Group:BESYO

Group Members:

Kemal Bora Bayraktar

Emir Şahin

Süleyman Balyer

Yusuf Demirtaş

Ömer Faruk TAL

Table Of Contents

Introduction……………………………………………………………03

TeamWork Organization……………………………………………….04

UML Use Case Diagram…………………………………………………06

Use Cases…………………………………………………………………07

Domain Model……………………………………………………………20

System Sequence Diagram……………………………………………21

Operation Contracts…………………………………………………….28

Sequence/ Communication Diagram…………………………………32

UML Class Diagram………………………………………………………43

UML Package Diagram…………………………………………………...44

Design Discussion…………………………………………………………45

Supplementary Specification……………………………………………47

Non-Functional Requirements……………………………………………48

Glossary……………………………………………………………………49

Introduction and Vision

Introduction

We aim to create Outer Space Game, which is a fan-produced and improved version of the game Breakout.

Business Requirements

Our main intention to make Outer Space Game is to enhance the enjoyable elements of the game Breakout. In Breakout game, players usually stop playing the game after a couple of trials since the game does not have enough properties to attract the players. Breakout game only contains static brick and moveable paddle and ball, which is the main reason of its failure of keeping players play more. Breakout game also does not have any feature to make gameplay experience different for other players. Breakout's lack of randomness and interaction with player, also prevent players from playing the game much longer. The game also has no ability to recreate the game scenes, which prevent its reusability.

Product Solution/Overview

In this version the game Breakout, Outer Space Game comes within its innovational attributes which can make game more playable and increase gameplay experience. Our first solution is to make game more playable in one sitting; to solve this issue in our game, we present additional features to challenge the player; so that the game itself can give impression it never ends. We include motion to game by changing the bricks stationary features, and we include chance-based gameplay factors that player can get benefit or harm. We also add interactive objects, Aliens that can alter the situation in the game grid; the presence of the Alien in the game also gives players more concrete 'enemies' than bricks. We also implement a login system and save/load system to game for players; with this player will be able to recreate the situation in their previous games, and they will have chance to replay the games they stuck.

TeamWork Organizations

Kemal Bora Bayraktar

-Ball Paddle Collision Improvement

-Basic Game Loop (Start game, restarting game when ball exceed paddle)

-Refactor in asteroid creation area

-Improving UI design

-ScoreBoard Implementation and UI

-Bugfix (collisions)

-Pause Resume (With buttons)

Emir Şahin

-Ball Paddle Collision Improvement

-Ball Asteroids Collision Improvement

-Paddle Rotation

-Builder Mode (UI, Building Screen, Drag and drop feature)

-Login Screen (UI, Login and Register Operations, Database part)

-Save/Load (Saving game (all features), loading game from database)

-Wrap PowerUp

-Bugfix (database issues, explosion bug)

Süleyman Balyer

-Paddle Ball Collision refactor

Yusuf Demirtaş

-Worked on Fixing Asteroid ball collision (impact from X horizontal) (later on Emir completed this part)

-Text based help screen (not implemented in the Game, just class exist)

-Worked on adding pause and resume with buttons (Bora fully implemented this part)

Ömer Faruk TAL

-GamePanel draw objects (all features)

-Asteroids (all asteroids and their moves and rersponsibilites)

-Aliens (all Aliens and their responsibilities)

-Taller Paddle, Magnet, Chance Power-Up

-Ball and Paddle and Wall Collision and movement (Very primitive, they have been improved by Emir and Bora)

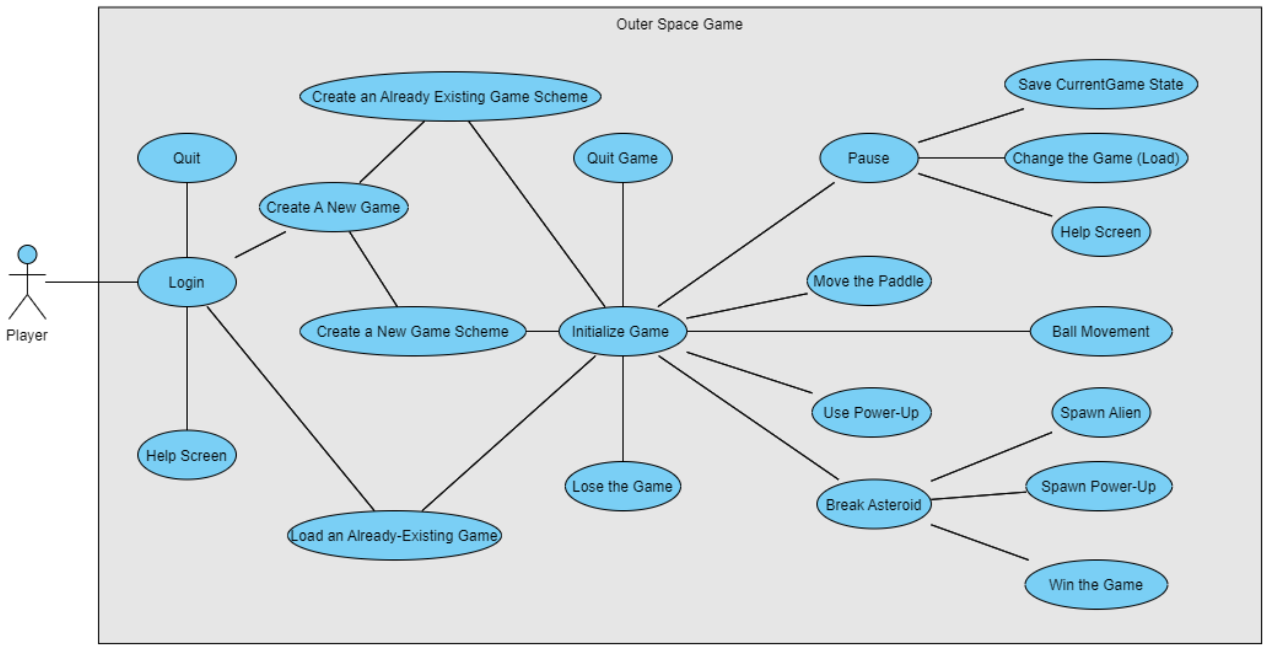
-Asteroid Ball Collision

-Pause and Resume (by keyboard)

-Introduction of lives

-Remaining asteroid and Lives UI

UML Use Case Diagram



Use Cases

Use Case UC1: Quit Game

Scope: Outer Space Game
Level: User-goal
Primary Actor: Player
Stakeholders and Interests:
Player → Wants to quit the game
System → Closes the game
Preconditions:
No precondition required. The player can close the application right after they start it, and they can do it at any given time during the game.
Success Guarantee (or Postconditions):
Application is closed. The game is saved automatically.
Special Requirements:
No special requirement needed.
Frequency: Once during a game.
Main Success Scenario:

1- The user is shown “save” or “quit without save” option.
2- The game is saved into data base if user wants to save.
3- The game is closed.

Extensions:
*a. If the system fails:

1- The player is informed with an error message.
2- The player goes back to main menu.

Use Case UC2: Save Game

Scope: Outer Space Game
Level: User-goal
Primary Actor: Player, System, Data Base
Stakeholders and Interests:
Player → Wants to save the game
System → Saves the game state
Preconditions:
The player must have logged in with their own unique username.
Success Guarantee (or Postconditions):
The game is successfully saved, and a save is created at the data base or file.
Special Requirements:
The game should be paused while trying to save.
Frequency: Infinite. The player can save as many times as they can during the game.
Main Success Scenario:

1- Save with the necessary data is created, including player health, power-up status, power-ups, asteroid count and their types, and cooperative alien appearance status depending on the user database and file choice.

Extensions:
*a. If the user choice is file option:

1- The data will be saved into a text file.

*a. If the user choice is database option:

1- The data will be saved with MongoDB database system.

*a. If the system fails:

1- The player is informed with an error message.
2- The player goes back to main menu.

Use Case UC3: Help Screen

Scope: Outer Space Game

Level: User-goal

Primary Actor: Player

Stakeholders and Interests:

Player → Wants to receive help

System → Manages the menu states

Preconditions:

The game should be paused and help screen should be selected from pause menu. Or the player can see the help screen before game initialization.

Success Guarantee (or Postconditions):

Help screen is opened and help status is shown to the player. Goes back to the pause screen when closed.

Special Requirements:

The game should be in pause mode or game building mode.

Frequency: Up to player. The player can open the help screen as many times as they want.

Main Success Scenario:

1- The help screen displays explanations for controls, power-ups, aliens, and asteroids. They can take as much time as they want to spend on help screen. When closed, it goes back to pause or game building menu.

Extensions:

*a. If player decides to quit the game:

1- System asks to player whether the player wants to save.

*b. If the system fails:

1- The player is informed with an error message.
2- The player goes back to main menu.

Use Case UC4: Continue Game

Scope: Outer Space Game

Level: User-goal

Primary Actor: Player

Stakeholders & Interests:

Player → Wants to continue the game

System → Continues the game

Preconditions:

The game should be paused. Continue Game option only appears at pause menu.

Success Guarantee (or Postconditions):

The game keeps going from right where the player paused previously.

Special Requirements:

The game must be initialized and started first to be able to open the pause menu, so that the player can continue afterwards.

Frequency: Up to player. The player can pause and continue as many times as they want.

Main Success Scenario:

1- The player can keep playing where they left.

Extensions:

*a. If player decides to quit the game:

1- System asks to player whether the player wants to save.

*b. If the system fails:

1- The player is informed with an error message.

2- The player goes back to main menu.

Use Case UC5: Login

Scope: Outer Space Game

Level: User-goal

Primary Actor: Player

Stakeholders and Interests:

Player → Wants to login

Preconditions:

Player must login to start a new game or load an existing one.

Success Guarantee (or Postconditions):

The player has logged in.

Special Requirements:

The player is got into the main screen and have two options whether start a new game or load existing game.

Frequency: Occurs exactly once before the game.

Main Success Scenario:

1- Player enters username.

2- Player gets into main screen.

Extensions:

*a. If the system fails:

1- The player is informed with an error message.

2- The player goes back to main menu.

1a. If the player enters wrong username:

1- The player is asked to reenter a username.

Use Case UC6: Create a New Game

Scope: The New Game Portal

Level: User-goal

Primary Actor: Player

Stakeholders and Interests:

Player → Achieves the Game-scheme screen

Preconditions:

Player needs to login the game to create a new game.

Success Guarantee (or Postconditions):

The player opened the game scheme choosing screen.

Special Requirements:

The choose an already-existing game scheme and create a new game-scheme options must be displayed on screen.

Frequency: Occurs exactly once before the game.

Main Success Scenario:

1- The player reached choosing game scheme screen.
2- Player has not started the game.

Extensions:

*a. If the system fails:

1- The player is informed with an error message.

2- The player goes back to main menu.

Use Case UC7: Load an Already Existing Game Scheme

Scope: The Existing Game
Level: User-goal
Primary Actor: Player
Stakeholders and Interests:
Player → Loads an already existing game scheme
Preconditions:
Player needs to login the game and chose “load an already existing game scheme”.
Success Guarantee (or Postconditions):
The player loaded an existing game scheme instead of new game scheme.
Special Requirements:
The already-existing game scheme chosen so the already existed game scheme loaded and must be displayed on screen.
Frequency: Occurs exactly once before the game.
Main Success Scenario:

1- The player chooses an already existing game scheme.
2- The chosen game scheme is loaded.

Extensions:
*a. If the system fails:

1- The player is informed with an error message.
2- The player goes back to main menu.

Use Case UC8: User Chooses an Already Existing Game Scheme

Scope: The Newly Created Game

Level: User-goal

Primary Actor: Player

Stakeholders and Interests:

Player → Chooses an already existing game scheme

Preconditions:

The player needs to login the game and chose starting a new game instead of loading a saved game. There must be already created game schemes.

Success Guarantee (or Postconditions):

The game scheme is selected.

Special Requirements:

The number of asteroids at each game scheme shouldn’t be less than the threshold. There must be at least 75 simple asteroids, at least 10 firm asteroids, at least 5 explosive asteroids and at least 10 gift asteroids. 6 of the gift asteroids contain 6 power-ups. 3 of them contain signals to aliens and the last one contains one of them chosen randomly. All asteroids should be visible on the screen.

Frequency: Occurs exactly once during the game.

Main Success Scenario:

1- The player chooses one of the existing game schemes.

Extensions:

*a. If player decides to quit the game:

1- System asks to player whether the player wants to save.

*b. If the system fails:

1- The player is informed with an error message.
2- The player goes back to main menu.

Use Case UC9: User Creates a New Game Scheme

Scope: The Newly Created Game

Level: User-goal

Primary Actor: Player

Stakeholders and Interests:

Player → wants to create a new game scheme

Preconditions:

The player needs to login the game and chose starting a new game instead of loading a saved game.

Success Guarantee (or Postconditions):

A new game scheme is created.

Special Requirements:

All asteroids should be visible on the screen. Asteroids should not overlap each other. Player should know English to understand the error messages.

Frequency: Occurs exactly once during the game.

Main Success Scenario:

1- Player places at least 75 simple asteroids.
2- Player places at least 10 firm asteroids.
3- Player places at least 5 explosive asteroids.
4- Player places at least 10 gift asteroids.
5- Player chooses either saving the game scheme or not saving it.

Extensions:

*a. If player decides to quit the game:

1- System asks to player whether the player wants to save.

*b. If the system fails:

1- The player is informed with an error message.
2- The player goes back to main menu.

*c. If the player places less than minimum number of asteroids from any kind:

1- An error message is shown to the player.
2- Player adds the missing number of asteroids.

*d. If the asteroids overlap each other:

1- An error message is shown to the player.
2- Player is asked to change the place of overlapping asteroids.

Use Case UC10: Initializing the Game

Scope: The Game
Level: User-goal
Primary Actor: System
Stakeholders and Interests:
System → Generates the chosen game scheme
Preconditions:
The player needs to login the game and chose starting a new game or loading a saved game.
Success Guarantee (or Postconditions):
The game screen is initialized with the selected game scheme or loaded from a saved game. The paddle is placed to its initial place.
Special Requirements:
All asteroids and the paddle should be visible on the screen. The player must be ready to fire the ball.
Frequency: Occurs exactly once during the game.
Main Success Scenario:

1- Asteroids are placed to their places.
2- The paddle is placed at the bottom middle of the game screen horizontally.
3- The game waits player to start the game by firing the ball.

Extensions:
*a. If player decides to quit the game:

1- System asks to player whether the player wants to save.

*b. If the system fails:

1- The player is informed with an error message.
2- The player goes back to main menu.

Use Case UC11: Starting the Game by Firing the Ball

Scope: The Game
Level: User-goal
Primary Actor: Player
Stakeholders and Interests:
Player → Starts the game by pressing "W" key.
Preconditions:
The game screen is initialized. Asteroids and paddle are at their initial place.
Success Guarantee (or Postconditions):
The game starts. The ball starts moving.
Special Requirements:
The player can use keyboard and mouse.
Frequency: Occurs when the player is initially starting the game, or the player couldn't catch the ball and firing the ball again.
Main Success Scenario:

1- System waits player to press "W" key.
2- Player presses "W" key on the keyboard.
3- The ball starts to move.
4- Player can pause the game any time.

