COMP302 Software Engineering
Term Project: Bricking Bad
Final Project Report

Group Name: Monopoly
Team Members:

Arda Duman
Arjin Renas Akgül
Cansu Doğanay
Doğa Can Özil
Elhan Oğuz
Taluhan Öneş

Content:

Vision

Introduction: We aim to create Bricking Bad Game, a similar game to old school DX-Ball game.

Business Requirements: Main reason for us to do this project is missing the old school games. Old School games were so much fun to play and were not that complex. Although, there is a problem with the old school games, most of them are not compatible with todays' computer systems. Therefore, Bricking Bad game will resolve this problem and revive the old school games.

Product Solution/Overview: Bricking Bad Game will be compatible with the current computer systems. With this solution, we aim to success on reviving the old school games. DX-Ball is one of the best examples and similar one to Bricking Bad Game. But in addition to DX-Ball game, we have some features. The most exciting and different part will be the Building Mode where players can build their own games and the aliens that will change the course of the game during the gameplay. To conclude, Bricking Bad Game is a gamechanger since it's an old school concept game with brand new features.

Teamwork Organization

We worked together throughout the entire project. We were almost always together while doing something for it. Sometimes, we worked in pairs; however, when a group member feels in trouble, the whole group was there for to support him/her.

For writing the report, we distributed the use cases. System sequence diagrams, communication diagrams and sequence diagrams are also shared according to the use cases.

The Report Part

Use Cases;

- Login to the game - Arda Duman
- Loading the game – The whole team
- Building mode – Cansu Doğanay and Doğa Can Özil
- Launching the ball – Arjin Renas Akgül
- Move the paddle – Arda Duman
- Hit the ball – The whole team

- Rotation of the paddle – Doğa Can Özil
- Use the power-ups – Elhan Oğuz
- Save/Load – Taluhan Öneş
- Pause/ resume – Cansu Doğanay
- Displaying the help screen – Elhan Oğuz
- Hitting bricks – Arjin Renas Akgül
- Hitting aliens – Doğa Can Özil
- Quiting the game – Taluhan Öneş
- Using the laser gun – Cansu Doğanay

- Domain model was drawn by Elhan Oğuz
- Operation contracts were written by Arjin Renas Akgül and Taluhan Öneş

Diagrams;

- Interaction/Communication Diagram 1-5 by Cansu Doğanay, Arjin Renas Akgül, Arda Duman
- Interaction/Communication Diagram 6-8 by Doğa Can Özil, Taluhan Öneş.
- Interaction/Communication Diagram 9-10 by Elhan Oğuz
- Everyone drew his or her use case's system sequence diagram and sequence diagram.

Supplementary Specifications;

- FURPS+ were written by Arda Duman, Doğa Can Özil and Taluhan Öneş
- Vision was written by Arjin Renas Akgül, Cansu Doğanay and Elhan Oğuz
- Glossary was written by Elhan Oğuz

Testing;

- Testing part was done by mostly Elhan Oğuz but rest of the team helped him for their parts for to learn and give him specific information.

Design Alternatives;

- Model view separation, Grasp and GoF patterns were discussed as a group because patterns and MVS principle are essentials of the project. They are crucial, even vital, for to write and efficient code. So that, we did not want to distribute them for individuals.

The Code

- Paddle and ball were created by the whole team.
- Buttons were created by Taluhan Öneş
- Power-ups were created by Doğa Can Özil and Elhan Oğuz
- Aliens were created by Arda Duman, Cansu Doğanay and Doğa Can Özil
- Bricks were created by Arda Duman and Arjin Renas Akgül
- Building mode was created by the whole team.
- Rotating paddle was done by Arda Duman, Cansu Doğanay and Doğa Can Özil
- Launching the ball was done by Arjin Renas Akgül, Elhan Oğuz and Taluhan Öneş
- Hitting the bricks were done by Arjin Renas Akgül, Cansu Doğanay and Elhan Oğuz
- Moving the ball was coded by Arda Duman, Arjin Renas Akgül, and Elhan Oğuz
- Launching the paddle was coded by Doğa Can Özil and Taluhan Öneş

Use Case Diagram

Use Cases

Use Case UC1: Launching the Ball

Scope: Running mode
Level: Subfunction
Primary Actor: Player
Stakeholders and Interests:
-Player: Wants to launch the ball to make the ball move towards bricks.
Preconditions:
-User must be logged in.
-Game must be loaded and started successfully.
-There must be at least one ball.
-User should have at least one "life".
-Ball must be landed on paddle.
-Ball must be on steady situation for launching.
Success Guarantee: Ball draws away from the paddle to the bricks.
Main Success Scenario:
1. Ball launches with the interaction of the player.
2. Ball moves vertically in a line that makes an angle of 90 degrees with the surface of the paddle.
Extensions:
1a. Ball can be launched from different points of the paddle according to the landing point.
Frequency of Occurrence: Launching the ball occurs at the start of the game and every start after missing a ball. It also occurs after catching a ball with a Magnet power-up.

Use Case UC2: Moving the Paddle

Scope: Running Mode
Level: User goal
Primary Actor: Player
Stakeholders and Interests:
-Player: Wants to move the paddle left or right in order to catch the ball.
Preconditions:
-User must be logged into the game.
-Game must have been loaded and started successfully.
-There must be a paddle for catching the ball.
-The paddle must be between the screen bounds.
Success Guarantee: Paddle moves according to the players button pushes.
Main Success Scenario:
1. Player pushes the right-arrow button.
2. Paddle moves to the right successfully.
Extensions:
1a. Player pushes and holds the button.

2a. If the paddle is at the boundaries of the screen, the paddle shouldn't move.

1. If the paddle is at the rightmost side of the screen, it mustn't move right anymore.
2. If the paddle is at the leftmost side of the screen, it mustn't move left anymore.

Special Requirements:
-If the left or right arrow is pressed and released the paddle should move by an offset equal to L/2 with a speed of L/second**.**
-If the button is down, it should move with the speed of 2*L/second**.**
Frequency of Occurrence: As long as the player continues to the game.

Use Case UC3: Hitting the Ball

Scope: Running mode
Level: Subfunction
Primary Actor: Player
Stakeholders and Interests:
-Player: Wants to hit the ball in order to continue the game and send the ball through the bricks.
Preconditions:
- User must be logged in.
-The game must be on the running mode.
-The ball must be launched.
-The ball's direction must be towards down.
-The player should catch the ball with the paddle.
Success Guarantee: Player catches the ball with the paddle and the ball makes a proper rebound.
Main Success Scenario:
1. Game is started with the launch of the ball.
2. Player moves the paddle in order to send the ball back to the bricks.
3. The ball moves to the paddle.
4. The ball hits to the non-moving paddle with 90 degrees and bounces back with the same speed.

Extensions:
1a. This step is the UC1. All extensions for the UC1 are valid for this case.
2a. Player can move the paddle to the right or left.
2b. Player can rotate the paddle.
3a. The ball moves downwards but not towards paddle.
4a. The ball cannot be caught by the user.
4b. The magnet power-up is on, and the ball doesn't bounce back.
4c. If the ball hits the paddle with an angle between 0 and 90 degrees around the norm of its surface, it will bounce and move back with the same speed and the symmetric angle with the norm of the paddle surface.
4d. If the ball hits the corner of the paddle, the same rule applies as explained in extension 6c, but reference will be an imaginary line making an angle of 45 degrees with the corner of the paddle.
4e. If the paddle moves with the same direction of the component of the ball velocity that is parallel to that movement direction, the ball will reflect as explained in extensions 6c or 6d, but with a speed increase of 5px/second.

1. If the paddle moves with the opposite direction of the component of the ball velocity that is parallel to that movement direction, the ball will reflect with an angle of 180 degrees relative to the line of the original ball movement direction and the speed of the ball stays the same.

6f. If the paddle moves with a direction which is perpendicular to the direction of the ball velocity, the ball will reflect with an angle of 45 degrees relative to the line of the paddle movement direction and the speed of the ball stays the same.

Special Requirements:
- All actions must apply defined physical laws.
Frequency of Occurrence: Could be nearly continuous.

Use Case UC4: Rotating the Paddle

Scope: Running Mode
Level: User Goal
Primary Actor: Player
Stakeholders and Interests:
-Player**:** Wants to rotate the paddle between 45 degrees and 135 degrees in order to change the direction of the ball.
Preconditions:
-User must be logged in.
-Game must have been loaded and started successfully.
-There must be a paddle for catching the ball.
-The paddle must be between the screen bounds.
Success Guarantee: Paddle rotates according to the players button pushes.
Main Success Scenario:
1. Player wants to rotate the paddle to the 135 degrees.
2. Player pushes the D button.
3. Paddle rotates up to 135 degrees with a 20 degrees/second.
4. Player wants to rotate the paddle to the initial rotation.
5. Player release the button.
6. Paddle returns to its initial rotation with a 45 degrees/second.
Extensions:
1a. Player wants to rotate the paddle to the 45 degrees.
2a. Player pushes the A button.
3a. Paddle rotates up to 45 degrees with a 20 degrees/second.
Special Requirements:
-Maximum and minimum angle for the rotation is 45 degrees and 135 degrees.
Frequency of Occurrence: As long as the player continues to the game.

Use Case UC5: Using the Power-ups

Scope: Running Mode
Level: User-goal
Primary Actor: Player
Stakeholders and Interests:
-Player**:** Wants to use a power-up. But before using it, the player must catch the power-ups, which are appeared at game screen after breaking a wrapper brick, with the paddle.
Preconditions:
1. Player successfully logins to the game.
2. Player loads a game.
3. Game is started with the launch of the ball.
4. The game must be on the running mode.
5. The player must break a wrapper brick for power-up appearance.
6. Once a power-up appeared, it will fall down. The player must catch it by hitting it with the paddle.
7. Caught power-ups are ready for using by the player.
Success Guarantee:
Success Guarantee (Postconditions) depends on which power is used.

- ➔ **Taller Paddle:** If this power-up is used, paddle's length will be doubled for 30 seconds.
- ➔ **Magnet:** If this power-up is used, paddle will catch the ball instead of reflecting it.
- ➔ **Destructive Laser Gun:** If this power-up is picked up, paddle will have laser guns attached to it on corners which have 5 shots and controlled by player.
- ➔ **Fireball:** If this power-up is picked up, balls in the game will turn into the fireballs.

- ➔ **Chemical Ball:** If this power-up is used, ball can go through brick and break them for one minute.
- ➔ **Gang-of-Balls:** If this power-up is picked up, the ball number in the game will be 10 till player keeps them in the game.

Main Success Scenario:

1. The ball hits a wrapper brick and a Taller Paddle power-up appeared on the screen.
2. The power-up will fall with the speed of L/4, where L is equal to screen length/10(Paddle length).
3. Power-up caught by the player by hitting it with the paddle.
4. Player activated the power-up by hitting the designated button on the game screen or "T" button on the keyboard.
5. Paddle's length doubled for the next 30 seconds after usage of power-up.

Extensions:

1a. The ball can hit any other brick then the wrapper bricks, in that case no power-up will appear in the game screen therefore success scenario is back to step 4 until ball hits a wrapper brick.

1b. *Appearance of different power-ups:

1. Instead of Taller Paddle power-up, Magnet power-up can appear at game screen.
2. Instead of Taller Paddle power-up, Destructive Laser Gun power-up can appear at game screen.
3. Instead of Taller Paddle power-up, Fireball power-up can appear at game screen.
4. Instead of Taller Paddle power-up, Chemical Ball power-up can appear at game screen.
5. Instead of Taller Paddle power-up, Gang-of-Balls power-up can appear at game screen.

2a. The extension 1a is applicable for this situation too. If there is no power-up appeared, there is no power-up to fall with given speed.

3a. The power-up will disappear for good if the player cannot hit it with the paddle.

4a. The power-ups won't be activated if the player will not click designated button or keyboard buttons which will be mentioned below.

4b. *Activation of different power-ups:

1. If player caught Magnet power-up, players can activate the power-up by hitting the designated button on the game screen or "M" button on the keyboard.
2. If player caught Destructive Laser Gun power-up, power up will be activated immediately for 5 shots of laser. Player can shoot lasers with "W" button on the keyboard or the left mouse button click.
3. If player caught Fireball power-up, the power up will be activated immediately.
4. If player caught Chemical Ball power up, the player can activate the power-up by hitting the designated button on the game screen or the "C" button on the keyboard.
5. If player caught Gang-of-Balls power-up, the power-up will be activated immediately.

5a. *Power-up effects while Taller Paddle power-up is active:

1. If the player activates another Taller Paddle power-up while Taller Paddle is active, the length of the paddle will be double again until it reaches 4*length of the screen/5 and the time limit will be reset. To sum up, player can active 3 Taller Paddle power-ups at the same time because of screen length limit.

2. If the player activates Magnet power-up while Taller Paddle is active, the paddle gain attributions of the Magnet power-up, length of the paddle or time limit of Taller Paddle power-up will not be affected.

3. If the player activates Destructive Laser Gun power-up, the paddle gain attributions of the Destructive Laser Gun power-up, length of the paddle or time limit of Taller Paddle power-up will not be affected.

4. If the player activates other ball related power-ups, like Fireball, Chemical Ball and Gang-of-Balls power-ups, the length of the paddle or time limit of Taller Paddle power-up will not be affected.

5b. *Power-up effects while Magnet power-up is active:

1. Activating Taller Paddle power-up while Magnet is active extension is the same extension with 5a-2.
2. Activating Magnet power-up while Magnet is active will not affect anything.
3. Activation of Destructive Laser Gun while Magnet is active will add attributions of Destructive Laser Gun power-up to the paddle.
4. Activating Fireball and Chemical Ball power-up will not affect anything on paddle. The paddle can catch these balls too.
5. If Gang-of-Balls activated at the same time with the Magnet power-up, The paddle will catch and hold the all balls in the game and launch all caught balls at the same time.

5c. *Power-up effect while Destructive Laser Gun power-up is active:

1. Activating Taller Paddle while Destructive Laser Gun power-up is active is the same extension with the 5a-3.
2. Activating Magnet while Destructive Laser Gun power-up is active is the same extension with the 5b-3.
3. Activation Destructive Laser gun while itself is active will reload the 5 laser shots if any shot is used by the player. If any shot is not used, the laser gun power-up will not be activated.
4. Activation of the power-ups related with the ball, like Fireball, Chemical Ball and Gang-of-Balls, will not affect attributions of Destructive Laser Gun power-up.

5d. *Power-up effect while Fireball power-up is active:

1. Activation of the power-ups related with the paddle, like Taller Paddle, Magnet and Destructive Laser Gun, will not affect the attributes of the Fireball power up.
2. Activation of the Fireball power-up while itself is active will not affect anything in the game if all the balls in the game are fireballs, if not, then it changes these non-fireballs to the fireballs.
3. Activating Chemical Ball while the Fireball power-up is active will gave the ball both attributions of the Chemical Ball and the Fireball power-ups at the same time for one minute.
4. Activation of the Gang-of-Balls while the Fireball power-up is active will make number of the ball in the game is 10 which all will be copies of the fireball.

5e- *Power-up effect while Gang-of-Balls power-up is active:

1. Activation of the Taller Paddle or the Destructive Laser gun power-up will not affect the attributes of the Gang-Of-Ball power-up
2. Activation of the Magnet power-up while Gang-of-Balls active is the same extension with the 5b-5.
3. Activation of the Fireball power-up while Gang-of-Balls active will turn all the balls in the game into the fireballs.
4. Activation of the Chemical Ball power-up while Gang-of-Balls active will turn all the balls in the game into the chemical balls for one minute.
5. Activation of the Gang-of-Balls power-up while itself is active will fix the ball number to 10 if it is not 10. If the current ball number in the game is 10, then the power-up will not affect anything.

5f. *Power-up effect while Chemical Ball power-up is active:

1. Activation of paddle related power-ups, like Taller Paddle, Magnet and Destructive Laser Gun, will not affect attributions of the Chemical Ball power-up.
2. Activation of the Fireball while Chemical Ball is active is the same extension with the extension 5d-3. But it will not affect the time limit of Chemical Ball.
3. Activation of the Chemical Ball while itself is active will reset the time limit of the Chemical Ball for all balls in the game.
4. Activation of the Gang-of-Balls while Chemical Ball is active will not affect the time limit of the Chemical Ball power-up, but all the ball created will be chemical balls in the time left of the power-up.

Special Requirements:
-A bag like data structure for holding power-ups that can be stacked, like Magnet, Taller Paddle and Chemical ball.
- Chemical Ball time limit is 1 minute, and it is final cannot extend.
-Unstackable power-ups, like Destructive Laser Gun and Gang-of-Balls, have limits that cannot extend. Ball number of the game cannot be bigger than 10. Player cannot have laser shots more than 5.
- Paddle cannot grow more than the screen.
Frequency of Occurrence: For unstackable power-ups like Destructive Laser Gun, Gang-of-Balls and Fireball, it depends on the number of the wrapper bricks that contains them. For stackable power-ups like Taller Paddle, Magnet and Chemical Ball, it depends on the decision of the player.

Use Case UC6: Pausing the Game/Resuming the Game

Scope: Game screen
Level: Subfunction
Primary Actor: Player
Stakeholders and Interests:
-Player: Wants to pause the game without losing anything and wants to give a break, without any errors.
-Player: Wants to resume the game from the exact same point that she or he gave a break, without any errors.
Preconditions:
-User must be logged in successfully.
-There must be a running game at the background to pause.
-There must be a flawless pausing process to make an impeccable resuming act.
Success Guarantee:
Pause: The act of pausing is about instructing the software to stop processing new information. The logic on that paused screen waits for a user input before returning back to the previous game state. Player could stop any progression in the game when he or she pushes the pause button.
Resume: The act of resuming the game simply returns to the previous state and continues executing from the point where you stopped. Player could continue the progressions in the game when he or she pushes the resume button where the exact same place he or she left the game.
Main Success Scenario:
Pause:
1. Player presses the pause button during any time of the game.
2. System converts the pause button symbol to the resume button symbol.
3. The game stops at the state that the player presses the button.
4. The game waits for the user input to resume the game.

