COMP302 Project Final Report

Kaan Dai

Table of Contents

Introduction (Including the Vision)

1. Introduction

My vision is to develop a game called Outer Space which is an easy to play game that combines fun and challenge. In this game, an alien civilization is trying to destroy life on earth by building a wall of asteroids that surrounds the earth and prohibits sunlight. There are different kinds of asteroids which have special features. The player's goal is saving the planet by destroying the wall. The only available weapon that can be used is a paddle and metal-balls. The player tries to destroy the wall by sending a ball through the help of the paddle. While the player is demolishing the wall, some aliens will appear to protect it. There are also good aliens that will help the player by destroying some parts of the wall. The player has a limited time and limited number of balls to break all the asteroids. If the player could not, he/she will lose the game. If the player manages to finish all the asteroids, then she/he wins the game.

2. Positioning

Since retro games are gaining more popularity over the last few years. The game we will develop, Outer Space, will be the remastered, improved and upgraded version of the original brick breaker game, which was originally developed by Steve Wozniak and Steve Jobs for an arcade machine.

2.1 Product Position Statement

For	the gamers
Who	still loves playing the retro games
The	Outer Space is the remastered, improved and upgraded version of the original brick breaker game

That	was originally developed by Steve Wozniak and Steve Jobs for an arcade machine.
Unlike	the original brick breaker game, Outer Space has more features, some upgrades, and better graphics.
My product	is easy to play, fun and challenging.

3. Stakeholder Descriptions

3.1 Stakeholder Summary

Name	Description	Responsibility
Group 6: Avengers	Students who are developing the game	The students who are developing the game should be sure that the game is released without any bugs and errors.
Supervisor	The instructor and the TA's of the course	The supervisors should answer the questions of the group and warn the students about the mistakes they made.

3.2 User Environment

Individual Work and 2 supervisors which are the instructor and a TA. I will arrange weekly meetings with my TA which will be on every Friday at 2.30 pm. Right now we are using Google Docs as the only tool but we are planning to use Eclipse IDE for coding in the near future.

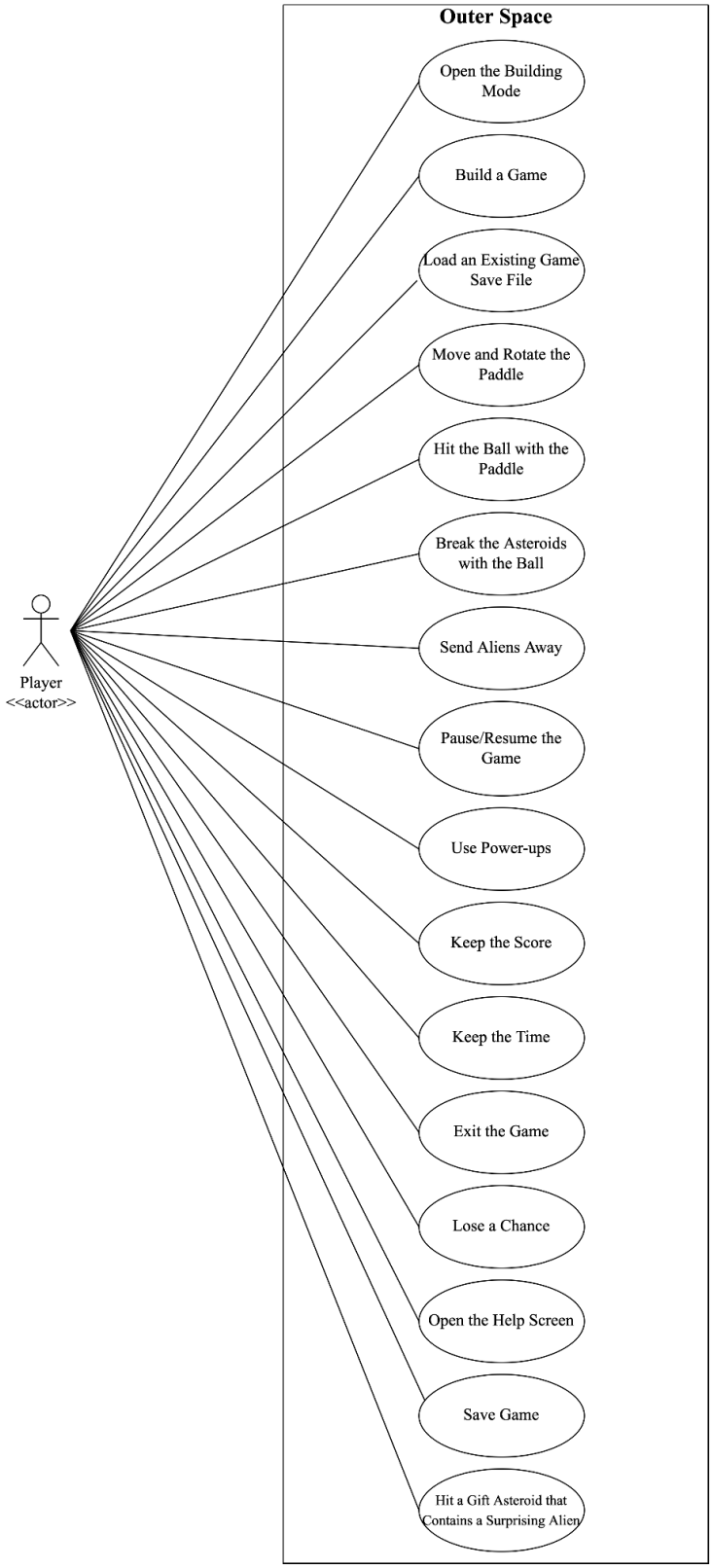
4. Product Overview

4.1 Product Perspective Outer Space is a competitive and enjoyable single player game. Outer Space can be played only if the player has a working keyboard and if the player's computer or the device can run Java programs. Outer Space is an independent product.

4.2 Assumptions and Dependencies Outer Space is a game that is written in Java language, so the computer or the device should be able to run Java programs. The player controls the game with a keyboard and the player must ensure that he/she has a functional keyboard.

4.3 Other Product Requirements: For usability, performance, design constraints and more; see Use Cases and Supplementary Specifications.

Use Case Diagram



Use Case Narratives

Use Case UC1: Build the Game
Level: User goal
Primary Actor: Player
Scope: Building Mode

Stakeholders and Interests:

- Player: wants to create a new game.

Preconditions: Player is logged in to the System and pressed the "build a new game" button.
Success Guarantee (Post Conditions): Player entered the numbers correctly, and made necessary changes regarding the placement of asteroids and the game is ready to start.

