selects the “Button Settings” button

   • Player decides the buttons that will be used (right, left, space or A, D, W on the keyboard)

   • Player repeats the steps 5 and 6

5- Player selects “Back to Menu” button

6- System displays menu

Alternative Event Flow:

1- Player selects the “Sound Setting” button from the pause menu

2- Player changes the music or volume

3- Player selects the “Resume Game” button

4- Game resumes

Use Case: Pause Game

Actor: Player

Stakeholders and Interests:

• Player wants to pause the game

• System displays the pause menu

Pre-conditions:

• Player must be playing the game

Post-condition: -

Entry-conditions:

• Player selects the “Pause” button from the game screen

Exit-condition: Player selects the “Resume Game” button from the pause menu

Event Flow:

1- Player selects the “Pause” button from the game screen

2- Game pauses

3- System displays the pause menu

Alternative Event Flow:

1- Player selects the “Resume Game” button from the pause menu

2- Game resumes

Use Case: Exit Game

Actor: Player

Stakeholders and Interests:

• Player wants to exit

• System closes the game

Pre-conditions: Player must be in the menu

Post-condition: System closes

Entry-condition:

• Player selects the “Exit” and “Yes” buttons respectively

Exit-condition: Player selects “No” button from the exit menu

Event Flows:

1- Player selects the “Exit” button from the menu

2- Player selects the “Yes” button from the menu

3- The game closes

Alternative Event Flow:

1- Player selects the “Exit” button from the menu

2- Player selects the “No” button from the exit menu

3- System displays the menu

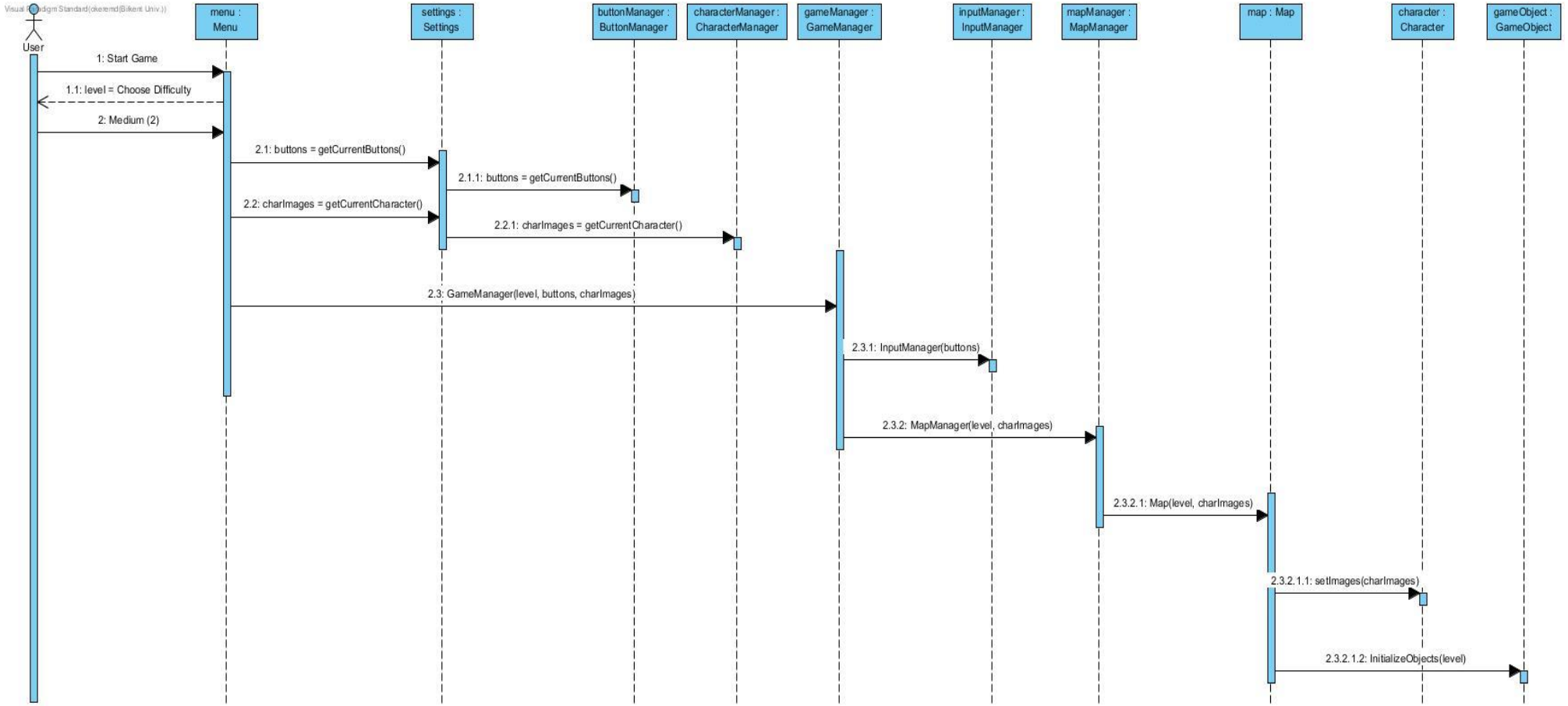
5.2 Dynamic Models

5.2.1 Sequence Diagrams

5.2.1.1 Start Game

Scenario: User starts the game

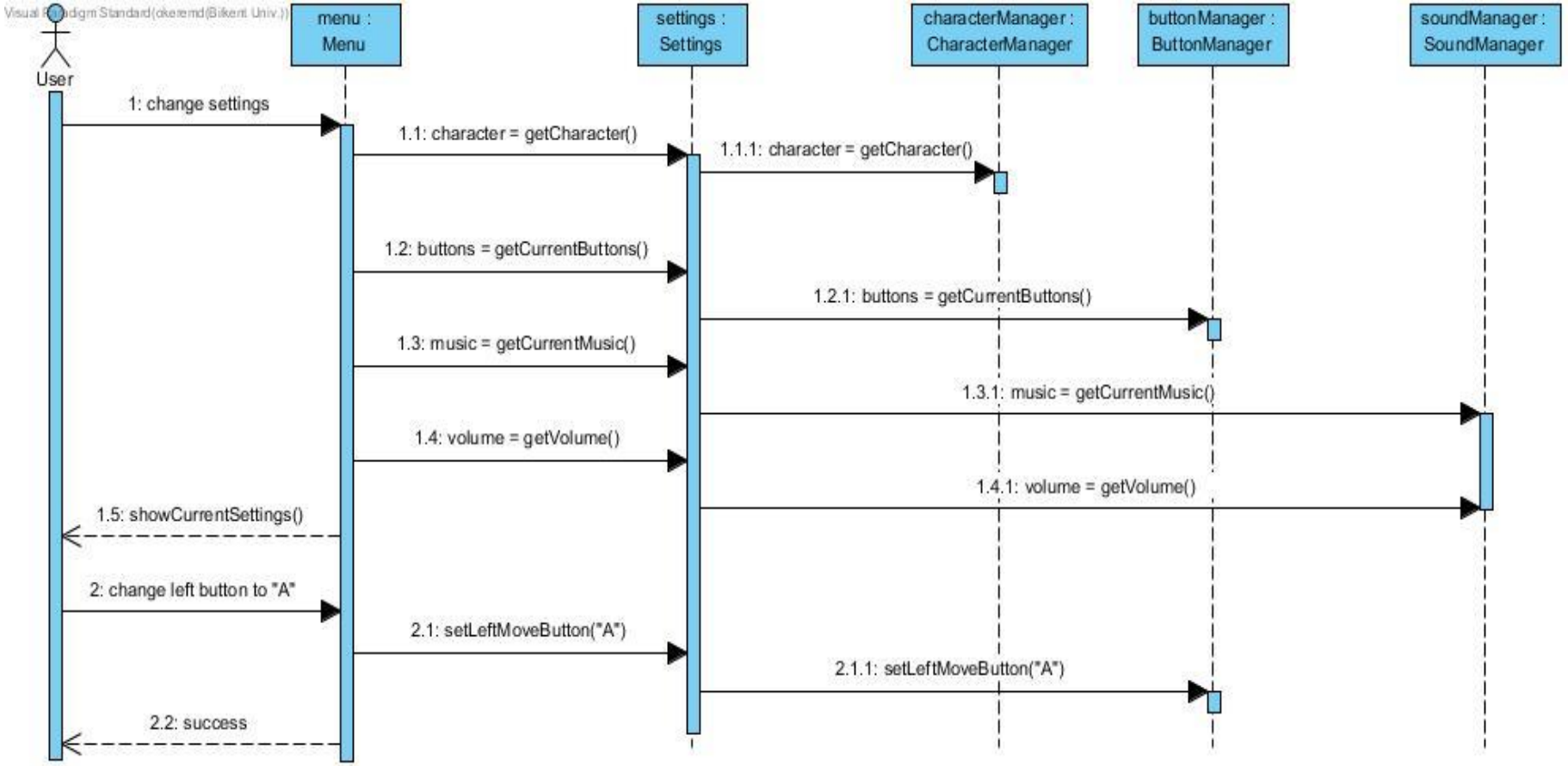
User runs the executable icy tower. S/he encounters with the main menu options and chooses “Start Game” option. Menu presents 3 options to start game which are difficulty options “Easy, Medium and Hard”. Player selects the “Medium” difficulty. Menu gets the Buttons to be used in game, Music that will be played in game and Character that will be play the game from Settings class. Settings class uses Button Manager, Sound Manager and Character Manager to retrieve these features. Then, menu initializes the Game Manager with level, sounds, buttons and character images. Game Manager makes Input Manager start with the provided buttons, makes the Map Manager start with the provided level and character. Map Manager initializes the camera depending on the level of the game. The harder the game, the faster will be camera move up. Map Manager also initializes the Map depending on the level. Map Manager passes the character images to Map, which will pass them to the Character class. Map initializes the start environment of the game and game starts.


User
menu : Menu
settings : Settings
buttonManager : ButtonManager
characterManager : CharacterManager
gameManager : GameManager
inputManager : InputManager
mapManager : MapManager
map : Map
character : Character
gameObject : GameObject
1: Start Game
1.1: level = Choose Difficulty
2: Medium (2)
2.1: buttons = getCurrentButtons()
2.1.1: buttons = getCurrentButtons()
2.2: charImages = getCurrentCharacter()
2.2.1: charImages = getCurrentCharacter()
2.3: GameManager(level, buttons, charImages)
2.3.1: InputManager(buttons)
2.3.2: MapManager(level, charImages)
2.3.2.1: Map(level, charImages)
2.3.2.1.1: setImages(charImages)
2.3.2.1.2: InitializeObjects(level)

5.2.1.2 Settings

Scenario: User wants to change a setting

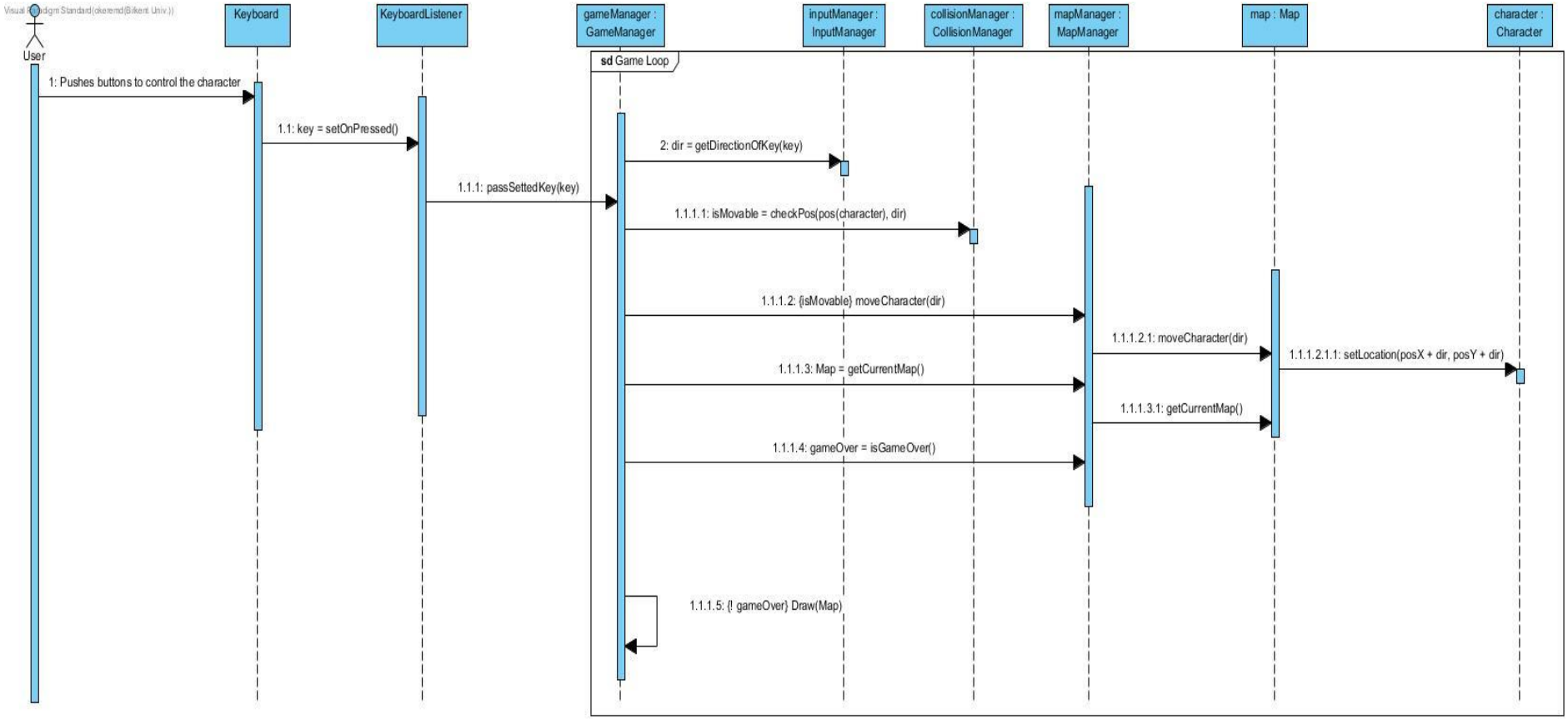
At the menu screen, user chooses settings. Settings screen will provide current settings to Menu. Settings class get current buttons from Button Manager, current music and volume from Sound Manager and current character from Character Manager. Menu retrieves these information from those and present them to user. User wants to change the left button to “A”. He enters “A” to area provided by Menu. Then Menu passes that to Settings and Settings passes to Button Manager which will eventually change the in game setting to control the character.


