Bilkent University

Department of Computer Engineering

CS-319 Object Oriented Software Engineering Project
Sword & Shield: A Space Adventure

Analysis Report

Iteration II

Group Members:
Akın Berkay Bal
Eren Aslantürk
Mehmet Enes Keleş
Sadık Said Kasap

Instructor: Uğur Doğrusöz

TA: Hasan Balcı

Analysis Report

1. Introduction

We decided to design and implement a game called Sword and Shield. It is a two dimensional top-down strategy/tower defense game inspired from Bloons: Tower Defense. The game will be playable by two players: the attacker and the defender. The map will consist of a base for attack units to spawn, a lane for them to walk and a lot of space for defense units to be located. There will be offensive and defensive units to be used during two different stages of the game. At the first stage, defensive player will buy and put his desired items on the map. During the second stage, attacker player will try to breach through the end of the map by spawning attack units in a limited time.

This report includes the overview of the game, describes how the game will be played, it will give a brief look on the gameplay and rules of the game. It also contains functional, non-functional and pseudo requirements, use-case models with scenarios, use-case diagrams, dynamic models, object and class model and a mockup of Graphical User Interface.

2. Game Overview

Sword and Shield will contain three turns for every game. Turns will have different maps to reduce repetitiveness and hinder the ability of one player to get too comfortable. Turns will have a time limit of 3 minutes. There will be an attacker and a defender in the game. They will start the game with a predetermined amount of coins and they will try to set up their base with the given budget and they will earn some coins throughout the game. Defender earns coins by destroying attacker's ships. Attacker will earn coins by passing the defense line and destroying some health of the defender and through reactors which earns money to player. Coins won't get resetted every turn. Coins can be used to upgrade/buy turrets, factories or reactors.. Also, coins can be used to buy additional tiles to build more of turrets or factories.

Players can pause the game anytime they desire. If we can we want to include a save and a load function but it is not in priority list for the time.

On the top part of the GUI, there will be a timer, scores and money for each player so that they can see who is winning and how much money they have everytime they check. On the right side, defender will be able to see the turrets that can be bought and these turrets can be upgraded by clicking on them.

The first Defender will get their defense ready by using the mouse and after that, attacker will be available to build factories/reactors for attack. Timer will start after attacker builds the first factory/reactor. All turrets, factories and reactors have a maximum level of three.

2.1 Defense Buildings

Buildings that will be used only by the defender.

2.1.1 Turrets

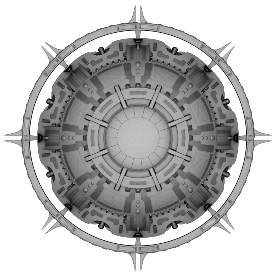
There will be four kinds of turrets with different properties to let player decide the tactic.

Turrets' properties are the following:

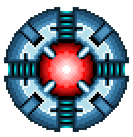
- Cost
- Dimensions
- Cost of Leveling Up
- Attack Speed
- Range
- Armor Penetration

Miner

Miner will produce mines which will be placed in front of the miner. It can produce at most 3 mines until at least one of them get destroyed. They will be placed at first exactly in front, then one to the left tile of the first, then the last one will be on the right tile of the first one. Mines will have area damage effect. To balance it, it will be produced in a low rate. It does not have armor penetration. Moreover it gives less damage to armored units.



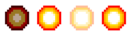
Miner



Mine

Lazerion

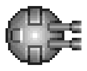
Lazerion will be the fast shooter turret. Therefore, it will have the least range but second powerful projectiles for balancing. It does not have armor penetration.



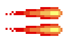
Lazerion Projectile

Good Ol' Gun

Good Ol' Gun will be a machine gun. As technology is kinda old with this one, it will have the least damage, but an average range and a good rate of firing. No armor penetration with this gun.



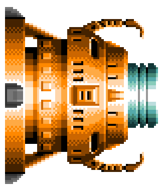
Good Ol' Gun



Good Ol' Gun Projectile

Fryinator

Fryinator is the most powerful weapon for defender. It will have the most damage and most range. But as expected its rate of fire is low, it takes up two tiles and it is the most expensive one of the turrets. It has armor penetration.