Extensions:
*a. If player decides to quit the game:

1- System asks to player whether the player wants to save.

*b. If the system fails:

1- The player is informed with an error message.

2- The player goes back to main menu.

Use Case UC12: Pause the Game

Scope: Outer Space Game

Level: User-goal

Primary Actor: Player

Stakeholders and Interests:

Player → Wants to pause the game.

System → Pauses the game.

Preconditions:

The game must be running.

Success Guarantee (or Postconditions):

Game is paused. The ball stops moving. Player is not able to move paddle anymore.

Special Requirements:

All interactions are disabled.

Frequency: Up to player. The player can pause as many times as they can during the game.

Main Success Scenario:

1- Player presses the pause hotkey/button.

2- The game is paused.

Extensions:

*a. If player decides to quit the game:

1- System asks to player whether the player wants to save.

*b. If the system fails:

1- The player is informed with an error message.

2- The player goes back to main menu.

Use Case UC13: Power-up

Scope: Paddle and Ball

Level: User-goal

Primary Actors: Player

Stakeholder and Interests:

Player → Wants to take power-up, and use them for his/her advantage

Preconditions:

The ball needs to interact with Gift Asteroid trigger a power-up event and player should catch this power-up while falling with touching his/her paddle with power-up. To use active of them, the player should interact with keyboard.

Success Guarantee (or Postconditions):

Power-up should show its effects if the conditions are satisfied.

Special Requirement:

Only one power-up can be used at a time.

Frequency:

Occurs a few times in the playtime.

Main Success Scenario:

1- The paddle should catch the power-up icon first

2- Then power-up show its effect on the ongoing game

3- When their functionality is over, they will disappear, and game will continue like it is in the beginning

Extensions:

2a- If the power-up type is active.

1- If the power-up is Taller Paddle, when the key 'T' is pressed it will double the length of paddle for 30 seconds.
2- If the power-up is Magnet when the key 'M' is pressed it will be activated. It will allow paddle to catch ball instead of reflecting. Then with pressing 'W' or using mouse player can release ball again.
3- If the power-up is Destructive Laser, laser is Attached to both ends of the paddle.
4- If the power up is Wrap, when the key 'V' or click the related icon on the screen paddle will be able go through the side of screen and reappear other side for 2 minutes.

2b- If the power-up type is passive.

1- If the power-up is Chance, player lives will increase by 1. (At the beginning player has 3 lives)

2c- If the power-up is Gang-of-Balls, 9 additional balls will be created. Original ball should keep its speed and direction, other balls will have the same speed as original ball and 10 ball direction should divide 360-degree angle to equal degrees. All balls will have same property. Balls will stay if player is able to maintain them up.

Use Case UC14: Win the Game

Scope: Player

Level: User-goal

Primary Actor: Player

Stakeholders and Interests:

Player → Wants to destroy the final asteroid.

Preconditions: There must be only one asteroid left.

Success Guarantee (or Postconditions):

The game pauses. A message is displayed stating that the player won.

Special Requirements:

The number of remaining balls and amount of time the player has is displayed along with their score.

Frequency: Occurs either 1 or 0 times in each game.

Main Success Scenario:

1- The final asteroid is destroyed, either by the player or the cooperative alien.

Extensions:

*a. If the system fails:

1- The player is informed with an error message.

2- The player goes back to main menu.

Use Case UC15: Lose the Game

Scope: Player

Level: User-goal

Primary Actor: Player

Stakeholders and Interests:
Player → Wants to destroy the final asteroid.
Preconditions:
The player must have run out of balls, or the time must have run out.
Success Guarantee (or Postconditions):
The game pauses. A message is displayed stating that the player lost.
Special Requirements:
The reason for the player's loss (whether they ran out of balls or time) is displayed, along with their score.
Frequency: Occurs either 1 or 0 times in each game.
Main Success Scenario:

1- The timer runs out, or the player loses all their balls.

Extensions:
*a. If the system fails:

1- The player is informed with an error message.
2- The player goes back to main menu.

Use Case UC16: Move Paddle

Scope: Paddle
Level: User-goal
Primary Actors: Player
Stakeholder and Interests:
Player → Wants to move the paddle for controlling ball
Preconditions:
Player initialized the game, and then presses 'W' to start the game.
Success Guarantee (or Postconditions):
Player's interaction with the keyboard to move paddle is visible on the screen as properly.
Special Requirement:
There should be not delay in the game responses to the player interaction. Paddle length is 'L'.
Frequency: Occurs during the playtime.
Main Success Scenario:

1- The player interacts with a keyboard input for moving paddle.
2- Computer receive these inputs.
3- If the keyboard input is Arrow Left or Arrow Right, the paddle moves left or right correspondingly.
4- If the keyboard input is 'A' or 'D', the paddle rotates to left or right correspondingly.

Extensions:
*a. If the system fails:

1- The player is informed with an error message.
2- The player goes back to main menu.

3-4. If the user presses another key:

1- The pressed key does not affect state of the paddle.

Use Case UC17: Break Asteroid

Scope: Asteroid

Level: User-goal
Primary Actors: Player
Stakeholder and Interests:
Player → Wants to break the asteroids to gain points and finish the game.
Preconditions:
The ball needs to interact with asteroid by touching them.
Success Guarantee (or Postconditions):
The ball impact on the asteroid should be displayed, and the state of the asteroid should update.
Special Requirement:
There must be asteroids.
Frequency: Occurs during the playtime.
Main Success Scenario:

1- The ball move along its way and finally touch with asteroid.
2- The collision between ball and asteroid should update asteroid state.
3- The ball then should change its path and avoid from collision scene.

Extensions:
2a- If the asteroid type is Simple Asteroid:
    1- asteroid should be broken and disappear.
2b- If the asteroid type is Firm Asteroid:
    1- it requires one than more hit depending on asteroid radius (radius should be determined randomly).
    1- If it has live:
        1a – It will shrink down.
    2- If it's at its final size:
        2a - it cannot shrink once more (which is determined by a threshold), it should be broken with one it.
2c- If the asteroid type is Explosive Asteroid:
    1- it should be broken; this asteroid also destroys other asteroids within radius (2 * L)
2d- If the asteroid type is Gift Asteroid:
    1- it should be broken; it should also trigger an alien event or power-up
        1a -If alien event triggered:
            1- a random type of alien should be included in the game
        1b- If power-up event triggered:
            1- a random power-up should be created and fall with speed L/(4*second)
2e- If the asteroids are frozen by Time-Wasting Alien:
    1- ball will have no effect on them in the collisions. The asteroids lives can only be decreased by the Destructive-Laser Gun power-up if they hit by the laser.

Use Case UC18: Alien

Scope: Alien
Level: User-goal
Primary Actors: Player, Alien Objects
Stakeholder and Interests:
Player → Wants to get rid of the alien if it is detrimental to player; if it is beneficial for player, player wants to keep this alien if possible.
Preconditions:
The ball needs to interact with Gift Asteroid, and trigger and alien event.

Success Guarantee (or Postconditions):

Alien should be included in the game, and alien should display its functionality depending on its type.

Special Requirement:

2 aliens cannot appear at the same time.

Frequency: Occurs a few times in the playtime.

Main Success Scenario:

1- The ball should contact Gift Asteroid; and an alien event which calls random type of alien.

2- Alien should perform its functionality in the game.

3- They disappear when they are hit by ball or complete their functionality.

Extensions:

2a- If the alien type is Repairing Alien:

- 1- They will create random number and located simple asteroids. They can create an asteroid in 5 seconds. They cannot create more than the number of destroyed asteroids.

2b- If the alien type is Protecting Alien:

- 1- It moves the asteroid with the speed of (3*L)/seconds.

2c- If the alien type is Cooperative Alien:

- 1- it will pick a random row, that contains at least 1 asteroid, and destroy all asteroids.

2d- If the alien type is Time-wasting Alien:

- 1- They will select 8 asteroids remained and freeze them for 15 seconds.

2e- If the alien type is Surprising Alien:

- 1- It will behave differently with the percentage of the destroyed asteroid percentage
  - 1a- If the destroyed asteroid percentage is over %70:
    - 1- It will act as Repairing Alien at first, after it repaired; it will act as Time-wasting alien and freeze newly created 8 asteroid for 15 seconds.
  - 1b- If the destroyed asteroid percentage is between %60 and %70:
    - 1- It will stay in it's place and will disappear after 5 seconds.
  - 1c- If the destroyed asteroid percentage is between %50 and %60:
    - 1-It will act like Protecting Alien
  - 1d- If the destroyed asteroid percentage is between %40 and %50:
    - 1- It will act like Repairing Alien
  - 1e- If the destroyed asteroid percentage is between %60 and %70:
    - 1- It will stay in it's place and will disappear after 5 seconds.
  - 1f- If the destroyed asteroid percentage is less than %30:
    - 1- It will act like Cooperative Alien

3a- If the alien type is Repairing Alien:

- 1- when the ball is hit or fulfill their duty they will escape.

3b- If the alien type is Protecting Alien:

- 1- if the ball hits the top they will escape.

3c- If the alien type is Cooperative Alien:

- 1- if the ball hits they will escape and they will never be created further.

3d- If the alien type is Time-wasting Alien:

- 1- if the ball hits or fulfill their duty they will escape.

3e- If the alien type is Surprising Alien:

- 1- Then it will disappear like it's the Alien Type that it transformed.

Use Case UC19: Ball Movement

Scope: Ball
Level: User-goal
Primary Actors: Player
Stakeholder and Interests:
Player → Wants move ball to in an understandable manner
Preconditions:
Player initialized the game, and then presses 'W' to start the game.
Success Guarantee (or Postconditions):
Ball will move exactly as it is stated in the Rule of the Game's, every collision will be calculated accordingly.
Special Requirement:
Frequency: Occurs during the playtime.
Main Success Scenario:

1- The player interacts with a keyboard 'W' for releasing ball.
2- Ball is moving along straight lines with constant speed.
3- Ball interacts with an object and change its path accordingly.
4- After ball changes its path it continues to its straight-line movement.

Extensions:
1a- After ball is released ball will move with constant speed, and its direction will make 90 degrees with the paddle.
3a- Ball interacts with a stationary object from side:
1- It will bounce and move back with same speed and it will make an angle that is symmetric around the norm of the surface.
3b- Ball interacts with a stationary object from corner:
1- It will bounce and move back with same speed and it will make an angle that is symmetric around an imaginary line around corner.
3c- Ball interacts with a moving object:

1- If the direction of the movement of the object and is the same as the direction of the component of the ball velocity that is parallel to that movement direction:
    1a- The ball will reflect as it is stated in stationary object scenario. And the ball velocity will increase 5 pixel/second.
2- If the direction of the movement of the object and is the opposite of the direction of the component of the ball velocity that is parallel to that movement direction:
    1a- The ball will reflect with an angle of 180 degrees relative to the line of original ball movement. Speed of ball will not change.
3- If the direction of the ball and object is perpendicular:
    1a- the ball will reflect with an angle of 45 degrees relative to the line of the moving object movement. Speed of ball will not change.

Use Case UC20: Asteroid Movement

Scope: Asteroid
Level: User-goal
Primary Actors: Player

Stakeholder and Interests:
Player → Wants move asteroids to move in a predictable manner
Preconditions:
Player initialized the game and logged in.
Success Guarantee (or Postconditions):
Asteroids will move exactly as its stated in the rules of the game.
Special Requirement:
Asteroids should move if they are non-stationary asteroids. There must exist asteroids.
Frequency: Occurs during the playtime.
Main Success Scenario:

1- Player starts the game.
2- Asteroids starts to move depending on their type with speed L / (4* seconds).
3- They continue to this behavior until they explode.

Extensions:
2a- If asteroid is Simple or Mine Asteroid or Gift Asteroids:

1- With %90 probability asteroid will be stationary.
2- With %10 probability they will move horizontally back and forth.
  2a- If it is enough space:
    1- it will continue its movement.
  2b- If its route include asteroid:
    1- they will collide and asteroid change direction.

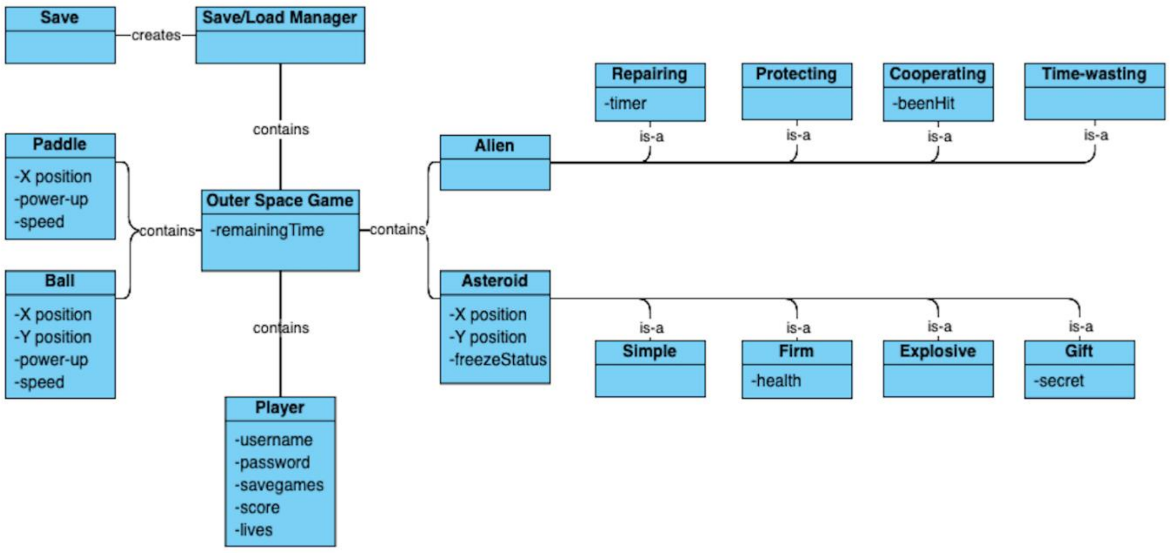
2b- If asteroid is Explosive Asteroid:

1- If space allow it will move in circular fashion, which has radius 1.5*L .
2- If space is not enough it will be stationary.

