Bilkent University

Department of Computer Engineering

CS319 - OBJECT ORIENTED SOFTWARE ENGINEERING SPRING 2015 – 2016 FALL

Break Bricks
Analysis Report

Group 14 | Section 01
Berire Gündüz
Kemal Büyükkaya
Kaan Akıncı
Barış Çelik

28.10.2015

Analysis Report

1. Introduction

Break Bricks is actually very well-known arcade game that has a many different versions. In our project we will develop our own version of this game which enriched by many power-ups, penalties and different kinds of bricks to break. For the implementation of the game Java, which is an object-oriented language, will be used.

The purpose of this game, like many other versions of it, will be to break as many bricks as possible in restrictive conditions. In our version these restrictive conditions are limited time and limited amount of lives that a player can have through a game. The game will be a desktop application and will be controlled by left and right buttons on the keyboard.

As many different objects such as 30 different bricks and their relationship among each other requires deep awareness about the course's main topic, we thought this game can be a great tool to understand and apply what we learned in this course. We aim to design a consistent game that abides by OOP fundamentals which properly models the real-life objects, interactions and events.

In this report two main categories about project analysis are included, requirement analysis and system analysis.

2. Requirement Analysis

2.1. Overview

Break Bricks is a ball-and-paddle arcade game. Like its many versions, it is quite easy to play and fun at the same time. Basically there 30 different kinds of bricks and the player tries to destroy each brick by sending a ball through them by the help of the paddle at the bottom. Every hit from the ball either destroys a brick or just crack it depending on the type of the brick or/and the ball. To complete the game the player should destroy all the bricks, in limited time and without running out of lives that given at the start of the game. At the end of the each game the point that player had won will be added to the scores list so that the list can be used to detect the highest score ever received.

2.1.1. Gameplay

Gameplay is aimed to be easy as possible because of that a player can simply control the paddle on the play screen by just two buttons, left button and right button, on the keyboard. The aim of the player should be steer the paddle, to make paddle go left player have to use left button and to make it go right player have to use right button, so that the ball bounces off from the paddle towards a selected brick. The ball can bounce off from the walls as well

however player should be careful about not to dropping the ball below the paddle. In case of missing the ball the player will be losing a life.

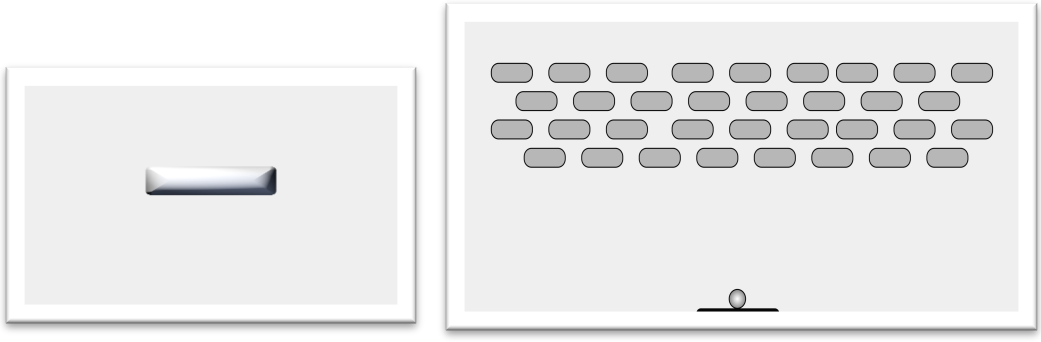
2.1.2 Challenge of the Game

Although we didn't design different levels for this game there is an indirect challenge to make a player keep wanting to play. That is the list of scores that gained through other rounds of the game. By showing the high score in the high scores menu which can be accessed by clicking the "High Scores" button at main menu, we aimed to create feeling of competition for players. As the score that gained through a game depends on power-ups that gained by luck, as the power-ups randomly distributed among bricks, and the amount of time plus life that is left after game is over, in each game player has a chance to have better score than ever.

2.1.3. Ball & Paddle

The paddle is the only tool that the player can control; so the size and usage of it are important to take control over the game. While a short kind of paddle would lose the ball to the ground, a long kind of paddle might have to catch penalties. Also the player might catch some power-ups that effect the paddle to destroy the bricks by its own. The size and the usage of the ball is also important. While a small kind of ball would make you cannot see the ball, a ball with increased strength might destroy more than one brick at the same time.

2.1.4. Brick Types



The arena basically consists of 30 bricks that consist of 3 kinds of bricks, like glass brick that can be broken by 1 hit, normal brick that can be broken by two hit and steel brick

that can be broken by 3 hit. In addition to the breakability of bricks there will be other properties that will be randomly distributed among bricks which are penalties and power-ups.

Glass brick: Takes one hit to break it.

Normal brick: Takes two hits to break it.

Steel brick: Takes three hits to break it.

2.1.5. Power-ups and Penalties

The power-ups and penalties are the essential part of the game to make it so fun. The player will never know if there is a coming power-up or a penalty when the ball hits a brick. Also the player cannot know exactly what type of power-up or penalty coming. Some of them would be rare and not be coming as much as the other ones and that makes this kind of power-ups interesting. The power-ups and the penalties are:

Faster paddle: Makes paddle move faster.

Slower paddle: Makes paddle move slower.

Extra life: It gives one extra life to the player.

Super ball: Increases the power of the ball so that it can break bricks easily.

Weak ball: Decreases the power of the ball so that it takes more hit then usual to break a brick.

PointsX2: One that multiplies the score with 2 per brick for a limited time

Extra Time: Gives additional time to player.

Time Penalty: Shortens the time limit.

Instant death: It takes one life from the player.

Random power-up: It gives a random power-up/down.

Rebel paddle: Paddle goes to opposite direction, when you move it to right or left.

2.1.6. Settings

The settings of the game can be changed in the Options Menu which can be accessed by clicking on “Options” button at the main menu. The User can change the volume, the bar and the puck’s color and texture from one of the pre-determined one’s. Any changes that are saved by the user will be stored until user change the options again.

2.1.7. Victory Conditions

In order to win the game, the user has to break all the bricks in a limited time without running out of lives. To get the highest score in the game, a player should provide victory conditions in shorter amount of time than other trials. In addition to time element if a player was able to gain more power-ups than other trials, which depends on how lucky the player is, then again the player can have the highest score among other scores.

2.2. Functional Requirements

Break Bricks is a Java client-based, arcade game with one player. Break bricks is a very well-known game that player uses a paddle to break bricks and gains points depending on the total number of the bricks that have been broken.

Users can launch the game using the desktop shortcut to the executable. After launching, users can choose to go to options menu, high-scores screen, help screen, credits screen and play screen.

If the user clicks on the “Options” button, the options menu will be opened and user will be able to control sound effects by on and off choices, customize the paddle or ball colors depending to their own wish. Then user will be able to go back to the main menu by clicking “Back” button.

If the user clicks on the “High Scores” button, the high scores menu that displays the list of high scores and the highest score ever received, will be opened. Again player will be able to go back to main menu by “Back” button.

As by clicking “Help” button a screen which gives detailed information about gameplay will be displayed by clicking “Credits” button a screen that displays information about our team will be shown. In both case user will be able to go back to main menu by clicking “Back” button.

After user starts the game by pressing “Play” button, system prepares the arena for the user. The arena basically consists of 30 bricks that consist of 3 kinds of bricks, like glass brick that can be broken by 1 hit, normal brick that can be broken by two hit and steel brick that can be broken by 3 hit. In addition to the breakability of bricks there will be other properties that will be randomly distributed among bricks such as power ups (one that multiplies the score with 2 per brick for a limited time, one that speeds up the paddle and the one that increases the strength of the ball etc.) and penalties (one that slows down the paddle etc.).