Resume:
1. Player presses the resume button during the break.
2. The symbol in the resume button turned to pause symbol.
3. The game starts with the exact state that the player has pressed to pause button.

Extensions:
-Pause:
1a. Player may press a different button other than the pause button.
2a. The symbol may not turn into a resume symbol.
3a. The pause button may not work properly, and the game will not stop.
4a. Player may want to save or quit the game other than to resume the game.
-Resume:
1a. Player may want to save or quit the game other than to resume the game.
1b. The resume button may not work properly.
2a. The symbol may not turn into a pause symbol.

3a. To stop the game or saving the state may not work properly so that the game will not start at where it should be started.

Special Requirements:

-There should be a circular button that should be visible for the player.
-Button should be on the screen that will not affect the player by playing the game (On the side of the screen, preferably on the top left corner).
-Button should be visible on each screen after login screen.
-In the button there should be a clear pause symbol to demonstrate the pausing act.
-The symbol in the button should turn into a resume symbol right after when the user pauses the game.

Frequency of Occurrence: As much as the player wants.

Use Case UC7: Hitting the Bricks

Scope: Game Screen
Level: user goal
Primary Actor: Player
Stakeholders and Interests:
-Player: Wants to break bricks by hitting them.
Preconditions:
- Player successfully logs in to the game.
- Player loads a game.
- Game is started with the launch of the ball.
- There must be bricks on the game screen.
- There must be at least one ball.
- Ball must be moving.

Success Guarantee: The ball hits a brick.
Main Success Scenario:
1. Ball is on moving state.
2. A moving ball hits a brick.

Extensions:

2a. If the brick is type of simple brick, it breaks.

1. Player's score will increase.

2. If the brick is stationary, and the ball hits the paddle with an angle between 0 and 90 degrees around the norm of its surface, it will bounce and move back with the same speed and the symmetric angle with the norm of the paddle surface.

2a. If the brick is stationary, and the ball hits the corner of the paddle, the same rule applies as explained in extension 2a.2, but reference will be an imaginary line making an angle of 45 degrees with the corner of the brick.

3. If the brick moves with the same direction of the component of the ball velocity that is parallel to that movement direction, the ball will reflect as explained in extensions 2a.2 or 2a.2a, but with a speed increase of 5px/second.

3a. If the brick moves with the opposite direction of the component of the ball velocity that is parallel to that movement direction, the ball will reflect with an angle of 180 degrees relative to the line of the original ball movement direction and the speed of the ball stays the same.

4. If the brick moves with a direction which is perpendicular to the direction of the ball velocity, the ball will reflect with an angle of 45 degrees relative to the line of the brick movement direction and the speed of the ball stays the same.

5. After that brick is broken it creates a space in its place.

5a. If this new empty place gives a free place in x-axis for simple brick, simple brick will start to move back and forth with a probability of 0.1 with the movement speed of L/4*second, or stiff with a probability of 0.9.

5b. If this new empty place gives a free place in x-axis for simple brick, half-metal-brick will start to move back and forth with a probability of 0.1 with the movement speed of L/4*second, or stiff with a probability of 0.9.

5c. If this new empty place gives a circular free place which centered in (x, y) and with a radius of 1.5*L where x is the x-coordinate of the center of mine-brick and y is the y-coordinate of the center of mine-brick minus 1.5*L**,** brick will start to move in this circular free space with a probability of 0.1with the movement speed of L/4*second, or stiff with a probability of 0.9.

2b. If the brick type is type of half-metal-brick and the ball hits the non-metal side, it breaks.

1. Player's score will increase.

2. If the brick is stationary, and the ball hits the paddle with an angle between 0 and 90 degrees around the norm of its surface, it will bounce and move back with the same speed and the symmetric angle with the norm of the paddle surface.

2a. If the brick is stationary, and the ball hits the corner of the paddle, the same rule applies as explained in extension 2b.2, but reference will be an imaginary line making an angle of 45 degrees with the corner of the brick.

3. If the brick moves with the same direction of the component of the ball velocity that is parallel to that movement direction, the ball will reflect as explained in extensions 2b.2 or 2b.2a but with a speed increase of 5px/second.

3a. If the brick moves with the opposite direction of the component of the ball velocity that is parallel to that movement direction, the ball will reflect with an angle of 180 degrees relative to the line of the original ball movement direction and the speed of the ball stays the same.

4. If the brick moves with a direction which is perpendicular to the direction of the ball velocity, the ball will reflect with an angle of 45 degrees relative to the line of the brick movement direction and the speed of the ball stays the same.

5. After that brick is broken it creates a space in its place.

5a. If this new empty place gives a free place in x-axis for simple brick, simple brick will start to move back and forth with a probability of 0.1 with the movement speed of L/4*second, or stiff with a probability of 0.9.

5b. If this new empty place gives a free place in x-axis for simple brick, half-metal-brick will start to move back and forth with a probability of 0.1 with the movement speed of L/4*second, or stiff with a probability of 0.9.

5c. If this new empty place gives a circular free place which centered in (x, y) and with a radius of 1.5*L where x is the x-coordinate of the center of mine-brick and y is the y-coordinate of the center of mine-brick minus 1.5*L**,** brick will start to move in this circular free space with a probability of 0.1with the movement speed of L/4*second, or stiff with a probability of 0.9.

3. If the ball hits a half-metal-brick's metal side, it will not break the brick, and will rebound.

3a. If the ball is a fireball, it will crack the metal side, and brick will become a simple brick.

3b. If the ball is a chemical ball, it will break the brick and will keep going.

1. Player's score will increase.

2c. If the brick is type of mine-brick, it breaks/explodes.

1. Player's score will increase.

2. If the brick is stationary, and the ball hits the paddle with an angle between 0 and 90 degrees around the norm of its surface, it will bounce and move back with the same speed and the symmetric angle with the norm of the paddle surface.

2a. If the brick is stationary, and the ball hits the corner of the paddle, the same rule applies as explained in extension 2c.2, but reference will be an imaginary line making an angle of 45 degrees with the corner of the brick.

3. If the brick moves with the same direction of the component of the ball velocity that is parallel to that movement direction, the ball will reflect as explained in extensions 2c.2 or 2c.2a, but with a speed increase of 5px/second.

3a. If the brick moves with the opposite direction of the component of the ball velocity that is parallel to that movement direction, the ball will reflect with an angle of 180

degrees relative to the line of the original ball movement direction and the speed of the ball stays the same.

4. If the brick moves with a direction which is perpendicular to the direction of the ball velocity, the ball will reflect with an angle of 45 degrees relative to the line of the brick movement direction and the speed of the ball stays the same.

5. After that brick is broken it creates a space in its place.

5a. If this new empty place gives a free place in x-axis for simple brick, simple brick will start to move back and forth with a probability of 0.1 with the movement speed of L/4*second, or stiff with a probability of 0.9.

5b. If this new empty place gives a free place in x-axis for simple brick, half-metal-brick will start to move back and forth with a probability of 0.1 with the movement speed of L/4*second, or stiff with a probability of 0.9.

5c. If this new empty place gives a circular free place which centered in (x, y) and with a radius of 1.5*L where x is the x-coordinate of the center of mine-brick and y is the y-coordinate of the center of mine-brick minus 1.5*L**,** brick will start to move in this circular free space with a probability of 0.1with the movement speed of L/4*second, or stiff with a probability of 0.9.

6. Once brick is exploded, it will break the neighbor bricks with a radius of 2*L pixels.

3a. Player's score will increase for each brick that is broken.

3b. For each brick that is broken with this explosion, every brick creates a free place in its place.

1. If this new empty place gives a free place in x-axis for simple brick, simple brick will start to move back and forth with a probability of 0.1 with the movement speed of L/4*second, or stiff with a probability of 0.9.

2. If this new empty place gives a free place in x-axis for simple brick, half-metal-brick will start to move back and forth with a probability of 0.1 with the movement speed of L/4*second, or stiff with a probability of 0.9.

3. If this new empty place gives a circular free place which centered in (x, y) and with a radius of 1.5*L where x is the x-coordinate of the center of mine-brick and y is the y-coordinate of the center of mine-brick minus 1.5*L, brick will start to move in this circular free space with a probability of 0.1with the movement speed of L/4*second, or stiff with a probability of 0.9.

2d. If the brick is type of wrapper-brick, it breaks.

1. Player's score will increase.

2. If the brick is stationary, and the ball hits the paddle with an angle between 0 and 90 degrees around the norm of its surface, it will bounce and move back with the same speed and the symmetric angle with the norm of the paddle surface.

2a. If the brick is stationary, and the ball hits the corner of the paddle, the same rule applies as explained in extension 2d.2, but reference will be an imaginary line making an angle of 45 degrees with the corner of the brick.

3. If the brick moves with the same direction of the component of the ball velocity that is parallel to that movement direction, the ball will reflect as explained in extensions 2d.2 or 2d.2a, but with a speed increase of 5px/second.

3a. If the brick moves with the opposite direction of the component of the ball velocity that is parallel to that movement direction, the ball will reflect with an angle of 180 degrees relative to the line of the original ball movement direction and the speed of the ball stays the same.

4. If the brick moves with a direction which is perpendicular to the direction of the ball velocity, the ball will reflect with an angle of 45 degrees relative to the line of the brick movement direction and the speed of the ball stays the same.

5. After that brick is broken it creates a space in its place.

5a. If this new empty place gives a free place in x-axis for simple brick, simple brick will start to move back and forth with a probability of 0.1 with the movement speed of L/4*second, or stiff with a probability of 0.9.

5b. If this new empty place gives a free place in x-axis for simple brick, half-metal-brick will start to move back and forth with a probability of 0.1 with the movement speed of L/4*second, or stiff with a probability of 0.9.

5c. If this new empty place gives a circular free place which centered in (x, y) and with a radius of 1.5*L where x is the x-coordinate of the center of mine-brick and y is the y-coordinate of the center of mine-brick minus 1.5*L**,** brick will start to move in this circular free space with a probability of 0.1with the movement speed of L/4*second, or stiff with a probability of 0.9.

6. If the wrapper-brick hides power-up inside, a power-up will fall with a speed of L/4*second.

7. If the wrapper-brick hides alien inside, it will trigger an alien.

7a. If the alien is a repairing alien, it will react as explained in UC8.

7b. If the alien is a protecting alien, it will react as explained in UC8.

7c. If the alien is a cooperative alien, it will react as explained in UC8.

Special Requirements:

1. Simple bricks and half-metal-bricks are rectangles with dimensions L/5 and 20px.
2. The mine-bricks are circular with a radius equal to 10px.

Frequency of Occurrence: Could be nearly continuous until all the bricks are broken.

Use Case UC8: Creating a Game with Building Mode.

Scope: Game screen
Level: User goal
Primary Actor: The player
Stakeholders and Interests:
-Player: Wants to build a game to load and play.
Preconditions:
-The player is identified and authenticated.
-The player logs in successfully.
-The player decides to create a new game instead of loading a saved game.
Success Guarantee:
-The player specifies the number of each brick type.
-System puts these bricks to random places of the game screen.
-The player becomes able to change the brick's locations. Remove some of bricks and/or add more bricks by mouse clicks.
Main Success Scenario:
1. The player decides to create a new game.
2. The player pushes to create a new game button which appears on the starting window.
3. The player specifies the number of simple bricks meets the lower bound.
4. The player specifies the number of half-metal bricks meets the lower bound.
5. The player specifies the number of mine-bricks meets the lower bound.
6. The player specifies the number of wrapper-bricks meets the lower bound.
7. The player can see number of each bricks labeled on the top of the game screen.
8. The player sees the generated game and edits the position and number of generated bricks by using a mouse.
9. The player starts the game with the newly created game.

Extensions:

1a. The player decides to use an existing game rather than create a new one.
2a. The player pushes to load a game button.
3a. If the player specifies the number of simple bricks less than the lower bound, game does not accept the number and tells the player to enter a valid number.
4a. If the player specifies the number of half-bricks less than the lower bound, game does not accept the number and tells the player to enter a valid number.
5a. If the player specifies the number of mine-bricks less than the lower bound, game does not accept the number and tells the player to enter a valid number.

6a. If the player specifies the number of wrapper-bricks less than the lower bound, game does not accept the number and tells the player to enter a valid number.
7a. Number of bricks, that are labeled on top of the game window can be false.
8a. If the total number of the bricks are more than upper bound the, game does not accept the number and tells the player to enter a valid number for each type of the brick.
8b. If the player overlaps the bricks while editing the position of the bricks, the game does not allow user to do that.
8c. If the player puts a brick lower than a specified limit on the game screen, the game does not allow user to do that.

Special Requirements:
-There has to be at least 75 simple bricks.
-There has to be at least 10 half-metal bricks.
-There has to be at least 5 mine-bricks.
-There has to be at least 10 wrapper-bricks.
-None of the bricks overlaps with the others.
-The maximum number of the total bricks must be bounded with the screen's size.
-The paddle must be visible on the screen.
-The ball must be visible on the screen.
-There must be a save button at the top of the screen.
-There must be a pause/resume button at the top of the screen.
-There must be a quit button at the top of the screen.
-There must be a load button at the top of the screen.
-There must be a label which demonstrates the number of lives of the player.
-There must be a label which demonstrates the score of the player.
Frequency of Occurrence: Every time the player decided to create a new game screen.

Use Case UC9: Hitting the Aliens

Scope: Game Screen
Level: User Goal
Primary Actor: Player
Stakeholders and Interests:
-Player: Wants to repel harmful aliens by hitting them.
Preconditions:
- Ball must hit the special wrapper brick which, when destroyed, sends a signal to the aliens to appear.
- There must be an alien on the game screen.
- There must be at least one ball.
- Ball must be moving.
Success Guarantee: The ball hits the vulnerable side of an alien.
Main Success Scenario:
1. Ball is on a moving state.
2. A moving ball hits an alien.
3. Alien escapes from the game screen and stops its task.
Extensions:
2a. If the alien is a type of repairing alien and hit by the ball, it stops building and escapes from the game screen.
2b. If the alien is a type of repairing alien and not hit by the ball, it remains in the game screen and keeps doing its task.
    1. Repairing alien rebuilds the simple bricks.
    2. Repairing alien builds a single block each five seconds.
        2a. The number of blocks to be built and their locations are decided randomly.

2c. If the alien is a type of protecting alien and the ball hits the top of the alien which is a vulnerable side of the protecting alien, it escapes from the game screen.
2d. If the alien is a type of protecting alien and the ball does not hit the alien or hit the bottom of the alien, it remains in the game screen and keeps doing its task.

1. Protecting aliens move horizontally under the bricks and with a speed of 3*L/second.
2. The ball hits the bottom of the protecting alien, ball bounces back.

2e. If the alien is a type of cooperative alien and hit by the ball, it stops its task and escapes from the game screen.

1. No more cooperative aliens come to the game screen after escaping from the game screen.

2f. If the alien is a type of cooperative alien and not hit by the ball, it remains in the game screen and does its task.

1. Cooperative alien picks a row of bricks at a random and destroys it.

1a. Cooperative alien, after destroying one row, disappears from the game screen.

2g. If the alien is a type of drunk alien and hit by the ball, it stops building and escapes from the game screen.
2h. If the alien is a type of drunk alien and not hit by the ball, it stops its task which is a task that is determined by brick number.

1. If the brick number >70 the drunk will firstly act like 2b.1 then 2d.1.
2. If the brick number >60 and <=70, the alien will appear and stays for 5 seconds. Then disappears.
3. If the brick number >50 and <=60, the alien will act like 2b.1.
4. If the brick number >40 and <=50, the alien will act like 2d.1.
5. If the brick number >30 and <=40, the alien will act like 2h.2.
6. If the brick number <=30, then the alien will act like 2f.1.

Special Requirements:
-The type of the alien is determined randomly.
-There can be only one alien on the game screen at a time.
-Aliens has to be square objects.
-Repairing aliens can rebuild only the simple bricks.
-The picked row for the cooperative alien should have at least one brick.
Frequency of Occurrence: Could be nearly continuous until all the bricks are broken.

Use Case UC10: Quitting the Game

Scope: Game Screen
Level: Subfunction
Primary Actor: Player
Stakeholders and Interests:
-Player wants to quit the game, the player might have finished the game, or the player simply does not want to keep playing.
Preconditions:
-There must be a game that is already on the running mode.
-There must be a way that gets the player out of the game.
-There must be a game screen for the player to choose quit option.
Success Guarantee:
The player decides to quit the game and the game is successfully closed.
Main Success Scenario:
1. Player opens the game screen and pushes the button for quitting the game.
2. System closes the game window.
Extensions:
1a. If the quitting button is not visible to the player, the game will still be on the running mode and the player can’t get out of the game.

1b. If the player clicks another button, the game will still be on running mode.
Special Requirements:
-There must be a button to quit the game.
-This button should be visible to the player.
Frequency of Occurrence: If the player didn't save the game first, the player can get out of the game only one time, but if the game was saved before, the player can get out of game multiple times.

Use Case UC11: Shooting Laser

Scope: Game screen
Level: Subfunction
Primary Actor: Player
Stakeholders and Interests:
-Player: Wants to use the laser gun to destroy full column of blocks.
Preconditions:
-The player successfully logins to the game.
-The player loads a game.
-Game is started with launching the ball.
-The player must have already caught the laser gun power-up and attach it to both ends of the paddle.
Success Guarantee: Player use the laser gun power-up without any errors. When the player uses it successfully, unless he or she hits a metal side of a brick, the player can destroy full column of blocks.
Main Success Scenario:
1. The ball hits a wrapper brick and a Taller Paddle power-up appeared on the screen.
2. Player attach a laser gun power-up to both ends of the paddle.
3. Player hits a brick and destroy a full column of bricks.