User
menu : Menu
settings : Settings
characterManager : CharacterManager
buttonManager : ButtonManager
soundManager : SoundManager
1: change settings
1.1: character = getCharacter()
1.1.1: character = getCharacter()
1.2: buttons = getCurrentButtons()
1.2.1: buttons = getCurrentButtons()
1.3: music = getCurrentMusic()
1.3.1: music = getCurrentMusic()
1.4: volume = getVolume()
1.4.1: volume = getVolume()
1.5: showCurrentSettings()
2: change left button to "A"
2.1: setLeftMoveButton("A")
2.1.1: setLeftMoveButton("A")
2.2: success

5.2.1.3 In Game Logic

Scenario: User wants to control the character with keyboard.

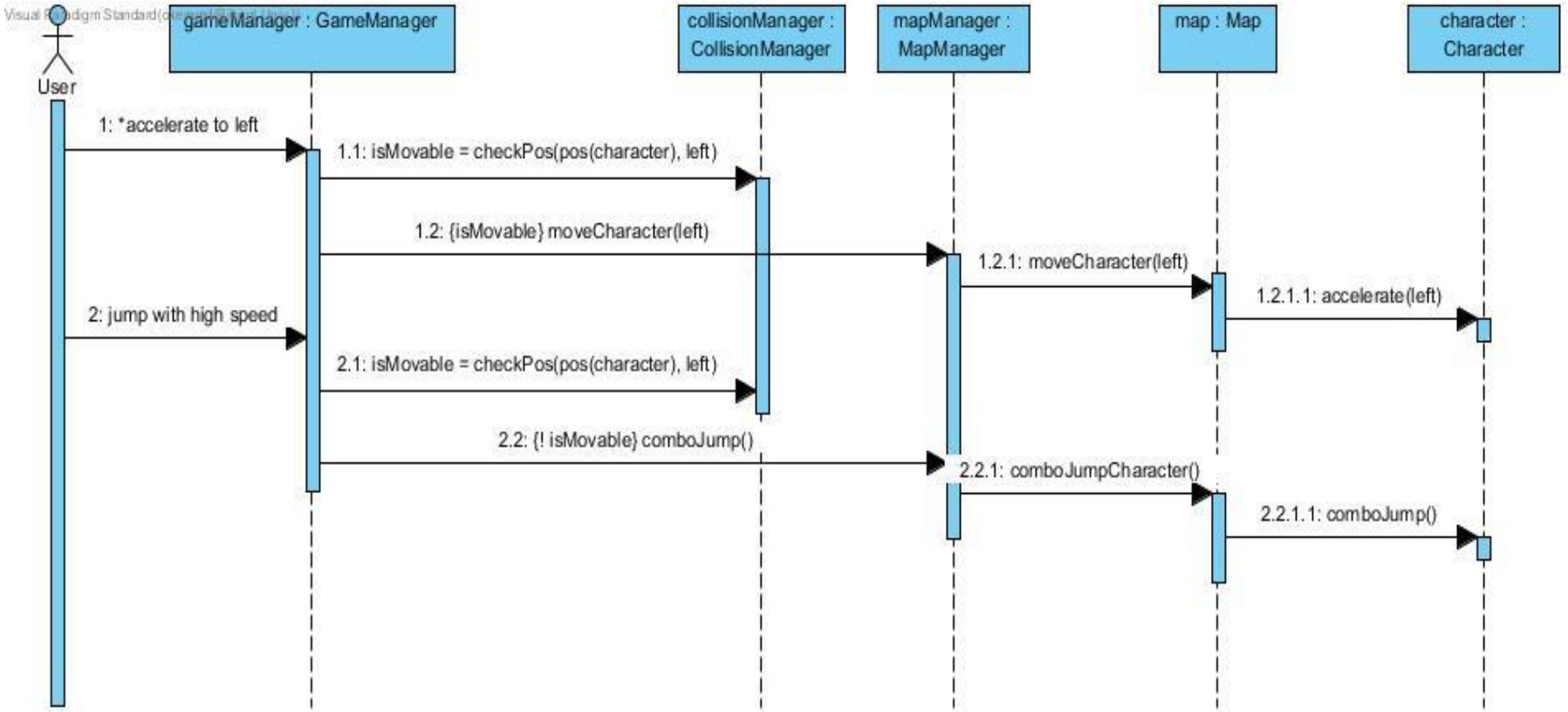
User presses the keyboard buttons. The Java Interface <<KeyboardListener>> gets the characters and passes them to Input Manager Class. Game Manager gets the meaning of the buttons for game from the Input Manager and checks if that movement is legal in game. If it is, passes the movement to Map Manager. Map Manager passes the movements to Map and Map updates the location of the Character then returns the current map to Map Manager. Map Manager updates the game situation by looking whether the character is fall down. If game is not over, Game Manager starts from the beginning of the scenario.


Visual Paradigm Standard(okeremd(Bilkent Univ.))
User
Keyboard
KeyboardListener
gameManager : GameManager
inputManager : InputManager
collisionManager : CollisionManager
mapManager : MapManager
map : Map
character : Character
sd Game Loop
1: Pushes buttons to control the character
1.1: key = setOnPressed()
1.1.1: passSettedKey(key)
2: dir = getDirectionOfKey(key)
1.1.1.1: isMovable = checkPos(pos(character), dir)
1.1.1.2: {isMovable} moveCharacter(dir)
1.1.1.2.1: moveCharacter(dir)
1.1.1.2.1.1: setLocation(posX + dir, posY + dir)
1.1.1.3: Map = getCurrentMap()
1.1.1.3.1: getCurrentMap()
1.1.1.4: gameOver = isGameOver()
1.1.1.5: {! gameOver} Draw(Map)