Fryinator

Fryinator Projectile

2.2 Attack Buildings

Buildings that will be used solely by the attacker.

2.2.1 Reactor

Reactor can be bought by the attacker to earn money throughout the stage. It is in the shape of a nuclear reactor for the game's concept. It will earn the attacker a certain amount of coins throughout the game at a predetermined rate that can be upgraded.

2.2.2 Factories

There will be four kind of factories which produces attack units with different properties to let player decide the tactic.

Factories' properties are the following:

- Cost
- Production Rate
- Dimensions
- Cost of Leveling Up

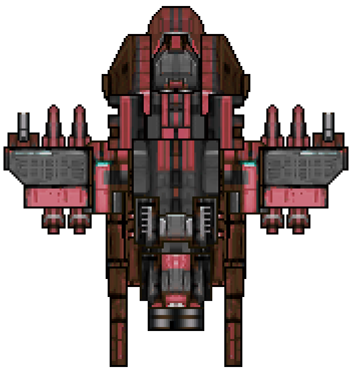
Units produced by this factories have these properties:

- Speed
- Armor
- Health

Muscular Spaceship Co.

As the name suggests this factory will produce heavy and powerful units. Therefore, it has a low production rate.

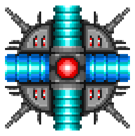
Units will be moderate in speed but will have heavy armor and high health.



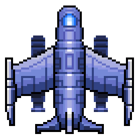
Muscular Spaceship

Athletic Spaceship Co.

This factory will produce units which will be fast, have light armor and average health at average production rate.



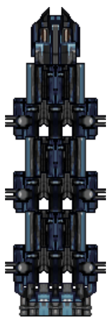
Athletic Spaceship Co.



The Athletic Spaceship

Swedish Spaceship Co.

If you want very fast things which will divide in three when first one is destroyed Swedish is the way to go. This units won't have any armor and they will have low health, but as expected they will be very fast. When the first one is destroyed it will spawn three more. Production rate is low.

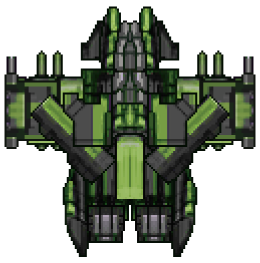


Swedish Spaceship

Swedish Spaceships After the Division

Tough Tough Co.

This is the most expensive factory, it will take 2 tiles and will have a low production rate. It will produce units with heavy armor and heavy health which can go at average speed.



Tough Tough Spaceship

3. Requirements

3.1 Functional Requirements

- User will be able to place buildings with mouse.
- User will be able turn on and off the game music.
- User will able to access the help menu. Help menu contains information about game mechanics and game elements such as building and attack units.
- User will able to pause the game.
- User will able to reach contact developers of the game.