Extensions:
5. Player may not be able to attach laser gun power-up to the paddle.
6a. Player may try to hit a brick which has metal sides and ball could hit the metal side so that he or she misses one shot and cannot destroy anything.
6b. Player may spend his or her five shots so that he or she cannot use the power-up.
Special Requirements:
- Player has only 5 shoots for one laser gun power-up. For instance, if the player finds a laser gun power up when he or she has already one, the gun behaves like it has been reloaded. For example, let's say the laser gun has 2 shots left and the player gets another laser gun, then laser gun has 5 shots, not 7.
Frequency of Occurrence: Five times in one power-up.

Use Case UC12: Login to the game

Scope: Starting window
Level: User goal
Primary Actor: Player
Stakeholders and Interests:
-Player: Wants to login to the game with a user ID without any problems.
Preconditions:
- User ID must be unique.
- User ID cannot include any special characters.
-The game must be initialized.
Success Guarantee: Player could login to the game successfully.
Main Success Scenario:
1. Player enters his or her user ID.
2. Player presses the login button.
3. System displays the load screen.

Extensions:

1a. If the user ID is not unique then user can reach his or her existing game loads.
1b. If the user ID is unique then the game creates a new account for the user.
Special Requirements:
1.There should be a rectangular login button that should be visible for the player.
2.There should be a bar where the user enters his or her user ID.
Frequency of Occurrence: Once in a run.

Use Case UC13: Loading the Game

Scope: Game Screen
Level: User goal
Primary Actor: Player
Stakeholders and Interests:
-Player: Wants to load an existing game that created earlier, without any errors by reaching the folder from the game client.
Preconditions:
- Player is identified and authenticated.
- There must be a saved game.
- There must be a directory on the load screen of the game in order to find an existing file.
- The file which player wants to load must be a suitable game file.
- The saved file should be matched with the player ID.
Success Guarantee: Existing and suitable file found and successfully loaded.
Main Success Scenario:
1. Player chooses to load an existing game instead of building a new one by clicking it.
2. Player selects a game to load from database.
3. The system checks if the ID of the player is the same with the credential in the selected game document.
4. System loads the document after checking out
Extensions:
1a. If player clicks on build a new game instead of load game system goes to building mode.
2a. If there is not a game file in the database, a player cannot upload any game.
3a. If the ID is not the same as the one in the credentials, then the player cannot upload the game.
4a. If the checkout fails, then the player cannot upload any game.
Special Requirements:
1. There should be a rectangular button that should be visible for the player.
2. Button should be on the screen that will not affect the player by playing the game (On the side of the screen, preferably on the top left corner).
3. Button should be visible on each screen after login screen.
Frequency of Occurrence: Once for each game.

Use Case UC14: Saving the Game

Scope: Game screen
Level: Subfunction-level
Primary Actor: Player
Stakeholders and Interests:
-Player: Wants to save the game to the file so that the player will be able to keep playing the game from where the player left.
Preconditions:
-User must be logged in.
-User loads a game to play.
-The game must be on the running mode.
-There must be a process for saving our game.

-There must be game screen for the instructions that includes saving feature.
-The file that we are trying to save the game must be a proper game file.
Success Guarantee: The game is not on the running mode anymore, and the game is saved to the proper file.
Main Success Scenario:
1. Player decides to save the game instead of continuing to play.
2. Player opens the game screen and clicks the button for saving the game.
3. The system creates a document with the credentials player's username, brick types/numbers, brick locations, the player's score, player's number of lives, alien types/numbers/positions and power-up types/numbers.
4. The system inserts the created document to the selected collection in the database.
Extensions:
1a. If the player decides to continue to play, the game is on the running mode again.
2a. If the player clicks another button, the game will not be saved.
3a. If any of the credentials is null. Document cannot be created.
3b. If the player has 0 life, then the document cannot be created.
4a. If the document cannot be created, the player cannot insert it to collection.
Special Requirements:
-There must be a button for the player to save the game.
-The button should be visible to the player.
-The button should not affect the player for playing the game. It should not interfere with the game.
Frequency of Occurrence: The player can save the game as many times as possible.

Use Case UC15: Displaying the Help Screen

Scope: Game Screen
Level: subfunction
Primary Actor: Player
Stakeholders and Interests:
-Player: Wants to open the Help Screen for getting knowledge about the game objects, instructions for using the keyboard keys and how to play the game.
Preconditions:
1. Player successfully logins to the game.
2. Player loads a game.
3. There must be a Help Screen, which is designed for covering all the knowledge about game to inform the player.
4. Game must be on running mode for displaying the Help Screen.

Success Guarantee:
Game is paused. Designed Help Screen appeared on the game screen.
Main Success Scenario:
1. Player wants to gain knowledge about game features and click the designated button on the game screen or the "H" button on the keyboard.
2. After clicking the designated button, the help screen appears on the game screen until the player closes it.
3. The player closes the help screen after usage, with the designated button on the game screen or the "H" button on the keyboard.

Extensions:
1a. When player clicked any other button then the buttons, which are mentioned above, the Help Screen will not appear on the game screen.
3a. When player clicked any other button then the buttons, which are mentioned above, the Help Screen will not close and the game will stay on the pause state.
Special Requirements:
-The Helping Screen must contain knowledge about the following features:

a) Keyboard Controls
b) Game Rules
c) Designated symbols for power-ups, bricks and aliens.

Frequency of Occurrence: Controlled by the player, whenever player wants to see the Help Screen.

Operation Contracts:

Contract CO1: Get Help

Operation: getHelp()

Cross References: Displaying the help screen

Preconditions:

-The player must log in to the game.

Postconditions:

-HelpScreen was created (instance creation).

Contract CO2: Quit Game

Operation: quitGame()

Cross References: Quitting the game

Preconditions:

-The player must log in to the game.

Postconditions:

-The game window was closed (instance deletion).

Contract CO3: Create New Game

Operation: createNewGame()

Cross References: Creating a Game with Building Mode

Preconditions:

-Being logged in to the game

-Choosing the option of creating a new game instead of loading a new game

Postconditions:

- Player.life is set to 3 and Player.score is set to 0 (attribute modification).

Contract CO4: Specify Simple Bricks

Operation: specifySimpleBrick(quantity: integer)

Cross References: Creating a Game with Building Mode

Preconditions:

- The player must choose to create a game instead of loading one.

-The player must log in to the game

Postconditions:

- SimpleBrick.number is set (attribute modification).

- Simple Bricks were created in the game world (instance creation).

Contract CO5: Specify Half-metal Bricks

Operation: SpecifyHalfMetalBricks(quantity: integer)

Cross References: Creating a Game with Building Mode

Preconditions:

- The player must choose to create a game instead of loading one.

-The player must log in to the game

Postconditions

- Half-MetalBrick.number is set (attribute modification).

- Half-Metal Bricks were created in the game world (instance creation).

Contract C06: Specify Mine Bricks

Operation: SpecifyMineBrıcks(quantity: integer)

Cross References: Creating a Game with Building Mode

Preconditions:

- The player must choose to create a game instead of loading one.

- The player must log in to the game

Postconditions:

- MineBrick.number is set (attribute modification).

- MineBricks were created in the game world (instance creation).

Contract C07: Specify Wrapper Bricks

Operation: SpecifyWrapperBrıcks(quantity: integer)

Cross References: Creating a Game with Building Mode

Preconditions:

- The player must log in to the game

- The player must choose to create a game instead of loading one.

Postconditions:

- WrapperBrick.number is set (attribute modification).
- Wrapper Bricks were created in the game world (instance creation).

Contract CO8: Initialize Game

Operation:

- initializeGame()

Cross References: Login to the game

Preconditions: None

Postconditions:

- GameScreen was created. (instance creation)

Contract CO9: EnterId

Operation: EnterId(userID: UserID)

Cross References: Login to the game

Preconditions:

-User ID must be unique.

-User ID can not include any special character.

Postconditions:

- Player class got created (instance creation).
- Player.userId is set (attribute modification).

Contract CO10: Launch Ball

Operation:launchBall()

Cross References: Launching the Ball

Preconditions:

- The player must log in to the game.
- The game must be on running mode
- The ball must be on the paddle steady.

Postconditions:

-Ball.velocity and Ball.direction is set (attribute modification).

Contract CO11: Move Paddle

Operation:movePaddle(direction: Direction)

Cross References: Hitting the Ball

Preconditions:

- The player must log in to the game.
- The game must be on running mode.
- Paddle must move within the boundaries.

Postconditions:

- Paddle.velocity and Paddle.direction is set (attribute modification).

Contract CO12: Rotate Paddle

Operation: rotatePaddle(direction: Direction)

Cross References: Rotating the Paddle

Preconditions:

-The player must log in to the game.

-The game must be on running mode.

Postconditions:

- Paddle.angle is set (attribute modification).

Contract CO13: Pick-up Power

Operation: pickUpPower()

Cross References: Using the Power-ups

Preconditions:

-The player must log in to the game

-The game must be on running mode

-There must a power-up appearance in the game screen.

Postconditions:

- powerUpBad.add(powerUp) new power up added to list (instance creation).

Contract CO14: Use Taller Paddle

Operation: useTallerPaddle()

Cross References: Using the Power ups

Preconditions:

-The player must log in to the game.

-The game must be on running mode.

-The power-up store must contain Taller Paddle power-up.

Postconditions:

- Paddle.length is set (attribute modification).

Contract CO15: Save Game

Operation: saveGame()

Cross References: Saving the Game

Preconditions:

- The player must log in to the game.
- The game must be paused by the player.

Postconditions:

- Document was created (instance creation).
- The credentials of the game were appended in Document class (attribute association).
- Collection class was created (instance creation).
- Collection.add(Document) a document is set to collection (attribute association).

Contract CO16: Load Game

Operation: loadGame()

Cross References: Loading the game

Preconditions:

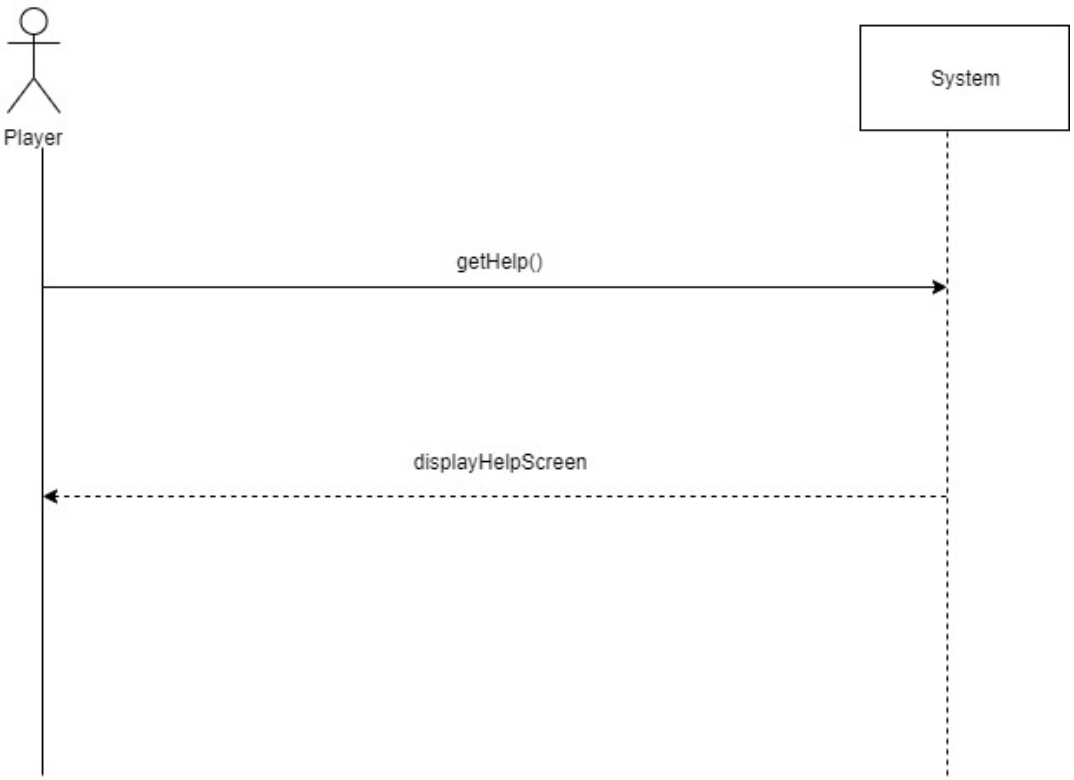
- The player must choose load game instead of new game.
- The player's ID must be the same with the one in the Document credentials.

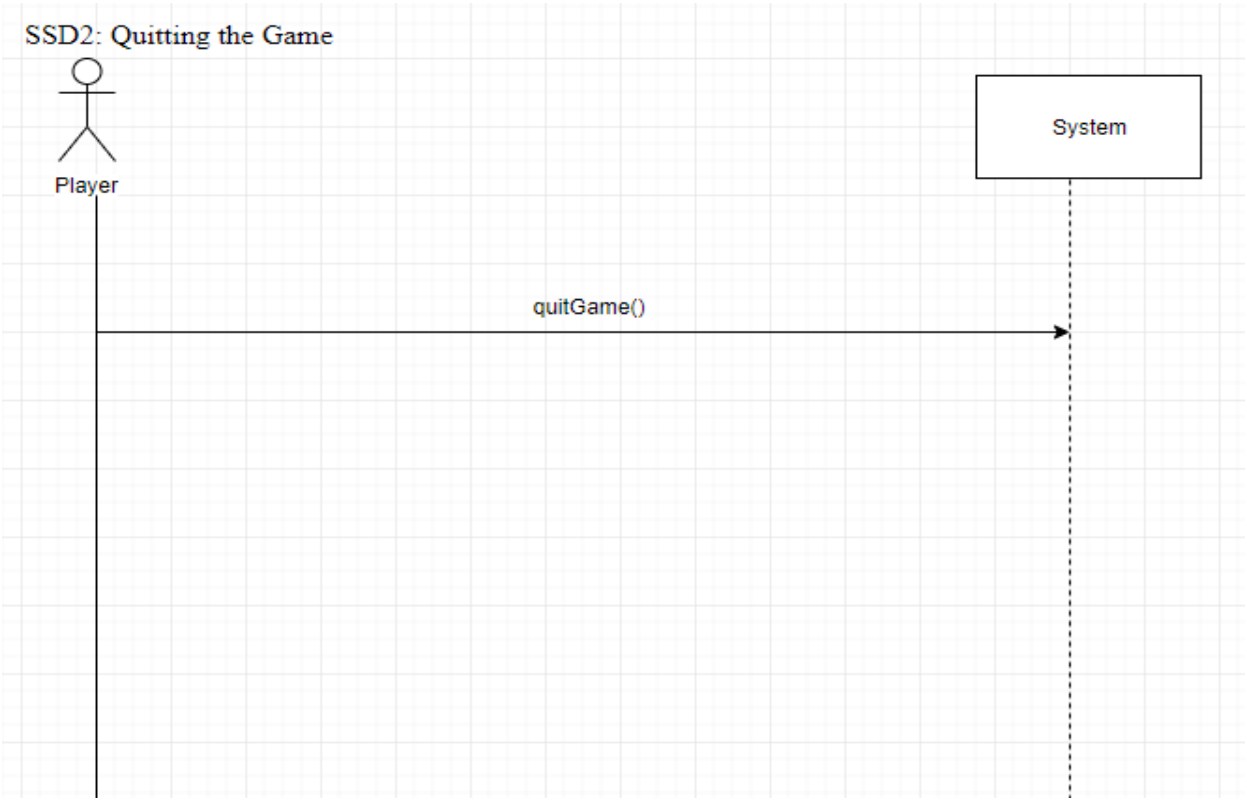
Postconditions:

- The Document of the game was changed into the checked Document (attribute modification).