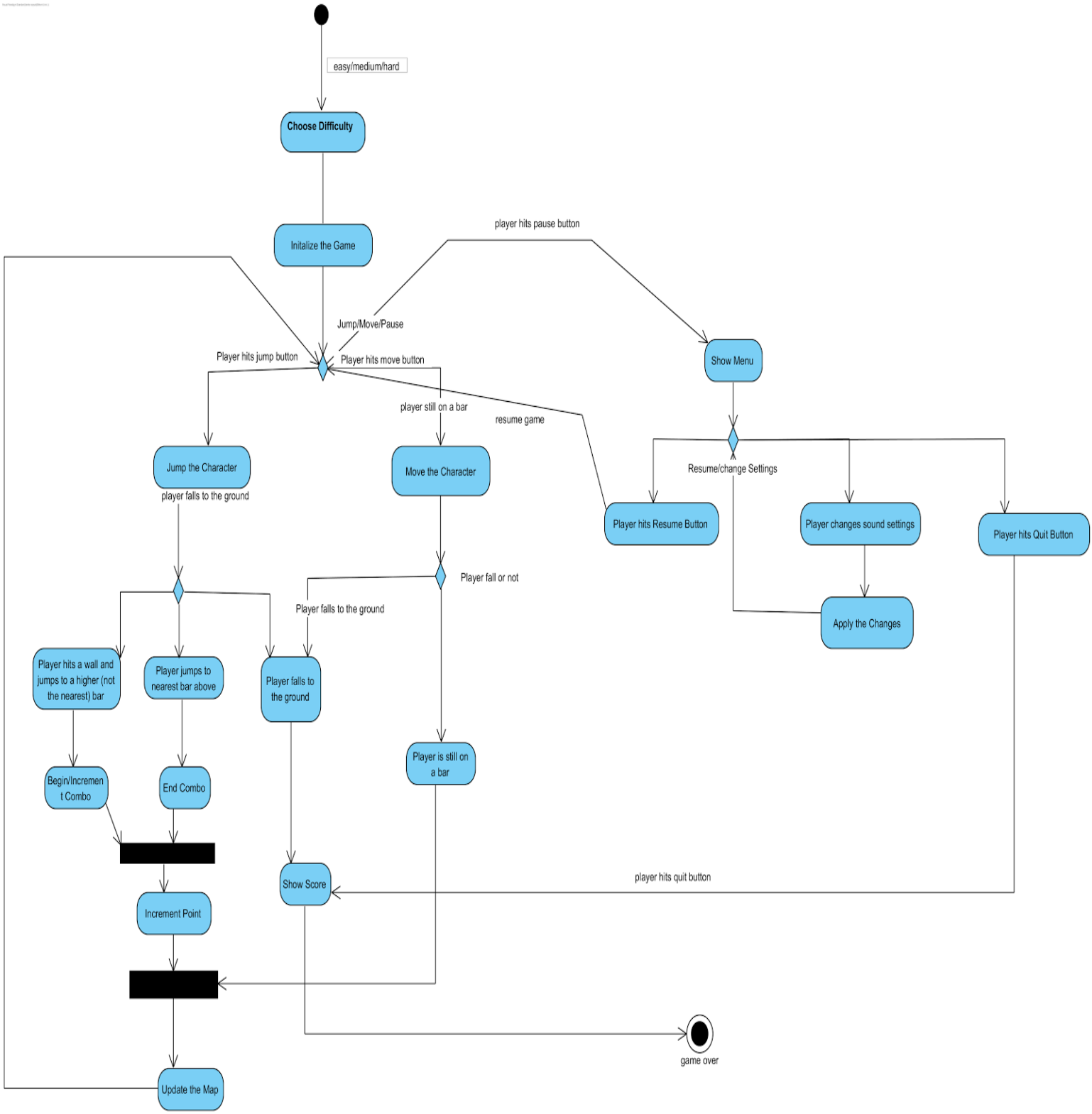
5.2.1.4 Combo Jump

Scenario: User wants to make combo jump

User holds the left button. CollisionManager checks whether user hits the wall. Since it does not, Game Manager passes the accelerate message to Map Manager. Map Manager passes to Map and eventually Map passes to Character. Thus, the character accelerates to the left. While accelerating is continuing, player presses the jump button before hitting the wall. CollisionManager checks if there is a wall which is an unmovable object. Player hits the wall. Then Game Manager passes the accelerate toward an unmovable object message to Map Manager. Map Manager passes to Map and Map passes to Character. Thus, this jump becomes combo jump which makes the progress easier for the user.


User
gameManager : GameManager
collisionManager : CollisionManager
mapManager : MapManager
map : Map
character : Character
1: *accelerate to left
1.1: isMovable = checkPos(pos(character), left)
1.2: {isMovable} moveCharacter(left)
1.2.1: moveCharacter(left)
1.2.1.1: accelerate(left)
2: jump with high speed
2.1: isMovable = checkPos(pos(character), left)
2.2: {! isMovable} comboJump()
2.2.1: comboJumpCharacter()
2.2.1.1: comboJump()

5.2.2 Activity Diagram

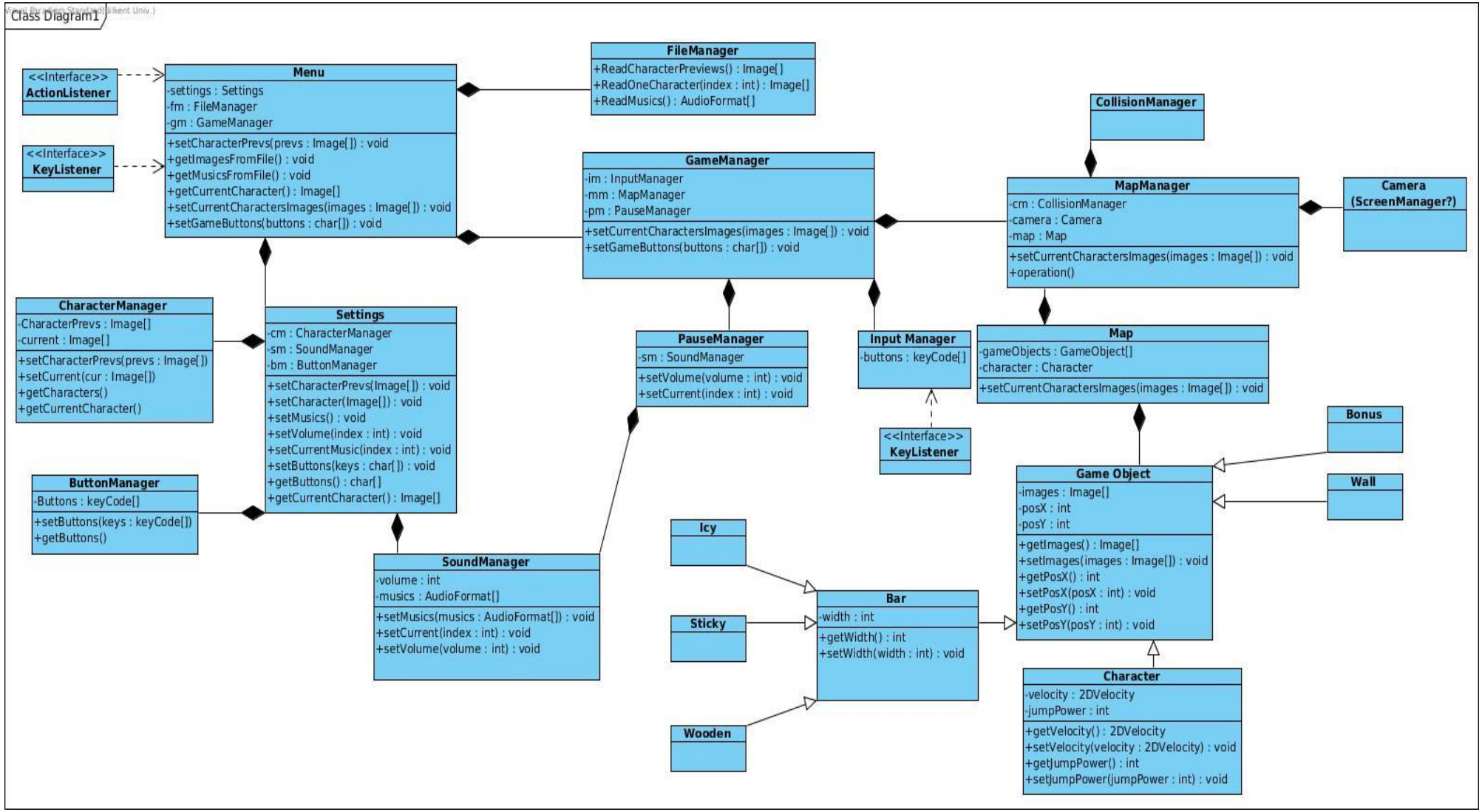


This diagram illustrates the aspects of “Play Game” use case. First the user has to select a difficulty play among three levels, easy, medium and hard. Then the game initializes, map and game objects are created and displayed. User can jump over bars, move on bars, or can pause the game. If he chooses to jump through the bars, the game checks the jump he made a combo jump, (**Combo Jump:** Player jumps to wall and then jumps to a “not the nearest bar”.) or he just jump to the current wall, or the nearest above wall. If it’s a combo jump, the game should begin the combo or increment it; otherwise it should end the combo. Then the point should be incremented and the map should be updated to display the game objects’ current positions. If the player drops to the ground, then the game should terminate and the score should be displayed.