Users will have a total Score point, which is a sum of all the points that user wins through the game. This total score will be shown on the top right of the screen until the game is over. In our version of the game player starts with 3 lives and there is also a time restriction (five minutes). The amount of lives and time left will be shown in play screen too. By pressing ESC user will be able to pause the game and from the pause screen user can choose whether to continue to game, to go back to main menu or to restart the game.

During the game play there is three conditions to game to end. One is when the player press the ESC and exits from the game, other is when player is run out of time and the last one is

when the player run out of lives. When player cannot hit the ball with paddle and causes the ball to fall under the paddle, he/she will lost a life. So the user's goal is to block the ball and give it the appropriate direction to break some bricks. While playing the game, the user will use the arrow keys to move their paddle right and left. User should not be able to move their paddle up and down for the sake of simplicity. The user can see their in-game score on the screen. Total score will be updated after every time when a brick is broken.
When the game is over user will be shown a new screen that displays Total Score and rank among the other high scores. This screen will also enable user to go back to main menu or to restart the game by appropriate buttons.

2.3. Nonfunctional requirements

2.3.1. Usability

• As break brick game has a lots of different versions user will be easily accustomed to our version of the game.
• Any user should be able to start a new game with less than 3 mouse clicks.
• User can change volume, paddle and ball color, all at once, via Options screen.
• Gameplay is sufficiently straightforward and very simple so that users can get used to it quickly.
• User can learn about any feature of the game from Help screen.

2.3.2. Reliability

• If system crashes in the middle of a task, which means during a game, no prior progress is lost due to this inconvenience.
• When a game is completed without any failure, system saves the score to the list of high scores and sorts the list in decreasing order so that the score at the top of the list will be shown as the highest score.

2.3.3. Performance

• System should respond to user's command no longer than 50 milliseconds.
• Graphical computations should be executed with an onboard GPU.
• System should run properly in any display supporting a 1100x720 resolution.

2.3.4. Supportability

• The system will be able to be developed, such new attributes and functionalities can be added without changing the core structure of the system. Changes imposed by new technologies and bug fixes will be easy to implement.

2.4. Constraints

2.4.1. Implementation

• The system must be implemented in Java because it is a widely used programming language that is perfectly suitable for object-oriented software implementation which is needed for this game which includes a lot of objects and interactions among them.

• Some of the graphic objects will be designed using Adobe® Photoshop CS4.

• Any system that has Java libraries installed in itself will be able to run the game executable.

• The system must be implemented in IntelliJ IDEA since it has functionalities that ease the coding process. In addition to that for the source code management GitHub will be used.

2.4.2. Interface

• Game graphics will be designed in order to be appealing

• User will be able to play game easily on the screen. The components of the game such as paddle, ball and different kinds of bricks etc. will be easy to distinguish and well designed. In addition to that, because the game will be played on the screen by using keyboard, the graphical user interface components such as buttons will be easily clickable and recognizable.

2.4.3. Legal

• The project should be licensed under Apache 2.0 open-source license, which is one of the most widely used license by open-source projects.

2.5. Scenarios

- **Scenario Name:** PlayerHitsBall

Participating Actors: Kemal: Player

Flow of Events:

- Kemal wants to save the ball in order to not to lose.
- Kemal presses the movement keys to move the paddle to the direction of the ball.
- Program responds and moves the paddle as long as he presses.
- The ball and paddle collides.
- When they collide program detects this collision and changes the direction of the ball by 90 ^{o}.
- Ball moves away with the calculation of the collision and program shows the result of this collision.

- **Scenario Name:** BreakBrick

Participating Actors: Kemal: Player

Flow of Events:

- Kemal hits the ball with his paddle.
- After Kemal hits the ball with paddle, with certain amount of time passed, ball collides with a brick object.
- Program catches the collision and makes the ball bounce off, similar to paddle-ball collision.
- When the calculations are done the brick disappears and number of bricks is decreased.
- Ball moves to its new direction and his score is updated.

- **Scenario Name:** DropPowerUp

Participating Actors: Kemal: Player

Flow of Events:

- When the ball that Kemal hits breaks a brick, program rolls a number and if the number is a hit, then the destroyed brick creates a powerup object.
- This object moves directly down to the bottom threshold line.
- Kemal catches it before the powerup touches the threshold line, Kemal's paddle will possess that random powerup.
- Kemal can't catch the other powerup and it collides with bottom threshold line. The powerup is destroyed and Kemal can't benefit from its power.

- **Scenario Name:** LoseLife

Participating Actors: Kemal: Player

Flow of Events:

- Ball comes at Kemal's paddle.
- Kemal can't time the movement and his paddle misses the ball.
- Ball moves past the paddle line.
- Program catches the collision with ball and threshold line.
- Program deletes the ball and decrements the lives and score he has.

- **Scenario Name:** NoLivesLeft

Participating Actors: Kemal: Player

Flow of Events:

- Kemal misses the ball and lost a life.
- Program detects that Kemal has no lives left.
- Program stops the game and presents the game over message.
- Program asks for a high score name.
- Kemal sees his high score in the list, and then clicks OK.

- **Scenario Name:** TimeUp

Participating Actors: Kemal: Player

Flow of Events:

- Kemal tries to break all the bricks.
- But the timer goes to 00:00 and he can't finish the level in the given time.
- Program stops the game and defines Kemal as loser by default.
- His score is recorded and game over message is presented.

- **Scenario Name:** PauseGame

Participating Actors: Kemal: Player

Flow of Events:

- Kemal needs to do his project, so he needs to exit the game.
- He pushes the escape button and program detects this key.
- The program pauses the game loop until Kemal decides to click the resume game button or Exit to Menu button.
- Kemal presses Exit to Menu button and quits the game.