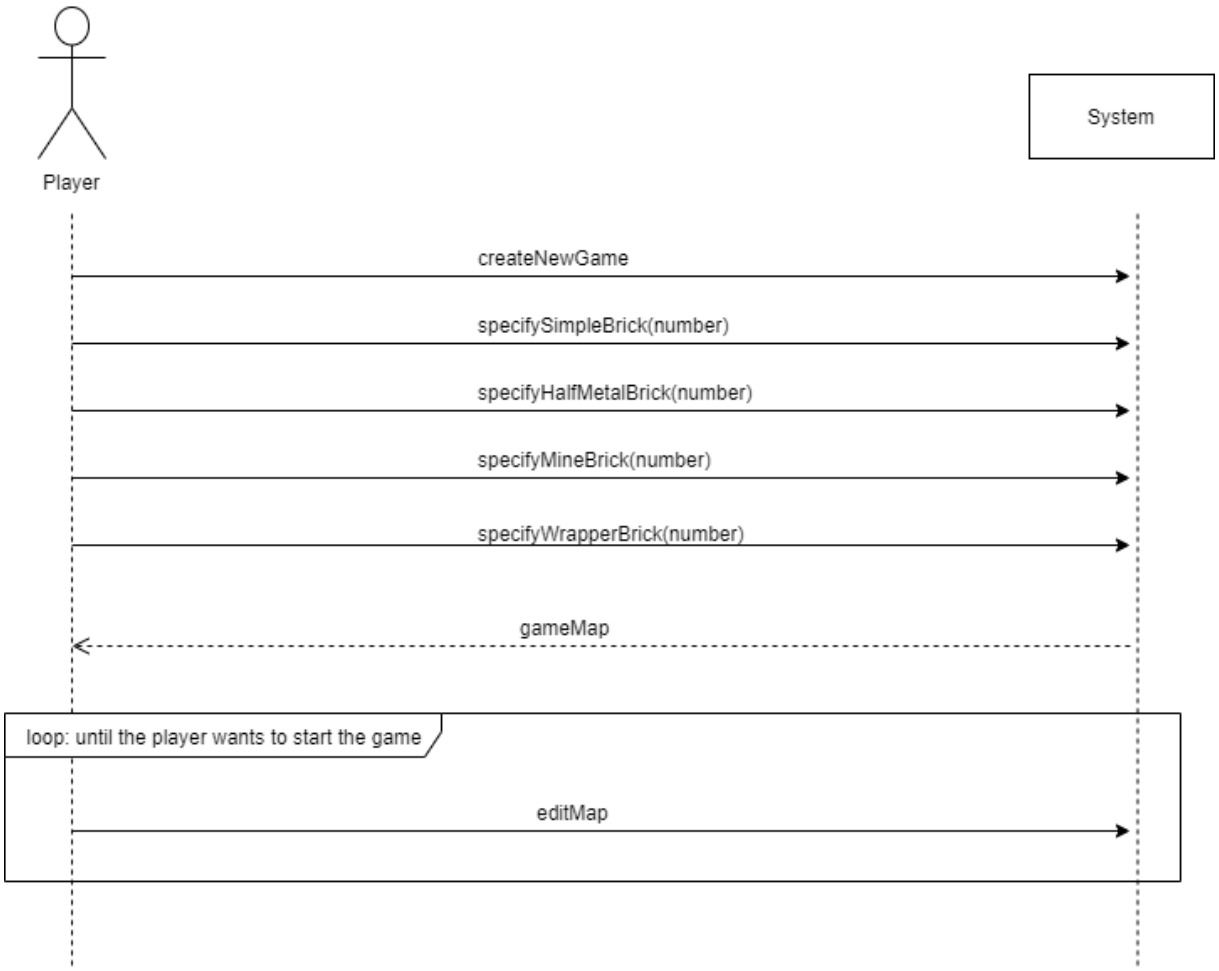
System Sequence Diagrams:

SSD1: Displaying the Help Screen

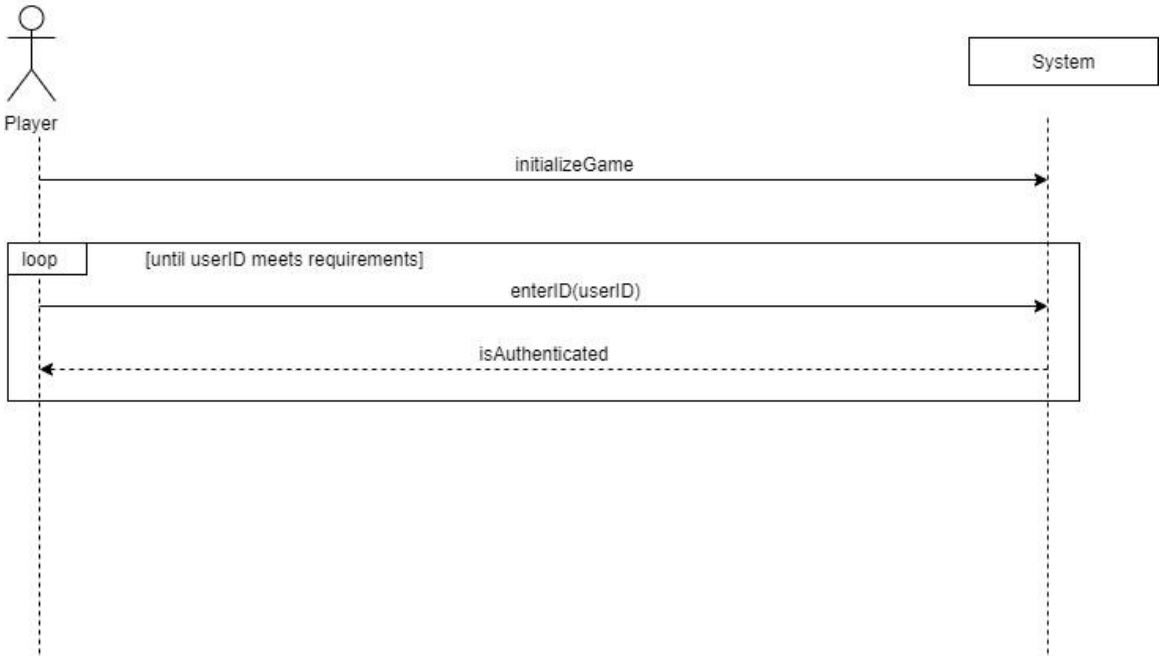



SSD2: Quitting the Game
Player
System
quitGame()

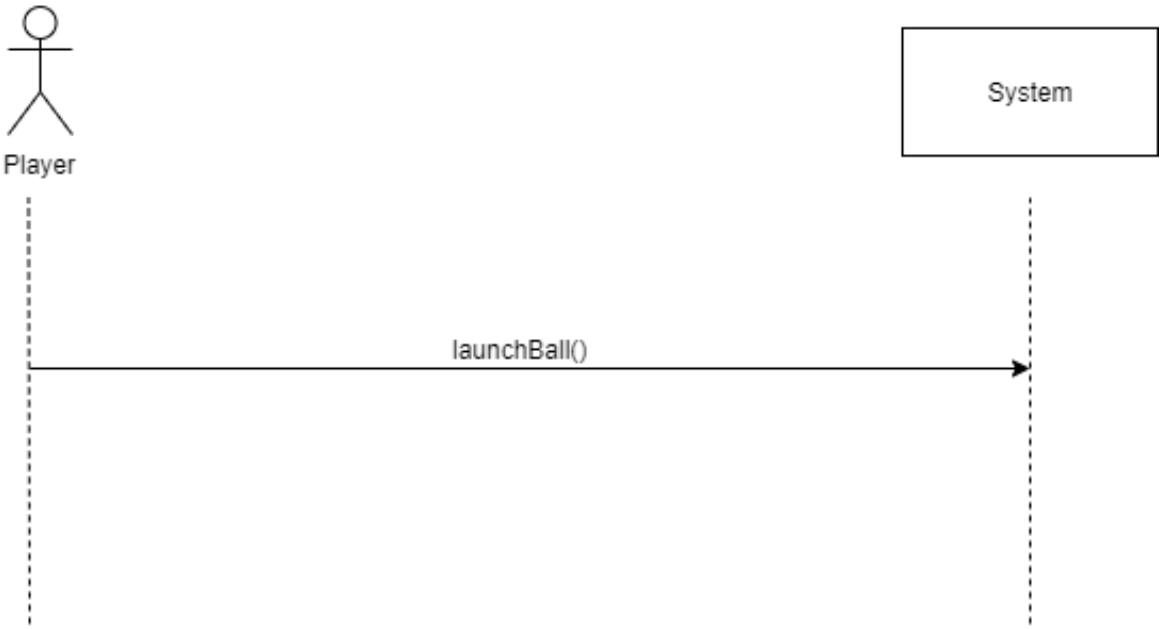
SSD3: Creating a game with building mode