If user chooses to move the player, the game should check if the player is still on a bar or not. Note that player can drop to a bar on the below, he can still continue the game. But if he didn’t fall to a below bar but to the ground, then the game should terminate, and the score is being displayed.

If user chooses to pause the game, then s/he can quit the game, change the settings, or resume to the game. Only settings the player can change on in-play settings is audio. If the user is quit the game, the game will show the score he made and then he can quit the application.

5.3 Object and Class Model



The class diagram of Icy Tower is shown above. Currently there will be at least 20 classes, as we can decide to increase the number of bar types, the number of classes would increase as well.

Menu

Menu will be the executable class that starts the all game. The main options such that playing the game, changing settings, looking to credits and how to play will be shown in the menu. Since it will be the main class of our project, the functional classes (managers) will have instances in the menu.

Settings

Settings class holds CharacterManager, ButtonManager and SoundManager to show in the settings page. User should be able to change the preferred buttons to play the game, change the music and volume and also can choose the character which s/he wants to play.

CharacterManager

CharacterManager class will show the preview of the selectable characters. User can choose one of them to play the game. There will be a default character that prevents to force the user to choose one of the characters.

ButtonManager

ButtonManager class will give the opportunity to the user to define her own buttons to play the game. There will be 3 buttons (left, right and jump) and ButtonManager will hold each of them individually. By default, left-reight arrows and space button are arranged.

SoundManager

SoundManager class will have methods to deal with the music and game sounds. User should be able to determine the volume of the music and game sounds, and can change the music as well. SoundManager will provide the features. SoundManager will be accessible from the Pause Menu as well.

FileManager

FileManager class will have methods to retrieve information from files. Since there will be musics and game sounds and images, a file manager will make the dealing with files easier for other manager classes. Additionally, the high scores should be maintained in a file and FileManager will know the proper file names and proper encoding/decodings in the files.

GameManager

Once the “Start Game” is selected from main menu, GameManager will start working since game ends. It will pass the proper buttons to InputManager, creates the MapManager and detects if there is a pause.

InputManager

InputManager class will detect the inputs from keyboard and signals the GameManager if the inputs are recognizable by the ButtonManager.

PauseManager

PauseManager will give user to change sound settings while stops the game.

MapManager

MapManager will have the CollisionManager, Camera and Map as subclasses. MapManager will integrate these 3 game logic related classes.

CollisionManager

CollisionManager will detect the interactions between map objects and the character.

Camera

Camera will be a factor to decide whether the game is over. If character is fall down from camera, the game is over. Also the camera will follow the character if character goes faster than current levels camera speed.

Map

Map will generate the GameObjects depends on the level and difficulty of the game. Higher levels and harder difficulties will cause harder map generations (with shorter or different types of bars).

GameObject

GameObject will be the abstract class that contains information every game object should contain such that x and y positions, images etc.

Bar

Bar will be an abstract class that holds the information that every bar should contain such that 2 images (ends and middle), length and height of bar.

StickyBar

StickyBar will be a type of bar that slows down the character.

IcyBar

IcyBar will be a type of bar that causes character to slip and accelerate.

WoodBar

WoodBar will be a type that does not change the speed of character.

Character

Character will be the object that user controls. It will have several images to create vision of motion. It will have speed and jump constants that determine the distance.

Wall

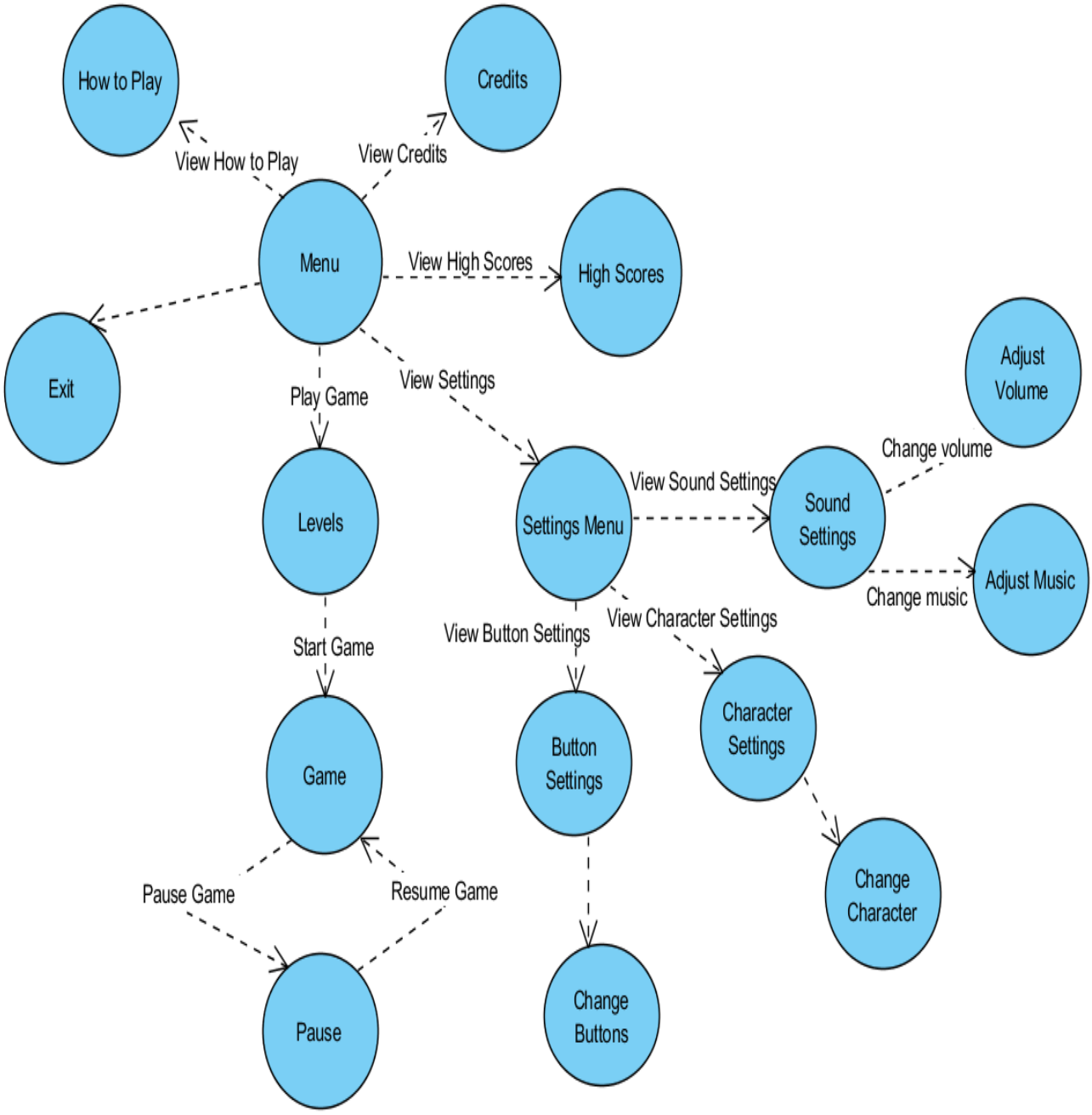
Wall will be the left and right ends of the map.