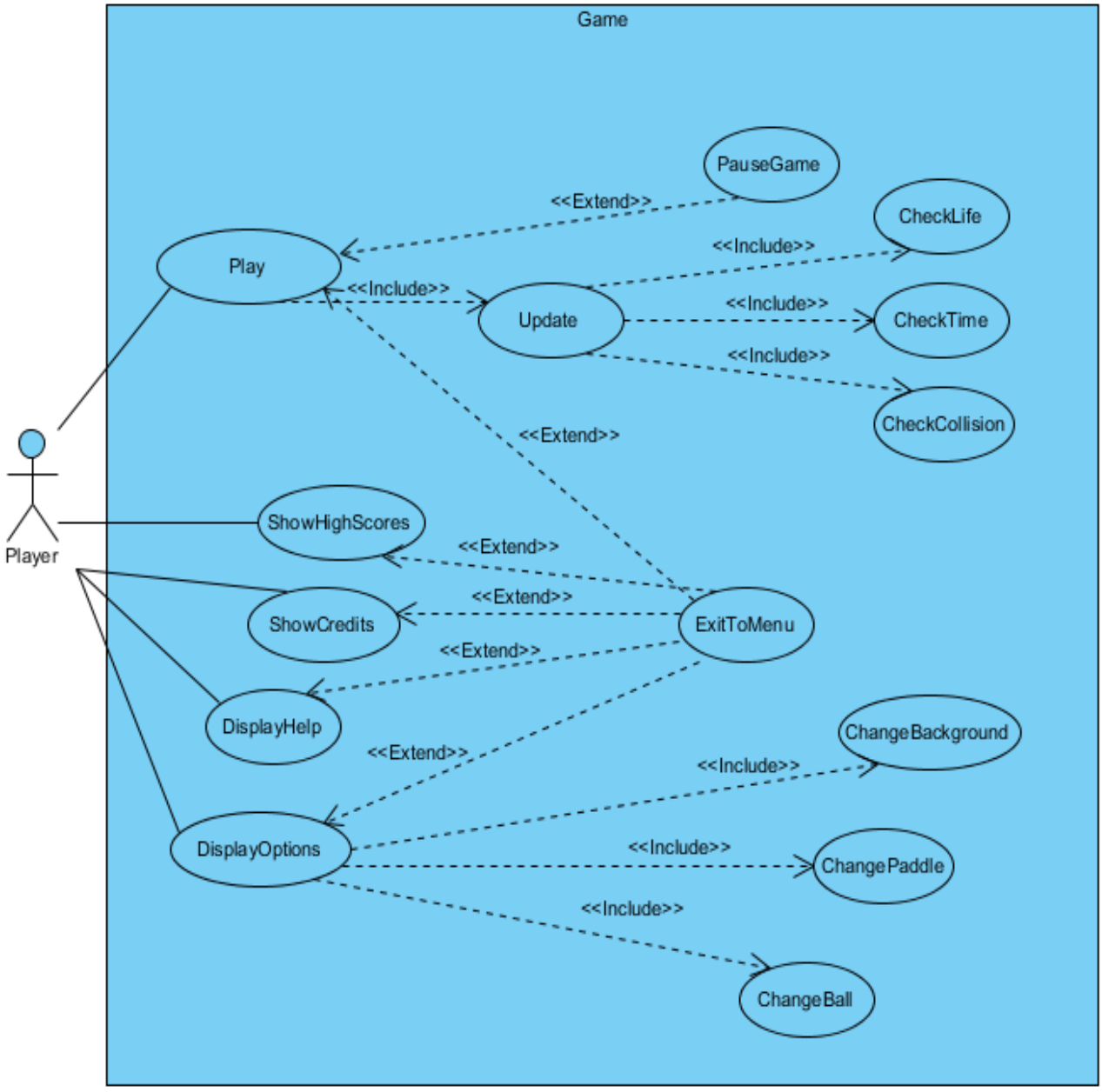
- **Scenario Name:** Customize

Participating Actors: Kemal: Player

Flow of Events:

- Kemal wants to change the appearances of the ball or paddle or background.
- He clicks the Options menu which opens the options and customization menu.
- In here he can turn the sound on or off and change the textures of the objects in game.
- After Kemal is done with customizations, he clicks the Back button which redirects him back to the previous menu.

2.6. Use Case Models



- **Scenario Name:** Play

Participating Actors: Initiated by Player

Flow of Events:

  - Program launches and the main menu appears.

- Player clicks Play.
  - Program opens the game at default settings. If the player changed the texture of the paddle or ball, they are set. Otherwise default textures are used. Brick objects are placed. Paddle and ball are placed. Score is set to 0 and Time is set for the according level.
  - Ball starts to move towards the player
- Player presses right or left arrow keys to navigate the paddle.
  - Program moves the player's paddle right or left and moves the ball. If ball collides with paddle or brick it bounces back. If a brick is broken program increases the score.
- Player breaks a brick with the ball and a powerup drops.
  - Program randomly generates a powerup object and starts moving it downwards. If the player's paddle collides with it, paddle will possess the attribute of the powerup. Else, program destroys that powerup.
- Player couldn't catch the ball, thus lost a life.
  - Program decreases the score of player and checks if player has any lives left. If player has lives left, ball's position is reset and paddle's position and powerups' are reset. Else program prompts game over message and asks for high score. Program stores the data in the txt file, sort it and open HighScoresMenu.
- Player destroys all the bricks.
  - Program checks if all the bricks are destroyed. After confirmation program prompts victory message and asks for high score. Program stores the data in the txt file, sort it and open HighScoresMenu.
- Player runs out of time.

    - Program detects that the time counter reached 00:00 and pauses game. Then prompts game over message and asks for high score. Program stores the data in the txt file, sort it and open HighScoresMenu.

Entry Condition: Player clicks Play.
Exit Condition: Player wins/loses/exits to menu.
Quality Requirements: The program's response time should be 0.1 second.

- **Scenario Name:** Update

Participating Actors: Initiated by Player
Flow of Events:

- Player clicks play and starts the game.
    - Program calculates ball's and paddle's next position, checks for collision and redraws the result in game loop
- Player ends the game.

Entry Condition: This use case is included in Play. It is initiated when the player clicks Play.
Exit Condition: Player pauses/loses/wins the game.
Quality Requirements: The program's response time should be 0.01 second.

- **Scenario Name:** CheckLife

Participating Actors: Initiated by Player
Flow of Events:

- Player is in game.
- Player misses the ball and loses a life
    - Program checks if the player has any lives left in game loop. If none left program prompts game over message and terminates.
- Player has no more lives left and loses the game.

Entry Condition: This use case is included in Update. It is initiated when the player clicks Play and Update is in use.
Exit Condition: Player pauses/loses/wins the game.
Quality Requirements: The program's response time should be 0.1 second.

- **Scenario Name:** CheckTime

Participating Actors: Initiated by Player

Flow of Events:

- Player is in game.
    - Program checks if the player has time left in game loop. If none left program prompts game over message and terminates.
- Player clicks ExitToMenu and returns to main menu.

Entry Condition: This use case is included in Update. It is initiated when the player clicks Play and Update is in use.

Exit Condition: Player pauses/loses/wins the game.

Quality Requirements: The program's response time should be 0.1 second.

- **Scenario Name:** CheckCollision

Participating Actors: Initiated by Player

Flow of Events:

- Player is in game.
    - Program checks if the ball hit the paddle or any bricks or bottom threshold line in game loop. If hit update the score and calculate next state of ball.
- Player exits the game.

Entry Condition: This use case is included in Update. It is initiated when the player clicks Play and Update is in use.

Exit Condition: Player pauses/loses/wins the game.

Quality Requirements: The program's response time should be 0.1 second.

- **Scenario Name:** PauseGame

Participating Actors: Initiated by Player

Flow of Events:

- Player is in game.
- Player presses the escape key.
    - Program gets out of game loop until player presses ResumeGame.
- Player presses ResumeGame.

    - Program gets back into the game loop and continues to update objects.

Entry Condition: This use case is extension of Play. It is initiated when the player clicks Play.
Exit Condition: Player clicks ResumeGame or ExitToMenu.
Quality Requirements: The program's response time should be 0.1 second.

- **Scenario Name:** ShowHighScores

Participating Actors: Initiated by Player
Flow of Events:
- Player finishes a game and enters high score.
- In main menu player clicks ShowHighScores.
    - Program opens up high scores and displays them to the player.
- Player clicks ExitToMenu.

Entry Condition: Player clicks ShowHighScores.
Exit Condition: Player clicks ExitToMenu.
Quality Requirements: The program's response time should be 0.1 second.

- **Scenario Name:** ShowCredits

Participating Actors: Initiated by Player
Flow of Events:
- Player launches the game.
    - Program greets the player with main menu.
- Player clicks on ShowCredits.
    - Program opens up the credits and displays the text.
- Player clicks ExitToMenu.

Entry Condition: Player clicks ShowCredits.
Exit Condition: Player clicks ExitToMenu.
Quality Requirements: The program's response time should be 0.1 second.

- **Scenario Name:** DisplayHelp

Participating Actors: Initiated by Player

Flow of Events:

- Player launches the game.
    - Program greets the player with main menu.
- Player clicks on DisplayHelp.
    - Program opens up the help window and displays the controls of the game.
- Player clicks ExitToMenu.

Entry Condition: Player clicks DisplayHelp.

Exit Condition: Player clicks ExitToMenu.

Quality Requirements: The program's response time should be 0.1 second.

- **Scenario Name:** DisplayOptions

Participating Actors: Initiated by Player

Flow of Events:

- Player launches the game.
    - Program greets the player with main menu.
- Player clicks on DisplayOptions.
    - Program opens up the options menu.
- Player changes some options and click ExitToMenu.

Entry Condition: Player clicks DisplayOptions.

Exit Condition: Player clicks ExitToMenu.