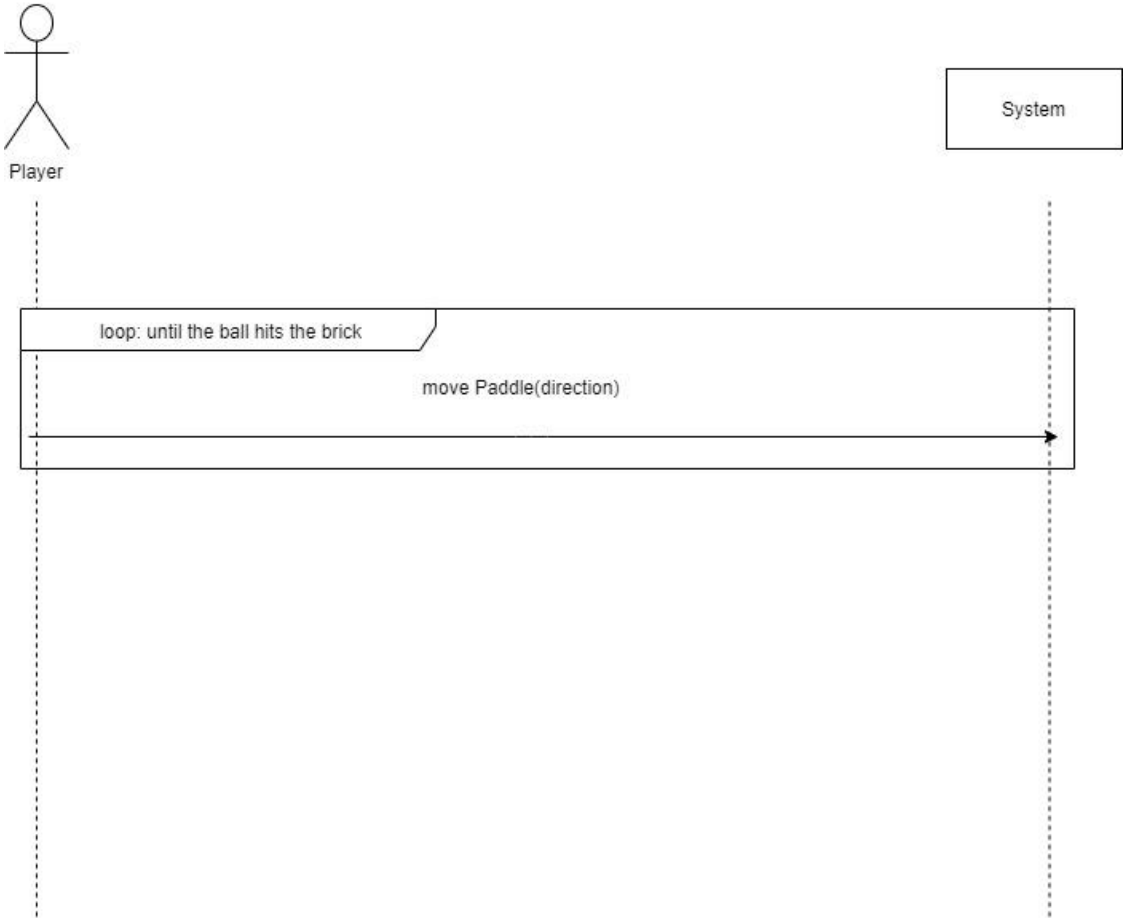
SSD4: Login to the game



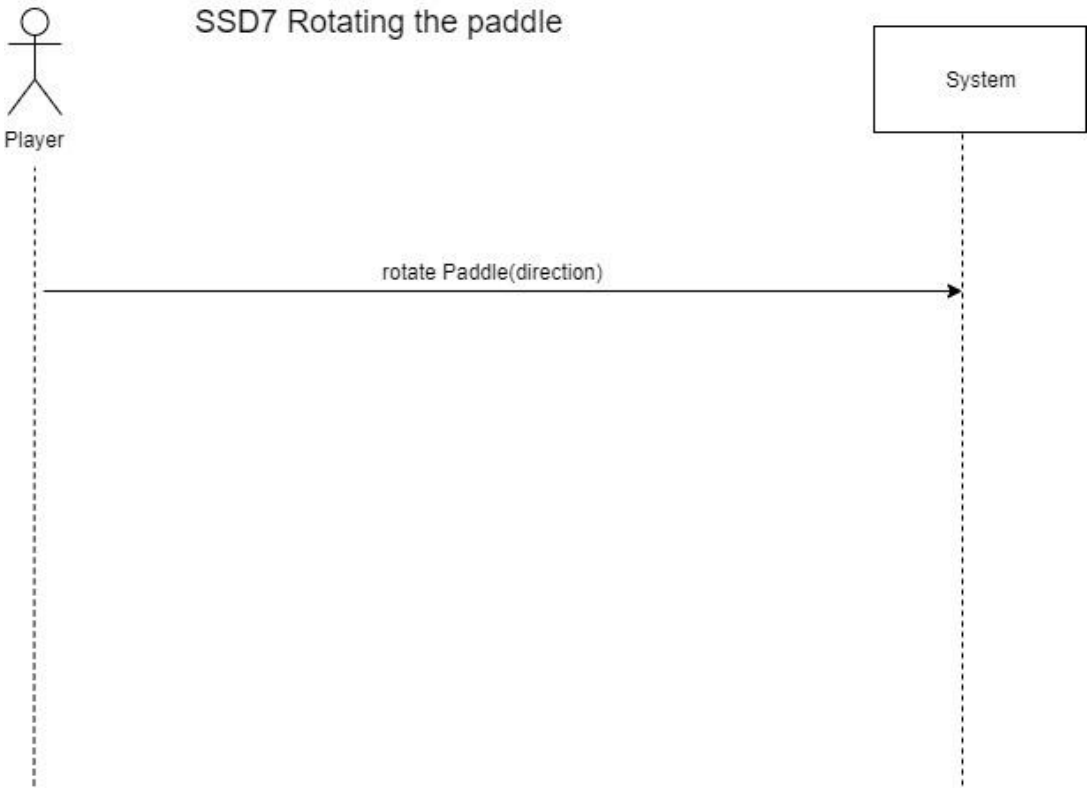
SSD5: Launching the Ball



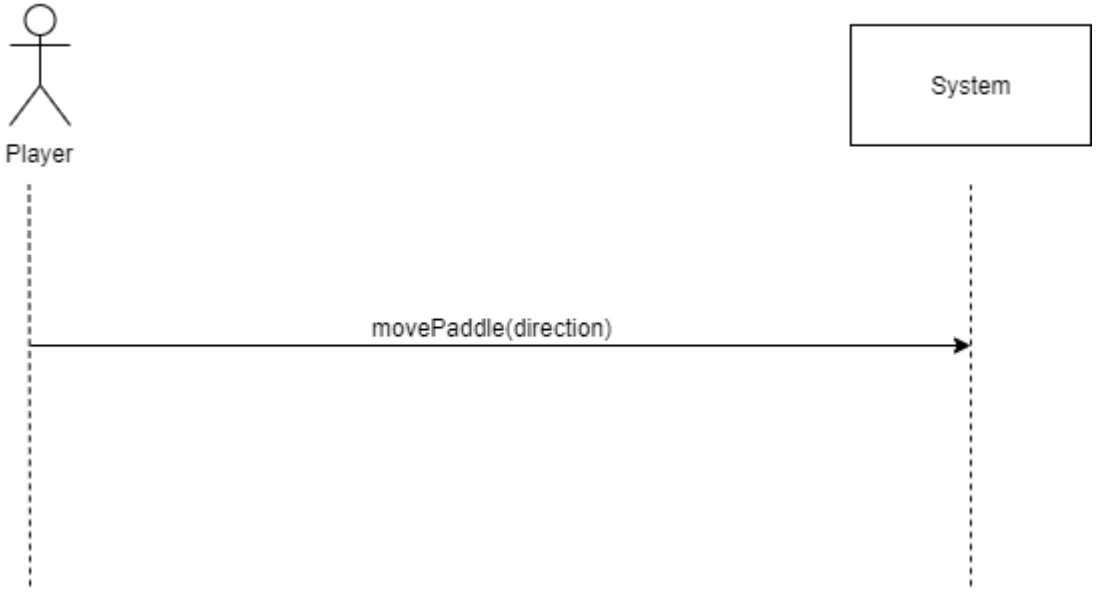
SSD 6 Hitting the Ball

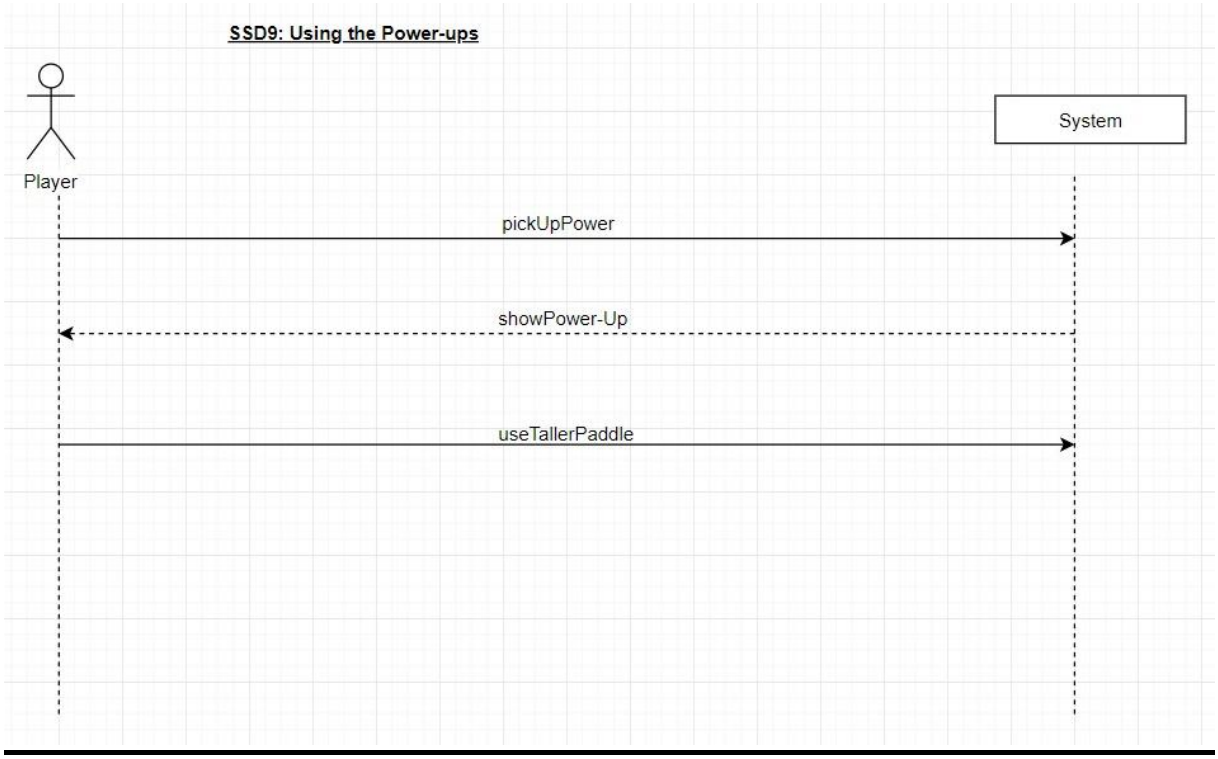


SSD7 Rotating the paddle

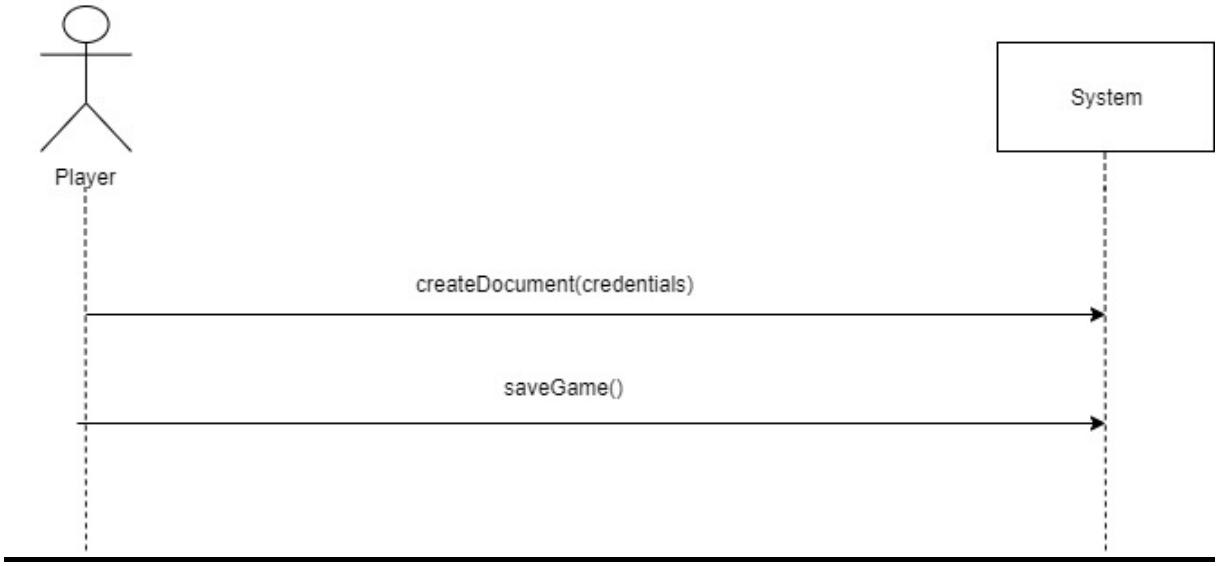


SSD8: Moving the Paddle

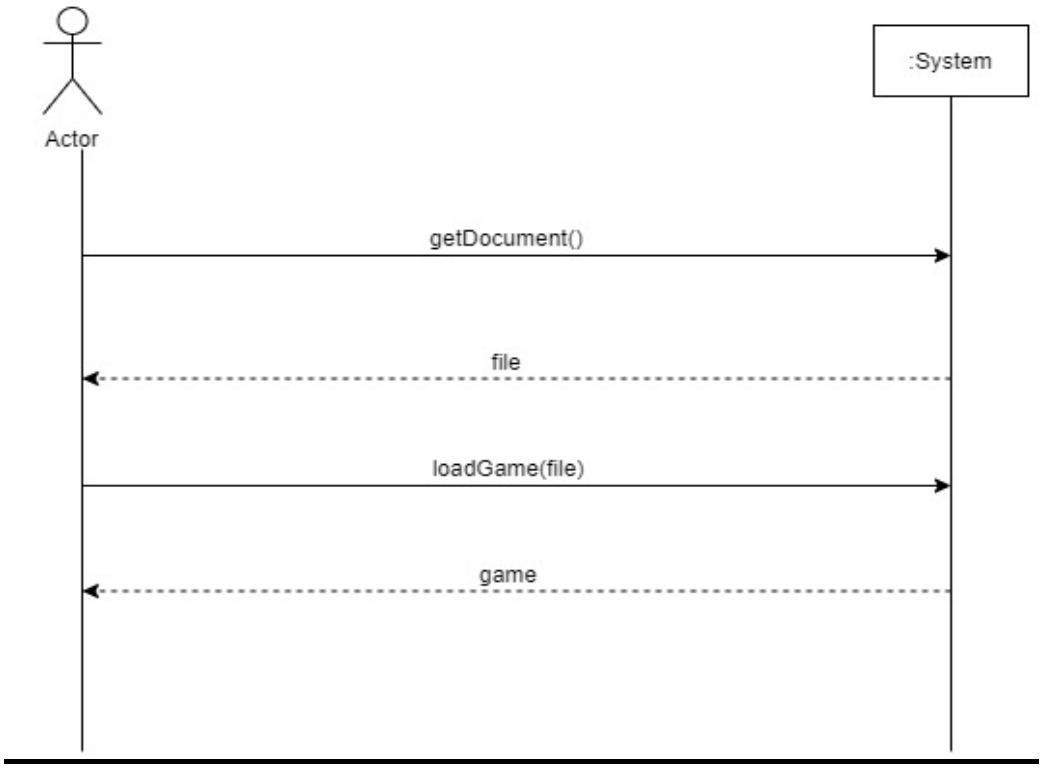




SSD 10: Saving the Game

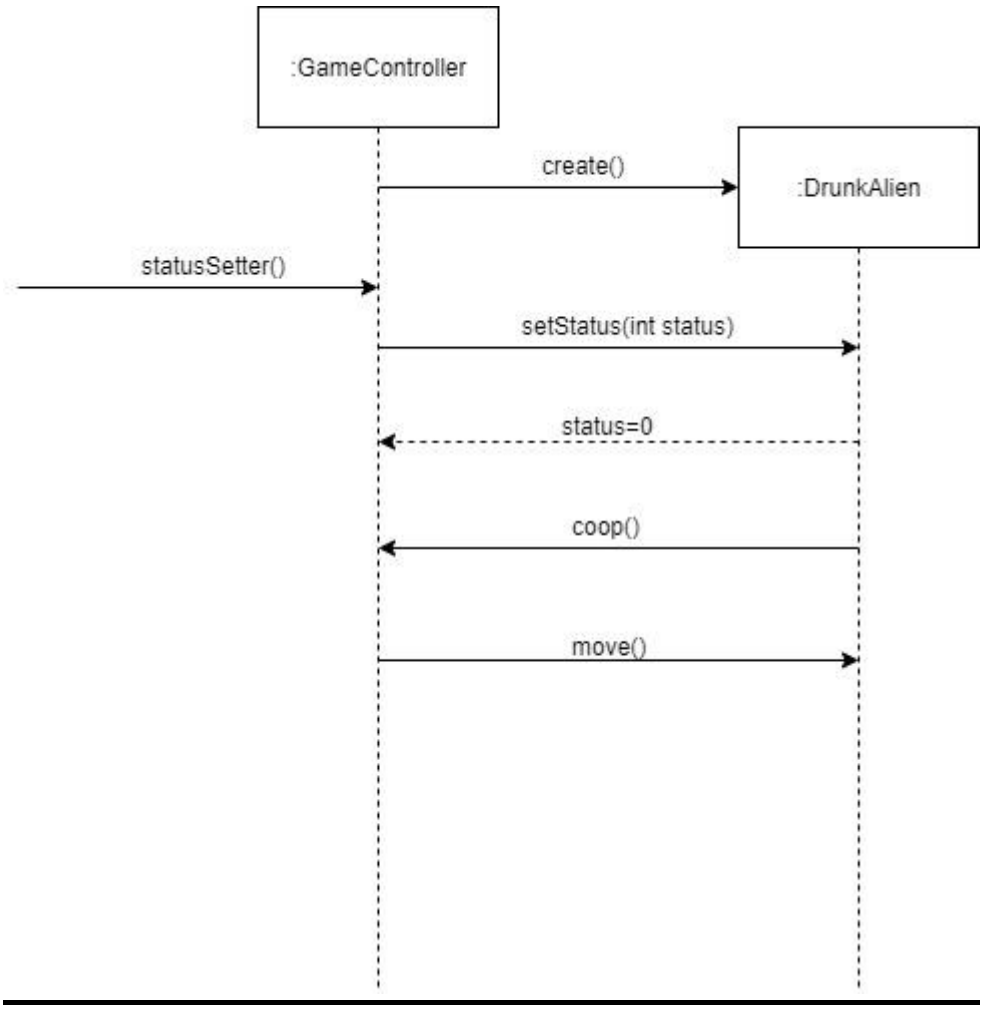


SSD 11: Load Game

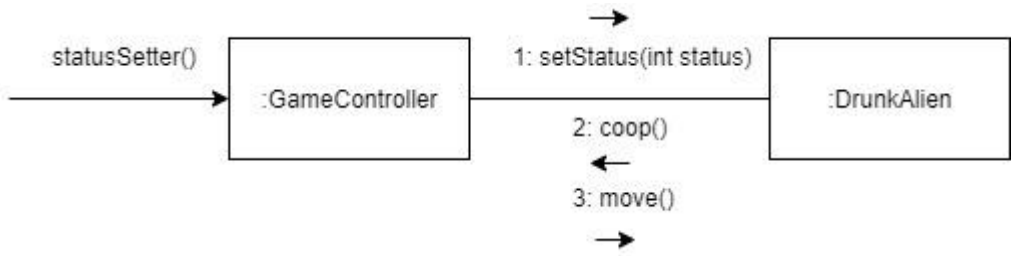


Sequence and Communication Diagrams:

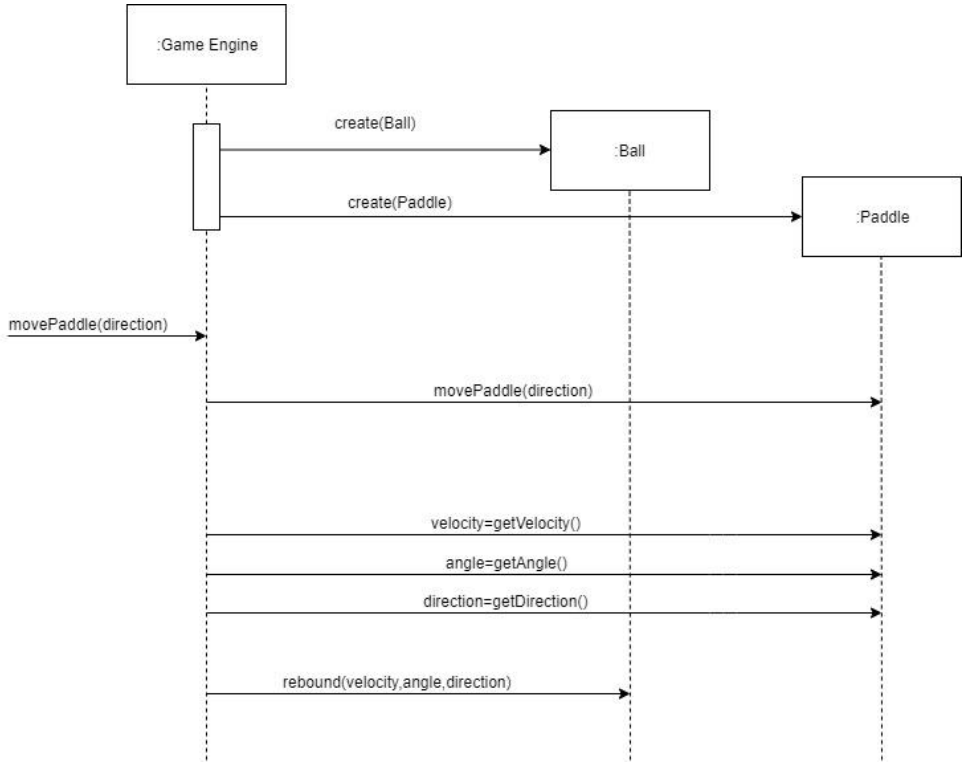
Drunk Alien Sequence Diagram:



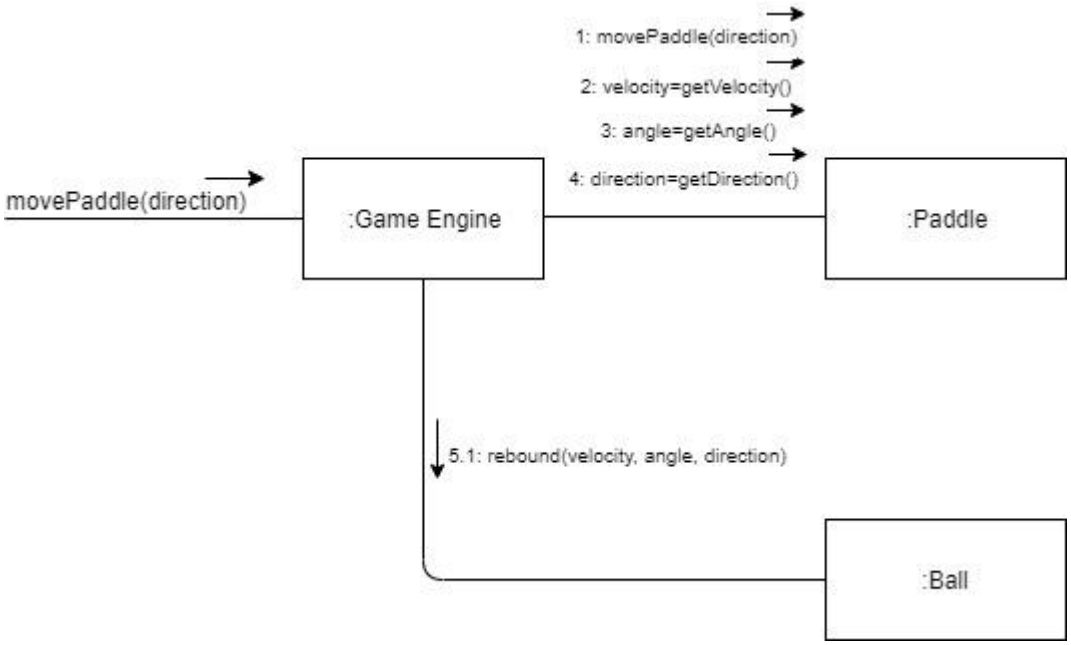
Drunk Alien Communication Diagram:



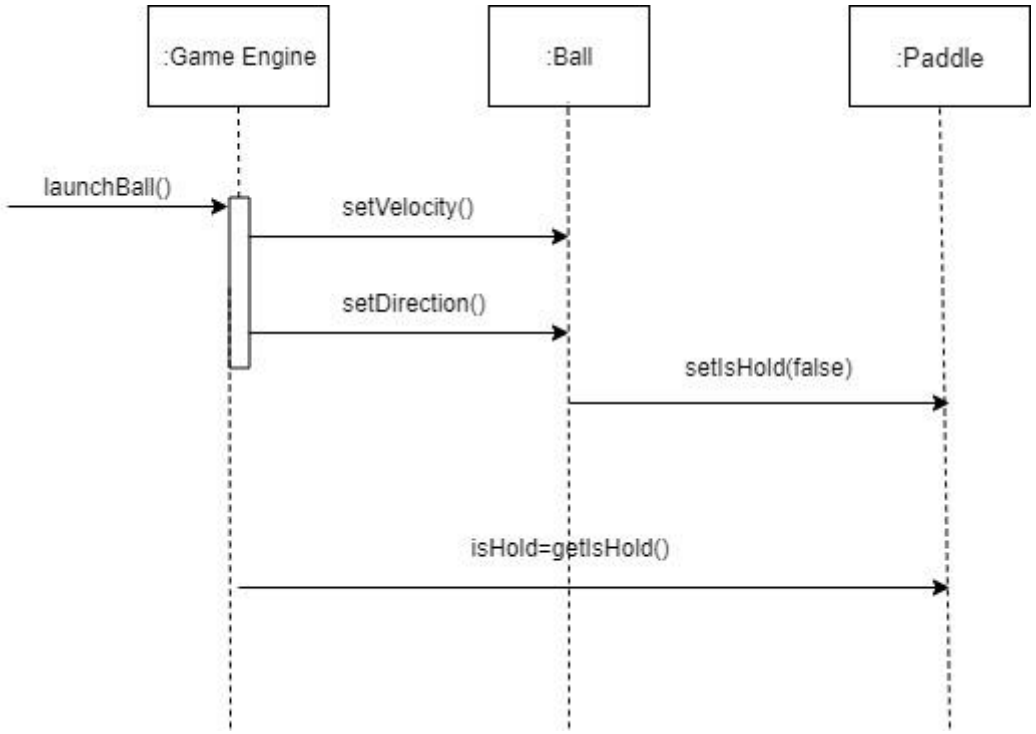
Hitting the Ball Sequence Diagram:



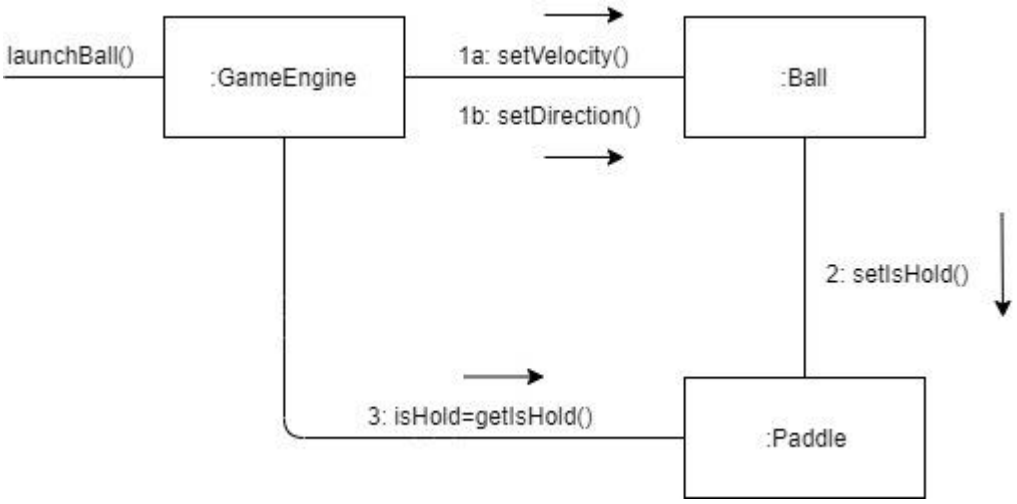
Hitting the Ball Communication Diagram:



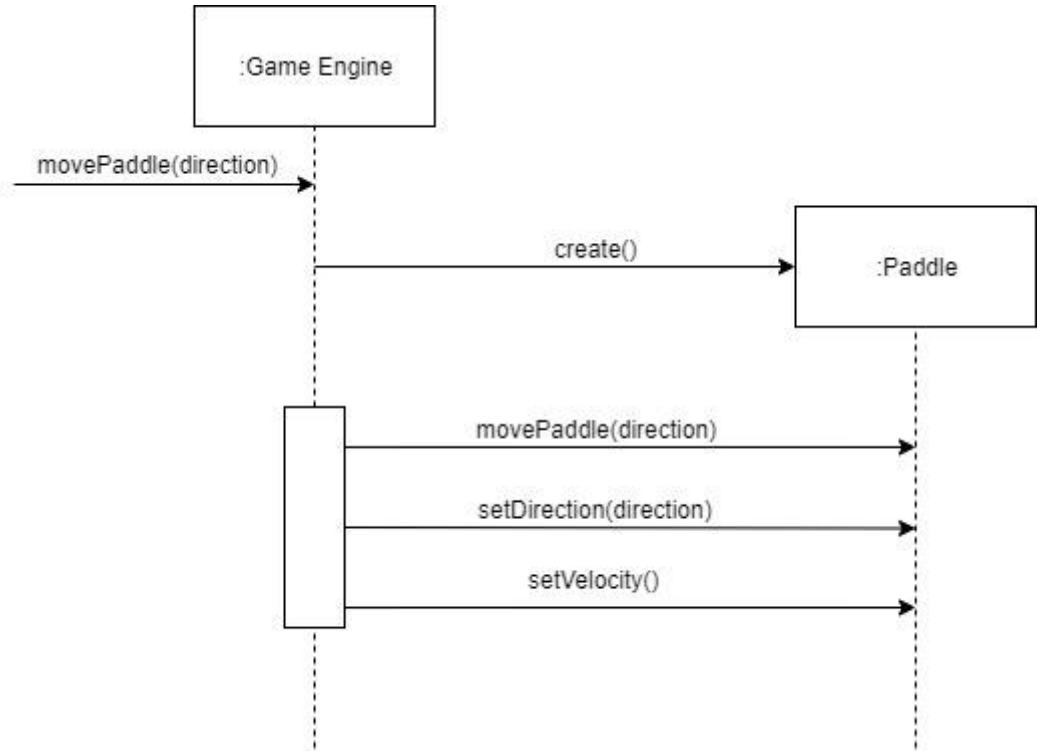
Launching the Ball Sequence Diagram:



Launching the Ball Communication Diagram:

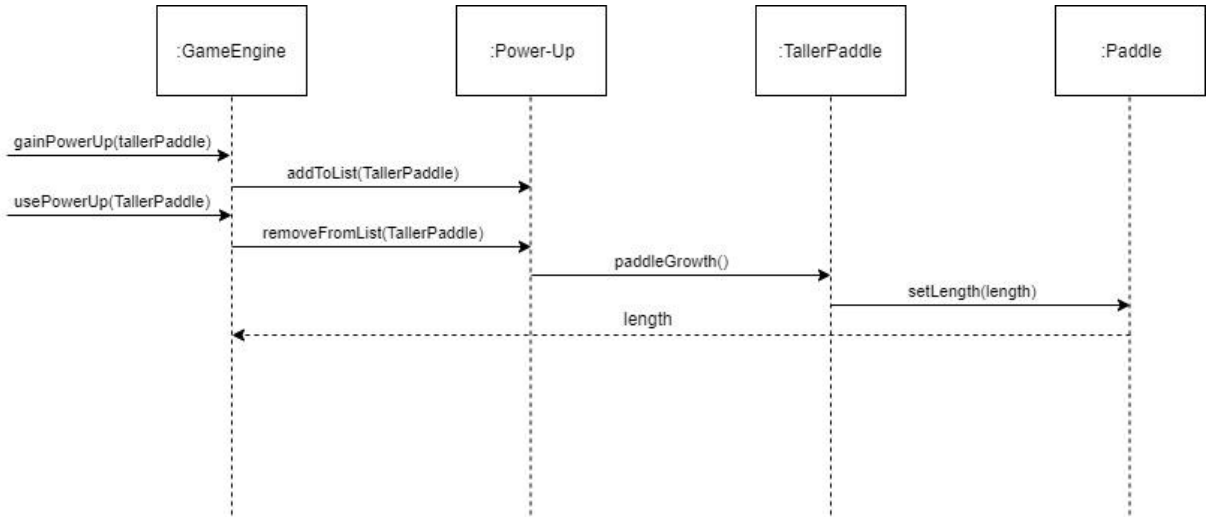


Move Paddle Sequence Diagram:

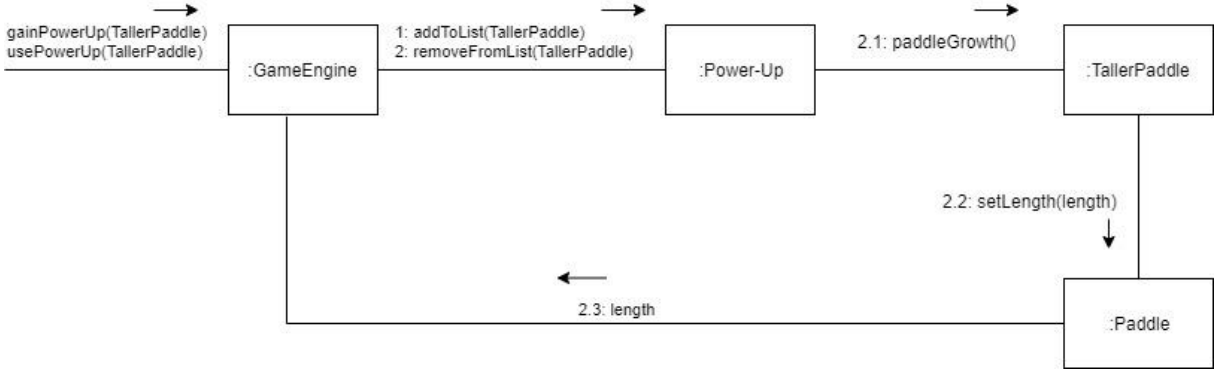


Move Paddle Communication Diagram:

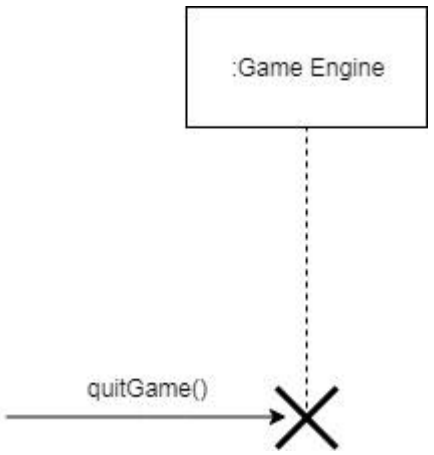
Power Up Sequence Diagram:



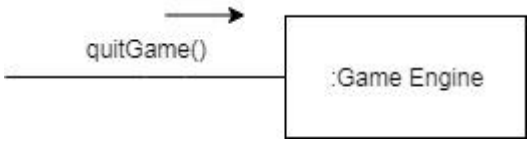
Power Up Communication Diagram:



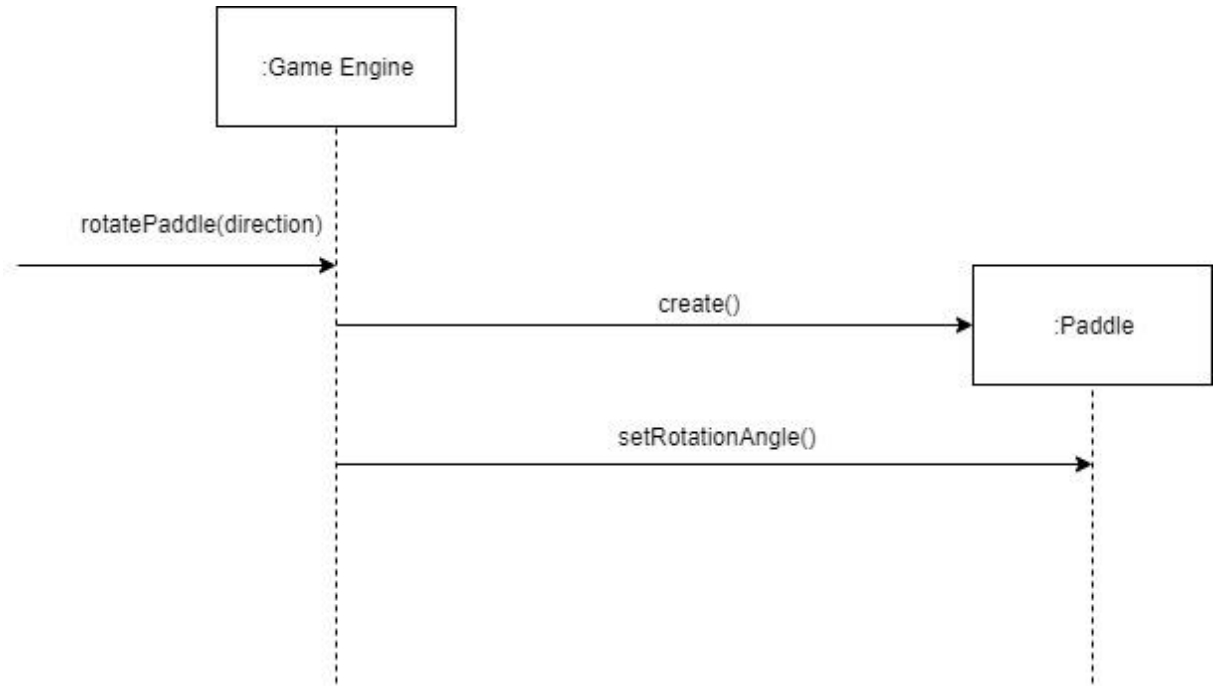
Quit Game Sequence Diagram:



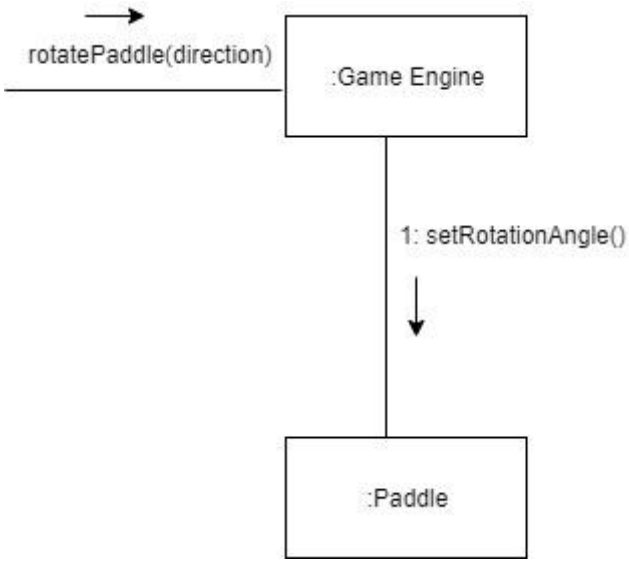
Quit Game Communication Diagram:



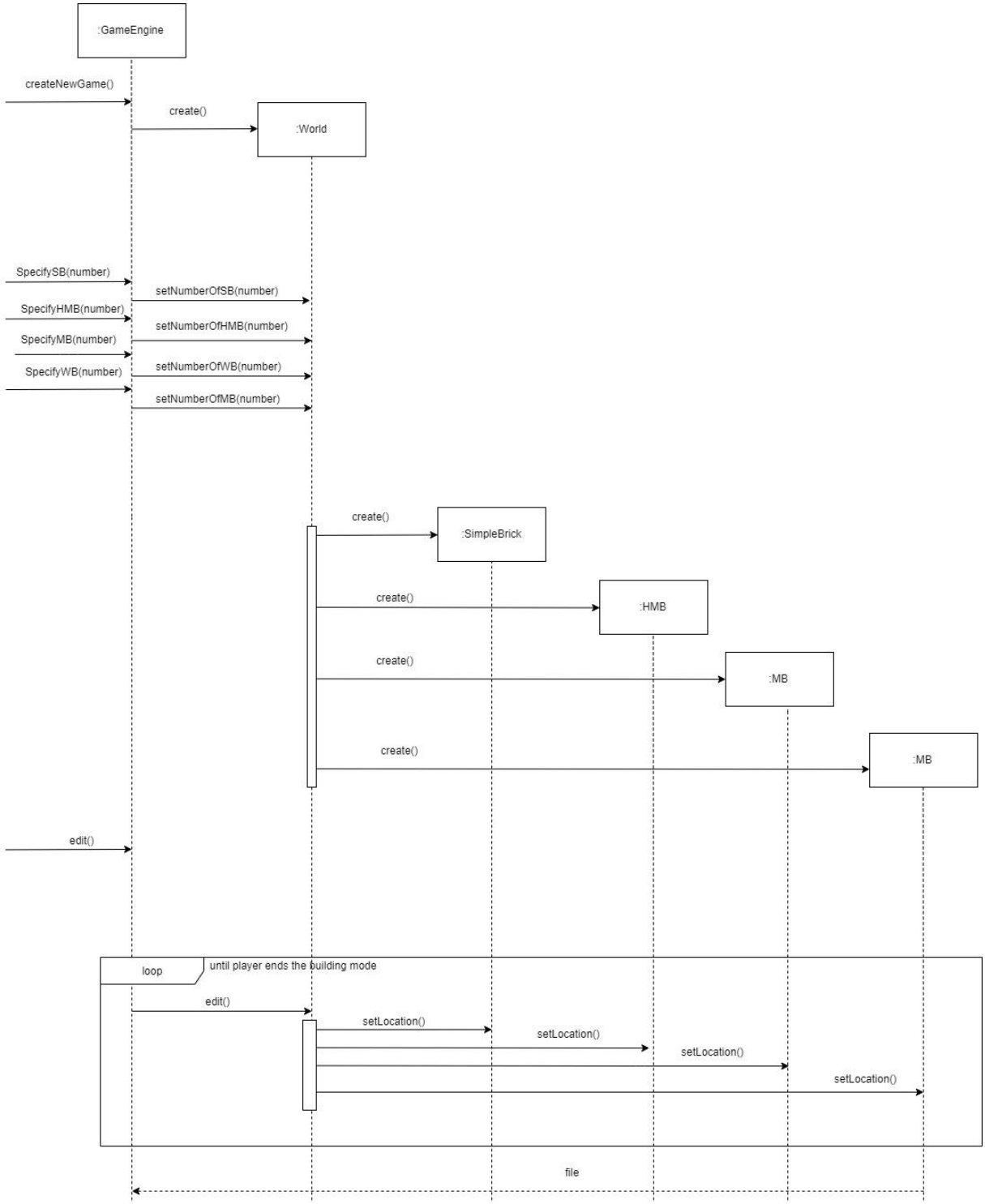
Rotate Paddle Sequence Diagram:



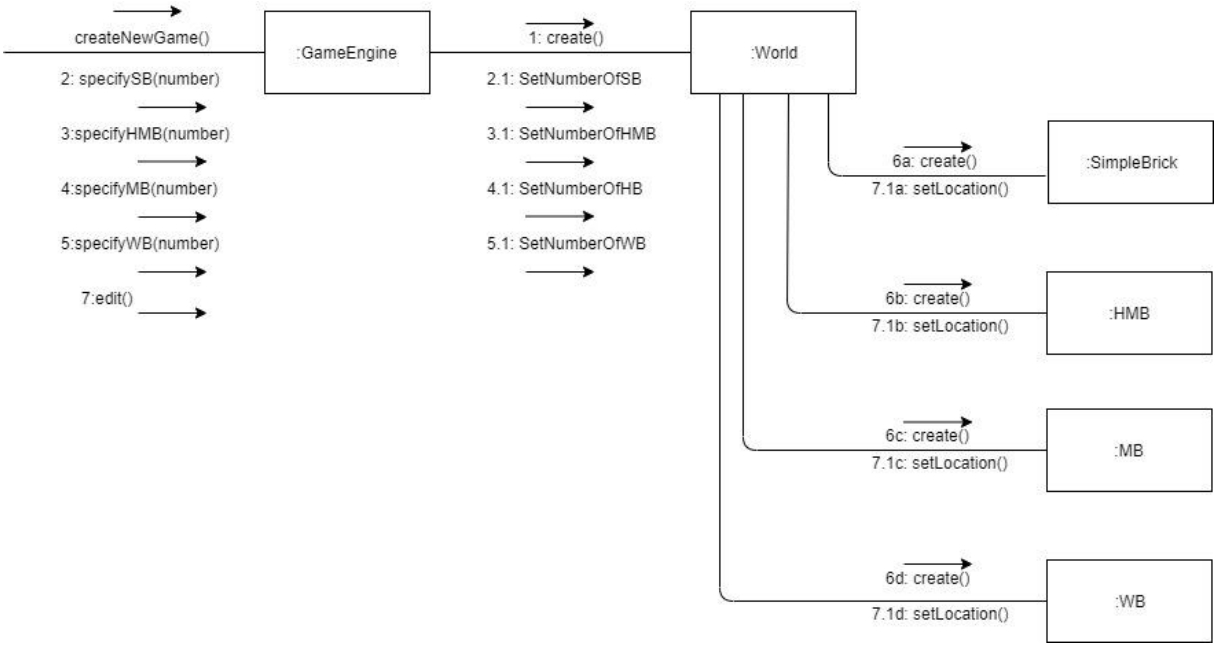
Rotate Paddle Communication Diagram:



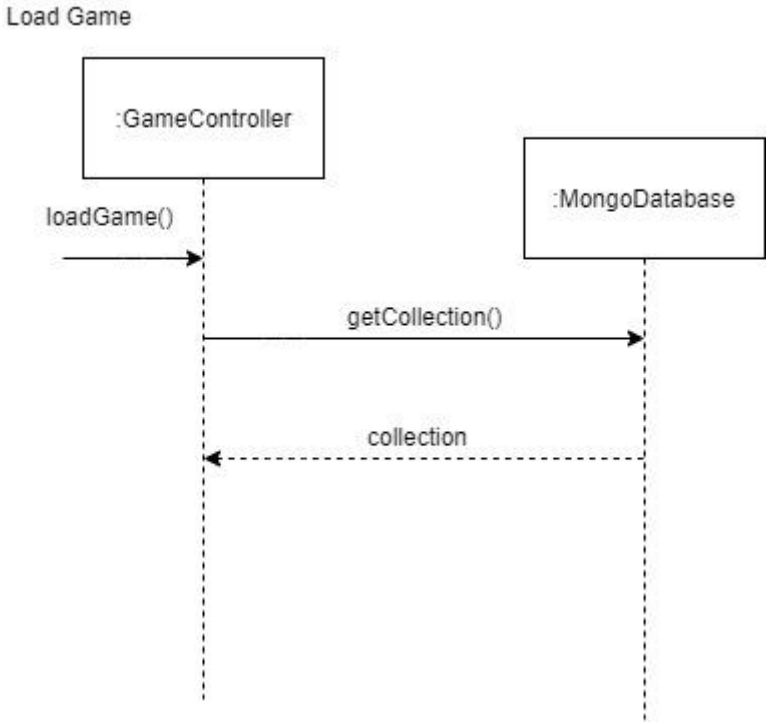
Building Mode Sequence Diagram:



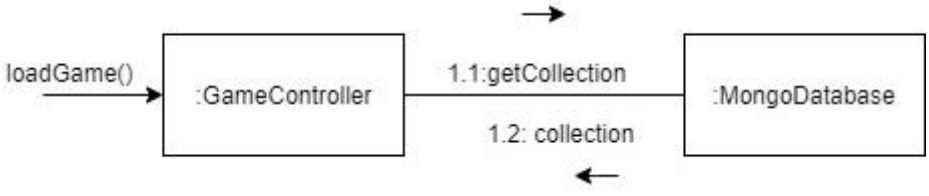
Building Mode Communication Diagram:



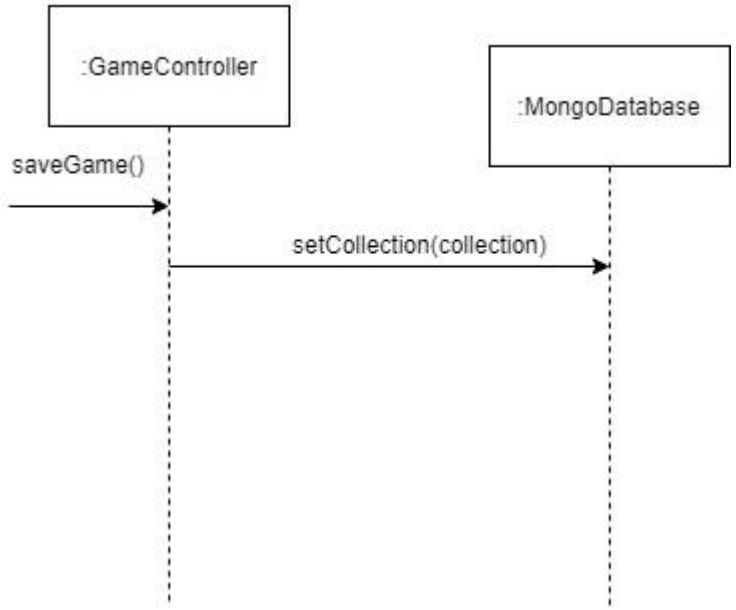
Load Game Sequence Diagram:



Load Game Communication Diagram:

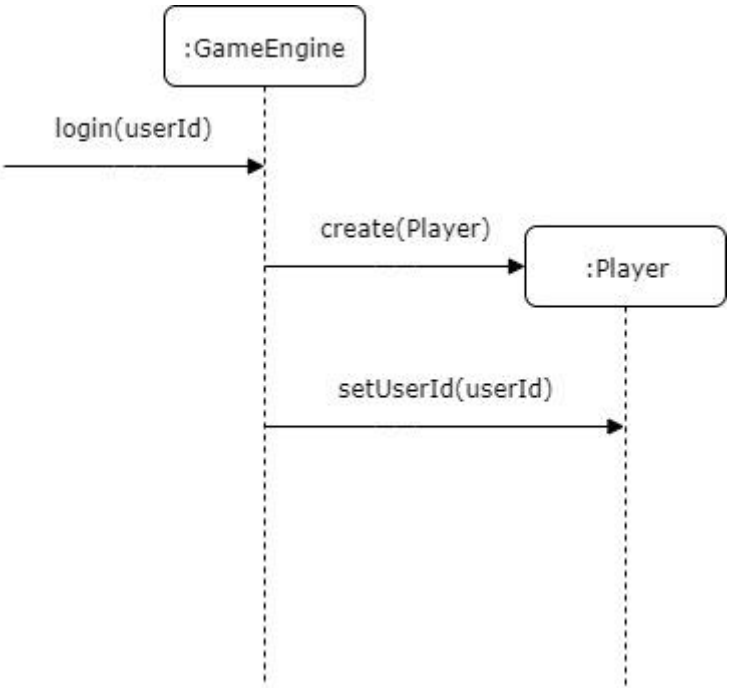


Save Game Sequence Diagram:

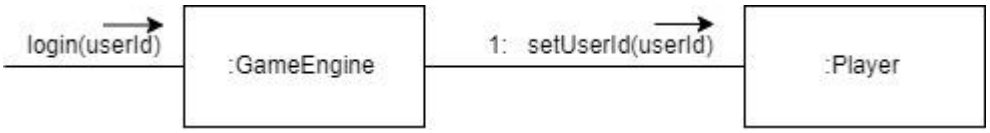


Save Game Communication Diagram:

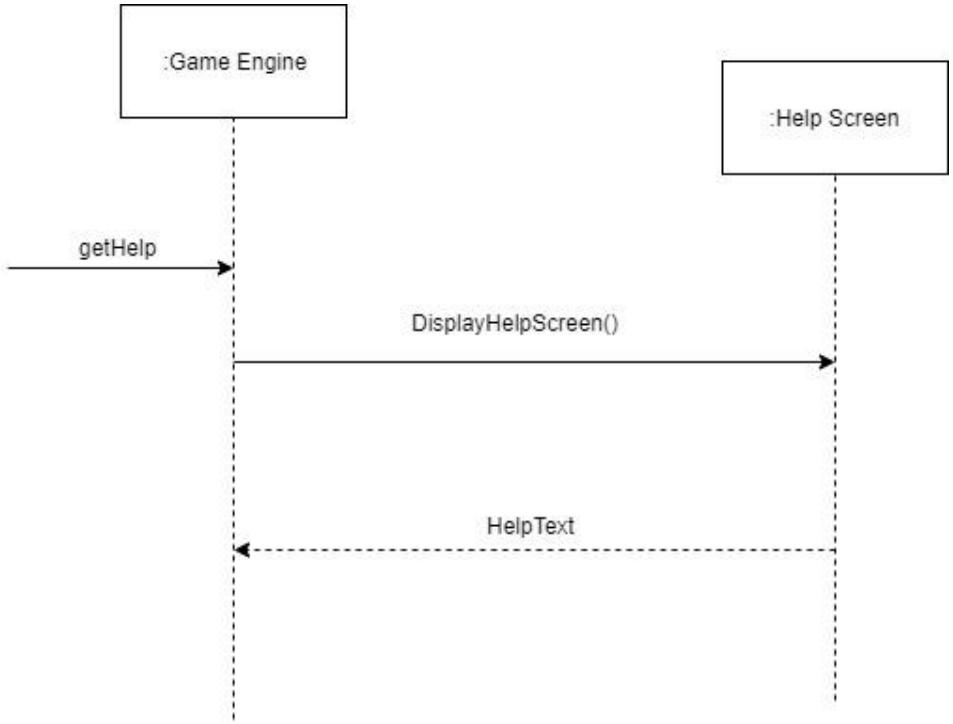
Login Sequence Diagram:



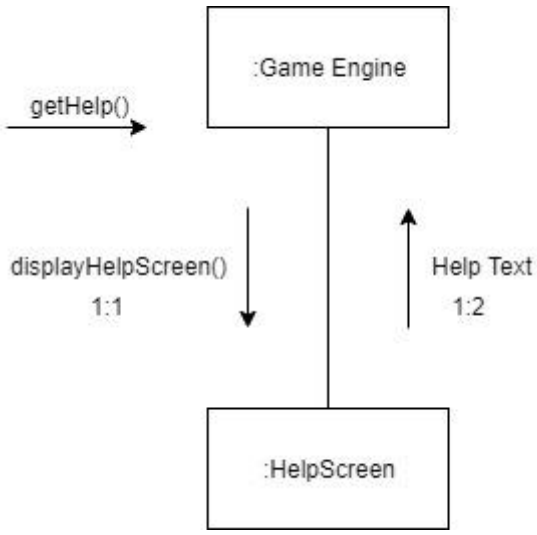
Login Communication Diagram:



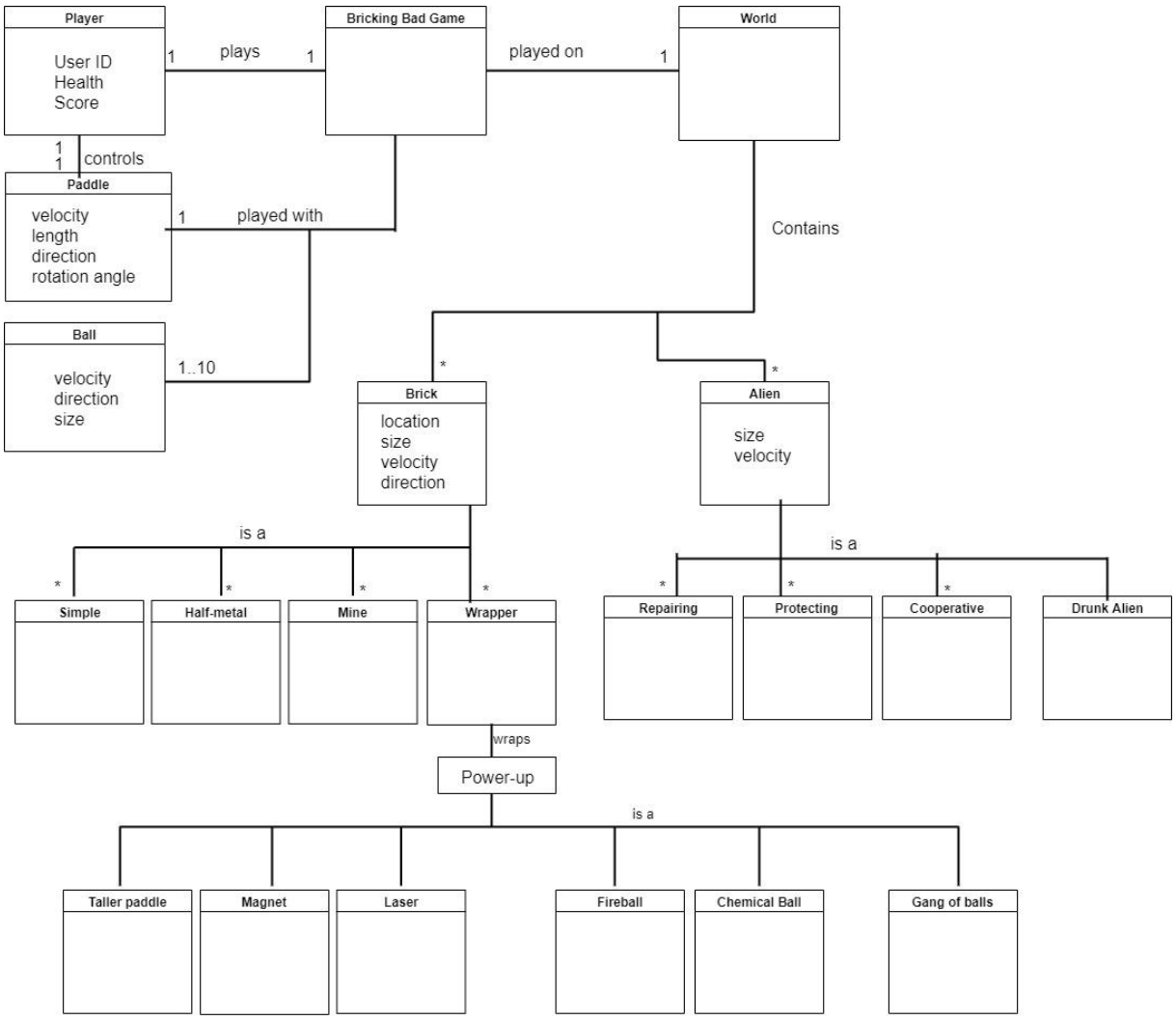
Help Screen Sequence Diagram:



Help Screen Communication Diagram:



Domain Model:



UML Class Diagram:

Board

-Ball : ball
-Paddle: paddle
-Player: player
-PlayGround: playground
+Alien: alien
-sbNumber: Int
-hmbNumber: Int
-mbNumber: Int
-wbNumber: Int
+xPositions: Int[]
+yPositions: Int[]
-numberBrick: Int
-rand: Random
+simpleBrickList: ArrayList<Brick>
+halfMetalBrickList: ArrayList<Brick>
+mineBrickList: ArrayList<Brick>
+wrapperBrickList: ArrayList<Brick>
+powerup: ArrayList<PowerUp>
+lasergun: ArrayList<LaserGun>
-isStatus3User: Int =1
+alienType: Int
+pocket: Int[]
-activeLaserGun: Bool==false
-totalSize: Int
+currentSize: Int

+<<construtor>>Board(sbNumber: Int, hmbNumber: Int, mbNumber: Int, wbNumber: Int)
+getLife()
+loseBall()
+movePowerUp()
+launchBall()
+moveBallX
+moveBallY()
+rotatePaddle(input: String)
+movePaddle(input: String)
+destroy(list: ArrayList<Brick>, i: Int)
+brickList()
+metalBrickList()
+mineBrickList()
+wrapperBrickList()
+powerup()
+rebound()
+statusSetter()
+coop()
+repair()
+protectAndRepair()
+laserShot()
+useMagnet()
+useChemicalBall()
+getBrickX(br: Brick)
+getBrickY(br: Brick)
+getXpositions()
+getyPositions()
+getP()
+getB()
+getUser()
+getNumberBrick()
+setNumberBrick(numberBrick: Int)
+getPaddleXPosition()
+getPaddleYPosition()
+gameAlien()
+moveAlien()
+getScore()
+activateLaserGun()
+useLaserGun()
+laserY()
+laserX()
+lasergunlist()
+activateTallerPaddle()
+alien()
+updateSize()

Game Controller

+board: Board

+<<construtor>>GameController(sbNumber: Int, hmbNumber: Int, mbNumber: Int, wbNumber: Int)
+getLife()
+loseBall()
+launchBall()
+moveBallX()
+gameAlien()
+getBrickList()
+metalBrickList()
+mineBrickList()
+wrapperBrickList()
+powerUpList()
+moveBallY()
+movepowerup()
+rotatePaddle(input: String)
+movePaddle(input: String)
+destroy(list: ArrayList<Brick>, i: Int)
+rebound()
+useChemicalBall()
+useMagnet()
+getNumberBrick()
+setNumberBrick(numberBrick: Int)
+getPaddleXposition()
+getPaddleYposition()
+moveAlien()
+statusSter()
+coop()
+laserShot()
+activateLaserGun()
+useLaserGun()
+laserY(i: Int)
+laserX(i: Int)
+lasergunlist()
+activateTallerPaddle()
+getScore()
+alien()
+updateSize()

ButtonDisplay

-quitButton
-loadButton
-saveButton
-pauseButton
-helpScreenButton

PaddleDisplay

-location
-angle
-size

BallDisplay

+xPos: Int=396
+yPos: Int=521
+xVel: Int=0
+yVel: Int=0

+<<constructor>> BallDisplay()
+drawBall(p: Graphics2D)

GameScreen

BrickDisplay

-field: type
-location

+getLocation
+setLocation

PowerUpDisplay

-field: type
-location
-amount

AlienDisplay

-field: type
-location

HelpScreen

-text

LoginScreen

-loginButton
-textBox

Playground

-width: Int=692
-length:Int=592
-upperbound: Int=0
-lowerbound: Int=592
-leftbound: Int=0
-rightbound: Int=692
-Random: rand
-BrickPositions: brickPos
-bricks:ArrayList<BrickPositions>

+<<Constructor>> Playground()
+getWidth()
+setWidth(width: Int)
+getLength()
+setLength(length: Int)
+getUpperBound()
+setUpperBound(upperbound: Int)
+getLowerBound()
+setLowerBound(lowerbound: Int)
+getLeftBound()
+setLeftBound(leftbound: Int)
+getRightBound()
+setRightBound(rightbound: Int)
+getBricks()
+setBricks(bricks: ArrayList<BrickPositions>)

BrickPositions

-x: Int
-y: Int

+<<Constructor>> BrickPositions(x: Int, y: Int)
+getX()
+setX(x: Int)
+getY()
+setY(y: Int)

<<interface>>

Brick

+getXpos()
+getYpos()
+getLength()
+getWidth()
+getXRightSide()
+getYBottomSide()
+setXpos(x: Int)
+setYpos(y: Int)

BrickFactory

+createBrick(type: String, xPos: Int, yPos: Int, length: Int, width: Int

Ball

-velocity: Int
-xDir: Int
-yDir: Int
-radii: Int
-isChemical: Bool
-xposition: Int
-yposition:Int
-isFireball: Bool
-playground: Playground