3.2 Non-Functional Requirements

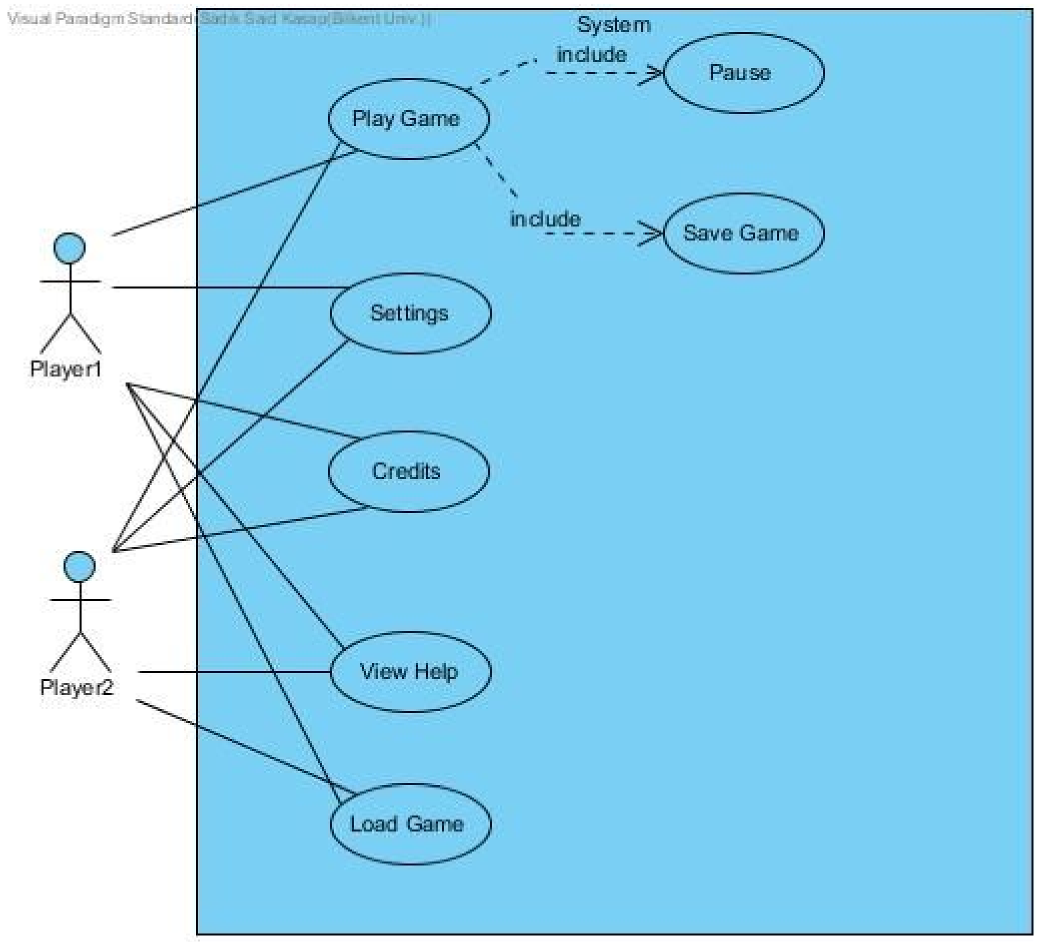
- Game graphics and user interface will be designed in a user-friendly manner. The colors will be designed so as not to disturb the user.
- Control mechanisms of the game will have low response time, which enables users to play with minimal delay.

3.3 Pseudo Functional Requirements

- Game will be implemented using Java Swing libraries.

4. Models

4.1 Use Case Models



use case diagram

Use Case#1

Use Case Name: Play Game

Participating Actors: Player1, Player2

Entry Condition: Player opened the game and click the play game in main menu

Exit Condition:

- Players have chosen to close the game, OR
- Players have chosen to return to main menu via in-game menu, OR
- Player can exit the game.

Main Flow of Events:

1. Players start the game.
2. System constructs the map.
3. System started the game in the defense stage.
4. Player2 locates the defense turrets.
5. Player2 clicks to Turn button.
6. System start the attack stage.
7. Player1 places the attack units.
8. Player1 destroys the defense units and wins the turn.
9. Players return the main menu.

Alternative Flow of Events:

- At any time player can exit the game.(go to step 9)
- At any time player can save the game.(go to step 9)
- Player1 did not destroyed the defense units and Player2 won the turn.(go to step 9)
- Players continue the game after first turn. (go to step 3)

Use Case#2

Use Case Name: Pause Game

Participating Actors: Player

Entry Condition: Player is already playing the game

Exit Condition:

- Player return main menu, OR
- Player continues the to play the game, OR
- Player can exit the game.

Main Flow of Events:

1. Player presses the pause button during the game.
2. The system displays the in-game menu.
3. Player continues to play the game.

Alternative Flow of Events:

- Player returns to main menu.

Use Case#3

Use Case Name: Save Game

Participating Actors: Player

Entry Condition: Player is already playing the game

Exit Condition:

- Player return main menu, OR
- Player continues the to play the game, OR
- Player can exit the game.