Quality Requirements: The program's response time should be 0.1 second.

- **Scenario Name:** ChangeBackground

Participating Actors: Initiated by Player

Flow of Events:

- Player is on the DisplayOptions menu.
- Player clicks on the background icon.
    - Program shows possible background images and gives player a choice to choose from these images.

- Player chooses one image and clicks on it.
    - Program sets the chosen image to be the new image of the background.

Entry Condition: Player clicks on the background icon on DisplayOptions menu.
Exit Condition: Player chooses a background image.
Quality Requirements: The program's response time should be 0.1 second.

- **Scenario Name:** ChangePaddle

Participating Actors: Initiated by Player
Flow of Events:
- Player is on the DisplayOptions menu.
- Player clicks on the paddle icon.
    - Program shows possible paddle textures and gives player a choice to choose from these images.
- Player chooses one texture and clicks on it.
    - Program sets the chosen image to be the new texture of the paddle.

Entry Condition: Player clicks on the paddle icon on DisplayOptions menu.
Exit Condition: Player chooses a paddle image.
Quality Requirements: The program's response time should be 0.1 second.

- **Scenario Name:** ChangeBall

Participating Actors: Initiated by Player
Flow of Events:
- Player is on the DisplayOptions menu.
- Player clicks on the ball icon.
    - Program shows possible ball textures and gives player a choice to choose from these images.
- Player chooses one texture and clicks on it.
    - Program sets the chosen image to be the new texture of the ball.

Entry Condition: Player clicks on the ball icon on DisplayOptions menu.
Exit Condition: Player chooses a ball image.
Quality Requirements: The program's response time should be 0.1 second.

- **Scenario Name:** ExitToMenu

Participating Actors: Initiated by Player

Flow of Events:

- Player is in an ongoing game or in any menu component.
- Player clicks on the ExitToMenu button.
    - Program destroys every object and clears any progress. Program saves the options.

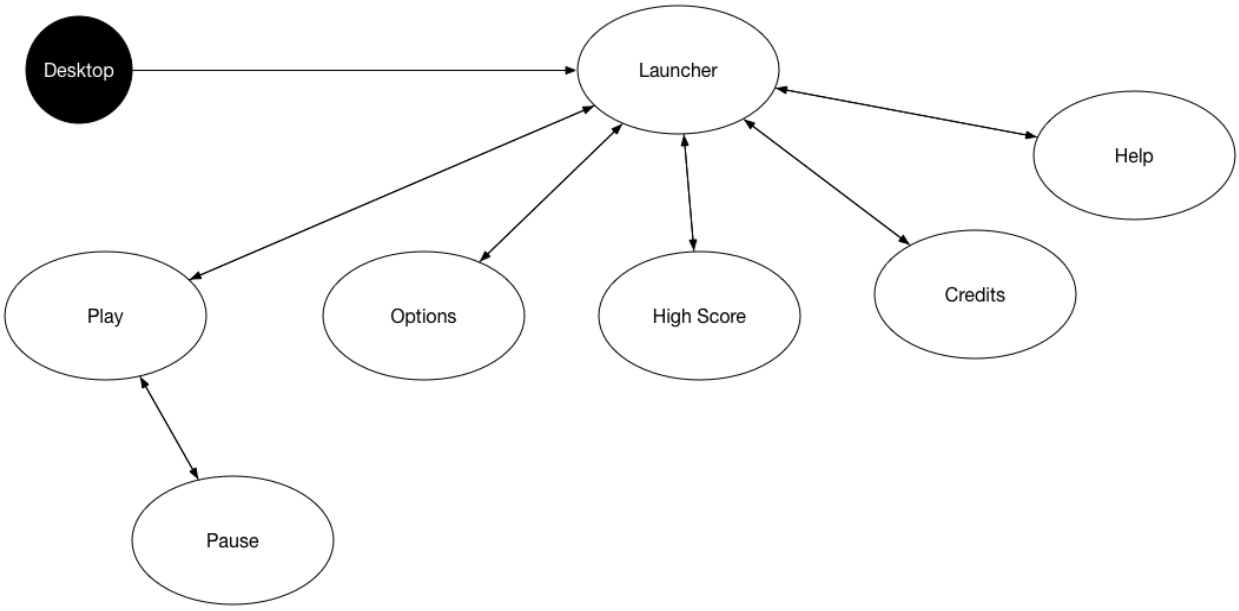
Entry Condition: Player clicks ExitToMenu.

Exit Condition: Player returns to previous menu.

Quality Requirements: The program's response time should be 0.1 second.