2c- If asteroids are frozen by Time-Wasting Alien:

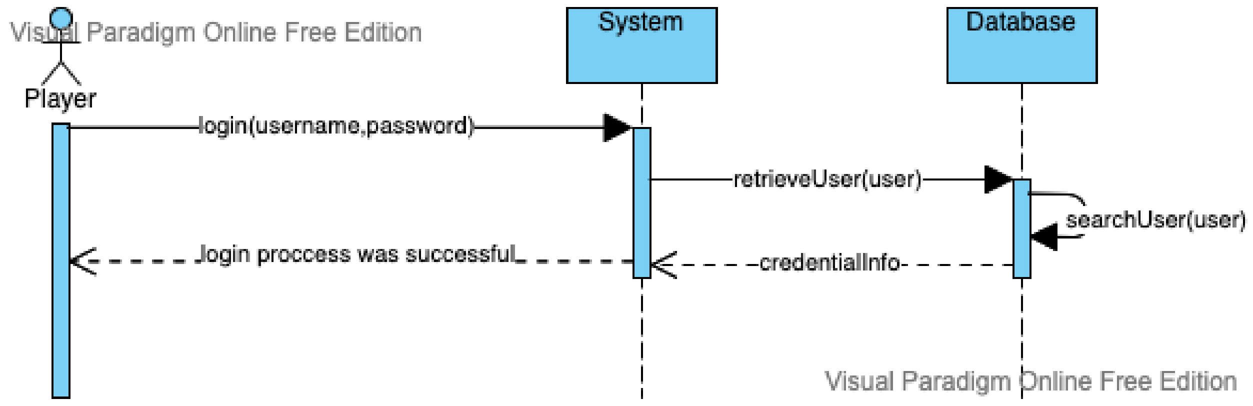
1- they cannot move.

Domain Model

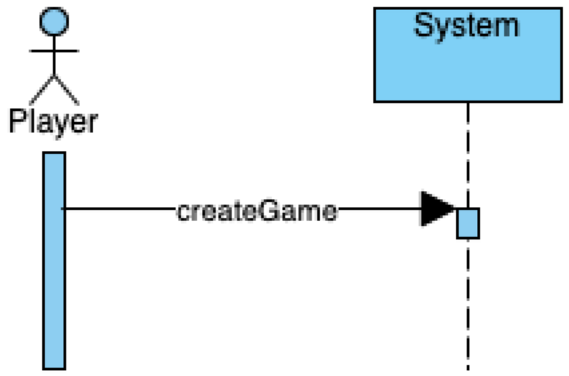


System Sequence Diagrams

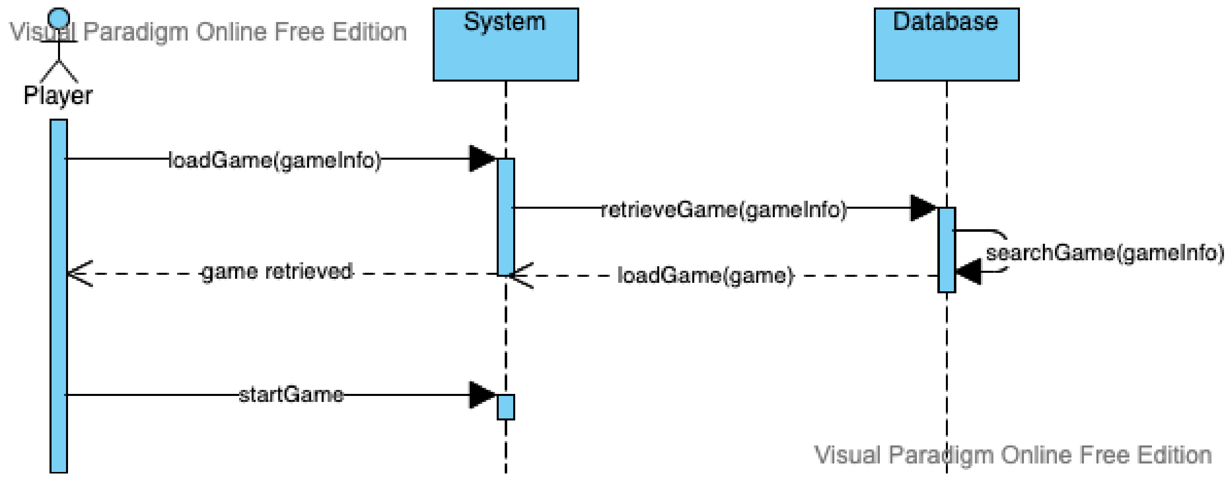
SSD1: Login


Visual Paradigm Online Free Edition
Player
System
Database
login(username,password)
retrieveUser(user)
searchUser(user)
login proccess was successful
credentialInfo
Visual Paradigm Online Free Edition

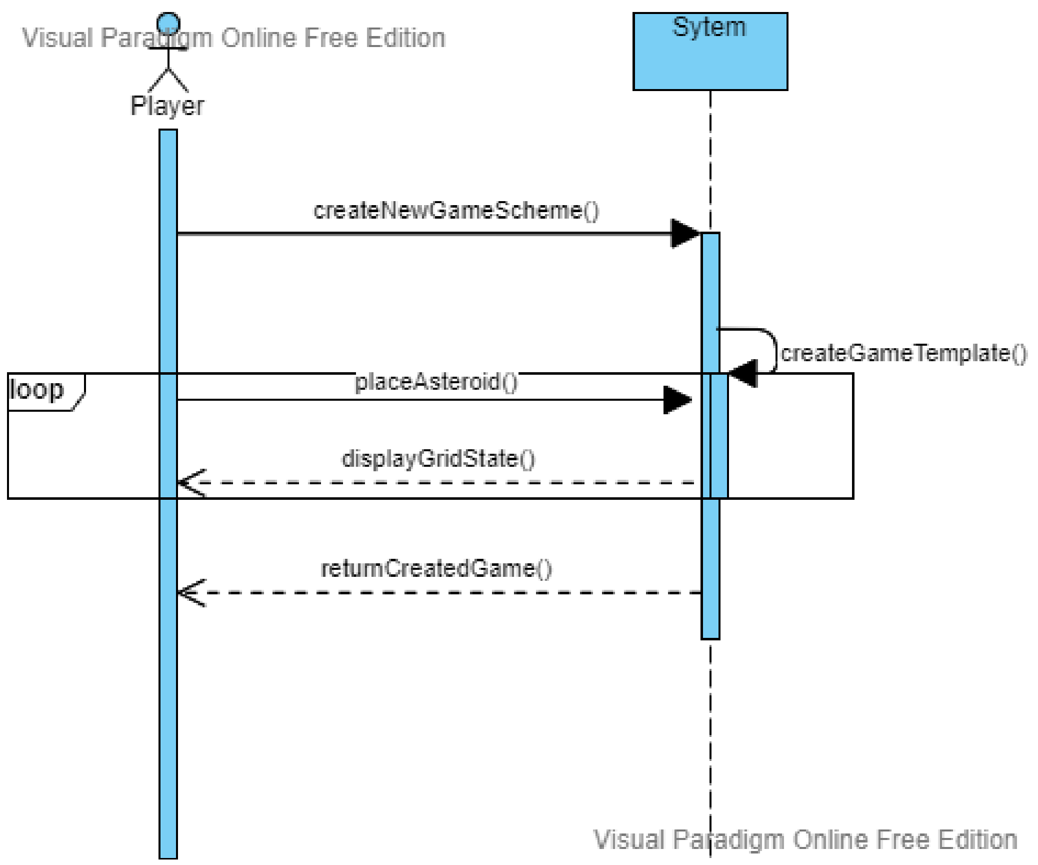
SSD2: Create a New Game



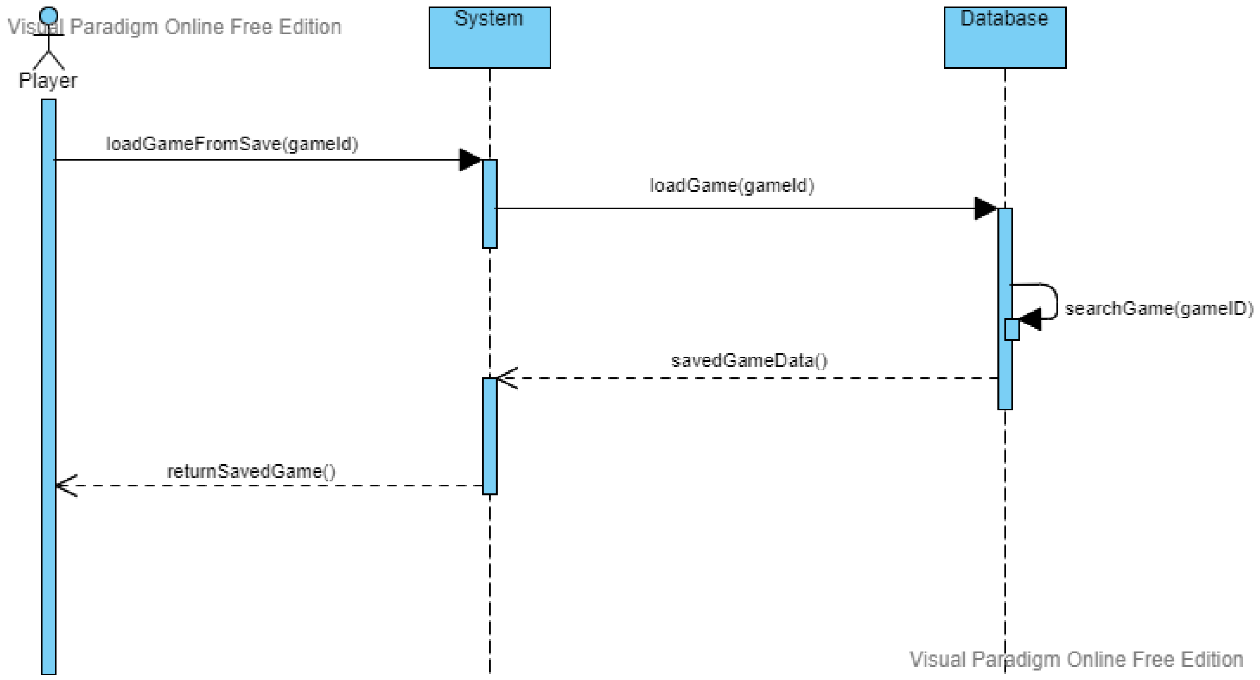
SSD3: Load an Already Existing Game



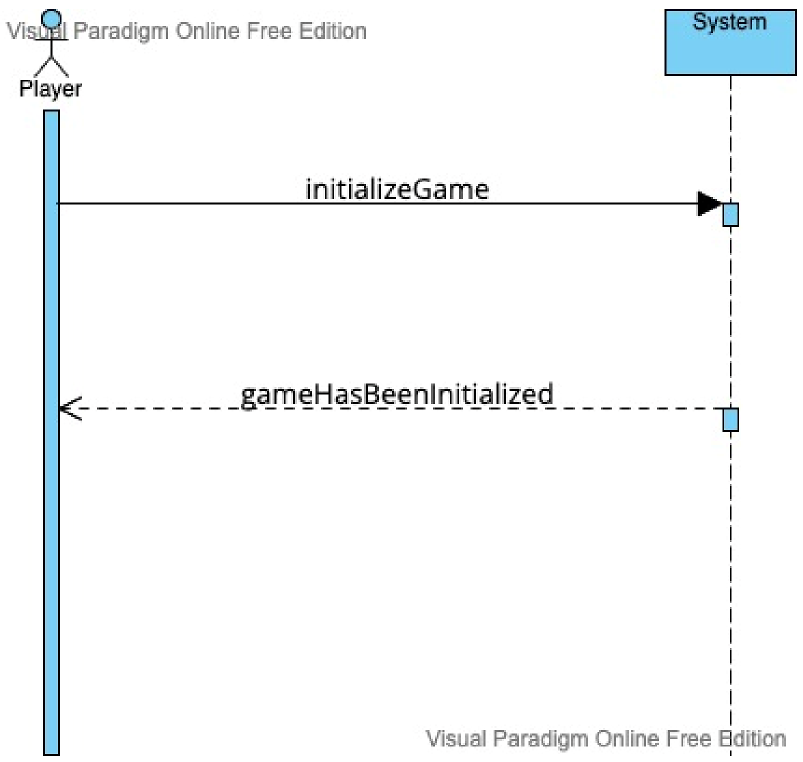
SSD4: Create a New Game Scheme



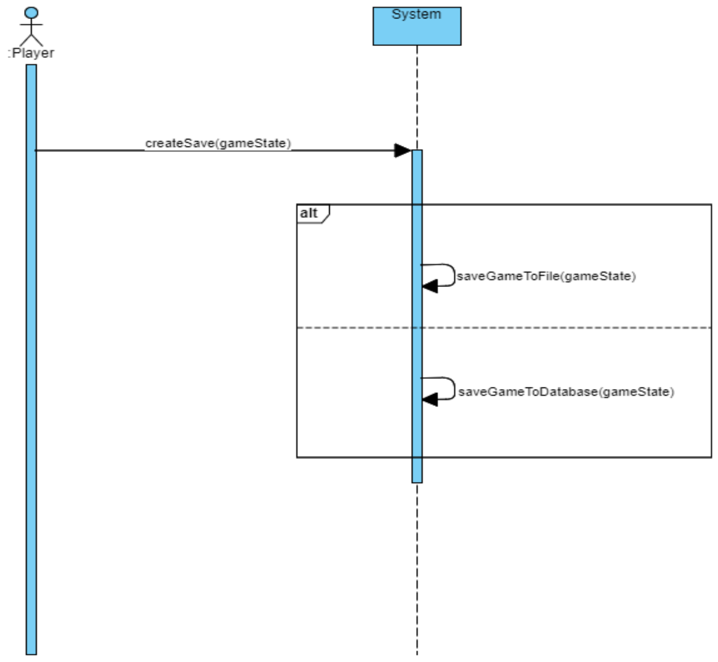
SSD5: Choose an Already Existing Game Scheme



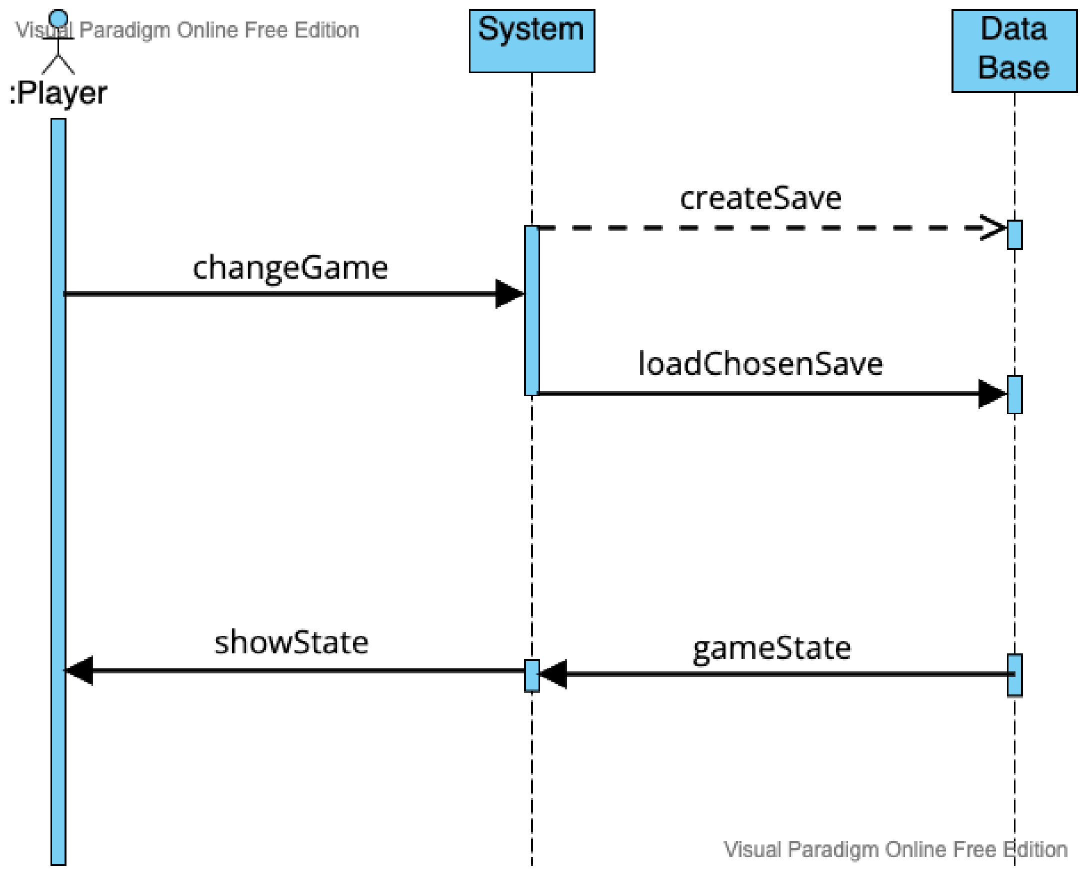
SSD6: Initialize Game


Visual Paradigm Online Free Edition
Player
System
initializeGame
gameHasBeenInitialized
Visual Paradigm Online Free Edition