Main Success Scenario:

1. System asks Player to specify the number of each asteroid type.
2. Player enters the number of each asteroid type.
3. System checks if the given numbers are greater than the minimum numbers.
4. System puts the given number of asteroids to random places.
5. Player edits the number and the places of asteroids.
6. Player starts the game by clicking the "start" button.

Extensions:

*a. At any time, the System fails.

1. System displays an error message.
2. Player restarts the game.

3a. Player does not satisfy the minimum number of each asteroid type.

1. System displays an error message that says "Please change the inputs".
2. Player enters the number of each asteroid type again.
   2a. Player satisfies the minimum number.
   2b. Player does not satisfy the minimum number.
   (Steps 1 and 2 are repeated until the Player enters the numbers that satisfies the criteria.)

5a. Player does not change anything.
5b. Player clicks "add" buttons.

1a. Number of asteroids on the screen reaches the maximum number.

1. System does not add asteroids.

1b. Number of asteroids on the screen does not reach the maximum number.

1. System adds more asteroids.

5c. Player removes some asteroids by right clicking.

1a. Remaining asteroids satisfy the minimum number criteria.

1b. Remaining asteroids do not satisfy the minimum number criteria.

1. System does not allow the attempt.

5d. Player changes the places of some asteroids.

1a. Player causes an overlap of asteroids.

1. System does not allow the attempt.

1b. Player does not cause any overlap.

1. System changes the placement of asteroids successfully.

Special Requirements:

- All text must be readable.

Frequency of Occurrence: Expected occurrence is once every time the game starts.

Use Case UC2: Move and Rotate the Paddle
Level: Subfunction
Primary Actor: Player
Scope: Running mode

Stakeholders and Interests:

- Player: Player wants to move the paddle horizontally and rotate by -45 or -135 degrees.

Preconditions: Player initialized the environment in building mode. Program is in running mode.
Success Guarantee (Post-conditions): Player moved the paddle horizontally. Player moved the paddle once or in a continuing movement. Also, the paddle can rotate by 45 or 135 degrees if desired.

Main Success Scenario (Basic Flow):

1. Player moves the paddle horizontally to prevent the ball from falling.
2. Player rotates the paddle to arrange the bouncing angle of the ball.
3. Ball bounces from the paddle in calculated angle and direction.
4. Player keeps his/her life since the ball did not fall.
5. Player may catch a power-up by paddle.

Extensions (Alternate Scenarios):

*a. At any time, the program fails.
    1. Error message is displayed.
    2. Player needs to restart the program.
    3. Achievements are gone because the game is not saved.

1a. Paddle hits the edges.
    a. Paddle hits to the left edge
        1. Player can move the paddle only to the right.
    b. Paddle hits the right edge
        1. Player can move the paddle only to the left.

1b. Player misses the ball.
    a. Player has remaining chances.
        1. Player loses one chance.
        2. New ball is located on the paddle.
    b. Player loses all chances.
        1. Player loses the game.
        2. Game finishes.

2a. Player rotated the paddle to the limits.

a. Player rotated the paddle 45 degrees in counter clock direction.
    1. Player cannot rotate the paddle in counter clock direction any more.
    2. Player can rotate the paddle in a clockwise direction.

b. Player rotated the paddle 135 degrees in clockwise direction.
    1. Player cannot rotate the paddle in a clockwise direction anymore.
    2. Player can rotate the paddle in counterclockwise direction.

3a. Paddle is not rotated.
    a. Ball bounces with a 90-degree angle from the paddle.

3b. Paddle is rotated.
    a. Ball bounces with the calculated reflection angle.

5a. Player catches a falling power-up with the paddle
    a. Player decides to use the power-up immediately.
        1. Paddle is modified according to power-up.
    b. Player decides to save the power-up for later.
        1. Power-up is saved to power-ups for later usage.

Special Requirements:

- Keyboard must be working properly.
- Player must be able to control his/her hands in order to press buttons.

Frequency of Occurrence: Starts when running mode starts and ends when the game ends. Thus continuous.

Use Case UC3: Hit a Gift Asteroid that Contains a Surprising Alien
Level: User goal
Primary Actor: Player
Scope: Running Mode

Stakeholders and Interests:
- Player: demands to hit and destroy an asteroid.

Preconditions: Player started the game.
Success Guarantee (Post Conditions): Surprising Alien has appeared.

Main Success Scenario:
1. Player uses the paddle to give direction to the ball.
2. Player hits a gift asteroid with the ball.
3. Player destroys the gift asteroid.
4. System displays a surprising alien.
5. System decides how the surprising alien will behave based on the Player's progress.

Extensions:
*a. At any time, the System fails.
1. System shows an error message.
2. Player restarts the game.

1a. Player fails to catch the ball.
1. Player loses one of the chances.
2. System displays a new ball.
2a. Player can catch the new ball.
2b. Player cannot catch the new ball.

5a. Player destroyed only 30% of the asteroids.
1. Surprising Alien behaves as a Cooperative Alien.

5b. Player destroyed 70% of the asteroids.
1. Surprising Alien behaves as a combination of a Repairing Alien and a Time-wasting Alien.

5c. Player still needs to destroy 40% to 50% of the asteroids.
1. Surprising Alien behaves as a Protecting Alien.

5d. Player still needs to destroy 50% to 60% of the asteroids.

1. Surprising Alien behaves as a Repairing Alien.

5e. Player still needs to destroy 30% to 40% of the asteroids.

1. Surprising Alien cannot decide how to behave.
2. Surprising Alien will stay in the exact place it appeared.
3. Surprising Alien will disappear after 5 seconds.

5f. Player still needs to destroy 60% to 70% of the asteroids.

1. Surprising Alien cannot decide how to behave.
2. Surprising Alien will stay in the exact place it appeared.
3. Surprising Alien will disappear after 5 seconds.

Special Requirements:

- Aliens must have square shapes.

Frequency of Occurrence: Once in every game.

Use Case UC 4: Use Power-ups
Level: Subfunction.
Primary Actor: Player
Scope: Running mode.

Stakeholders and Interests:

- Player: Player wants to use different power-ups at different times to enhance the abilities of the paddle and the ball. By using power-ups the player aims to destroy more asteroids in a shorter time.

Preconditions: Game should be in the running mode. Power-up should be caught by the paddle and no other power up should be in process.

Success Guarantee: Items in the game are affected in a way that Player can destroy more asteroids in a shorter time. Used power-ups are removed from the power-ups list of Player.

Main Success Scenario (Basic Flow):

1. Player catches a power-up with the paddle.
2. Player uses a power-up from the power-ups list.
3. Effects of the power-up materialized.

Extensions:

*a. At any time, the program fails.