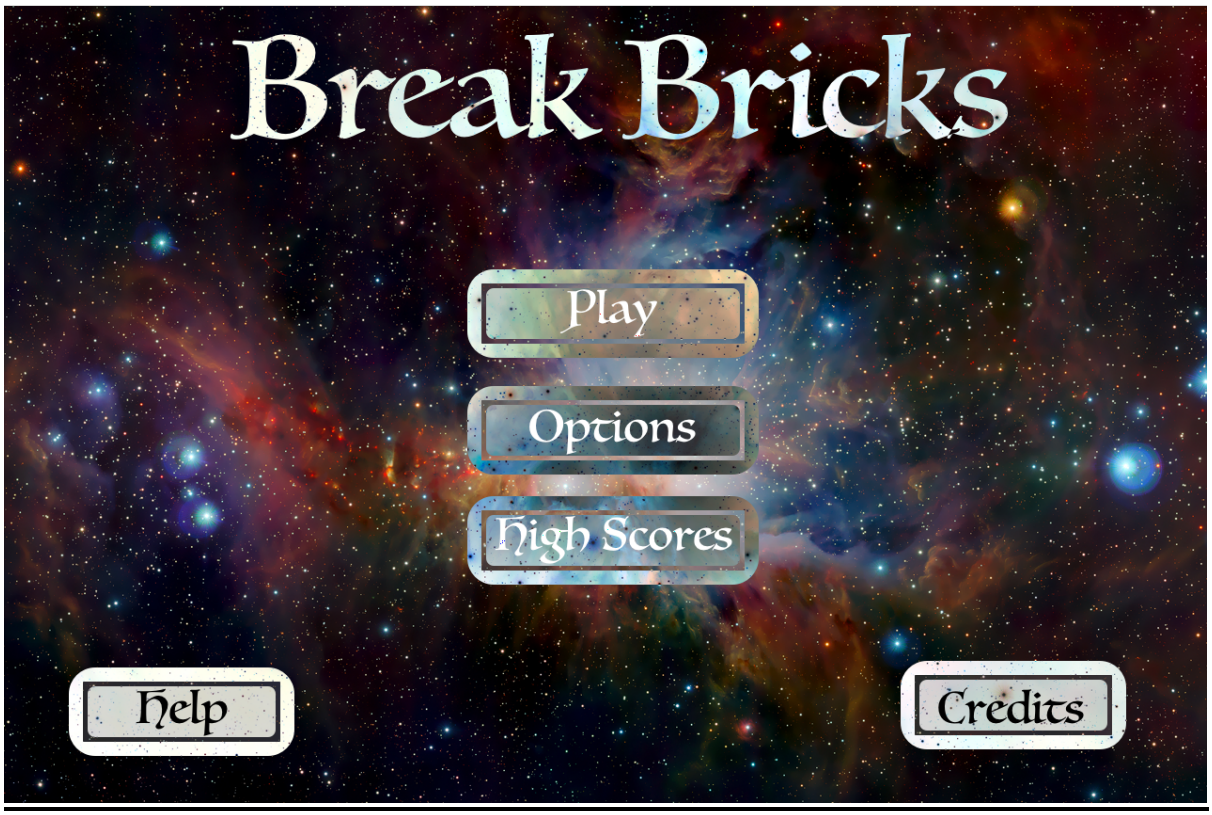
2.7. User Interface

2.7.1. *Navigational Path*



2.7.2. *Screen Mockups*

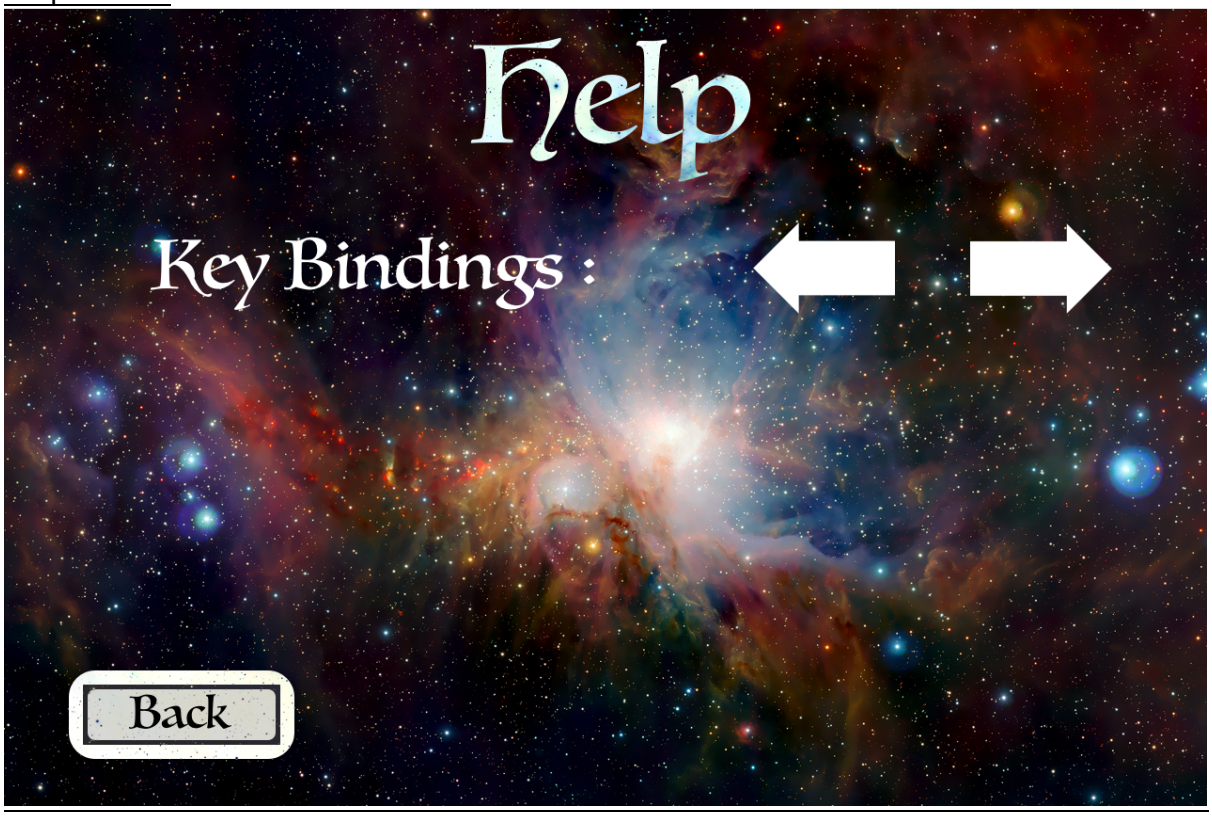
Main Menu:



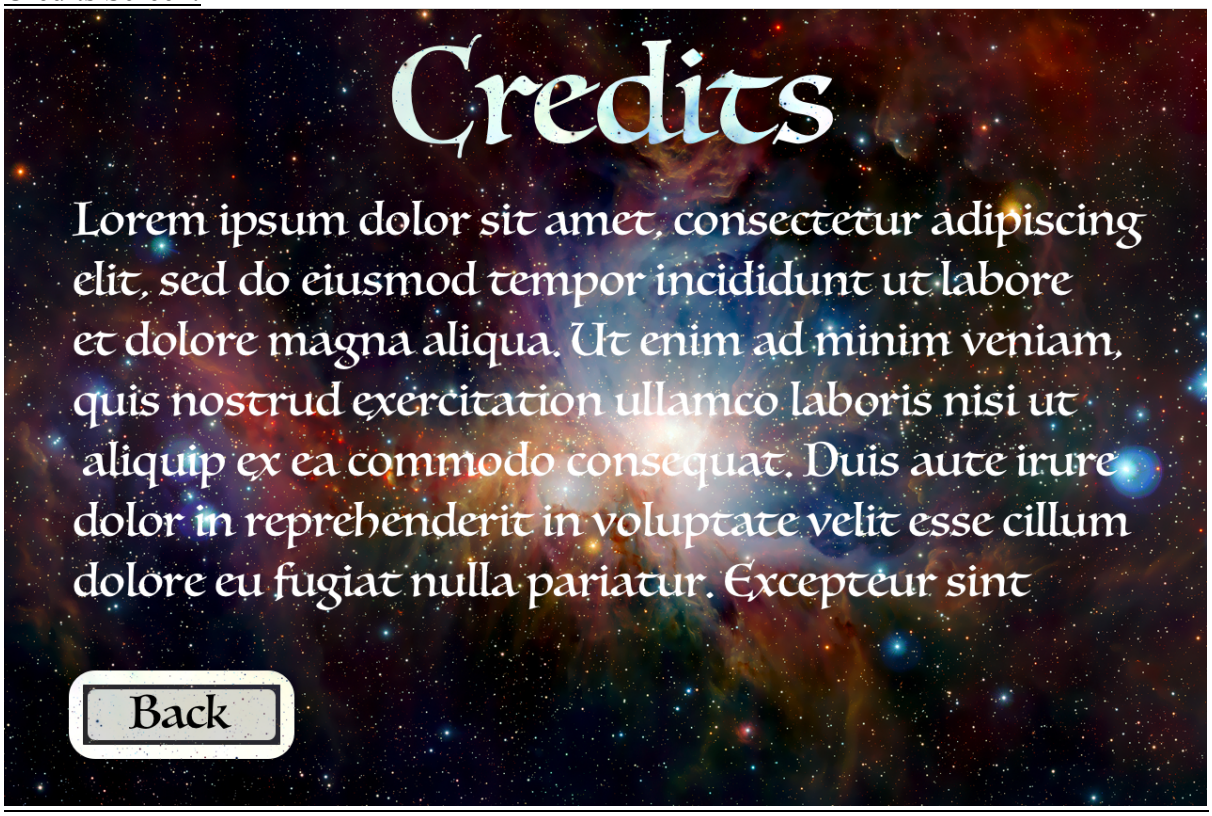
Options Menu:

High Scores Table:

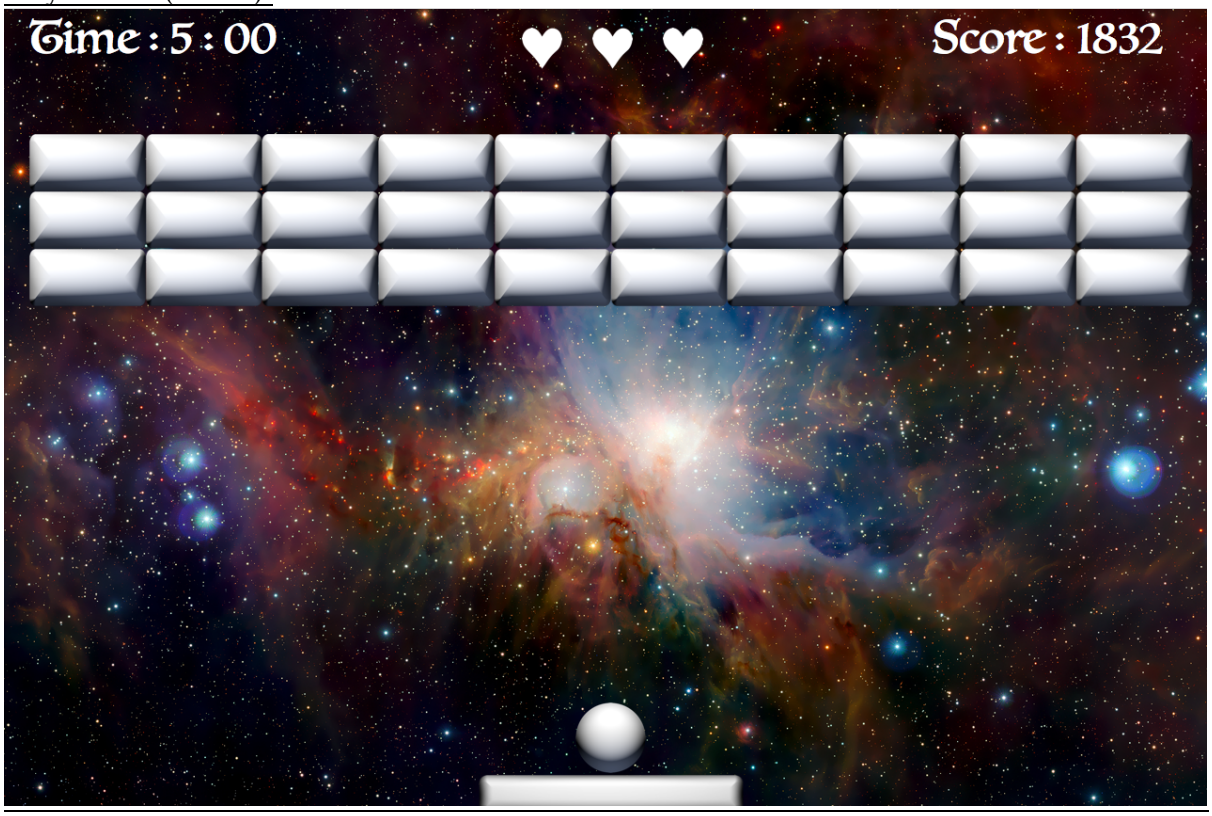
Help Screen:



Credits Screen:



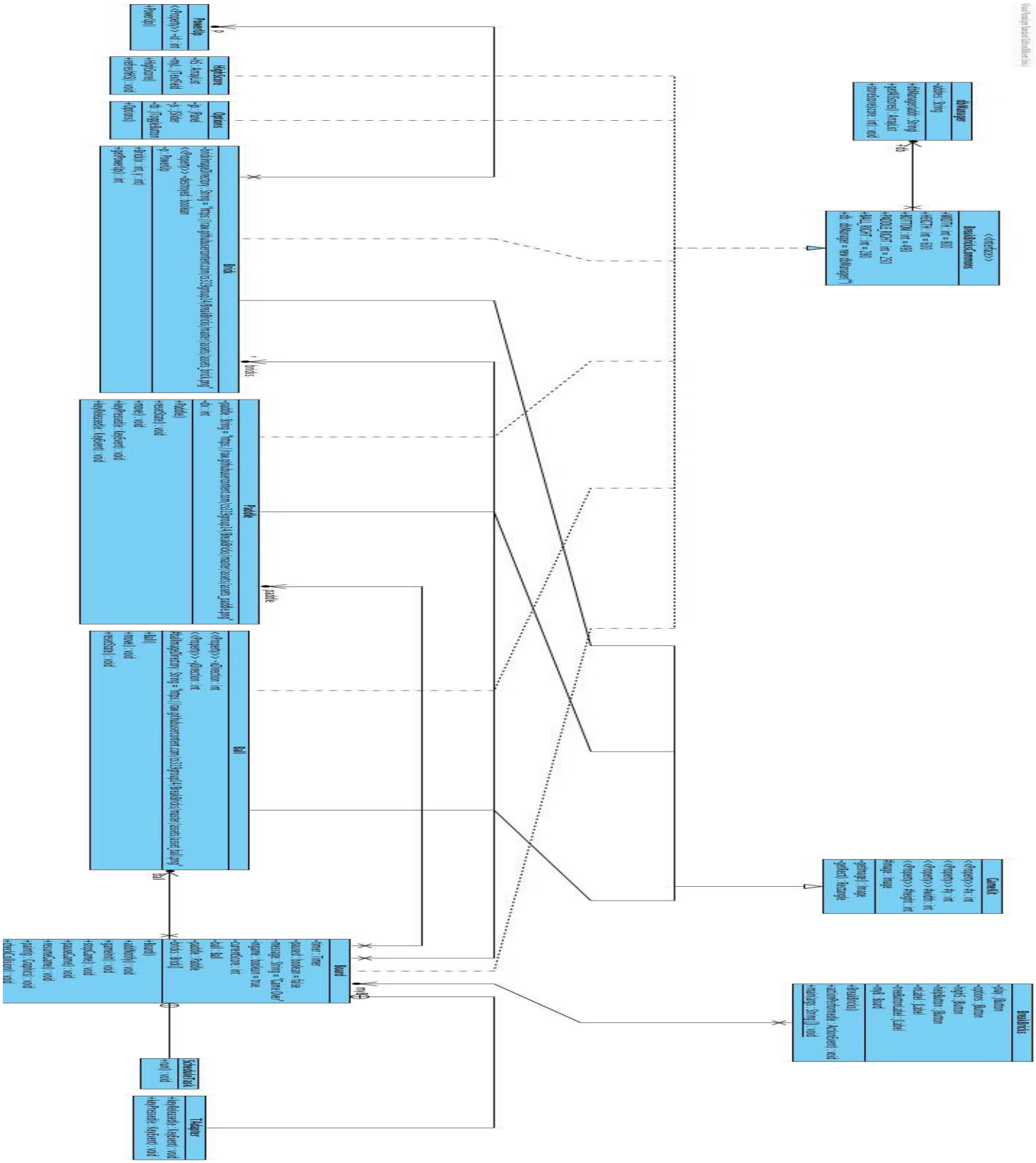
Play Screen (Arena):