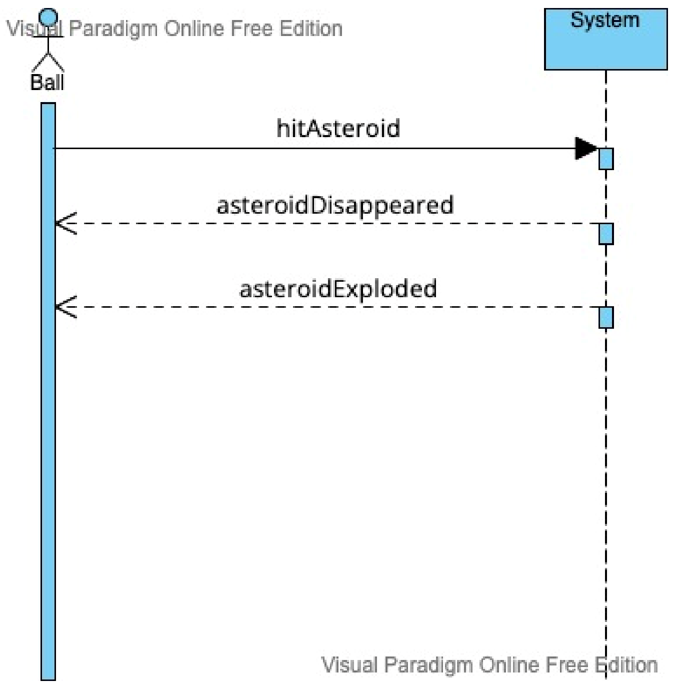
SSD7: Save Current State



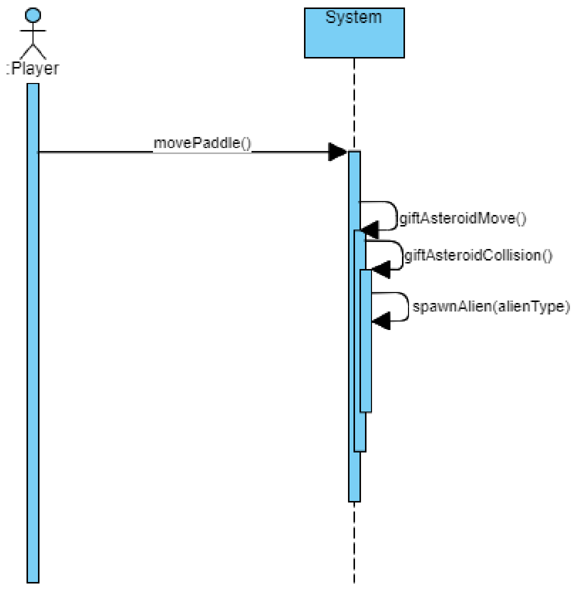
SSD8: Change the Game (Load)



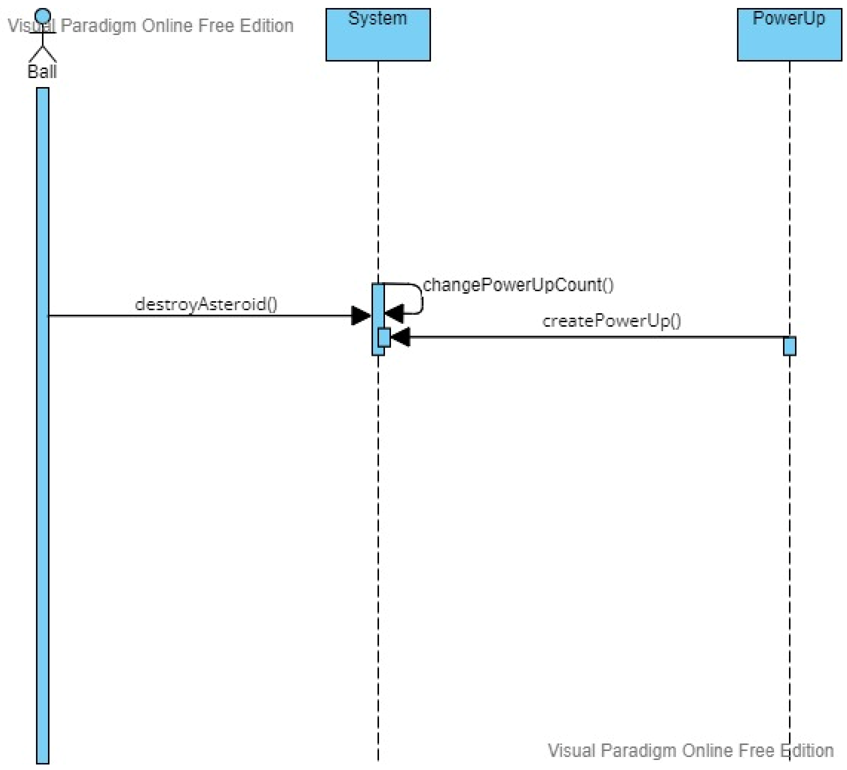
SSD9: Break an Asteroid


Visual Paradigm Online Free Edition
Ball
System
hitAsteroid
asteroidDisappeared
asteroidExploded
Visual Paradigm Online Free Edition

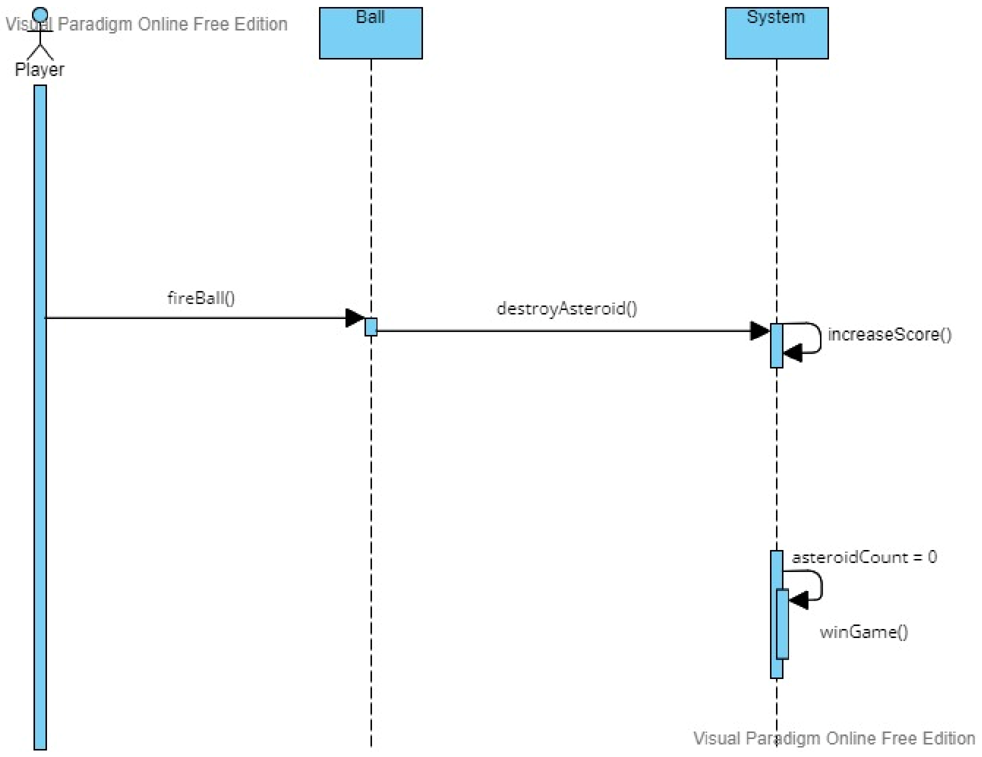
SSD10: Spawn Alien



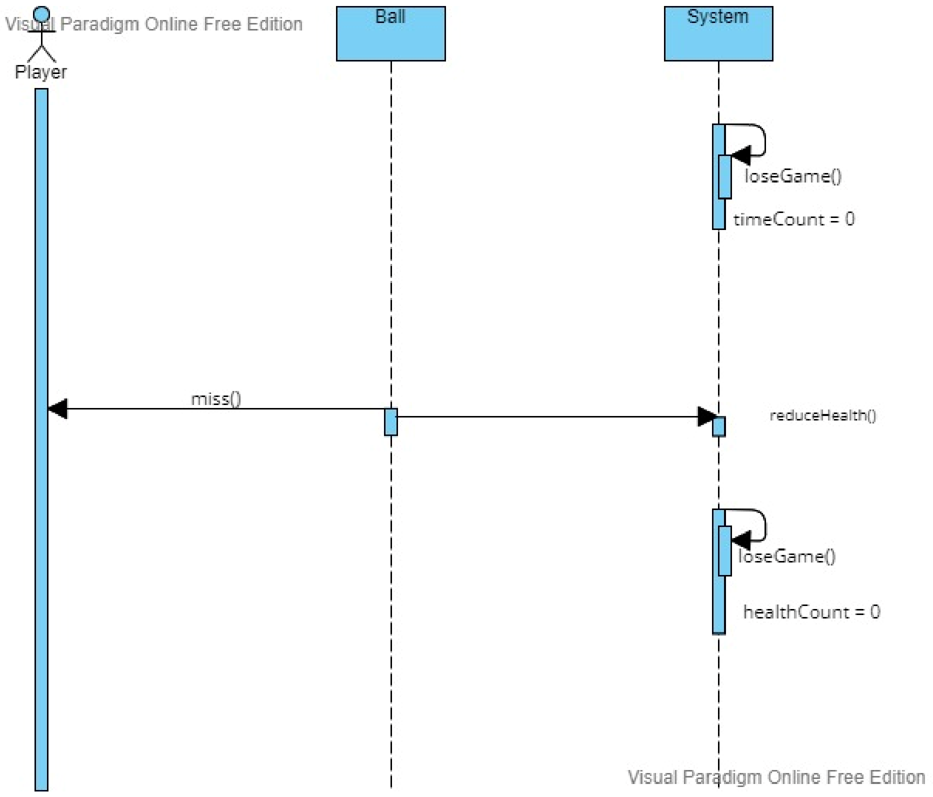
SSD11: Spawn Power-Up



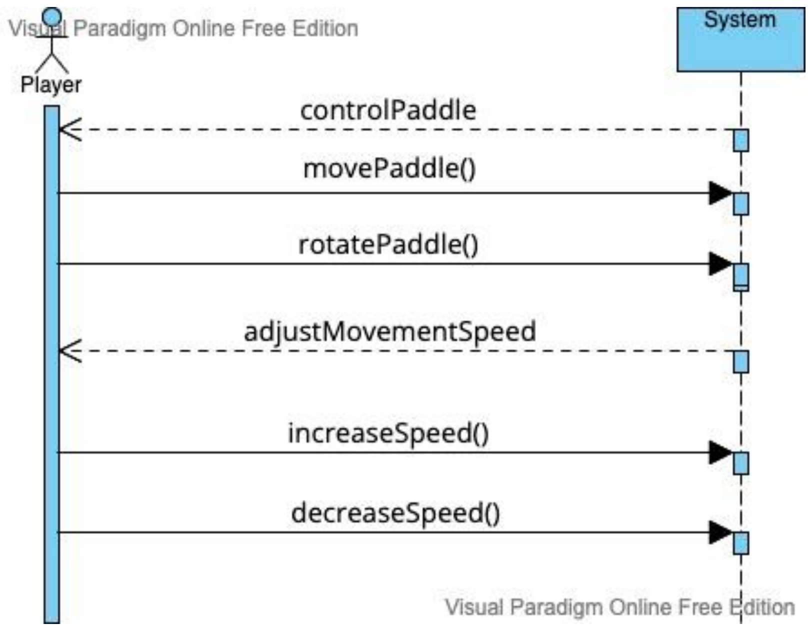
SSD12: Win the Game



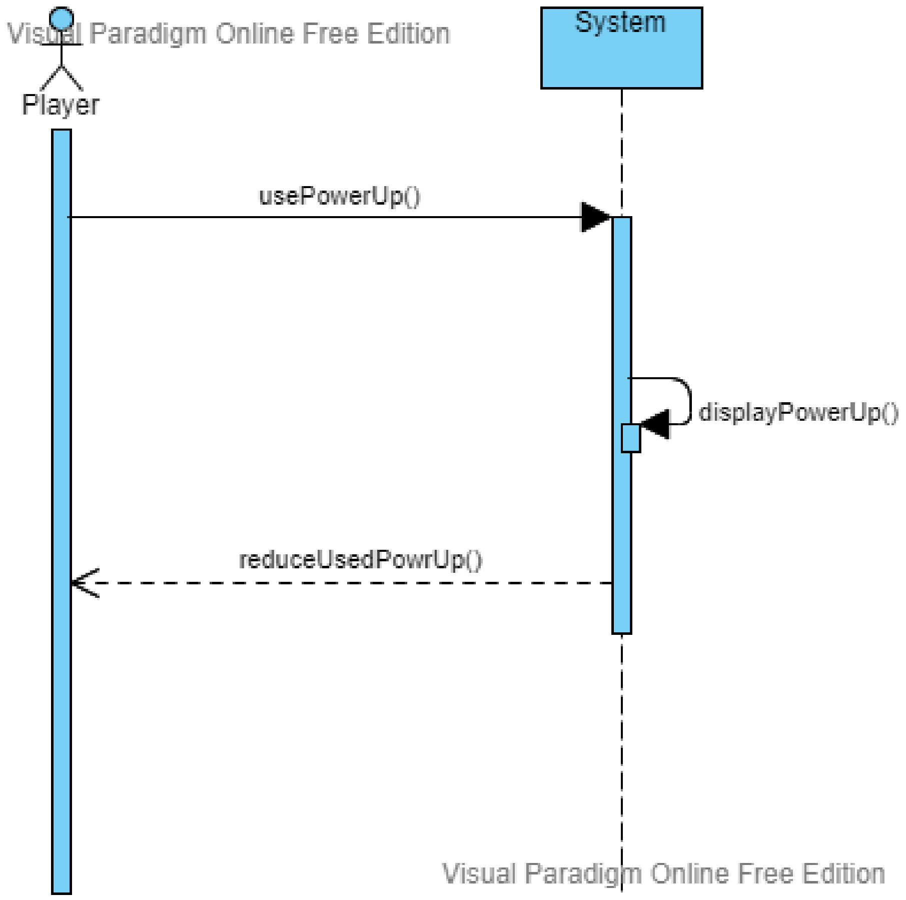
SSD13: Lose the Game



SSD14: Move Paddle



SSD15: Use Power-Up


Visual Paradigm Online Free Edition
Player
System
usePowerUp()
displayPowerUp()
reduceUsedPowrUp()
Visual Paradigm Online Free Edition

Operation Contracts

Contract CO1: login
Operation: login(username,password)
Cross References: Login
Preconditions:
- User has submitted their credentials.
Postconditions:
- The credentials are sent to the system to be compared to the database.