Main Flow of Events:

1. Player presses the pause button during the game.
2. The system displays the in-game menu.
3. Player select the save button
4. Player return main menu the game.

Alternative Flow of Events:

- Player exit the game.
- Player continue to play the game.

Use Case #4

Use Case Name: View Help

Participating Actors: Player

Entry Condition: Player opened the game

Exit Condition:

- Player return main menu, OR
- Player can exit the game.

Main Flow of Events:

1. Player click to help button.
2. The system displays the help page.
3. Player select the game object.
4. System show the specialities of gameobject.
5. Player returns to the main menu.

Use Case #5

Use Case Name: Change Settings

Participating Actors: Player

Entry Condition: Player opened the game

Exit Condition:

- Player returns to the main menu, OR
- Player can exit the game.

Main Flow of Events:

1. Player clicks to the settings button.
2. The system displays settings page.
3. Player chooses to turn off the sound
4. Player returns to the main menu.

Alternative Flow of Events:

- Player chooses to not to change the settings(go step 4)
- Player chooses to turn on the sound(go step 4)

Use Case #6

Use Case Name: View Credits

Participating Actors: Player

Entry Condition: Player opened the game

Exit Condition:

- Player returns to the main menu, OR
- Player can exit the game.

Main Flow of Events:

1. Player clicks to the credits button.
2. The system displays the help page
3. Player returns to the main menu.

Use Case #7

Use Case Name: Load Game

Participating Actors: Player

Entry Condition: Player opened the game

Exit Condition:

- Player returns to the main menu, OR
- Player can exit the game.

Main Flow of Events:

1. Player clicks to the load button.
2. The system displays the load page
3. Player select one of the saved game.
4. System open the saved game.

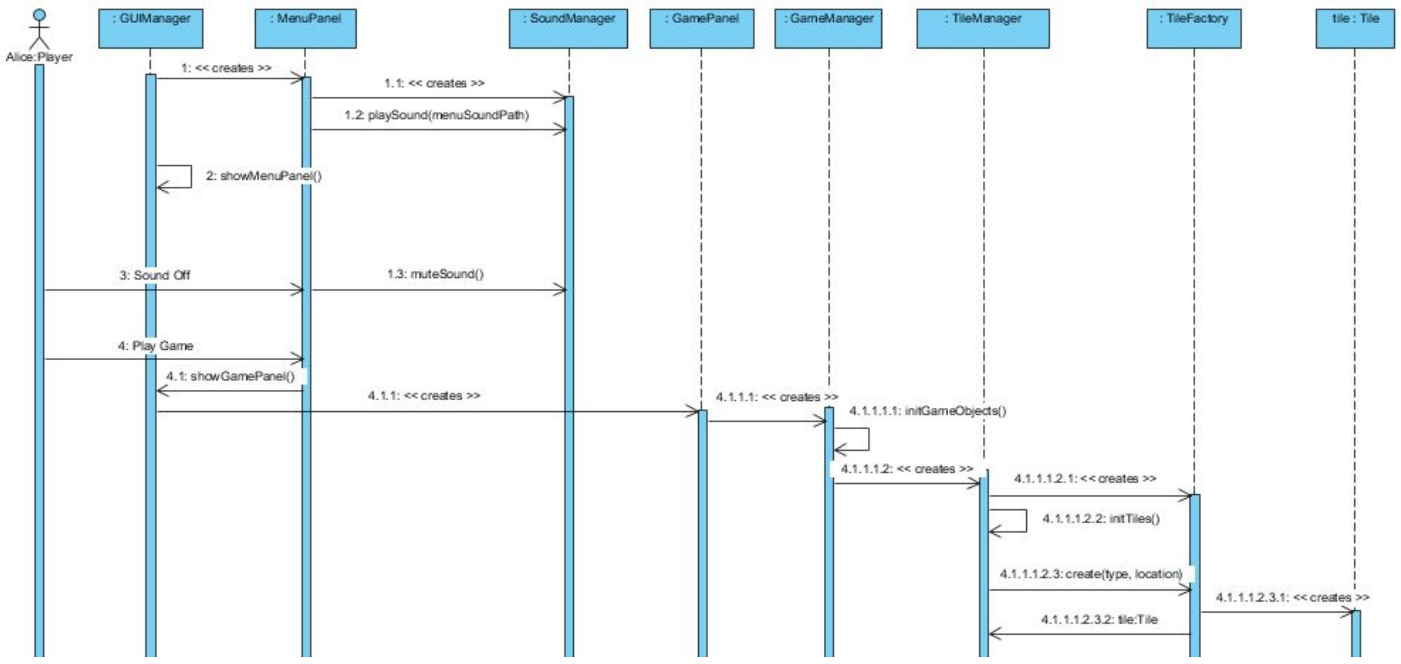
Alternative Flow of Events:

- Player return main menu.

4.2 Dynamic Models

Scenario I: Start the Game

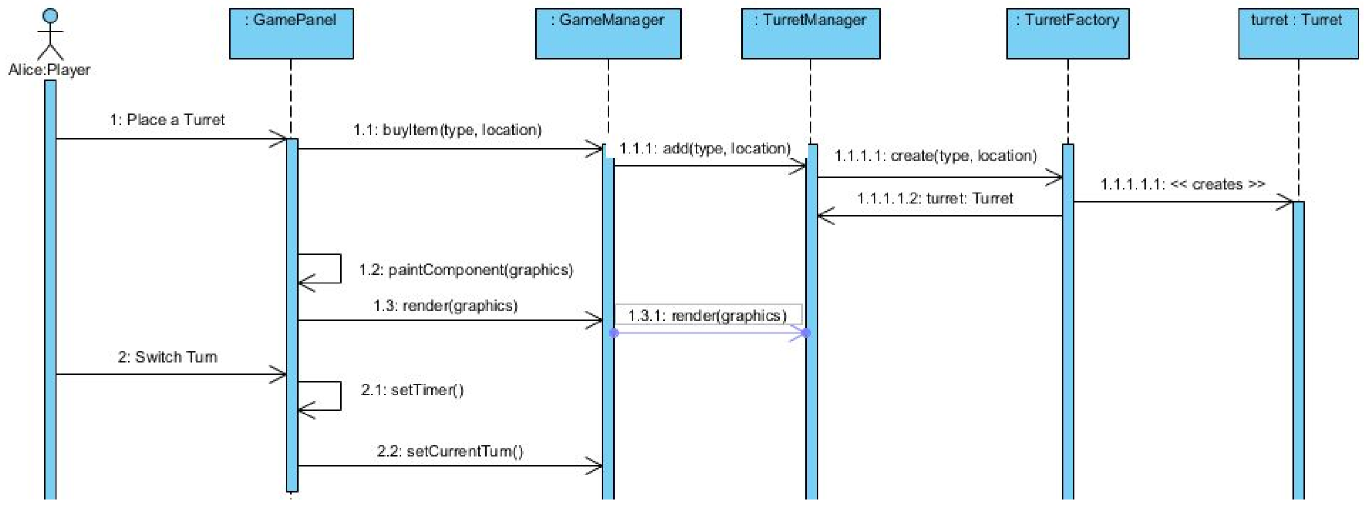
Plot: Alice doesn't know anything about the game. She happens to click to the game icon on the desktop. The menu appears on the screen. Alice gets tired of the music, clicks to "Sounds Off" button. After that, she starts a new game by clicking to "Play" button to see what the actual game looks like.



Description: When the game is opened, GUIManager is created automatically. GUIManager creates MenuPanel. MenuPanel creates SoundManager and SoundManager starts playing the music. When the player turns off the music muteSound() method of SoundManager is called. The player starts the game and GamePanel shows up. GamePanel initializes GameManager which holds the objects and the state of the game. initGameObjects() method is called for that purpose. This method creates tiles which will show up at the beginning of the game. TileManager is created and its add() method is called for adding the tiles. But TileManager needs a class that will create tiles. TileFactory's create() method creates a tile at the given location and returns it.