3. System Analysis

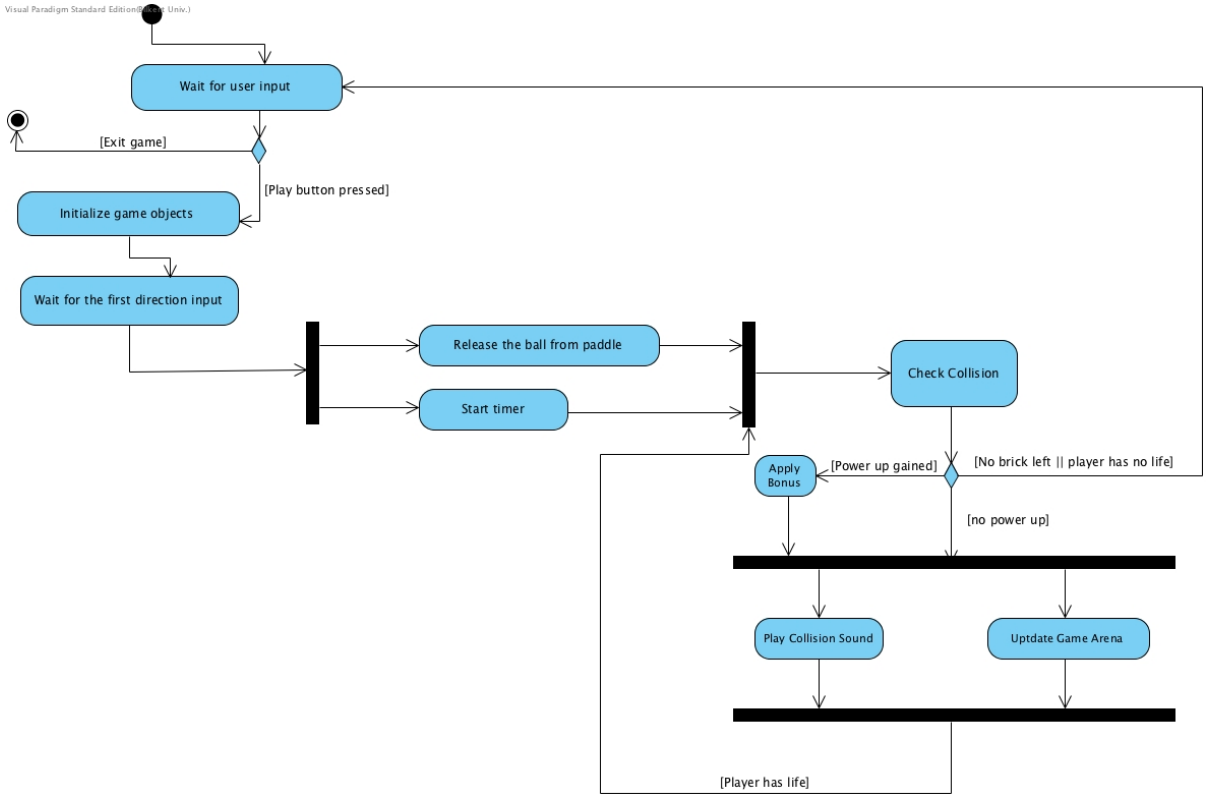
3.1. Object Model

3.1.1 Class Diagram



3.2. Dynamic Model

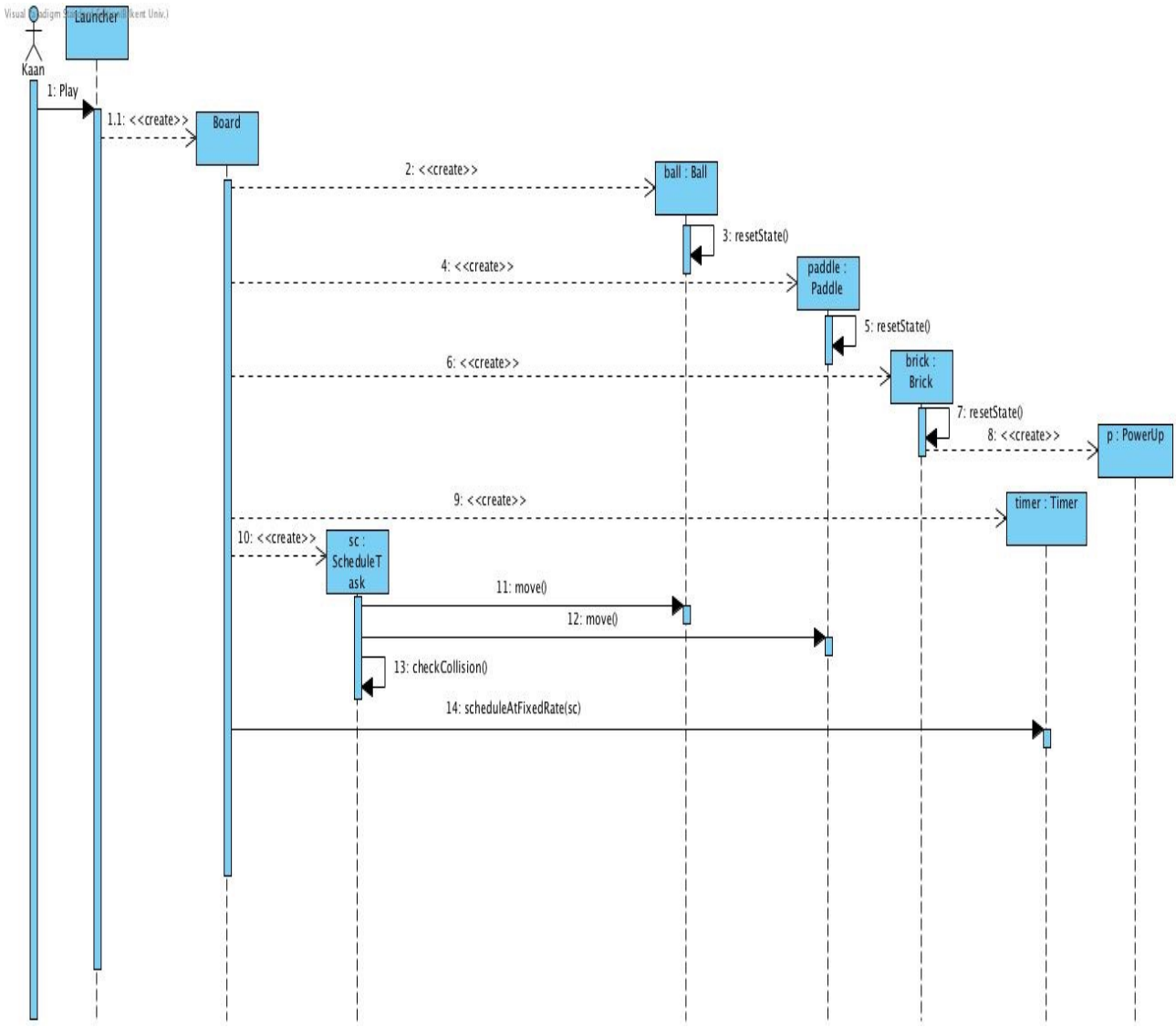
3.2.1. State Chart



3.2.2. Sequence Diagram

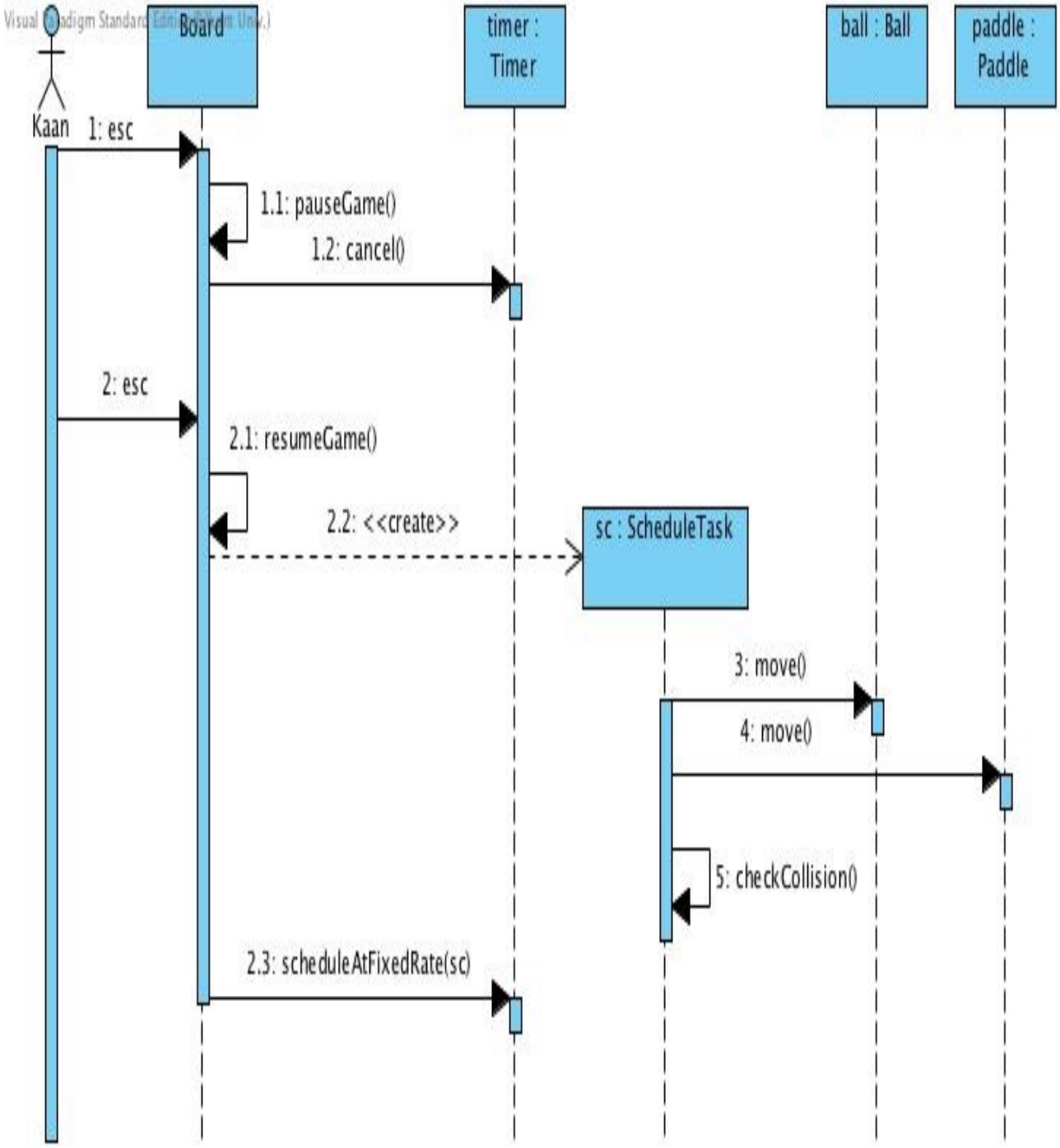
Start Game:

Scenario: Player Kaan presses Play button from launcher. Launcher calls Board object. Board object creates Ball, Paddle objects and also creates a Brick array. each Brick object creates a PowerUp object. PowerUp objects can have good , bad or none affect to the game. Then it resets the starting positions of the objects. Then it creates a Timer to periodically refresh the game. In order to do this it creates a ScheduleTask object. which controls movement of the ball and the paddle and also checks if the ball has collided.



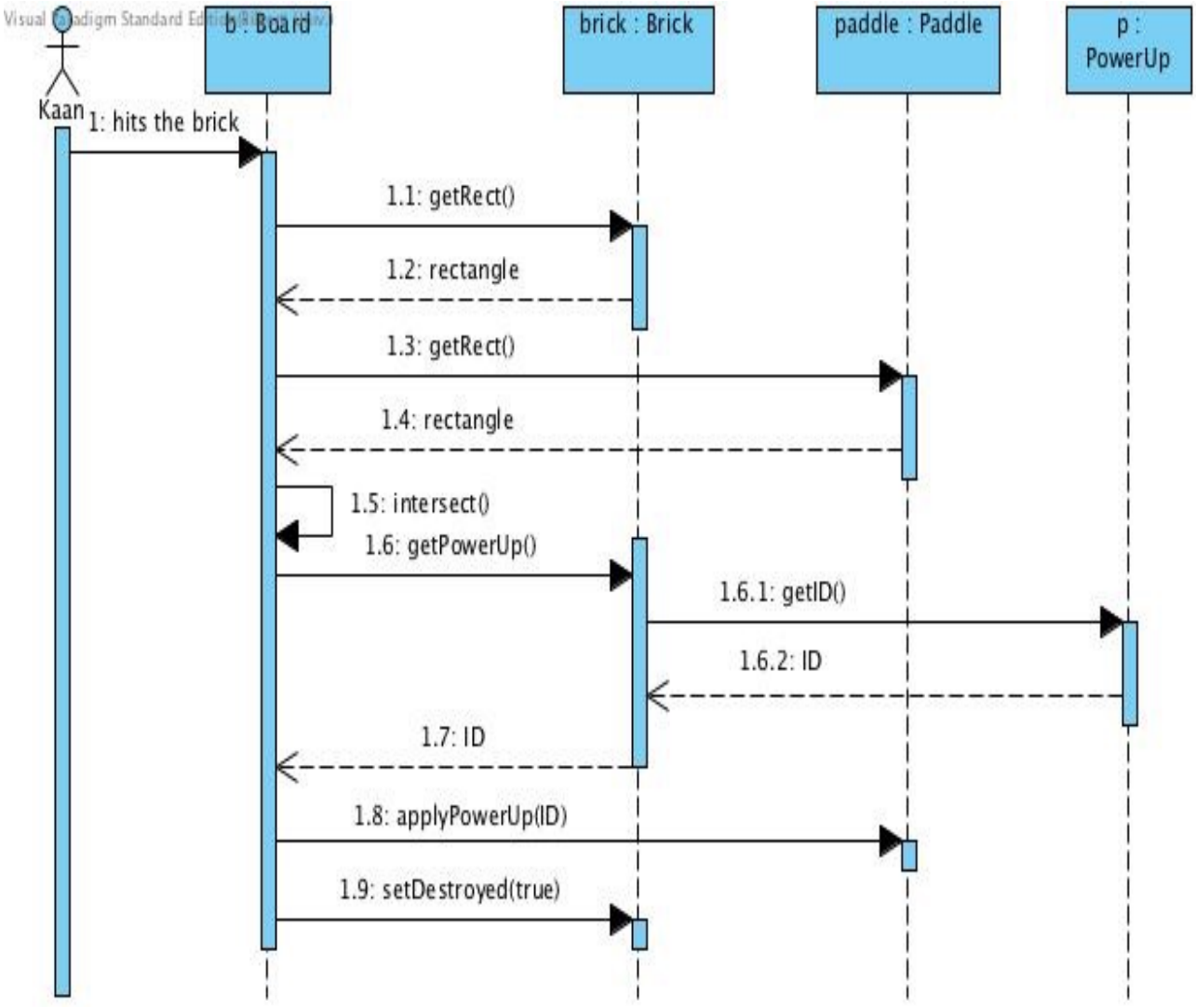
Pause Game:

Scenario: Kaan presses play button then previous diagram occurs. Then while playing the game he presses pause button and game pauses. Then after a delay Kaan presses esc again to resume the game.



Power Up:

Scenario: Kaan presses the play button and the game begins. The first diagram occurs then Kaan hits a brick system checks for any collision if yes then it looks for power up. There is always a power up which is determined from its id. The id 0 is for no power up and others for other power ups. In this game power up automatically gained and no need to pick it up. In this scenario we assume that we get fast paddle power up. The board takes the id of the power up from brick. Then it determines which one is it and sends the id to corresponding object. (For example if power up is about brick, it sends the ID to brick or if it is about paddle, it sends it to the paddle.)



4. Conclusion

Break Bricks is actually very well-known arcade game which in our version will be enriched by many power-ups, penalties and different kinds of bricks to break. For the implementation of the game Java, which is an object-oriented language, will be used.

The purpose of this game, like many other versions of it, will be to break as many bricks as possible in restrictive conditions. In our version these restrictive conditions are limited time and limited amount of lives that a player can have through a game. The game will be a desktop application and will be controlled by left and right buttons on the keyboard.

As many different objects such as 30 different bricks and their relationship among each other requires deep awareness about the course's main topic, we thought this game can be a great tool to understand and apply what we learned in this course. We aim to design a consistent game that abides by OOP fundamentals which properly models the real-life objects, interactions and events.

We learned very important lessons through writing this report which are using UML tools and diagrams to represent components of the project in the application domain analysis stage, generating a functional model using scenarios and producing use cases, transforming functional model into class model and dynamic model to represent the behavior and structure of the system in a real-world situation, decomposing a given model into meaningful subsystems where it produces a neat and organized relationship chain, deciding on the interactions and dependencies of subsystems and learning about architectural styles and design patterns.