Contract CO2: retrieveUser
Operation: retrieveUser(user)
Cross References: Login
Preconditions:
- The user has submitted their credentials.
- A user with the given credentials is on the database.
Postconditions:
- The user is logged in.

Contract CO3: loadChosenGame
Operation: loadChosenGame(chosenGame)
Cross References: Use Cases: Pause the Game
Preconditions:
- Player is in a game.
- Player chooses a saved game.
- Player is at the help screen.
Postconditions:
- Leaved game is saved.
- Another saved game is loaded.

Contract CO4: retrieveGame
Operation: retrieveGame(gameInfo)
Cross References: Load an Already-Existing Game
Preconditions:
- The user is logged in.
- A game with the given game-info exists on the database.
Postconditions:
- The game with the given game-info is returned to the System.

Contract CO5: createRandomGame
Operation:createRandomGame()
Cross References: User Chooses an Already Existing Game Scheme, Create a New Game, Initializing Game
Preconditions:
- Player must be logged into the system.
- Player must have selected the create already existed game scheme option.

- Player should choose random game option.

Postconditions:

- All asteroids type will be generated according to their type and number of type.
- The new added asteroid should not contact with the previous ones.

Contract CO6: placeAsteroid
Operation:placeAsteroid()
Cross References: User Creates a New Game Scheme, Create a New Game, Initializing Game
Preconditions:

- Player must be logged into the system.
- Player must have selected the create new game scheme option.
- Player must be in builder mod.

Postconditions:

- One more asteroid into gameplay grid added.
- Remaining asteroid count is updated for each category of asteroids.
- The new added asteroid should not contact with the previous ones.

Contract CO7: displayGridState
Operation:displayGridState()
Cross References: User Creates a New Game Scheme, Create a New Game, Initializing Game
Preconditions:

- Player must be logged into the system.
- Player must have selected the create new game scheme option.

Postconditions:

- The updates that have been made on the grid will be shown to player instantly.

Contract CO8: returnCreatedGame
Operation:returnCreatedGame
Cross References: Create a New Game, Initializing Game
Preconditions:

- Player must be logged into the system.
- Player should be done with game creating menu and hit start button.

Postconditions:

- The game that is created will send message to initialize the game
- The objects in the game grid will have to start to move, and game should be started.

Contract CO9: createSave
Operation: createSave(currentState)
Cross References: Use Cases: saveGame
Preconditions:

- Player is in a game.
- Game is paused.

Postconditions:

- A save at the database or file is created.

Contract CO10: initializeGame
Operation: InitializeGame(startGame)
Cross References: Use Cases: Initialize Game
Preconditions:
- Player login into game.
- Player creates new game scheme or loads an existing one.

Postconditions:
- Game Initialized.

Contract CO11: rotatePaddle
Operation: rotatePaddle(paddle)
Cross References: Use Cases: Move paddle
Preconditions:
- Player initializes the game.

Postconditions:
- Paddle is rotated between 45-135 degrees

Contract CO12: reflect
Operation: reflect()
Cross References: Use Cases: Ball Movement, Starting the Game by Firing the Ball
Preconditions:
- Player is in a game.
- Player’s number of lives is greater than 0.
- Ball interacts with any other object such as paddle or asteroid.

Postconditions:
- Ball is reflected with the same coming angle.

Contract CO13: spawnAlien
Operation: spawnAlien(String alienType)
Cross References: asteroidCount, time, asteroidDestroyed
Preconditions:
- Repairing Alien can only spawn if at least 1 asteroid is destroyed.
- Cooperative Alien will not spawn again if it spawned before and got hit by the player.
- If an alien is on screen, it will not spawn another one.
- They will get an alert if a gift asteroid is destroyed.

Postconditions:
- A spawned alien will perform whatever action they are supposed to do depending on their type. (Repairing, Cooperative, Time-wasting, Protecting, Surprising)

Contract CO14: increaseSpeed
Operation: increaseSpeed (paddle)
Cross References: Use Cases: Move Paddle
Preconditions:

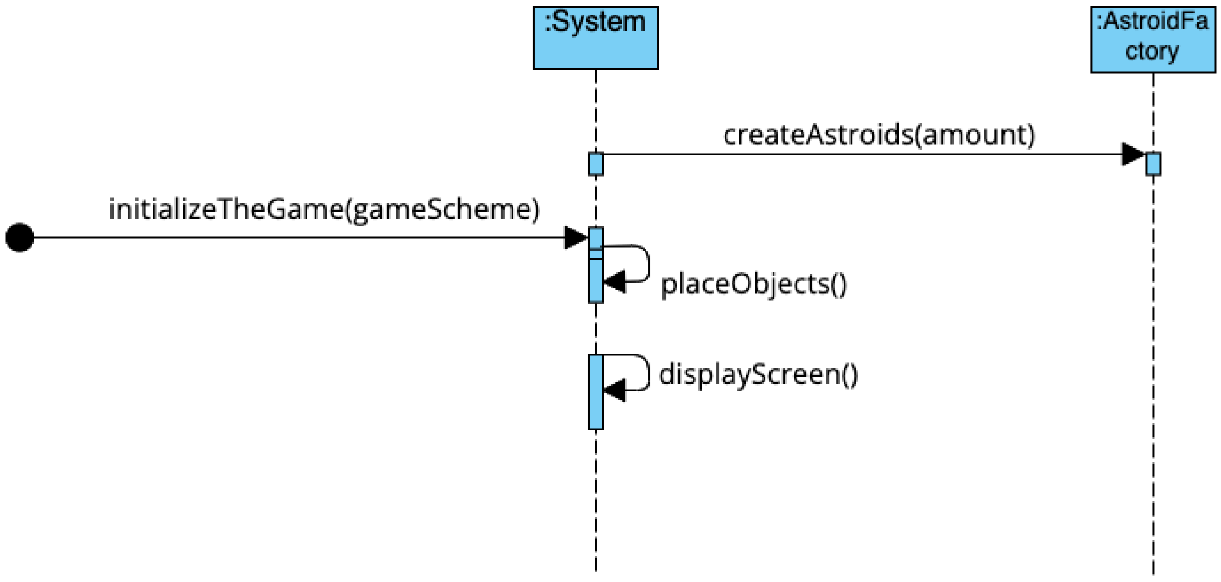
- Player is in the game.
Postconditions:
- The speed is increased.

Contract CO15: winGame
Operation: winGame()
Cross References: asteroidCount, time, score
Preconditions:
- Time left should not be zero.
- The player must destroy all asteroids.
Postconditions:
- “You win!” screen with player score is displayed.

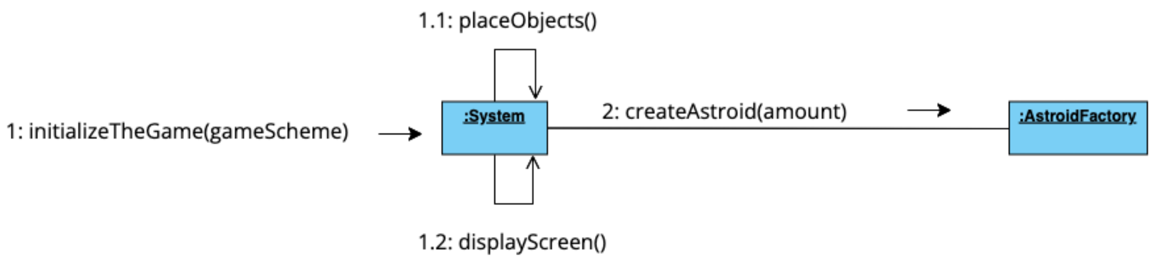
Contract CO16: loseGame
Operation: loseGame()
Cross References: asteroidCount, time, playerHealth,
Preconditions:
- Time left should hit zero or the player should not have any health left.
Postconditions:
- “You lose!” screen is displayed.
- Start a new game option is offered. (possibly)

Interaction Diagrams

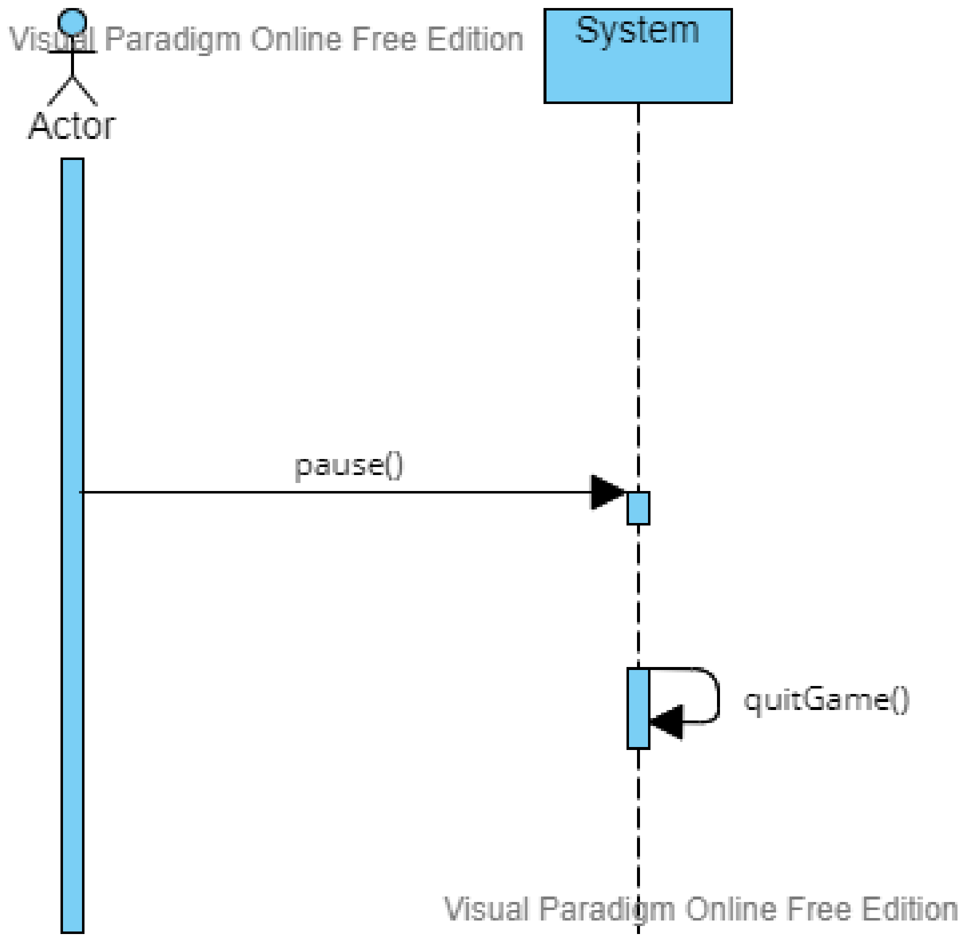
Sequence Diagram 1: Initialize the Game



Communication Diagram 1: Initialize the Game

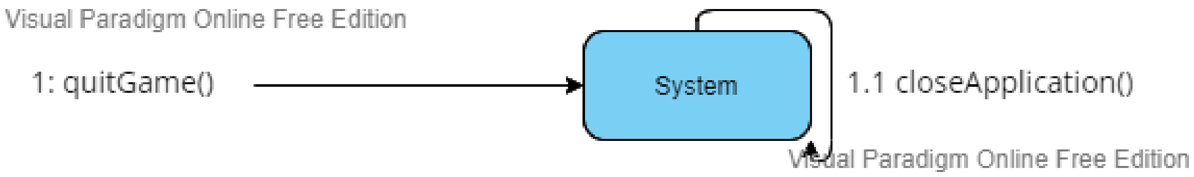


Sequence Diagram 2: Quit Game



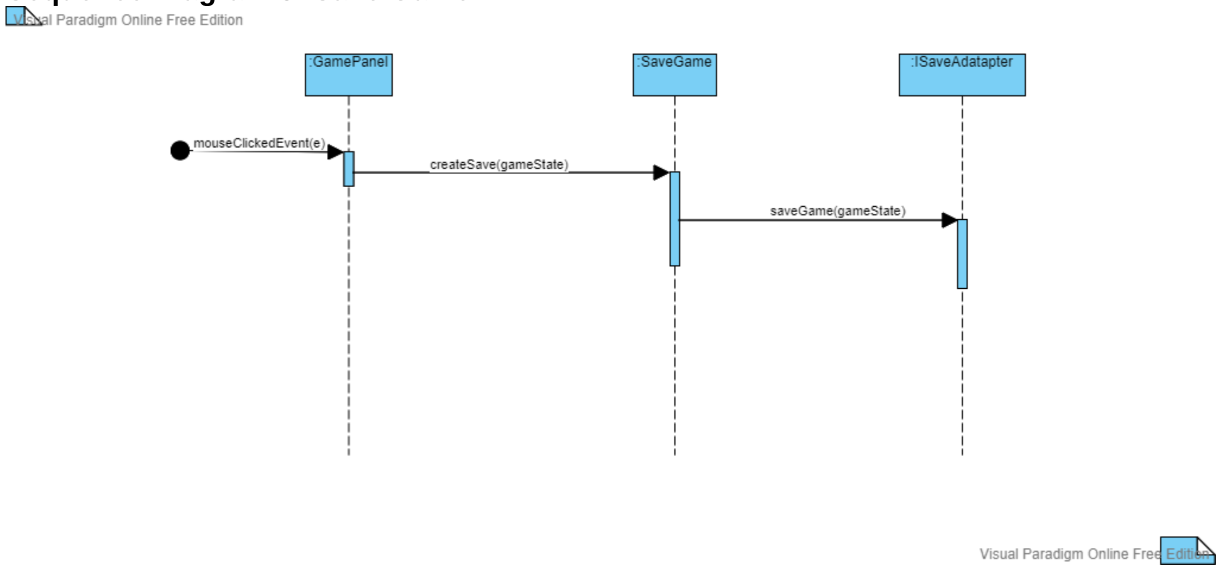
Visual Paradigm Online Free Edition
System
Actor
pause()
quitGame()
Visual Paradigm Online Free Edition