Bonus

Bonus will be the collectible that causes to player to boost its speed for a certain period of time.

5.4 User Interface- Navigational Path and Screen Mock-ups

5.4.1 Navigational Path

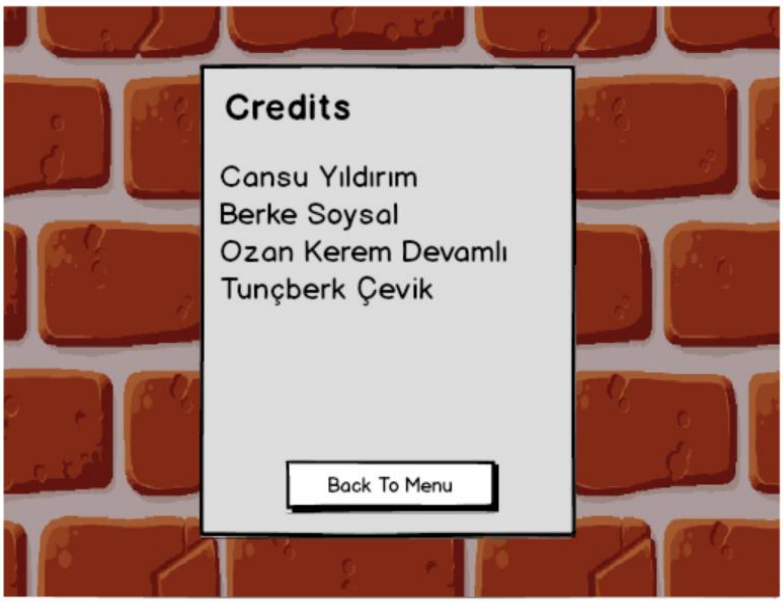


5.4.2 Mock-ups

Main Menu

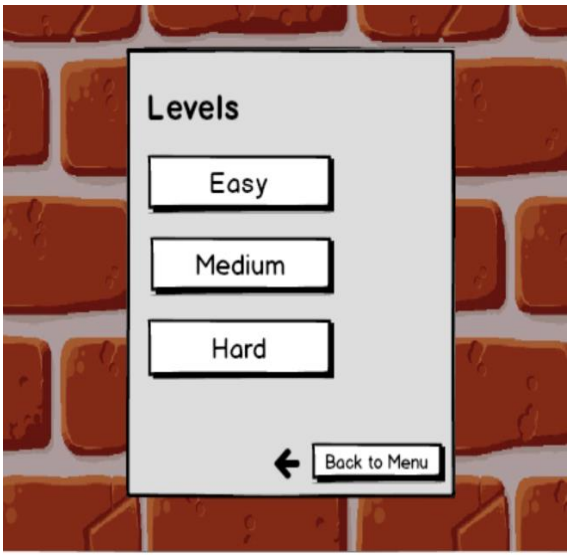
In the main menu there are 6 buttons. Play Game, High Scores, How to Play, Settings, Credits and Exit buttons. By clicking each of the buttons, relative pages open. When the play game button is clicked, icy tower will start.

Credits



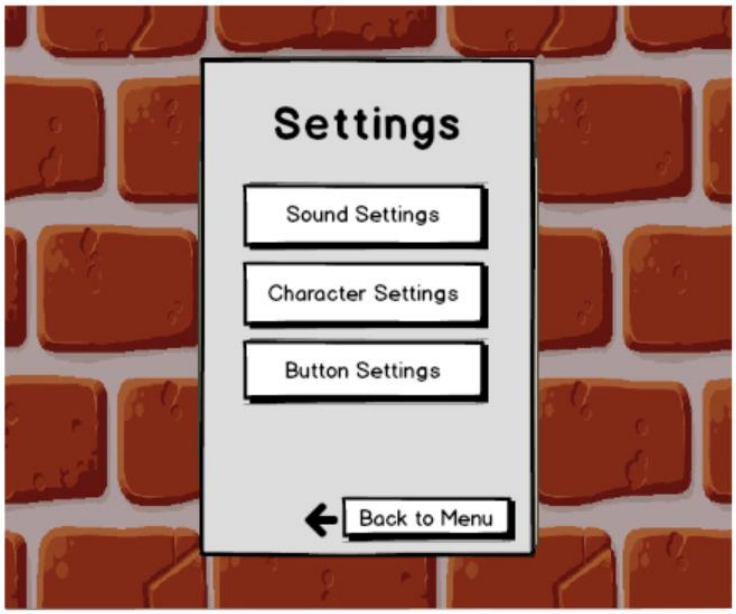
Credits screen shows the developers of the game.

Play Game

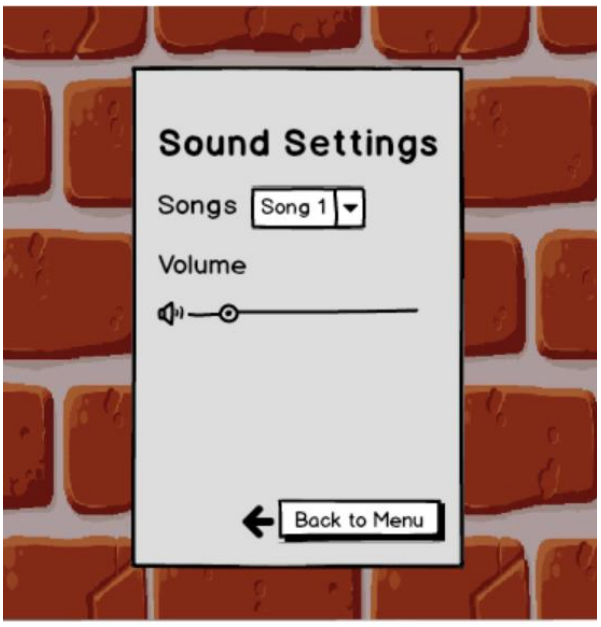


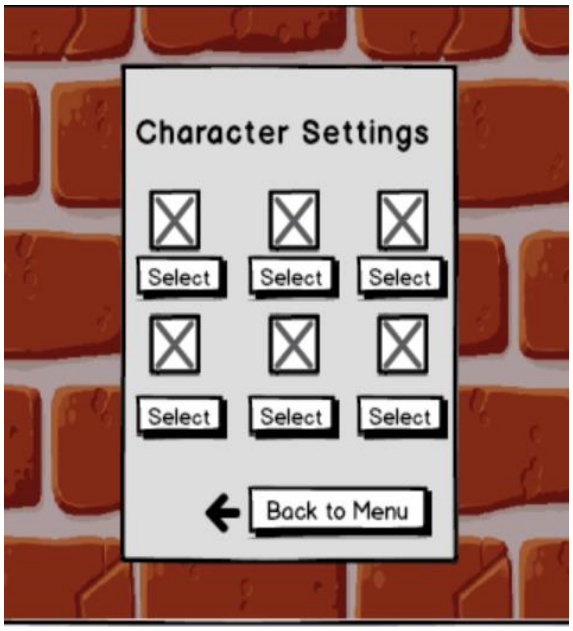
When the player selects “Play Game” button on the main menu, the difficulty level will be asked. After choosing the level, start game page will be seen. Player must click this “Start Game” button. Or player can go back the main menu.