1. Error message is displayed.
2. Player needs to restart the program.
3. Achievements are gone because the game is not saved.

1a. Player decides to use the power-up immediately.

1. Player types the first letter of the power-up or clicks with the left mouse button.
    a. Effects of the power-up initialized.

1b. Player decides to save the power-up for later usage.

1. Power-up is saved to the power-ups list of the user.
    a. Icon of the power-up can be seen on the screen.

2a. Player uses a power-up from the power-ups list.

1. Selected power-up is removed from the list.
2. Icon of the power-up can no longer be seen on the screen.
3. Effects of the power-up initialized.

2b. Player uses a power-up which is not in the list.

1. Nothing happens, effects of the power-up are not initialized.

3a. According to different power-ups, different visualizations are added.
3b. According to different power-ups different ability improvements are added.

Special Requirements:

- Keyboard buttons should be working properly.
- Player should be able to reach the keyboard.

Frequency of Occurrence: When a power-up comes out from an asteroid and the player caches it with the paddle. Not continuous.

Use Case UC5: Save the Game
Level: User goal
Primary Actor: Player
Scope: Running Mode

Stakeholders and Interests:
- Player: demands to save the current game

Preconditions: Player started the game.

Success Guarantee (Post Conditions): Player successfully saved the game using either database option or the file option.

Main Success Scenario:
1. Player clicks the “Pause” button.
2. System offers two saving options: “Save to Database” or “Save to TextFile”.
3. Player chooses one of the options.
4. System saves the game into either a database or a text file based on the Player’s choice.

Extensions:
*a. At any time, the System fails.
    1. System shows an error message.
    2. Player restarts the game.

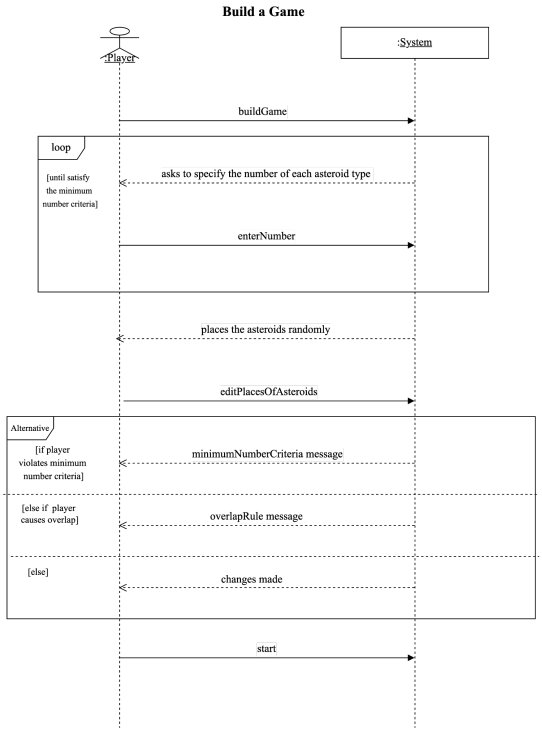
5a. Player chooses saving into a database.

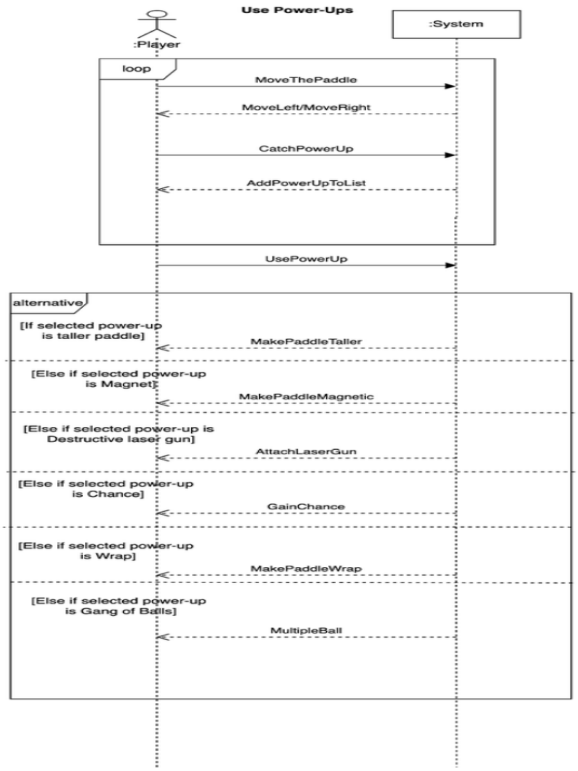
5b. Player chooses saving into a file.

Special Requirements:
- All text must be readable.

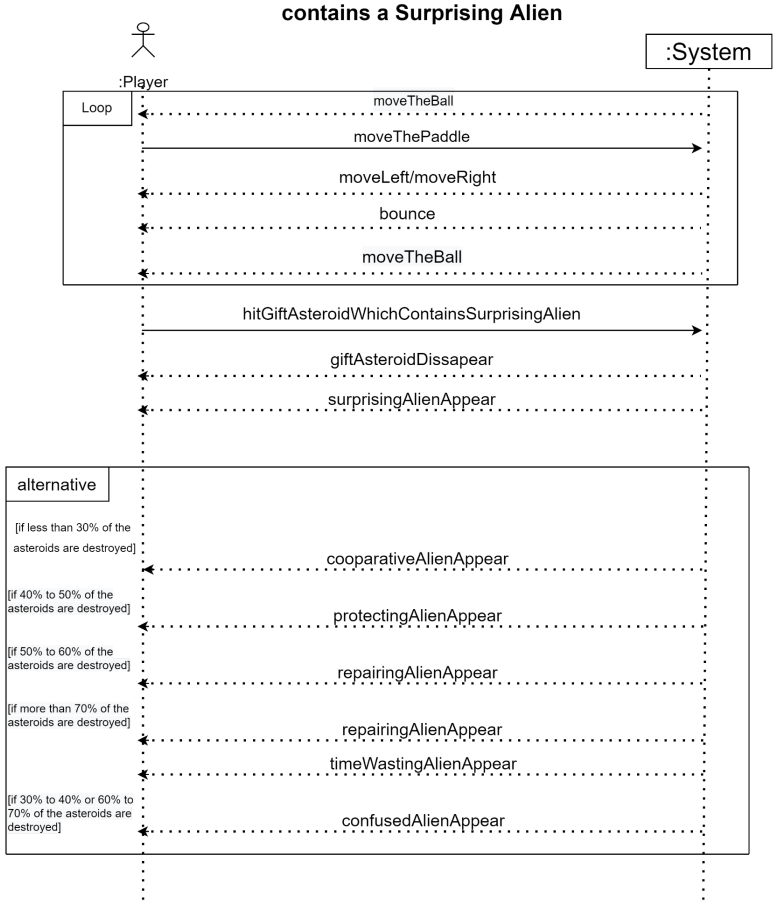
Frequency of Occurrence: Once in every game.