Scenario II: Defender Places Objects

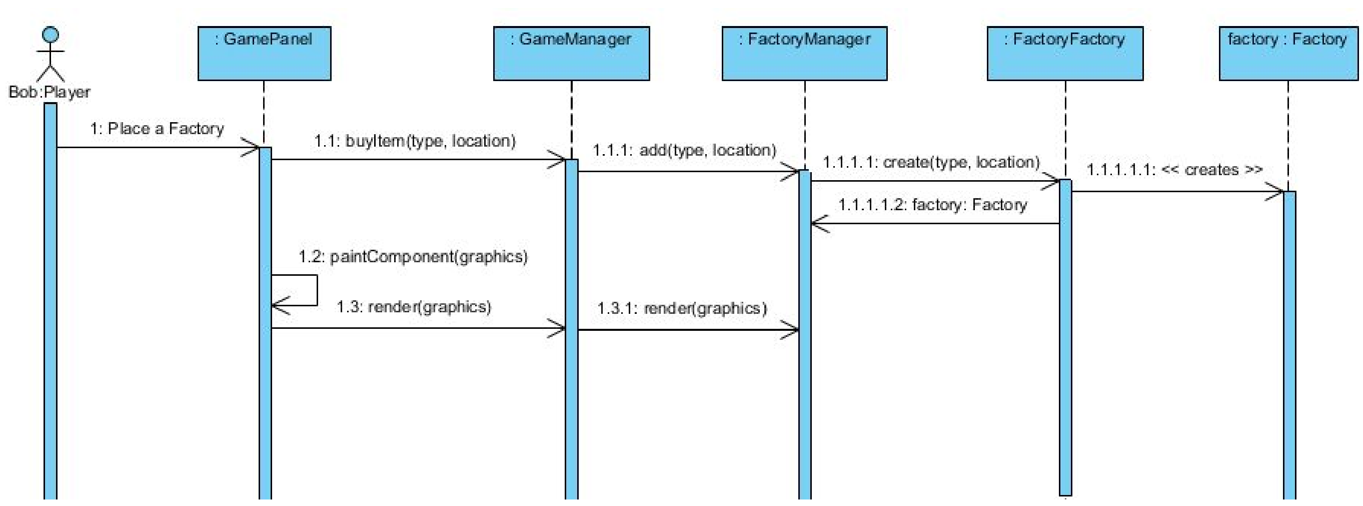
Plot: Alice starts a new game. She sees that the game is intended for two players. She invites her friend Bob to play with. At the beginning she has some gold to buy and locate the turrets from the GamePanel. So she does. Afterwards she clicks to "Turn" button in order to end the defense stage of the game.



Description: Alice puts her desired turrets on the map using the mouse. buyItem method of GameManager is called for this purpose. GameManager uses TurretManager which manages all the turrets on the map. TurretManager needs TurretFactory in order to create a turret instance. create() method of TurretFactory creates a turret and returns it. After that the timer tick comes and all the objects are drawn on the screen. When Alice clicks to turn button, the timer gets set for Bob's turn and turn value in GameManager also changed. Note that setCurrentTurn() method doesn't need a parameter because there are only two players.