Settings



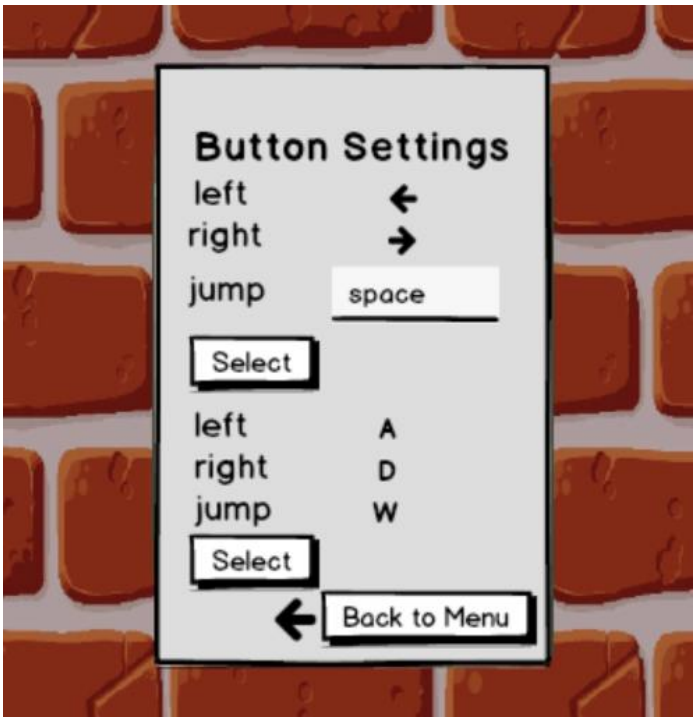
In the settings screen, there are 3 buttons for sound setting, character setting and button setting. Player can click these buttons to change the relevant settings. Or player can go back to the main menu by clicking the Back to Menu button.





In the Sound Settings screen, player can change the music that is played during the game. Also, player can arrange the volume of the sound.

In the Character Settings screen, player can select the character. The characters appearances will be different. Player must click the button under the chosen character.



In the Button Settings screen, player can choose the control buttons that will be used during the game. If player wants to control the character with right-left arrow and space, s/he must click the upper button. If s/he wants to control with A,D,W, s/he must click the below button.

To go back the main menu, player must select the “Back to Menu” button on the bottom of the screen.

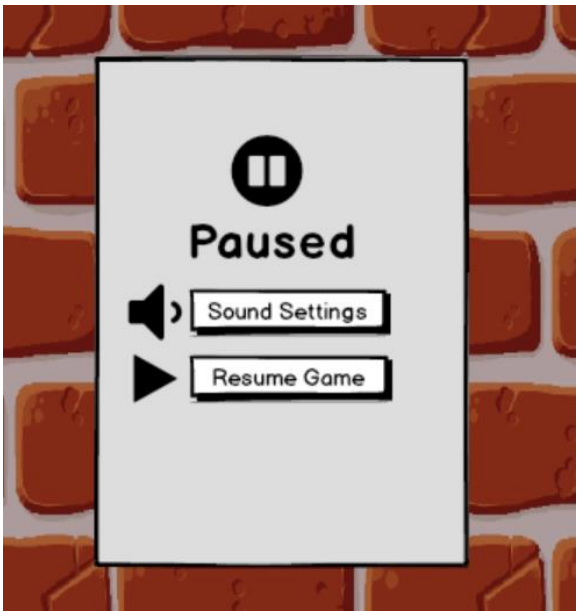
Exit

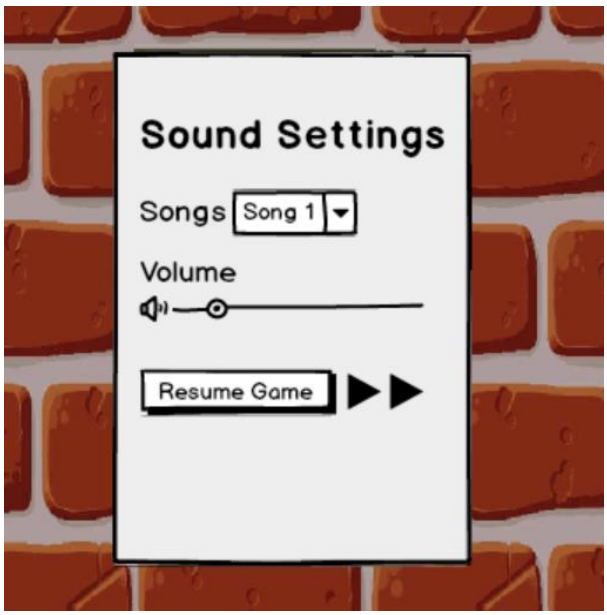
Exit screen displays the question in order to be sure about the player’s decision. If the player is sure about quitting the game, s/he must click the “Yes” button, otherwise “No” button to go back the main menu.

High Scores

High Scores screen shows the scores of the players. When a player gets higher score than the highest score in the list, a name will be asked. With this name, the score will take its place in the list. To turn the main menu, player must select the Back to Menu button.