System Sequence Diagrams

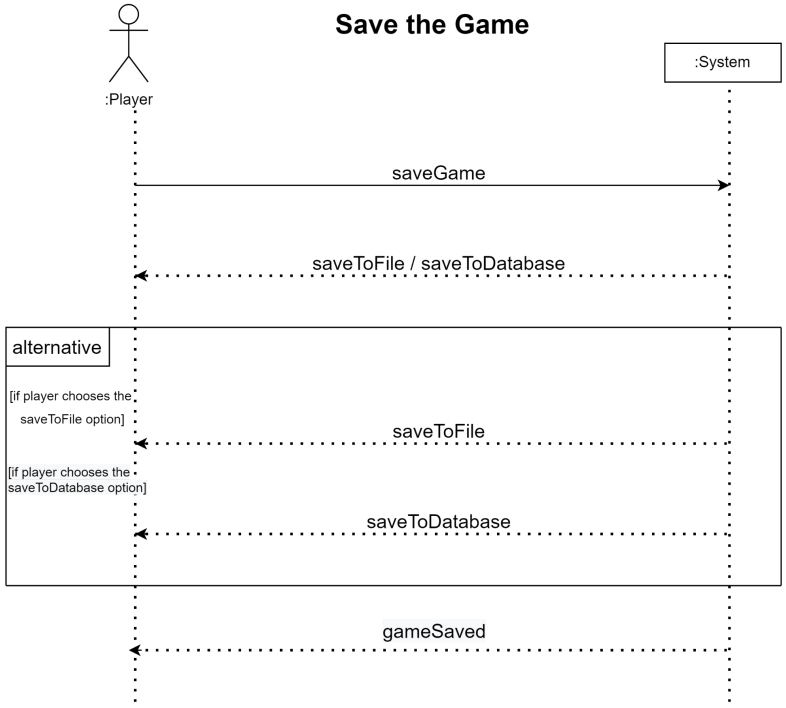


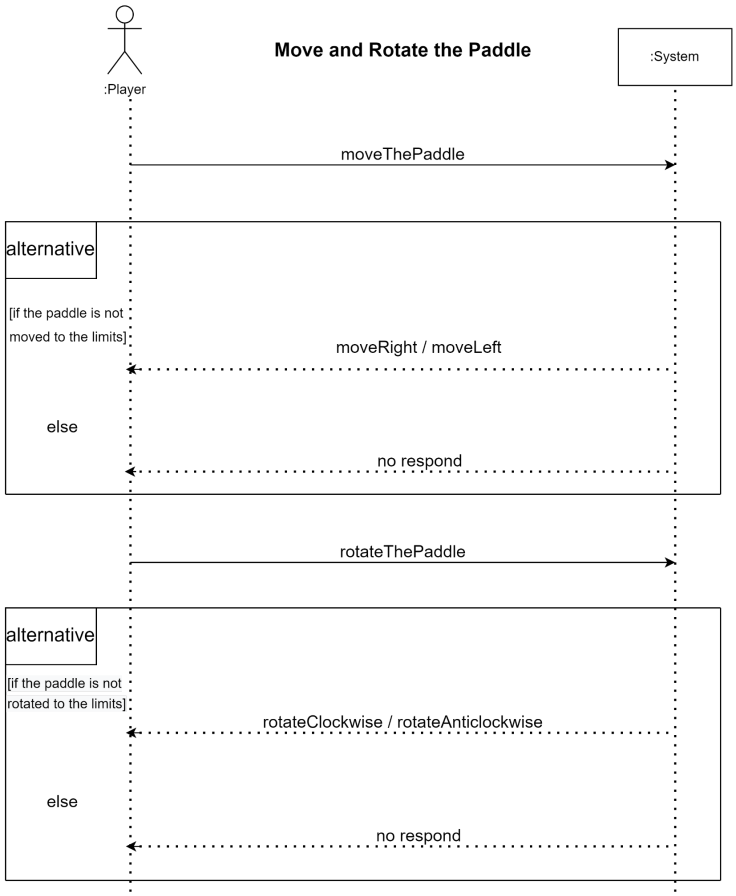


Hit a Gift Asteroid which contains a Surprising Alien



Save the Game




Move and Rotate the Paddle
:Player
:System
moveThePaddle
alternative
[if the paddle is not moved to the limits]
moveRight / moveLeft
else
no respond
rotateThePaddle
alternative
[if the paddle is not rotated to the limits]
rotateClockwise / rotateAnticlockwise
else
no respond

Operation Contracts

Operation: moveThePaddle()

Cross References: Use Cases: Move and Rotate the Paddle

Preconditions: Game should be in the running mode. Paddle is not adjacent to the edge in that direction.

Postconditions:

- Paddle.x became x (Attribute modification)
- Paddle.x2 became x2 (Attribute modification)
- Ball hits paddle if paddle.x == ball.x and paddle.y == ball.y (Association formed)

Operation: hitGiftAsteroidWithSuprisingAlien()

Cross References: Use Cases: Hit a Gift Asteroid that Contains a Surprising Alien

Preconditions: The game should be in running mode. There should be a gift asteroid in the ball's way.

Postconditions:

- The current instance "gift asteroid" was deleted (instance deletion).
- Surprising Alien instance is created (instance creation).

Operation: saveTheGame()

Cross References: Use Cases: Save Game

Preconditions: Game should be in running mode and the game should be started by the player in order to save the game. Player should decide on a saving option.

Postconditions:

- The game is saved with the database option.
- The game is saved with the file option.

Operation: useGangOfBalls()

Cross References: Use Cases: Use Power-ups

Preconditions: Gift asteroid destroyed, gang-of-balls power-up fell and the player catched it with the paddle (Gang-of-balls power-up is available in the power-ups list). Player pressed the “G” button on the keyboard.

Postconditions:

- Gang-of-balls power-up is removed from the power-ups list.
- 10 more balls have appeared.
- Usage of other power-ups have been blocked.