+<<constructor>> Ball(velocity: Int, xDir: Int, yDir: Int, xPosition: Int, yPosition: Int, radii: Int)
+nextPosX()
+nextPosY()
+getVelocity()
+setVelocity()
+moveX(int)
+moveY(int)
+getxDir()
+setxDir()
+getyDir()
+setyDir)
+getRadii()
+SetRadii()
+getxPosition()
+setyPosition()
+getxPosition()
+setyPosition()
+isChemical()
+setChemical(isChemical: Bool)
+isFireball()
+setFireball(fireball: Bool)
+repOk()
+toString()

<<actor>>
Player

1

-Player: instance
-life: Int=3
-score: Int=0
-ammo: Int=5

+<<constructor >>Player()
+getInstance()
+getLife()
+setLife(life: Int)
+getScore
+setScore(score: Int)
+getAmmo()
+setAmmo(ammo: Int)
+repOk()
+toString()

<<interface>>

IMongoDBAdapter

+getDataBase()
+setDataBase(database: MongoDataBase)

MongoDataBase

+collection: MongoCollection<Document>

+getCollection()
+setCollection()
+getDataBase()
+setDataBase(database: MongoDataBase)

Paddle

-velocity: Int=20
-length: Int
-width: Int
-direction: Int=0
-xPositions: Int[]
-yPositions: Int[]
-copyx: Int[]
-copyy: Int[]
-xposition: Int=346
-yposition: Int=530
-rotationAngle: Int
-isMagnet: Bool=false
-isTaller()
-activeLaserGun: Bool=false
-isLaunched: Bool
-status: Int=0;

+<<constructor>> Paddle(velocitiy: Int, length: Int, width: Int, direction: Int, xPositions: Int[], yPositions: Int[], xposition: Int, yposition: Int, rotationAngle: Int)
+rotate(angleSelection: Int)
+move(direction: Int, speed: Int)
+getCopyx()
+getCopyy()
+setCopyx(copyx: Int[])
+setCopyy(copyy: Int[])
+getXpositions()
+getYpositions()
+setXpositions(xpositions: Int[])
+setYpositions(ypositions: Int[])
+getVelocity()
+setVelocity(velocity:Int)
+getLength()
+setLength(length: Int)
+getWidth()
+setWidth(width: Int)
+getDirection()
+setDirection(direction: Int)
+getXPosition()
+setXPosition(xposition: Int)
+getyPosition()
+setyPosition(yposition: Int)
+getRotationAngle()
+setRotationAngle(rotationAngle: Int)
+isMagnet()
+setMagnet(isMagnet:Bool)
+isActiveLaserGun()
+setActiveLaserGun(isActiveLaserGun:Bool)
+isLaunched()
+setLaunched(isLaunched:Bool)
+getStatus()
+setStatus(n: Int)
+isTaller()
+setTaller(isTaller: Bool)
+repOk()
+toString()

SimpleBrick

-Xpos: Int
-Ypos: Int
-length: Int
-width: Int

+<<constructor>> SimpleBrick(xPos: Int, yPos: Int, length: Int, width, Int.
+getXpos()
+getYpos()
+getLength()
+getWidth()
+getXRightSide()
+getYBottomSide()
+setXpos(x: Int)
+setYpos(y: Int)

HalfMetalBrick

-Xpos: Int
-Ypos: Int
-length: Int
-width: Int

+<<constructor>> HalfMetalBrick(xPos: Int, yPos: Int, length: Int, width, Int.)
+getXpos()
+getYpos()
+getLength()
+getWidth()
+getXRightSide()
+getYBottomSide()
+setXpos(x: Int)
+setYpos(y: Int)

MineBrick

-Xpos: Int
-Ypos: Int
-length: Int
-width: Int

+<<constructor>> MineBrick(xPos: Int, yPos: Int, length: Int, width, Int.)
+getXpos()
+getYpos()
+getLength()
+getWidth()
+getXRightSide()
+getYBottomSide()
+setXpos(x: Int)
+setYpos(y: Int)

WrapperBrick

-Xpos: Int
-Ypos: Int
-length: Int
-width: Int

+<<constructor>> MineBrick(xPos: Int, yPos: Int, length: Int, width, Int.)
+getXpos()
+getYpos()
+getLength()
+getWidth()
+getXRightSide()
+getYBottomSide()
+setXpos(x: Int)
+setYpos(y: Int)

<<interface>>

PowerUp

usePower()
getXpos()
getYpos()
getRadii()
getType()
getIsActive()
setXpost(x: Int)
setYpos(y: Int)
setRadii(radii: Int)
setType(type: String)
setIsActive(isActive: Bool)
move()
isHit()
setHit(isHit: Bool)

DrunkAlien

-speed: Int=1
-xPosition: Int
-yPosition: Int
-dir: Int=1
-length: Int
-width: Int
-type: String
-status: Int
-alienAppeared: Bool
-Playground: playground

+<<Constructor>>DrunkAlien(xPos: Int, yPos: Int, length: Int, width: Int)
+getStatus()
+setStatus(status: Int)

<<interface>>

Alien

+getXpos()
+getYpos()
+getLength()
+getWidth()
+getType()
+setType(type: String)
+move()
+disappear()
+appear()
+perform()
+getStatus()
+setStatus(status: Int)
+isAppeared()
+setAppeared(appear: Bool)
+getDir()
+setDir(dir: Int)
+setXpos(x: Int)
+setYpos(y: Int)

AlienPowerup

-x: Int
-y: Int
-type: String
-isActive: Bool
-radii: Int
-isHit: Bool

+<<constructor>> AlienPowerup(x Int, y: Int, radii: Int,type: String, isActive: Bool, isHit:Bool)
+isHit()
+setHit(isHit: Bool)

TallerPaddle

-duration: Int

+paddleGrowth()

FireBall

+changeToFireBall(Ball)

MagnetPowerUp

-x: Int
-y: Int
-type: String
-isActive: Bool
-radii: Int
-isHit: Bool

+<<Constructor>>MagnetPowerUp(x: Int, y: Int, radii: Int, type: String, isActive: Bool, isHit: Bool)
+isHit()
+setHit(isHit:Bool)

ChemicalPowerUp

-x: Int
-y: Int
-type: String
-isActive: Bool
-radii: Int
-isHit: Bool

+<<Constructor>>ChemicalPowerUp(x: Int, y: Int, radii: Int, type: String, isActive: Bool, isHit: Bool)
+isHit()
+setHit(isHit:Bool)

RepairingAlien

+<<constructor>> Repairing(xPos: Int, yPos: Int, length: Int, width: Int)

CooperativeAlien

+<<constructor>> CooperativeAlien(xPos: Int, yPos: Int, length: Int, width: Int)

ProtectingAlien

-speed: Int=1
-xPosition: Int
-yPosition: Int
-dir: Int=1
-length: Int
-width: Int
-type: String

-Playground: playground

+<<Constructor>>ProtectingAlien(xPos: Int, yPos: Int, length: Int, width: Int)

LaserGun

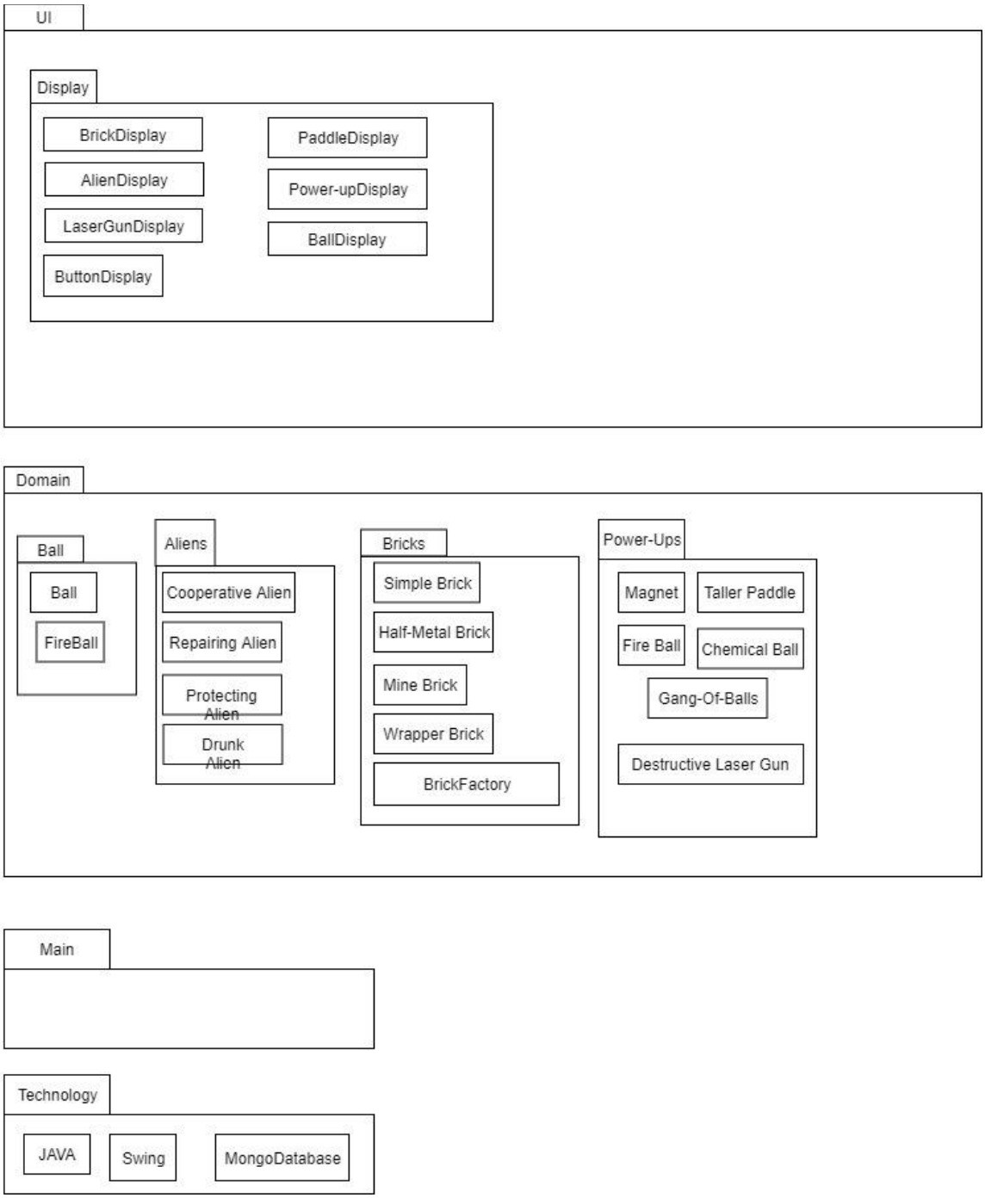
-x: Int
-y: Int
-velocity: Int
-isFired: Bool

+<<Constructor>>LaserGun(x: Int, y: Int, velocity: Int, isFired: Bool)
+moveLaser()
+getX()
+setX(x: Int)
+getY()
+setY(x: Int)
+getVelocity()
+setVelocity(velocity: Int)
+isFired()
+setFired(isFired:Bool)

Gang-Of-Balls

increaseAmount(ball.Amount())

Logical Architecture:



Test Plan

1. testBallMoveX()

We test that, if the location of the ball changes with respect to direction and velocity. Initially ball has velocity==2 and we move it with direction==1. So, we expect the X direction of the ball to be incremented by 2.

2. TestPaddleMove()

We test that, if the location of the paddle changes with respect to direction and velocity. Initially the paddle has velocity==5 and we move it with direction==1. So, we expect the location of the paddle to be incremented by 5.

3. TestBrickFactory()

We test that, if the brickFactory creates the bricks. We created a SimpleBrick with initial xposition ==600, yposition==300, length==80, width==40. Then using the brickFactory we created a simple brick with the same values. So, we expect that these two bricks are the same.

4. TestSimpleBrick()

We test that, if the x positions of the bricks are set correctly. We created a brick with an xPosition==600. Then using setXPos() method we set the xposition to be 500. We expect that the x position of the same brick becomes 500.

5. TestPlayer()

We test that, if the score of a player is set properly and whether the lives of the player are initially set to 3. Using setScore() method we set the score of the player to be 6500. We expect that the score of the same player becomes 6500 and the number of lives of the player is 3.

DESIGN ALTERNATIVES

Model-View Separation

The Model - View Separation principle states that domain objects (model) should not have direct knowledge of UI (view) objects, at least as view objects. In usage of model-view separation, application logic is should not be put in the UI object methods. UI objects should only initialize UI elements, receive UI events (such as a mouse click on a button), and delegate requests for application logic on to non-UI objects (such as domain objects). We applied this principle in our design such as in our project, domain objects did not have any coupling to UI objects are not aware of game logic.

General Responsibility Assignment Software Patterns (GRASP)

Creator Principle

Creator Pattern defines which class should be assigned to responsibility to create some other class. It has advantages and disadvantages. As an example of one of the advantages, the creator class is chosen to be the creator because this class has the knowledge about the created class. However, when creator class is allowed to create another class just because creator class has some knowledge or relation to the other class, we may create extra coupling or low cohesion. In our project, GameEngine has responsibility to create the game World, since it has information for creating a world. On the other hand, world creates bricks inside itself according to the Factory Pattern.

Controller Principle

Controller Pattern defines a controller as an object that receives and coordinates system operations. A controller has to delegate other classes to perform tasks. We created every single class differently for low coupling and the only controller that delegates other classes will be the GameEngine in our project, will act as a controller anonymously.

Information Expert Principle

Information expert (also expert or the expert principle) is a principle used to determine where to delegate responsibilities such as methods, computed fields, and so on. In our project, there is lots of usage of this principle. Wrapper Bricks, World and every power-up uses this principle. It helps us to delegate the classes and help them communicate with each other. But classes must only know the information that they need otherwise we will break the low coupling principle.

Low Coupling Principle

Low Coupling Pattern states the importance of low coupling such that impact of change will be very low. One of the best ways to achieve low coupling principle is having less classes linking to another classes. When we make classes as independent as possible, it becomes easier to make changes in classes which is why low coupling is really important. In our project every class is independent from each other and only the GameEngine reaches all of them and uses them. So only GameEngine will be affected by the changes in the other classes.

High Cohesion Principle

High cohesion is a software engineering concept. Basically, it says a class should only do what it is supposed to do, and does it fully. Do not overload it with functions that it is not supposed to do, and

whatever directly related to it should not appear in the code of some other class either. In our project every single thing has their own class and they do their own job does not interfere with others. Together with Low Coupling these 2 principles will raise the efficiency of our work in a good amount.

Gang of Four (GoF)

Singleton Pattern

In software engineering, the singleton pattern is a software design pattern that restricts the instantiation of a class to one "single" instance. This is useful when exactly one object is needed to coordinate actions across the system. The Paddle class is act according to singleton pattern since its a “single” instance in the project.

Observer Pattern

The observer pattern is a software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods. By this pattern, we seperate the UI and domain in our project. Since the Model-View separation is a huge part of the project, Observer pattern plays a crucial part in the project.

Simple Factory Pattern

The simple factory pattern allows the creation of objects without having to specify the exact class of object that will be created. This is achieved by using factory methods. World creates bricks according to the simple factory pattern since it creates bricks without the knowledge of their type with respect to the interface. Moreover, Wrapper Bricks creates object according to this pattern too, without knowing any knowledge of the alien or power-up types.

Adapter Pattern

The Adapter Pattern allows us to reach various APIs indirectly. As an instance, we will use MongoDB in our project for saving and loading the game. Our project need to talk with MongoDB and transfer mains features of our game to it for creating a document or loading that document. Since we use another component like MongoDB, adapter pattern is a crucial part of this project and help us to reach the MongoDB.

Supplementary Specifications

FURPS+

Functionality: When any error occurs, it will be displayed on the screen with text and the game is ended. There is only a player in this game. Authenticated player controls main objects in the game and decides course of procedure.

Usability: Aliens, bricks and ball will be visible and colors' of them will be different from the color of the background. Bricks will be designed due to their types but colors that admonishes any health
Reliability: When minor situations will be avoided.

errors happen in the system, like power-ups limit breakages or ball jams, the system will continue to work despite the minor errors.

Performance: Efficiency of the system is very crucial for this game. Since the player plays the game simultaneously, the system must keep up with the player. The paddle must not hold back or the ball must move with a constant velocity.

Supportability: The system must be very adaptable in this game. Players controls almost everything in the game and there are many variations and extensions like stated in the use case, so the system must adapt to every decision made by the player. Configurability(?)

Glossary

	Glossary	
Term	**Definition and Information**	**Format**
Player	**Authenticated user that controls the paddle.**	
Paddle	**Main object in the game, the player hits the ball with it.**	
Ball	**Main object in the game can hit brick or kill aliens.**	
Bricks	**Targets of the player**	

Simple Brick	**Brick that can be broken with one hit.**	
Half Metal Brick	**Can be broken with fireball or by a hit on top**	
Mine Brick	**Brick that explodes and breaks its neighbours.**	
Wrapper Brick	**Contains aliens or power-ups. Reveals them when broke.**	
Aliens	**Gamechangers that helpers or the enemies of the user**	
Repairing Alien	**Randomly creates simple bricks**	
Protecting Alien	**Protects the bricks by moving besides them**	
Cooperative Alien	**Helper alien that breaks a row of bricks**	
Power-ups	**Reveals when a wrapper brick is broken. Gives special abilities to paddle or the ball.**	
Fireball	**Gives power to ball to break a brick and its neighbours. It turns the half metal brick into the simple bricks.**	
Chemical Ball	**Gives power to ball to get through the bricks and broke them.**	
Gang-of-Balls	**Sets ball number to 10.**	

Taller Paddle	**Doubles the size of paddle.**	
Magnet	**Gives power to paddle to hold the ball.**	
Destructive Laser Gun	**Gives power to paddle to shoots laser shots**	
Laser shots	**Moves in one direction and break bricks except half metal ones.**	
Drunk Alien	**A random alien that acts due to brick number**	