Scenario III: Attacker Plays

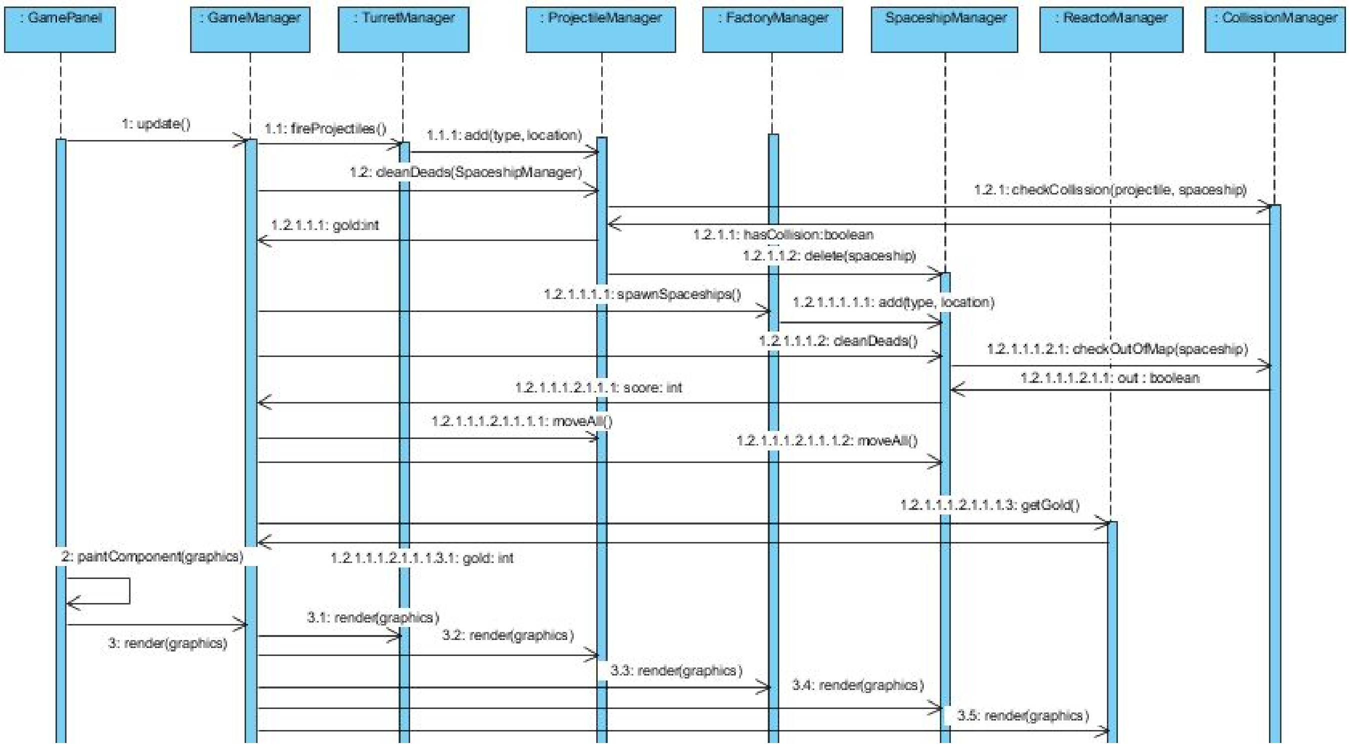
Plot: Bob needs to start putting factories immediately in order to pass through the defense Alice created in the previous turn. However his gold and time is limited. He selects these units from the GamePanel and puts them on the map.



Description: This diagram is quite similar to the previous diagram but this one is intended for the attacker player, Bob. When Bob puts a factory on the map, buyItem() method of GameManager is called. As in the previous diagram a Manager and a Factory class is used for adding the object to the game state.