Operation: addNewAsteroid()

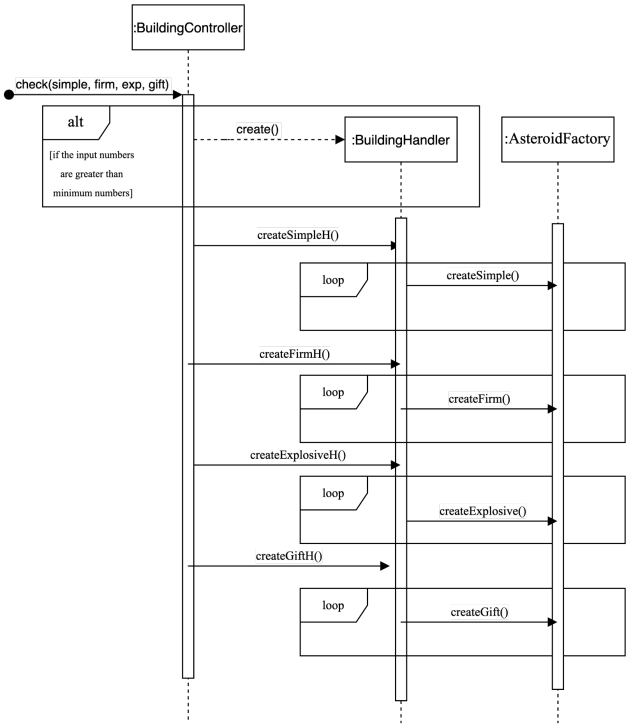
Cross References: Use Cases: Build the Game

Preconditions: Player should logged in and be on be building screen. Player should decide which asteroid to add and press one of the provided buttons accordingly.

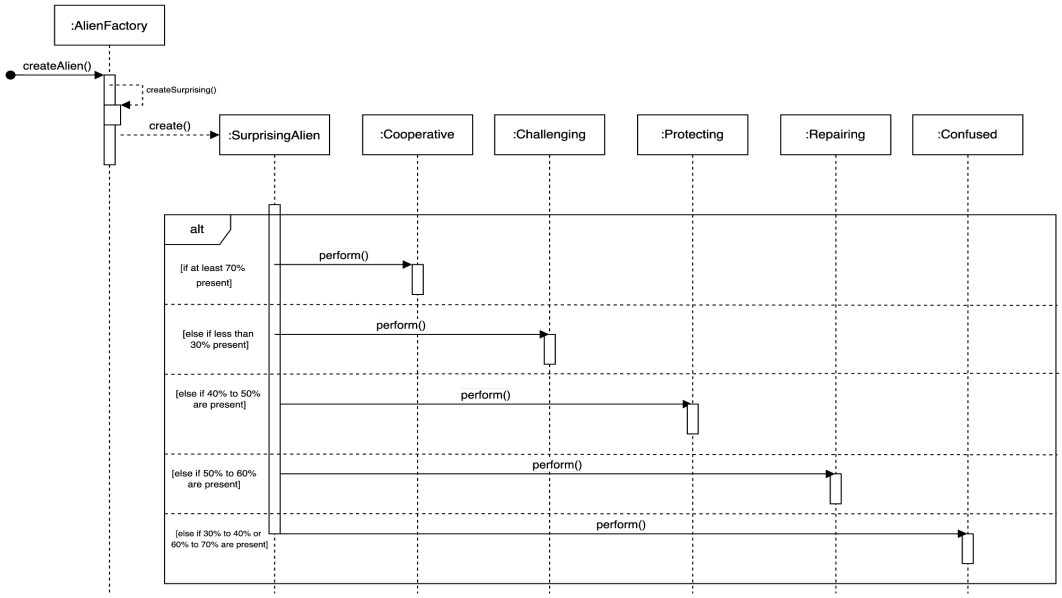
Postconditions:
- If player pressed the “add simple” button, a simple asteroid appears in a random location
- If player pressed the “add firm” button, a firm asteroid appears in a random location
- If player pressed the “add explosive” button, an explosive asteroid appears in a random location.
- If player pressed the “add gift” button, a gift asteroid appears in a random location.