Communication Diagram 2: Quit Game

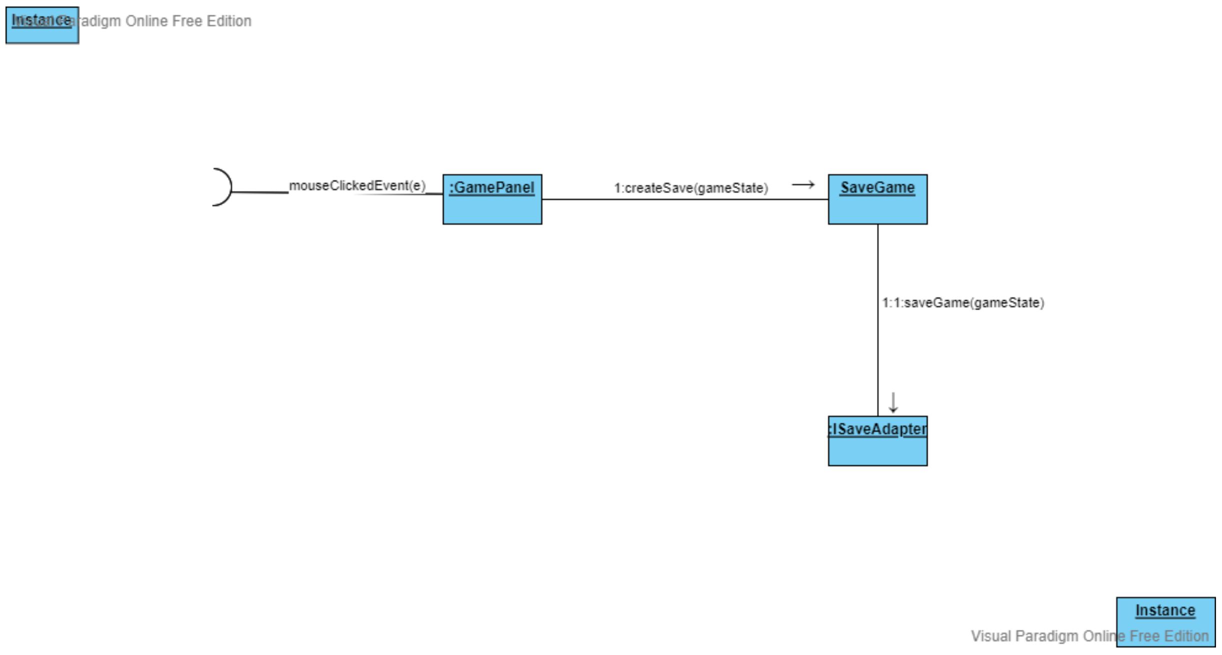


Visual Paradigm Online Free Edition
1: quitGame()
System
1.1 closeApplication()
Visual Paradigm Online Free Edition

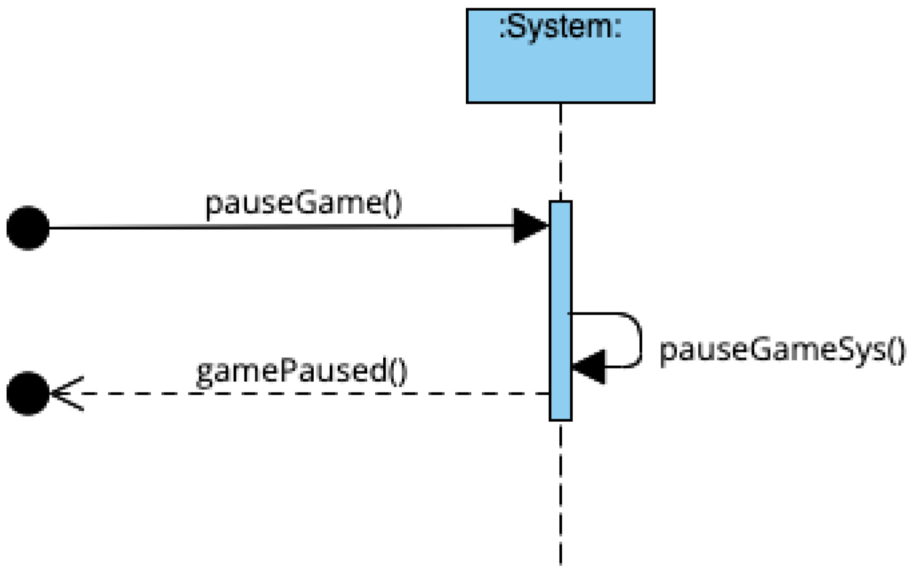
Sequence Diagram 3: Save Game



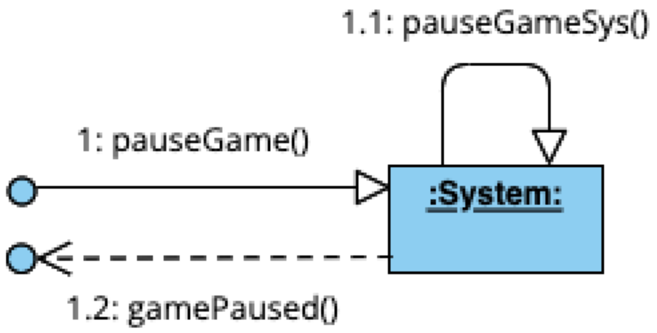
Communication Diagram 3: Save Game



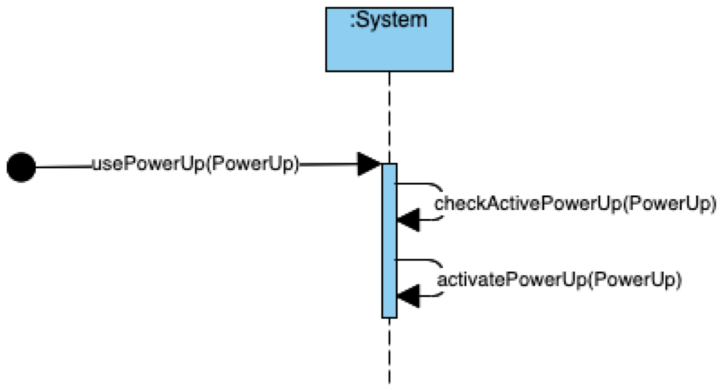
Sequence Diagram 4: Pause Game



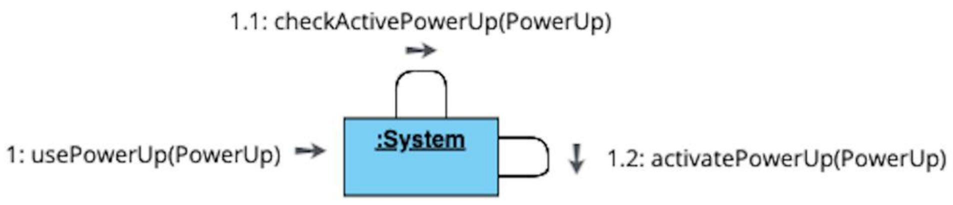
Communication Diagram 4: Pause Game



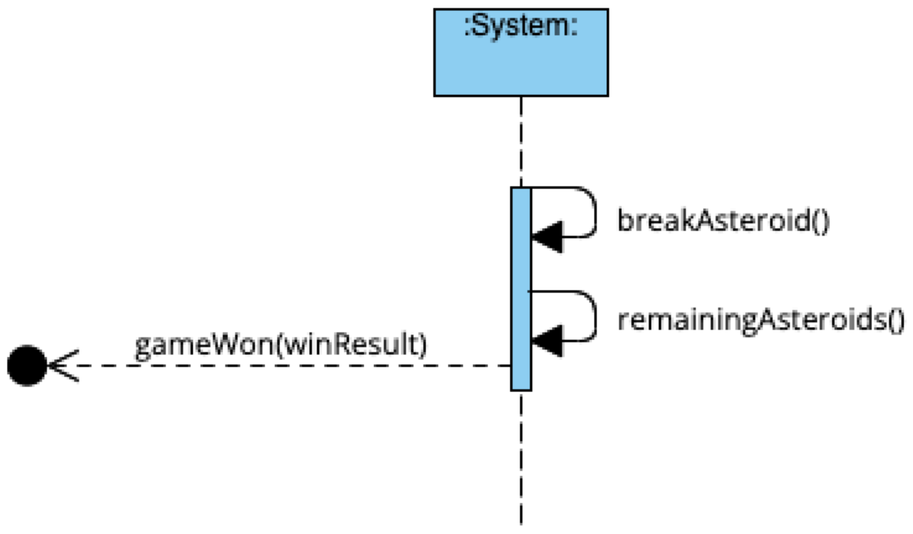
Sequence Diagram 5: Power Up



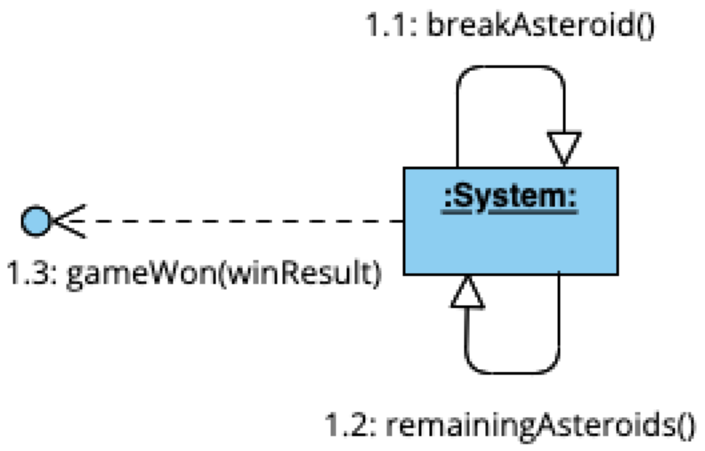
Communication Diagram 5: Power Up


1.1: checkActivePowerUp(PowerUp)
1: usePowerUp(PowerUp)
:System
1.2: activatePowerUp(PowerUp)

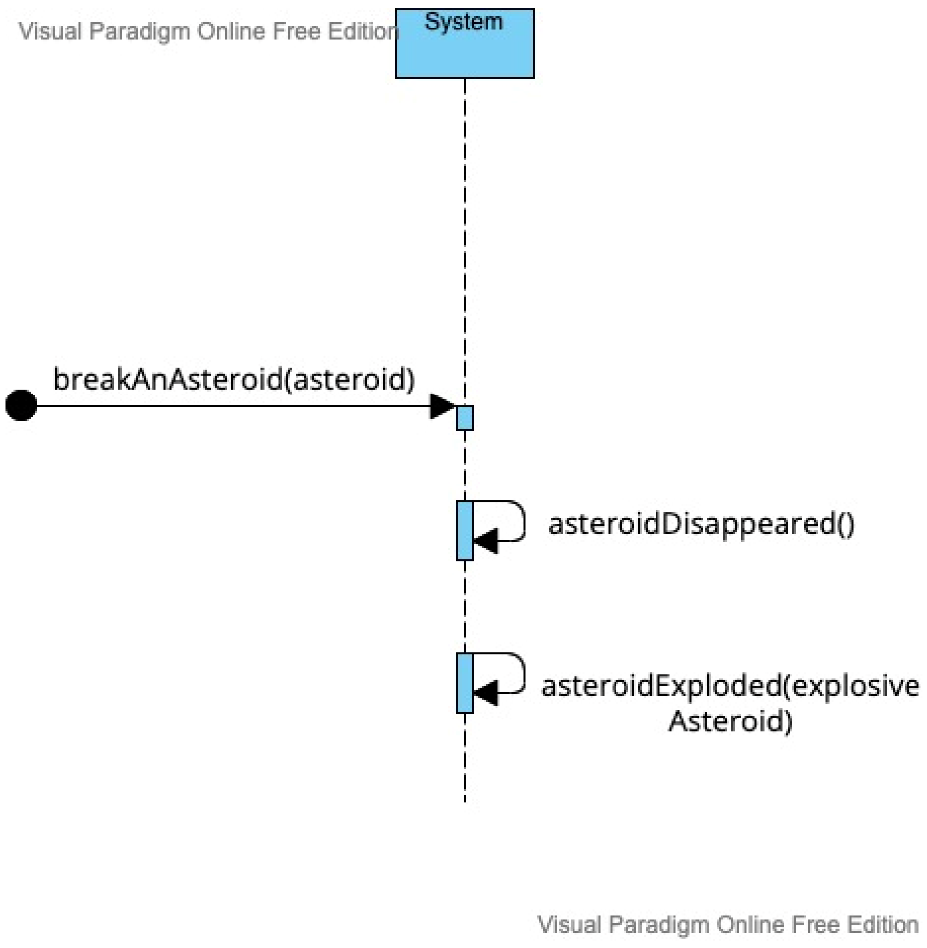
Sequence Diagram 6: Win Game



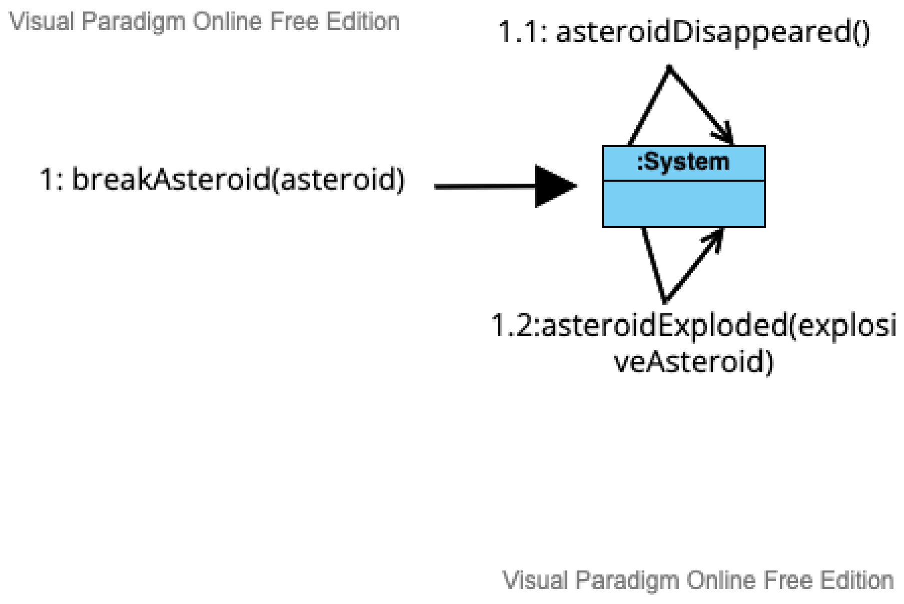
Communication Diagram 6: Win Game



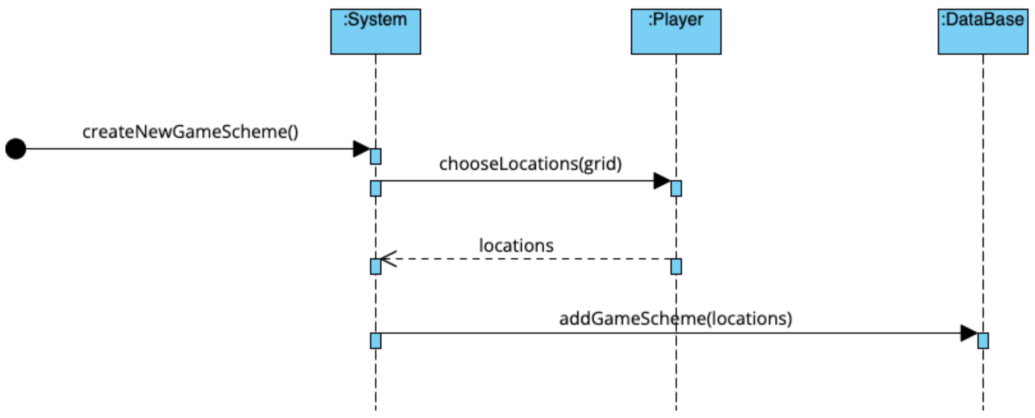
Sequence Diagram 7: Break Asteroid


Visual Paradigm Online Free Edition
System
breakAnAsteroid(asteroid)
asteroidDisappeared()
asteroidExploded(explosive Asteroid)
Visual Paradigm Online Free Edition