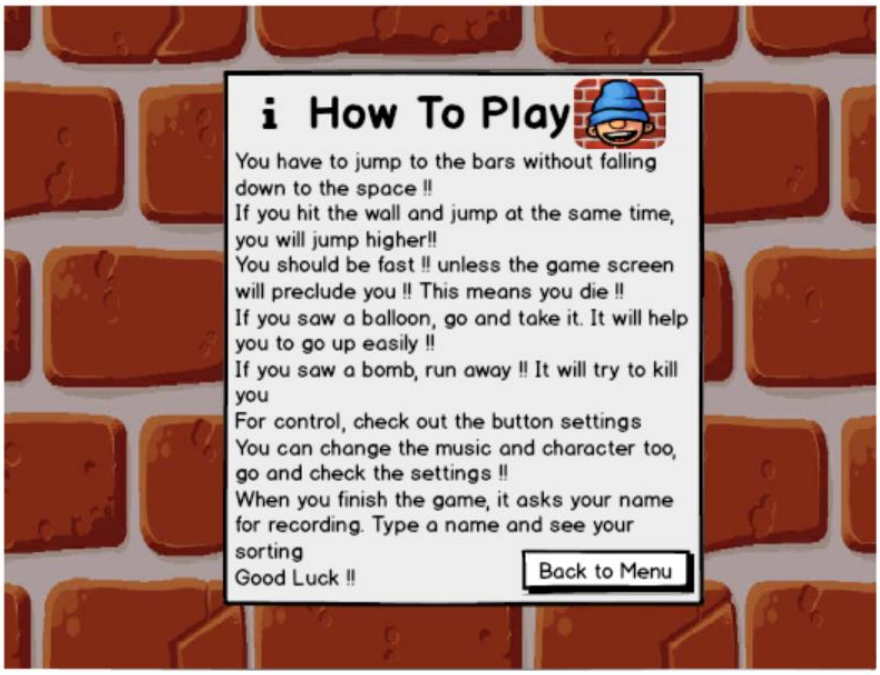
Pause Menu and Sound Setting





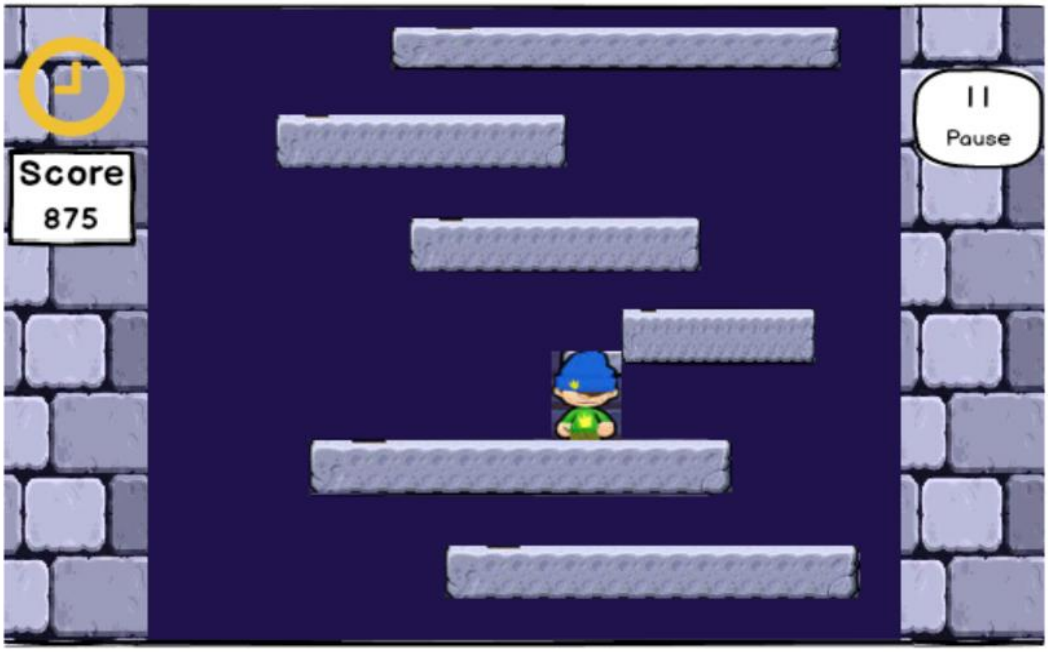
To open the pause menu, player must be playing the game. If the player selects the pause button on the game screen, pause menu will be opened. From this menu, player can go to the sound settings and adjust the sound. On the sound setting menu, player must click the resume game button to resume the game. Or player can click the resume game button from the pause menu, if s/he did not go to the sound settings.

How to Play

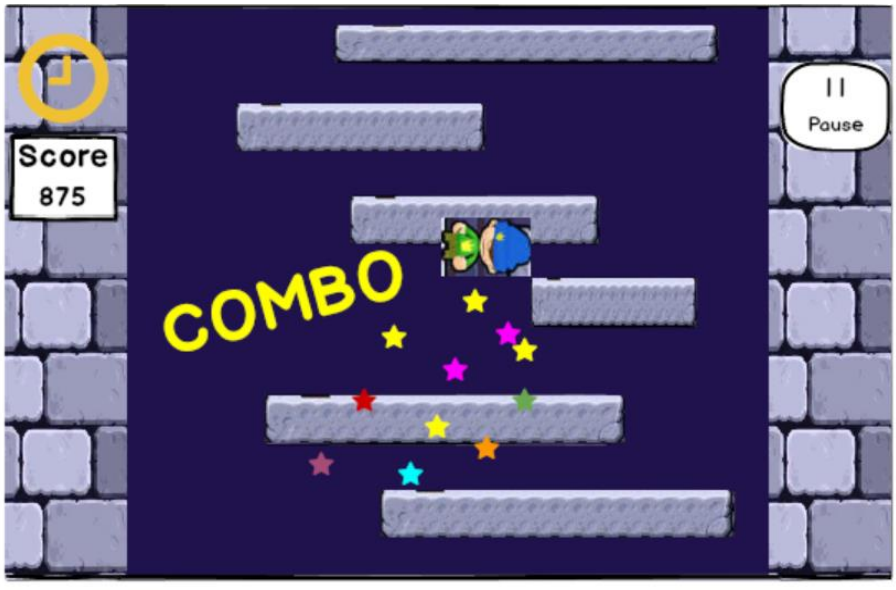


How to play screen displays the logic of the game and general information about the game. By clicking the button “Back to Menu”, player can go back to the main menu.

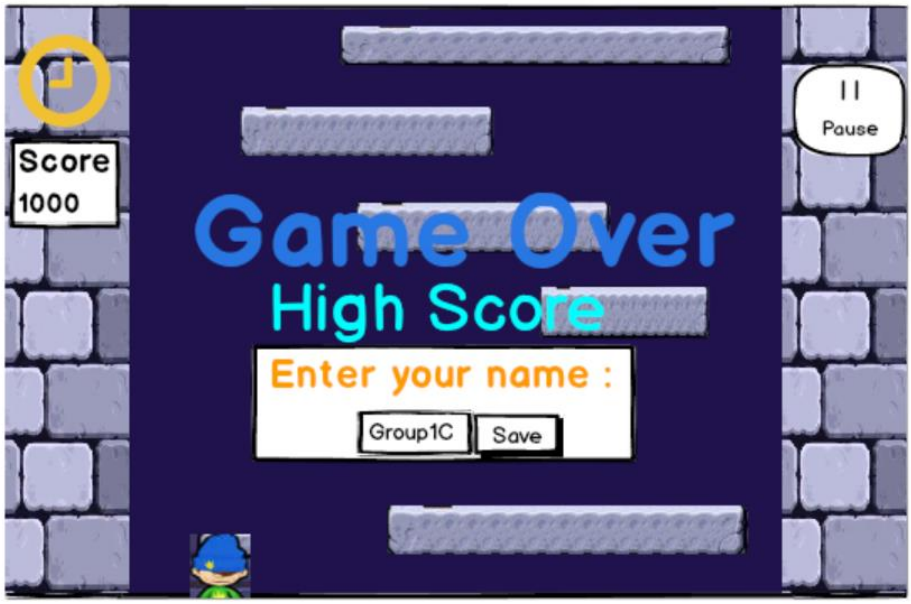
Game Screen



The game screen of our game will be like this. There will be bars that character can move above them. There will be pause button to pause the game and open the pause menu. The current score of the game can be seen while the game continues. Also, there will be a clock to calculate when the game screen increases its moving speed.



When player makes the combo jump, character will go upward spirally. “Combo” writing and colourful stars will appear on the screen in order to urge the player about making more combos.



“Game Over” writing will appear in the pursuit of the character’s fall. If the character makes new high score as explained more detailed above, player will be asked to type a name. After typing a name, player must click the “Save” button to have a place in the high scores list. After the name is saved, game returns to the main menu.

6. Glossary & References

http://gaming.wikia.com/wiki/Icy_Tower