Sequence Diagrams

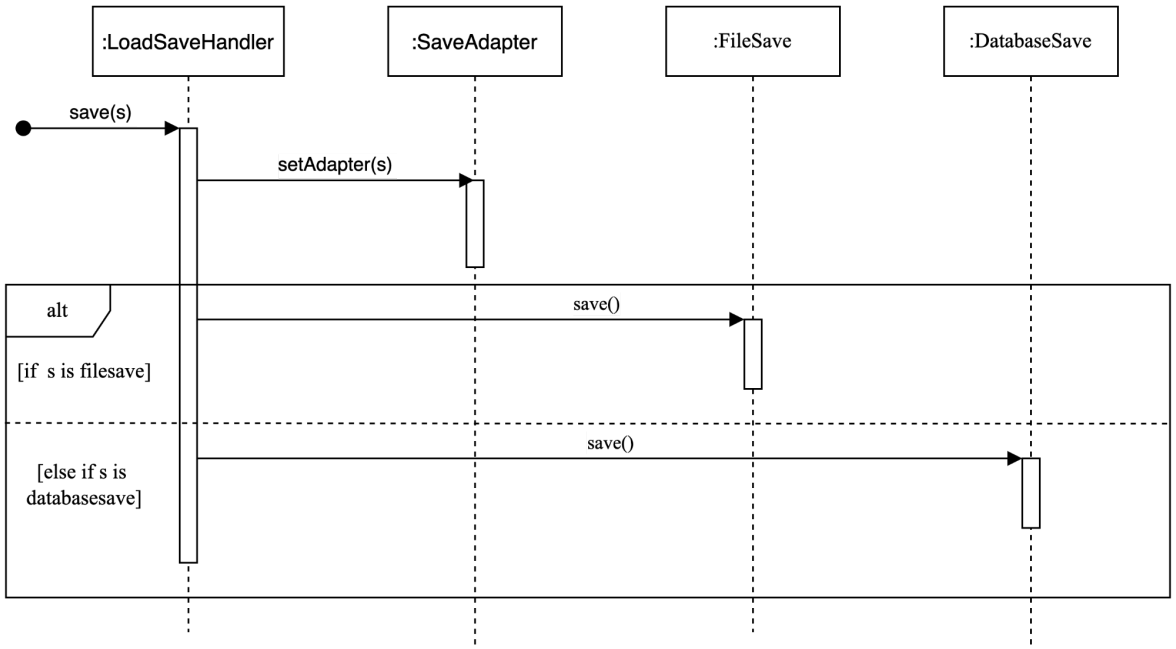
Build Game



Hit a Gift Asteroid that Contains a Surprising Alien

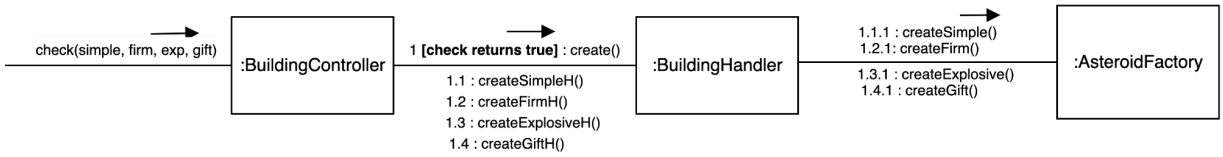


Save Game

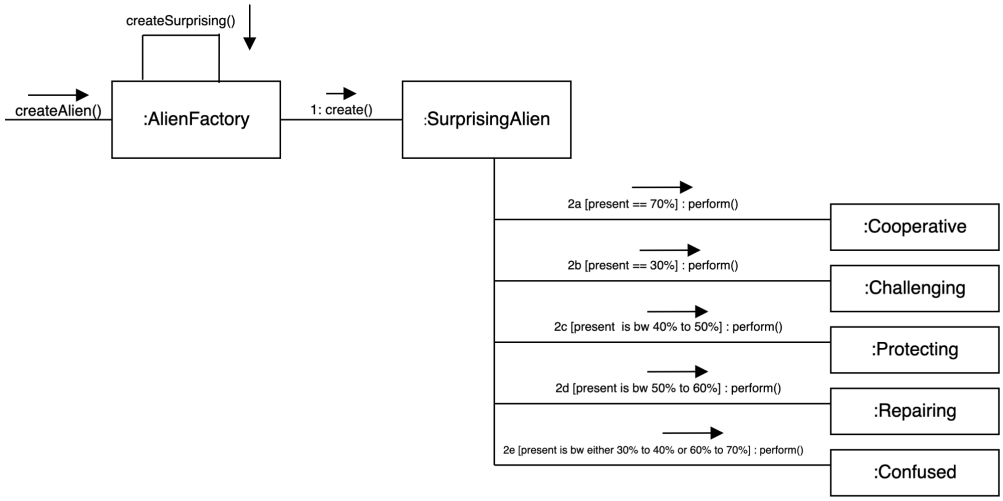

:LoadSaveHandler
:SaveAdapter
:FileSave
:DatabaseSave
save(s)
setAdapter(s)
alt
[if s is filesave]
save()
[else if s is databasesave]
save()

Communication Diagrams

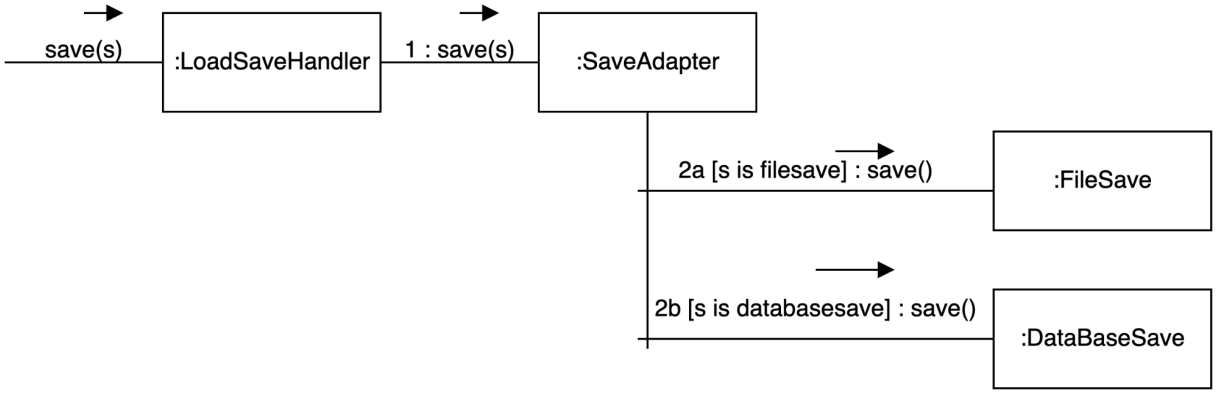
Build Game



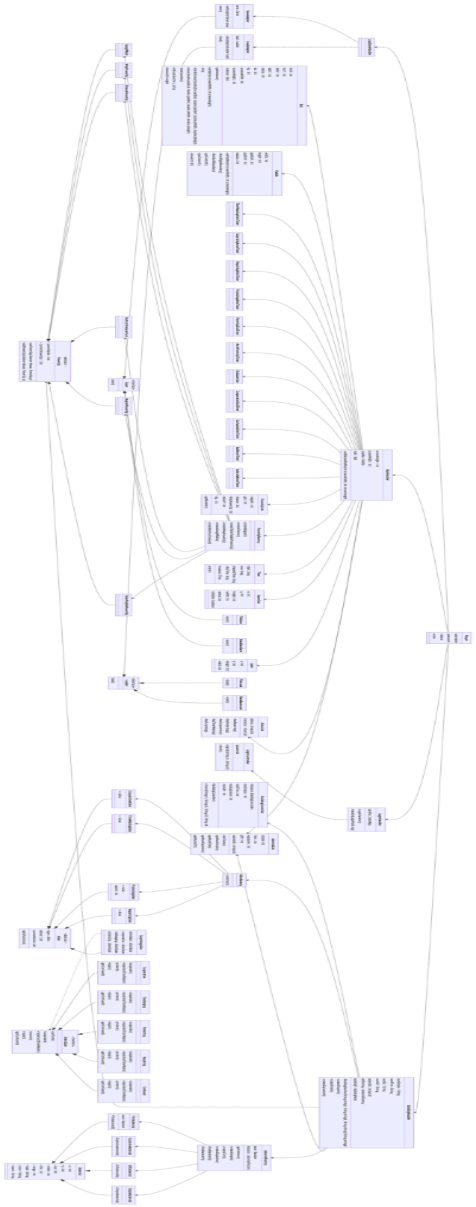
Hit a Gift Asteroid that Contains Surprising Alien