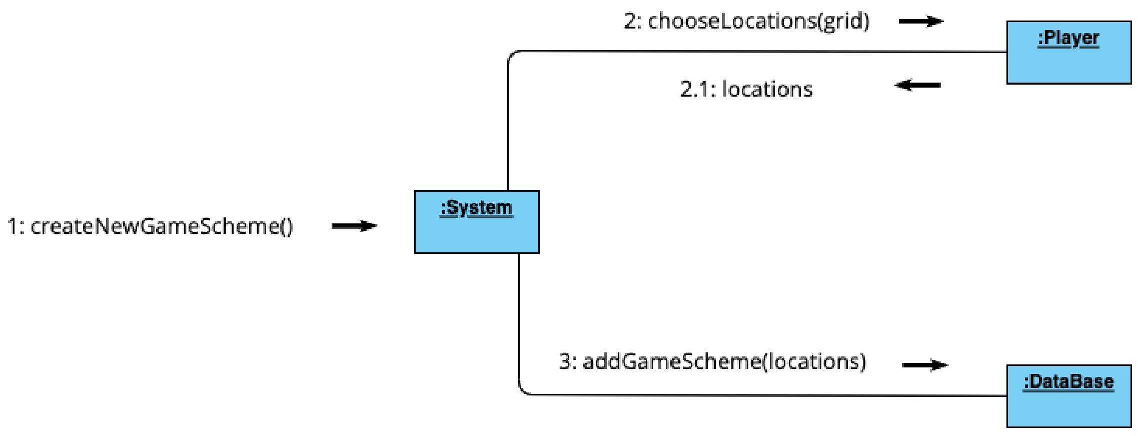
Communication Diagram 7: Break Asteroid


Visual Paradigm Online Free Edition
1.1: asteroidDisappeared()
1: breakAsteroid(asteroid)
:System
1.2:asteroidExploded(explosiveAsteroid)
Visual Paradigm Online Free Edition

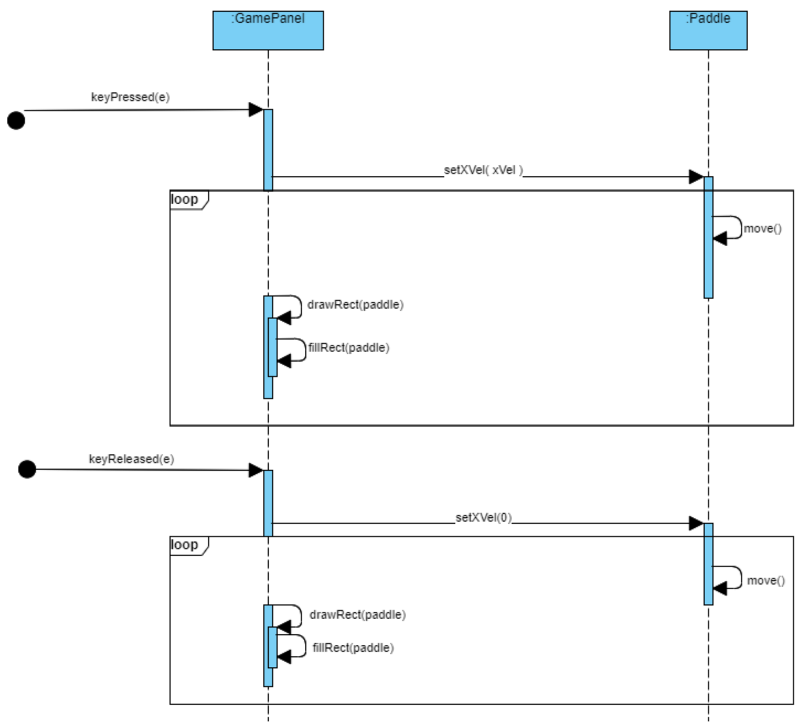
Sequence Diagram 8: Create New Game Scheme



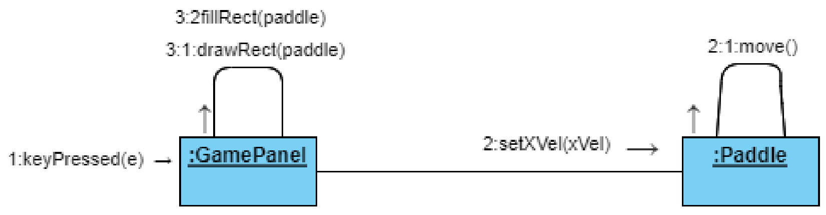
Communication Diagram 8: Create New Game Scheme