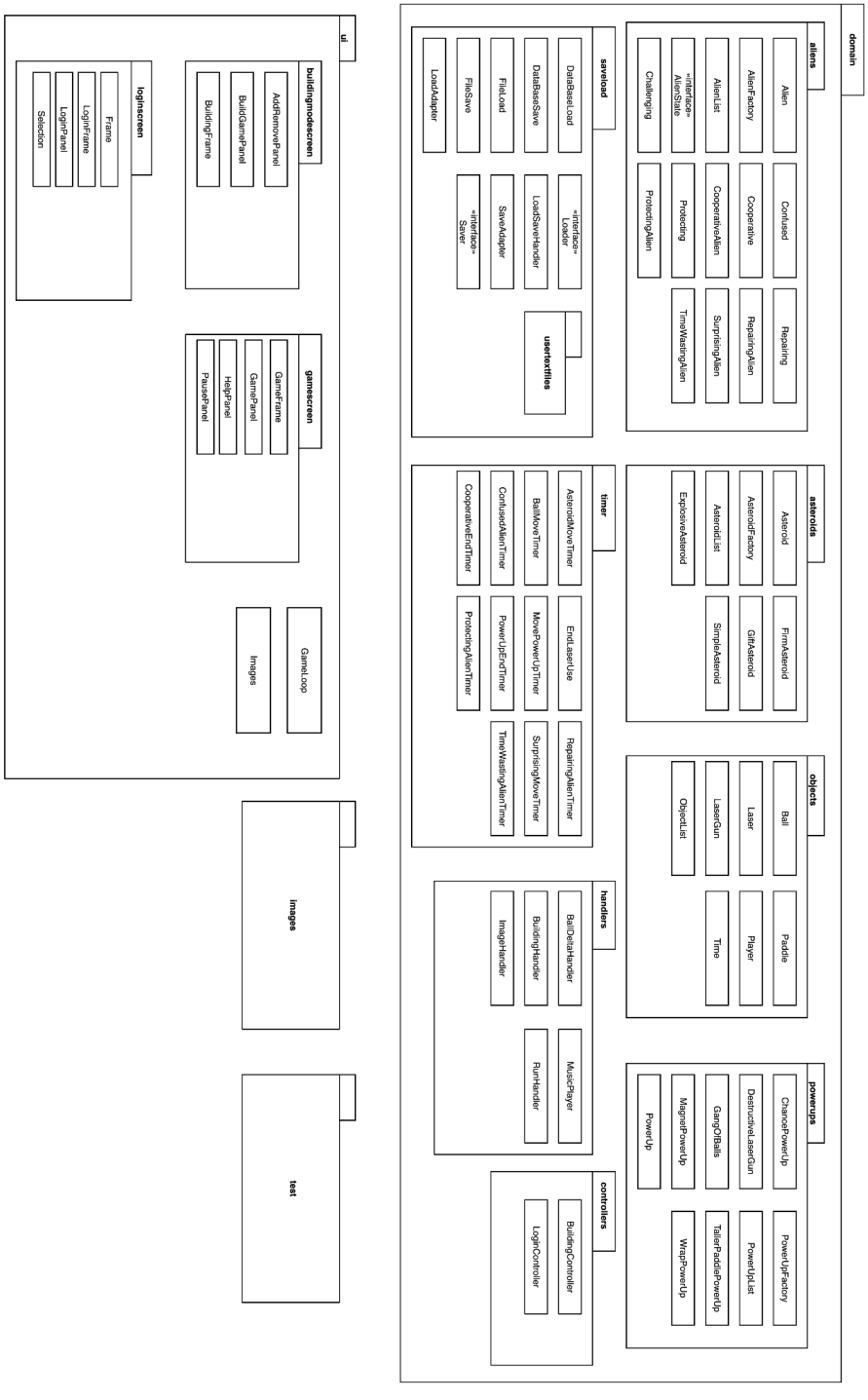
Save Game



Class Diagram



Package Diagram



Discussion of Design Alternatives and Design Patterns

I decided to use the "Simple Factory" pattern in object creation because I wanted the information of object creation to be limited to specific factory classes. I designed my project in a way that there are different factories for different types of object creation. The game has Alien Factory, Asteroid Factory and Power-up Factory. When a new object is created, the "BuildingHandler" or "RunHandler" classes will call the "create" functions of the "AsteroidFactory", "AlienFactory" or "PowerUpFactory" classes. Also, in order to make our design low coupled I created abstract classes for different object types eg: abstract mother class Alien and subclasses of it or abstract mother class Power-up and the subclasses for different types of power-ups that extend the Power-up class.

I tried to follow the Controller Pattern in order to separate the UI packages from the domain packages. According to the Object-Oriented Design principles the UI should not interfere with logic of the application. To achieve this with domain and UI packages, I thought of "LoginHandler" and "BuildingHandler" classes that will handle the input. The UI package exists for the sole purpose of managing the design and listening to the events coming from the user. For this reason we decided to implement the GameFrame, GamePanel, LoginFrame etc. in this package.

I decided to use the "State" pattern to dynamically change surprising alien's behavior based on the player's progress in the game. State pattern allows surprising aliens to show different behaviors that are unique to different states. It also decreases the complexity of if or switch statements. To apply the state pattern, I created a class for each state of the alien: Confused, Challenging, Cooperative, Repairing, Protecting. These classes implement the"AlienState" interface and override the "perform" method of the "AlienState" interface to display different actions in different states. Surprising Alien's state is decided in the surprising alien's constructor based on the percentage of remaining asteroids.

I used the "Singleton" design pattern in order to prevent the creation of multiple instances of objects which need to be created only once. These objects work in a way that all the responsibilities of the object should be taken care of by the same instance. For example, only one instance of the "AsteroidFactory" object is needed, and whenever an asteroid creation is needed that instance of AsteroidFactory takes care of it. In order to implement the "Singleton" design pattern, I designed singleton classes in a way that when the constructor of the class is called it checks if it is already initialized. If it is already initialized, the constructor returns the previously created instance otherwise it creates a new instance and returns the new one.

I used the “Adapter” design pattern in order to let an interface of a class be used as an interface for another class. For example in our project there is “Save and Load” functionality. Users can decide on saving on a “TextFile” or on a “Database”. When saving is intended in “PausePanel” class “save()” method of the “LoadSaveHandler” is called with the intended saving instance (filesave or databasesave). “save()” method of LoadSaveHandler sets the adapter according to the given saving technique and calls the “save()” method of the adapter. In this way by using only one interface “Saver” I can use two different saving techniques. My Load functionality also works in the same way by using one interface “Loader” we can use two different loading techniques.

Supplementary Specification

Introduction
This document is the repository of all OuterSpace requirements not captured in the use cases.

Functionality
Functional requirements are examined in the use cases. To sum up, players must have an account to play the game. Players can choose to load an already existing game or build a new game. Players can hit aliens and use power-ups to destroy asteroids. Also, players can pause, resume, save and quit the game.

Security
Each time the system is opened user authentication is required.

Usability

Human Factors
The user will be playing the game. Therefore:

- All text must be easily readable.
- Avoid colors that are hard to distinguish for people with color blindness.

Quick, comprehensible, and error-free processing is important for the game since the user plays the game to have fun.

Reliability

Recoverability
If the game fails to perform some operation, it shows an error message suggesting restarting the system.

Performance
As mentioned under human factors, the user plays the game for entertainment and the user aims to win. Not letting the ball fall and being able to catch power-ups before they fell is crucial to win the game. System should respond to players actions as quickly as possible. Thus, we want a system that responds in less than 5 seconds 90% of the time and we do not want it to crash.

Supportability

Adaptability

Each operation of the game has unique business rules. Thus, at multiple defined points in the scenario, pluggable business rule will be enabled. Also, we aim for a game that can be played on different operating systems.

Configurability

Arrow-left and arrow-right buttons are used to move the paddle and A and D keys are used to rotate the paddle up to 45- and 135-degrees. Mouse left button or the letter W is required to make the first shot. Player can activate a power-up either by clicking on its icon or by typing the first letter of the power-up.

Implementation Constraints

OuterSpace will use Java and Java Swing technologies.

Purchased Components

There are no extra components to be purchased.

Free Open Source Components

In general, we recommend using free Java technology open source components to implement this game.

Interfaces

Game Window Interface

Game window have the following criteria:

- Paddle length L is 10% of the screen width.
- Paddle thickness T is 20px.
- Simple and gift asteroids are rectangles with dimensions L/5 and 20px.
- Firm asteroids are circular with radius 10px + (number of hits required to destroy*1px).
- Explosive asteroids are circular with radius 10px.
- Ball has dimensions of 17x17px.

Domain (Business) Rules

ID	Rule	Changeability	Source
RULE1	Paddle can be rotated up to 45- or 135-degrees with A and D keys respectively. Paddle can be moved horizontally with arrow-left and arrow-right. The paddle moves by an offset equal to the paddle's length divided by 2 and with a speed equal to the 2 times paddle's length/second	Low	Project Info
RULE2	The simple and gift asteroids may be moving back and forth with a probability of 0.1, or stiff with a probability of 0.9.	Low	Project Info
RULE3	After destroying an asteroid, the new score is calculated by the following formula: New score = Old Score + (300/ (Current Time – Game Starting Time))	Low	Project Info
RULE4	Minimum number of asteroids to build a new game: - 75 simple asteroids - 10 firm asteroids - 5 explosive asteroids - 10 gift asteroids · 6 gift asteroids that contains power-ups · 3 gift asteroids that sends signal to aliens · 1 gift asteroid that contains surprising alien	Low	Project Info

RULE5	Initially, ball will be fired vertically relative to the paddle. When the ball hits a non-moving object, it will be reflected with the same angle and velocity. When the ball hits an object that moves in the same direction with the ball's velocity, the non-moving object rule applies but with an 5px increase in the speed. When the ball hits an object that moves in a different direction will the ball's velocity, the ball with reflect with an angle of 180 degrees relative to the ball's initial movement direction, and the speed does not change. When the ball hits an object that moves perpendicular to the direction of the ball's velocity, the ball will reflect with an angle of 45 degrees relative to the line of moving object, and the speed does not change.	Low	Project Info
RULE6	When a gift asteroid that contains a power-up is destroyed, the power-up will fall with the speed: Speed = Paddle's Length/4*second	Low	Project Info
RULE7	Surprising Alien's state is decided based on the player's progress in the game: - Surprising alien behaves as cooperative if at least 70% of the asteroids are present. - Surprising alien behaves as challenging if less than 30% of the asteroids are present. - Surprising alien behaves as protecting if 40% to 50% of the asteroids are present. - Surprising alien behaves as repairing if 50% to 60% of the asteroids are present. - Surprising alien behaves as confused if 30% to 40% or 60% to 70% of the asteroids are present.	Low	Project Info

Glossary

Term	Definition and Information	Format	Validation Rules	Aliases
player	The person who plays the game.			
building mode	The mode that existing games can be loaded and other games can be created.			
running mode	The mode that enables the player has the control over the paddle to destroy the asteroids to play the game.			
help screen	The screen which explains the game objects, hand features and how to play.			
login screen	The screen which appears when the game executable is run. Players can login their information and access to the game if their login information is valid. After that they can load and save the game by this screen.			
pause screen	The screen which appears when the player pauses the game. It disappears when the game is resumed. It contains the saving options as well.			

ball	The object used to destroy asteroids.			
paddle	The tool used to prevent the ball from dropping to the ground.			
asteroid	A wall that aliens built which surrounds the earth and prohibits sunlight. They do not follow gravity and can stay in the air.			
simple asteroid	Can be broken in one hit. When broken, an asteroid just disappears.			
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. It gets smaller and eventually gets destroyed.			
explosive asteroid	These asteroids have explosive gas in their components. These asteroids have circular shapes and they explode once being hit. Once exploded, they destroy the neighbor asteroids with a radius of 2*L pixels.			

gift asteroid	These asteroids can be destroyed in one hit like simple ones. However, they are hiding inside power-ups or triggers for the aliens to start repairing or protecting the wall. If there is a power-up inside, the power-up will fall. If the player cannot catch it by hitting it with the paddle it will be lost. If she/he catches it, it can be used accordingly.			
score	It is calculated with a function: NewScore = OldScore + 300/(CurrentTime-GameStarting Time) where, CurrentTime is the time of the asteroid destruction. The score is initially zero if we are in a new game and the times are measured in seconds.			
alien	Enemies of the player that are trying to destroy life on earth by building a wall of asteroids that surrounds the earth and prohibits sunlight.			
repairing alien	These aliens can create only the simple asteroids.			
protecting alien	These aliens try to protect the wall by moving under the asteroids in a horizontal fashion. This type of alien is protected against the ball hits except the upper part of its body.			

cooperative alien	These aliens decided to help the player fulfilling the mission since they are against the idea of building the wall.			
time-wasting alien	These aliens try to waste the player's time and protect the wall.			
surprising alien	This alien acts like one of the other aliens when it appears, based on the number of the asteroids.			
power-ups	Special abilities that the player can use when he/she destroy a gift asteroid.			
taller paddle power-up	This doubles the length of the paddle, the effect of this power-up stays for 30 seconds. It might not be used directly, and it can be stored for the future.			
magnet power-up	This feature allows the paddle to catch the ball instead of reflecting it, and then throwing it using the mouse or W, this can help to direct the ball to hit more beneficial positions of the wall. This feature can also be kept and not used directly.			
destructive laser gun power-up	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.			

chance power-up	At the beginning of the game, the player has three lives. This power-up increases the player's chances(lives) by 1.			
wrap power-up	This power-up allows player's paddle to go through the side of the screen and re-appear on the opposite side. This upgrade lasts 2 minutes after it is activated.			
gang-of-balls power-up	Once a gift asteroid having gang-of-balls is hit, the hitting ball will turn into 10 replicas. In other words, 10 more balls will appear, and each will have the same size as the original ball.			