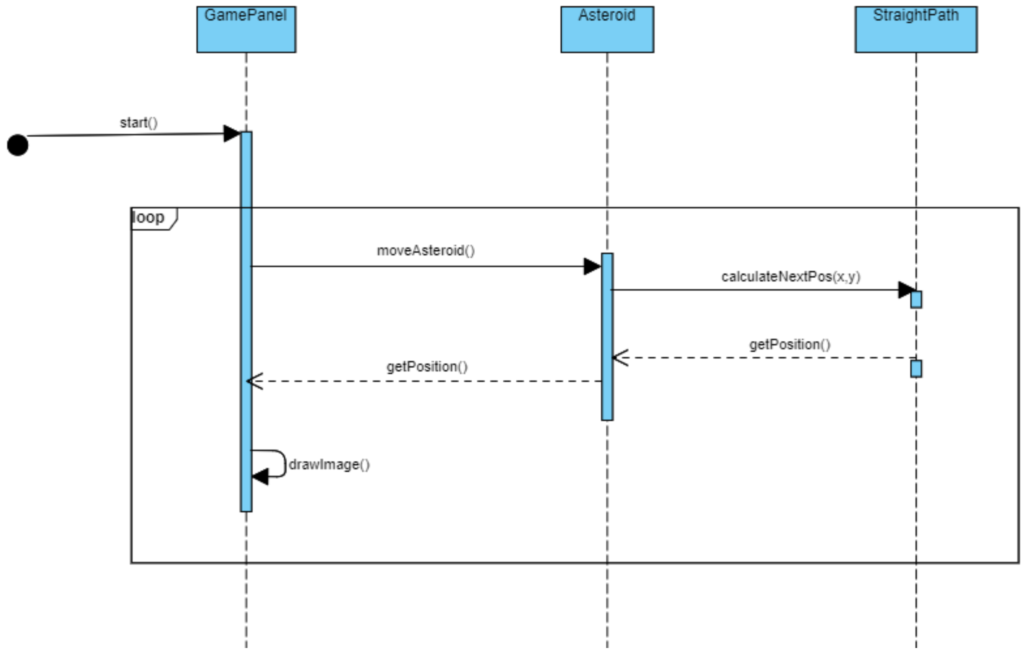
Sequence Diagram 9: Move Paddle



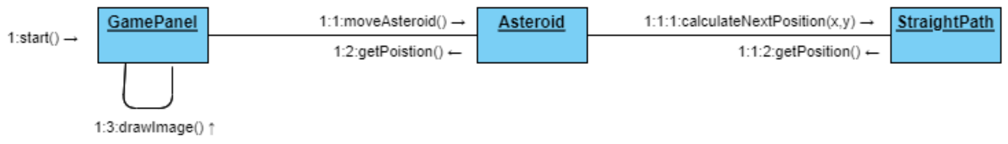
Communication Diagram 9: Move Paddle



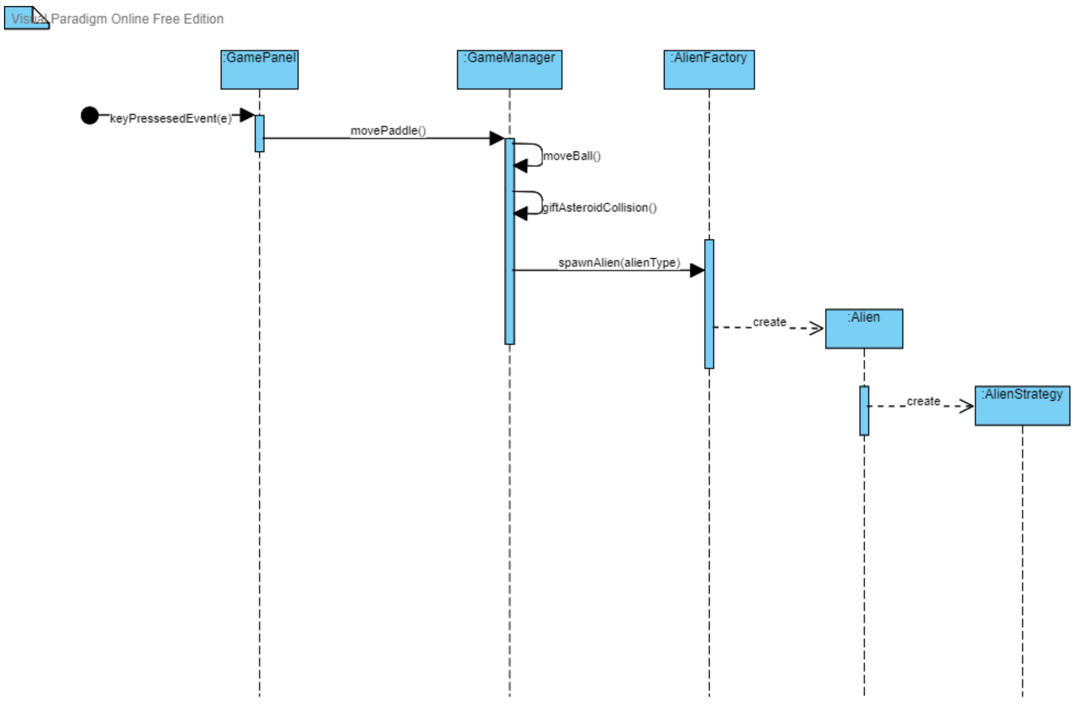
Sequence Diagram 10: Move Asteroid



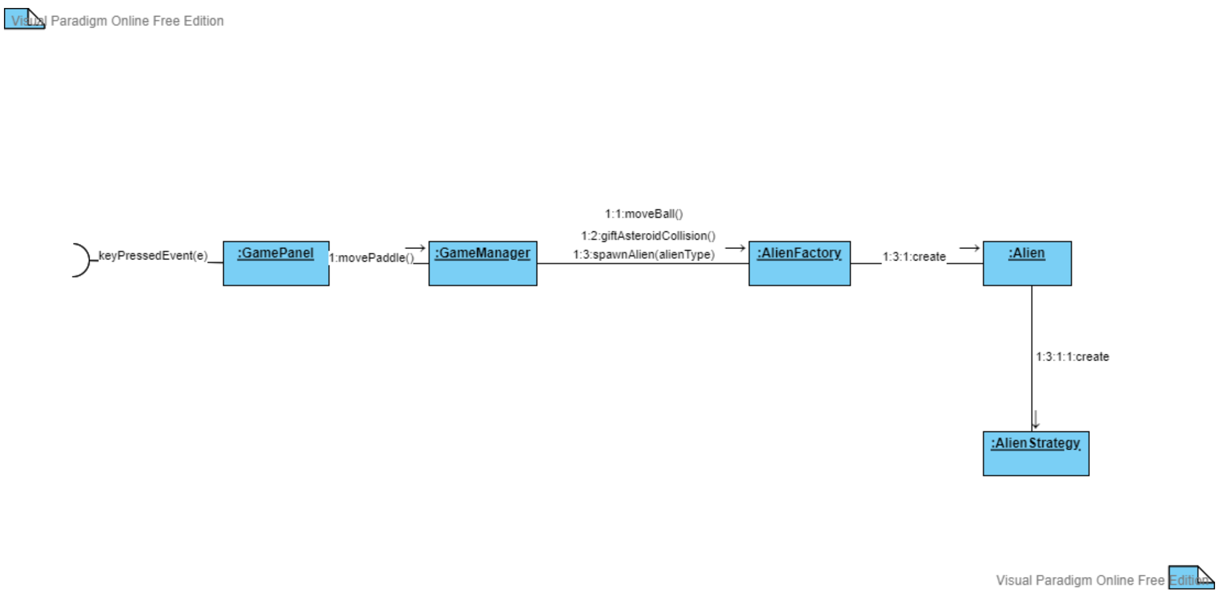
Communication Diagram 10: Move Asteroid



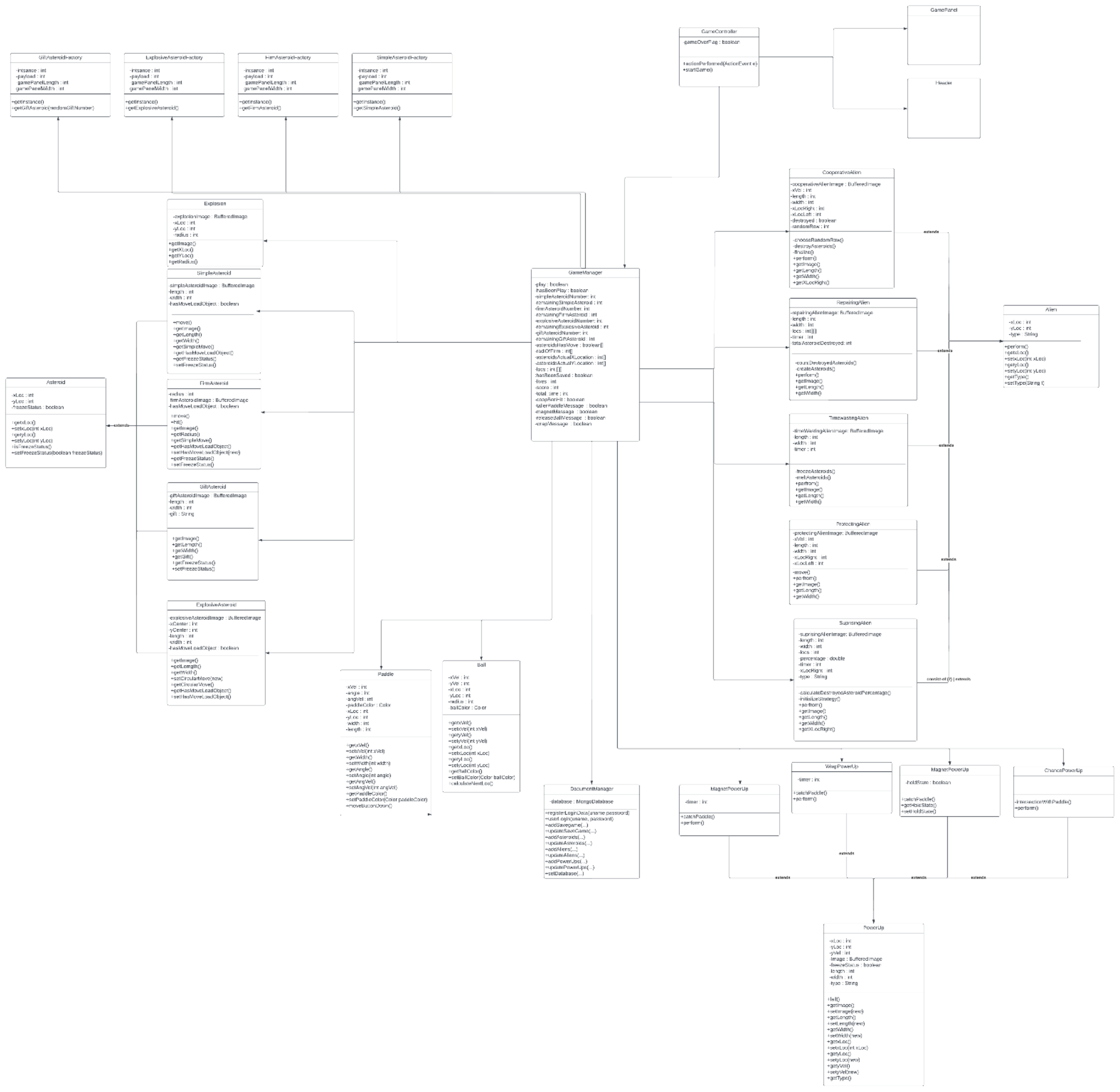
Sequence Diagram 11: Spawn Alien



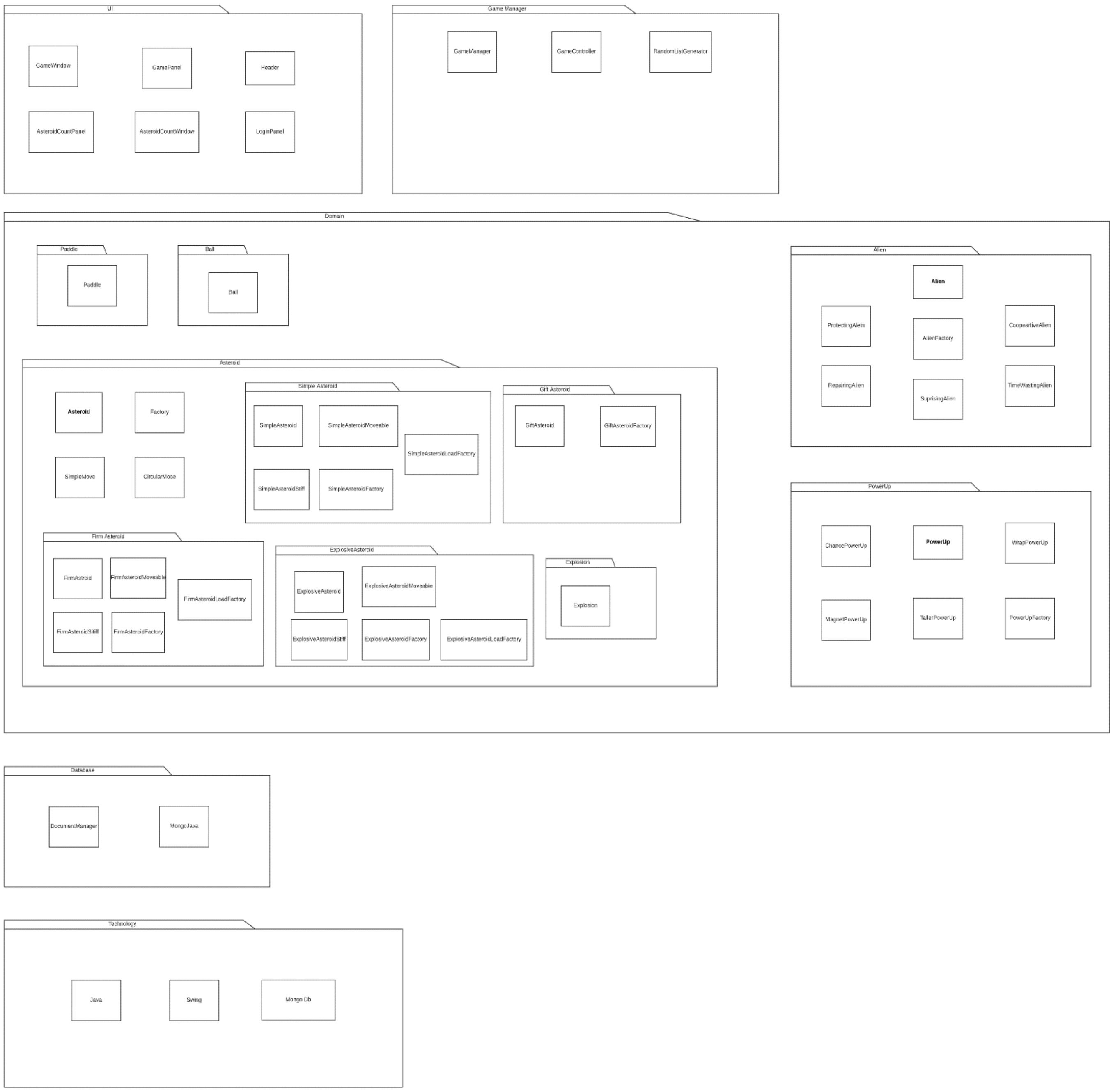
Communication Diagram 11: Spawn Alien



UML Class Diagrams



UML Package Diagrams



Design Discussion

Model-View Separation

Model-View Separation is the principle states that Domain Classes and UI Classes should not have knowledge about each other. An UI element should have nothing to do in the calculation, and logics in Domain objects; on the other hand, Domain element should not be aware of the method and implementations in the UI components of the Project. Domain Classes should be responsible for all the changes that will occur in the game, and UI Classes should be only responsible for transition of these changes into an interface that Users can interact.

General Responsibility Assignment Software Patterns (GRASP)

Creator Principle

Creator principle implies that the creation responsibility of one object type should be done by a class that has knowledge about the created class. Creator principle uses this idea to assign responsibility for creation of object types. When implementing Creator Principle into project, programmers should also consider the Low-coupling and the High-Cohesion into account. Programmers should not simply give responsibility of the creation of the object to a class easily because it has a relation with the created object. For instance, in out project the Simple Factory pattern will be used for creation of the Asteroid objects even the Grid has knowledge and relation with the Asteroid class.

Information Expert Principle

Information Expert principle argues that when an information is needed from one class, the information should be translated with considering the knowledge relations of the given objects. Advantage Information Expert Principle is that, it helps programmers to more organized code. It also helps programmers to share responsibility. The information Expert and the Creator Principle are similar in this fashion, since both of them suggest dividing creation and getting information responsibilities according to knowledge of classes with each other. In our project the information Expert Principle is used when it is needed to get Asteroid’s coordinates

Low-Coupling Principle

Low-coupling principle put a limit on the dependency of the objects to each other. With this principle it becomes easier to change replication of the objects since their dependency between them is limited. Disadvantage of the Low-Coupling pattern is, the relation restriction of the objects should be done in a way that project becomes easier to manage in the terms of the creation of the objects and method usage. We used this principle in out project when we need to define methods of the classes.

Gang of Four (GoF)

Singleton Pattern

Simple Factory design pattern restrict the creation of the same type of object more than one. It only allows programmers to instantiate same object type once and programmer work on the same object type in project. Factories, Controllers and Paddle, are implemented as singleton, since it the project we initiate them for once in the project. The advantage of the singleton pattern is that programmers can work unique class object in throughout the Project. The disadvantage is that Singleton pattern is hard to understand and implement, since it restrict the initiation of the same object type when it is need for testing.

Simple Factory Pattern

Simple Factory is a design pattern allow programmers to create instances of the classes, without specifying the exact class creation syntax. Simple Factory pattern simply takes the responsibility of the creation of objects in the project. In our project Asteroid objects and the ball objects in the case of gang-of-balls power-up is activated will be created with the Simple Factory Design Pattern. The advantage of the Simple Factory design Pattern is that it helps programmers to share responsibility among the classes of the project. Disadvantage of Simple Factory is, it makes code less readable and hard to write since it's an abstract layer of creation of objects. However; once it is fully implemented, this pattern can help programmers to reduce burden on the objects.

Strategy Pattern

Strategy is a design pattern that allow programmers to write more maintainable objects by separating object behaviors into different kind of classes. In our project the Asteroid classes will have different kind of Path classes to simulate behavior. The advantage of the Strategy Pattern is that the programmers do not have to worry about the inheritance related problems in the project. The disadvantage of Strategy Pattern is programmers have to create one class for each different behavior type; and this make the class number grow fast. However, with this pattern we will be able to write and change our implementation more smoothly.

Adapter Pattern

Adapter is a design pattern that help programmers to use different classes, that work similarly but have different methods for the same job, with generic commands. Adapter design pattern makes it easier to switch from one object type to another. In our Project we can use Adapter Design Pattern for implementation of different database's. With this when we need to switch from one database type to another we will not be dealing with the reimplementation of the same code. Adapter Design Pattern advantage is that it provides a generic structure for the programmers, and it makes code more maintainable for different implementations. The disadvantage of Adapter Pattern is that it makes hard to test different implementations for programmers.

Supplementary Specifications

FURPS+

- **Functionality**
  If an error occurs in the system, the errors are written to a text file and the player is informed with an error message.
  The user needs to login to play the game.

- **Usability**
  There will be GUI elements for aliens, ball, paddle, asteroids as well as buttons and texts. All the elements are visible from 1 meter. A space color theme will be chosen for game color theme.

- **Reliability**
  When a minor error or exception occurs in the game, the game continues to work. For example, when the score of the user calculated incorrectly, the game would continue.

- **Performance**
  The performance of the system is crucial for our system since there is a frequent interaction between the system and the player. The player should be able to move the paddle at the moment the player presses the key.

- **Supportability**
  The system should be adaptable to changing requirements. For example, the game can be changed in a way that user can spend his/her scores to buy power-ups or a different paddle/ball. Also, the system should be open to additions of new aliens or different types of asteroids.

Nonfunctional Requirements for Animation

- **Usability**

The animation of the Outer Space Game should have a proper refreshing rate, so that it does not make people feel dizzy. Also, the game should not lag because of its high refresh rate. Colors of the game should be appropriate for color blinds so that they should not have any difficulty during playthrough in the game. The help screen and login screen characters size should be readable from 75 cm away.

- **Reliability**

If an error occurs in the game, the game will simply flash an error message and save players game so that player can be able to continue from the situation that the game crashed. Animation system will be using thread, since it is the only component that access thread and alter shared data; there will be not synchronous threads going on, so there will be no concurrency-based errors. Therefore, thread synchronization is not necessary for Outer Space Game. If there occurs an error during accessing a saved game, the game will simply flash an error message and game will ask player to start a random game.

- **Performance**

During the playthrough of the game, the refreshing rate will have an enormous role on the game performance. The calculation and state updates should be done effectively, so that player should not be aware of the game lag. The collision calculation and movement of the asteroid, ball and paddle should also be synchronous, and they should be done in the time interval of the Outer Space Game refreshing rate. The login screen, however, should not be a burden on performance since there will be no multi calculations going on in the background. But accessing and retrieving saved game data should be done in a reasonable amount of time, and player should not wait too long for creating new game.

- **Supportability**

Outer Space Game animation should support flexible in terms of path organization. Different kind of path requirements should be supported during the animation of the game, and it should not fail during accessing different object types during creation and destruction of the objects. The game animation should be able to overcome further obstacles to maintain itself in different operations.

Glossary

	Glossary	
Term	**Definition and Information**	**Format**
Player	A person who is interacts with the game and trying to win.	Object
Ball	The object that player uses to destroy asteroids.	
Paddle	The object player uses to control the movement of the ball/balls.	
Help screen	Help screen explains the game objects and how to play the game.	
Grid	The space that asteroids, ball, paddle and aliens live within. Every interaction occurs in the grid.	
Pause/resume	The Player can pause the game whenever she/he decides to and resume it later.	
Score	The indicator of the success in the one game rally.	
Database	The file which player information and saved games are hold.	
Power-ups	The power-ups are abilities that can be collected by player during the game. Player could collect multiple power ups but can only use one at a time.	
(Power-ups) Taller paddle	This doubles the length of the paddle, the effect of this power-up stays for 30 seconds.	
(Power-ups) Magnet	This feature allows the paddle to catch the ball instead of reflecting it, and then throwing it using the mouse.	
(Power-ups) Destructive Laser Gun	It is attached to both ends of the paddle, it can destroy a full column of blocks, even the firm asteroids. The player has five shots per each power-up.	
(Power-ups) Chance	In the beginning of the game, the player has three lives. This power-up increases the player's chances(lives) by 1.	
(Power-ups) Wrap	This power-up allows player's paddle to go through the side of the screen and re-appear on the opposite side.	
(Power-ups) Gang-of-balls	Once a gift asteroid having gang-of-balls is hit, the hitting ball will turn into 10 replicas.	
Building Mode	In the building mode, the user can load an already existing game scheme or create a new one.	

	Glossary	
Term	**Definition and Information**	**Format**
Running Mode	In running mode, the user controls the paddle to break the asteroids furthermore user-system interactions are handled according to game rules.	Object
Login screen	Appears when the game executable is run. Every player should have a unique login name.	
Asteroid	The objects that player must destroy to increase score or win the game	Object
Simple asteroid	Can be broken in one hit. When broken, an asteroid just disappears.	Object
Firm-asteroid	These asteroids have strong metal inside them. To destroy these asteroids, the player needs to hit more than one, depending on the radius of the asteroid.	Object
Explosive-asteroids	These asteroids have circular shapes and they explode once being hit. Once exploded, they destroy the neighbor asteroids.	Object
Gift asteroids	These asteroids can be destroyed in one hit like simple ones. However, they are hiding inside power-ups or triggers for the aliens.	Object
Alien	System controlled objects that have an impact on the players gameplay either beneficial or detrimental.	Object
Repairing alien	These aliens can create only the simple asteroids and They can build a single asteroid each five seconds and If hit by the ball, this type of alien will stop building and escape.	Object
Protecting alien	These aliens try to protect the wall by moving under the asteroids and if the ball hits from the bottom or sides nothing will happen to the alien and ball bounces back and If the ball hits the top, the alien will escape.	Object
Cooperative alien	This feature allows the paddle to catch the ball instead of reflecting it, and then throwing it using the mouse.	Object
Time wasting alien	These aliens try to waste the player's time and protect the wall.	Object