Scenario IV: An Iteration in the Game

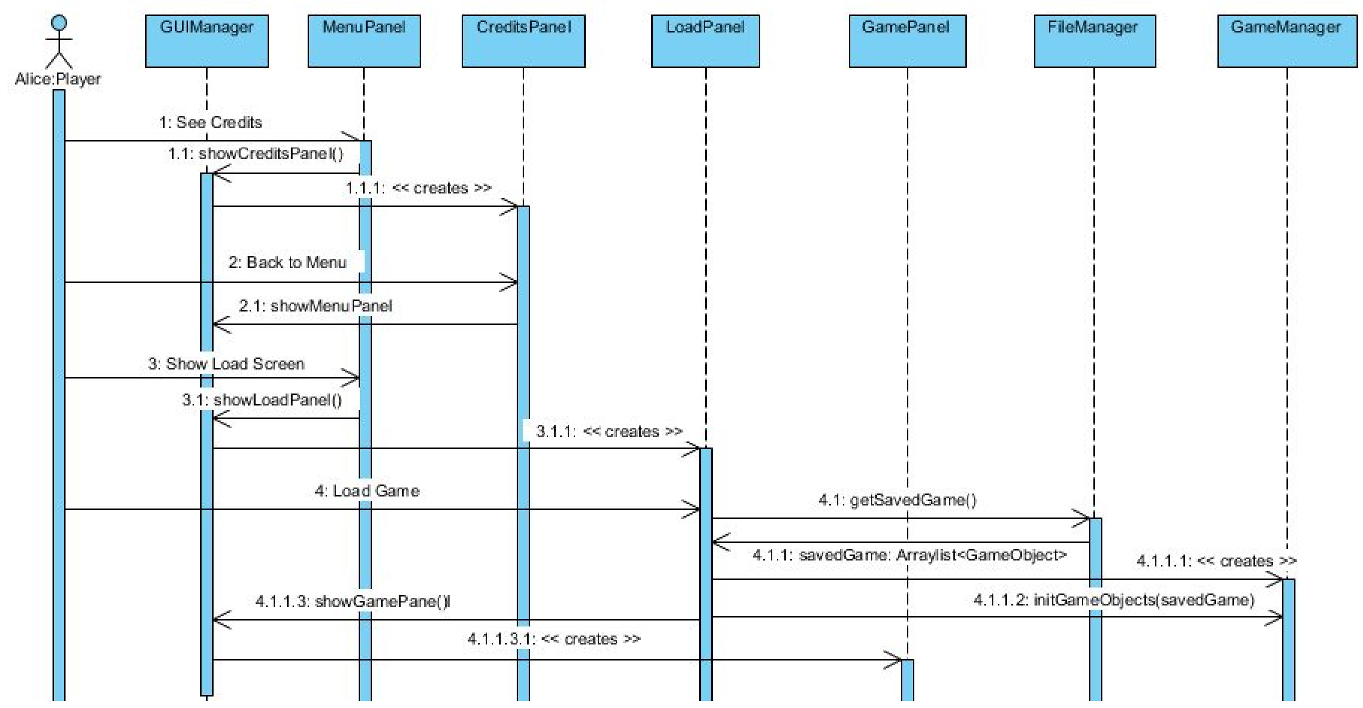
<u>Plot:</u> The timer ticks and an iteration in the game happens. These flow of events are repeated every 1/60 seconds.


: GamePanel
: GameManager
: TurretManager
: ProjectileManager
: FactoryManager
SpaceshipManager
: ReactorManager
: CollissionManager
1: update()
1.1: fireProjectiles()
1.1.1: add(type, location)
1.2: cleanDeads(SpaceshipManager)
1.2.1: checkCollission(projectile, spaceship)
1.2.1.1: hasCollision:boolean
1.2.1.1.1: gold:int
1.2.1.1.2: delete(spaceship)
1.2.1.1.1.1: spawnSpaceships()
1.2.1.1.1.1.1: add(type, location)
1.2.1.1.1.2: cleanDeads()
1.2.1.1.1.2.1: checkOutOfMap(spaceship)
1.2.1.1.1.2.1.1: out : boolean
1.2.1.1.1.2.1.1.1: score: int
1.2.1.1.1.2.1.1.1.1: moveAll()
1.2.1.1.1.2.1.1.1.2: moveAll()
1.2.1.1.1.2.1.1.1.3: getGold()
2: paintComponent(graphics)
1.2.1.1.1.2.1.1.1.3.1: gold: int
3.1: render(graphics)
3: render(graphics)
3.2: render(graphics)
3.3: render(graphics)
3.4: render(graphics)
3.5: render(graphics)

<u>Description:</u> To get the game going, update() and render() methods are called sequentially every 1/60 seconds by the timer. update() method in GameManager calls required methods from object manager classes such as fireProjectiles() from TurretManager, spawnSpaceships() from FactoryManager etc. All these methods needs to be called in the right order because undesired behaviors might happen otherwise. For instance we wouldnt't want spaceships to get hit by projectiles that are out of range. Note that CollissionManager is used for determining dead and damaged spaceships and out of range projectiles.

<u>Scenario V: Load Game</u>

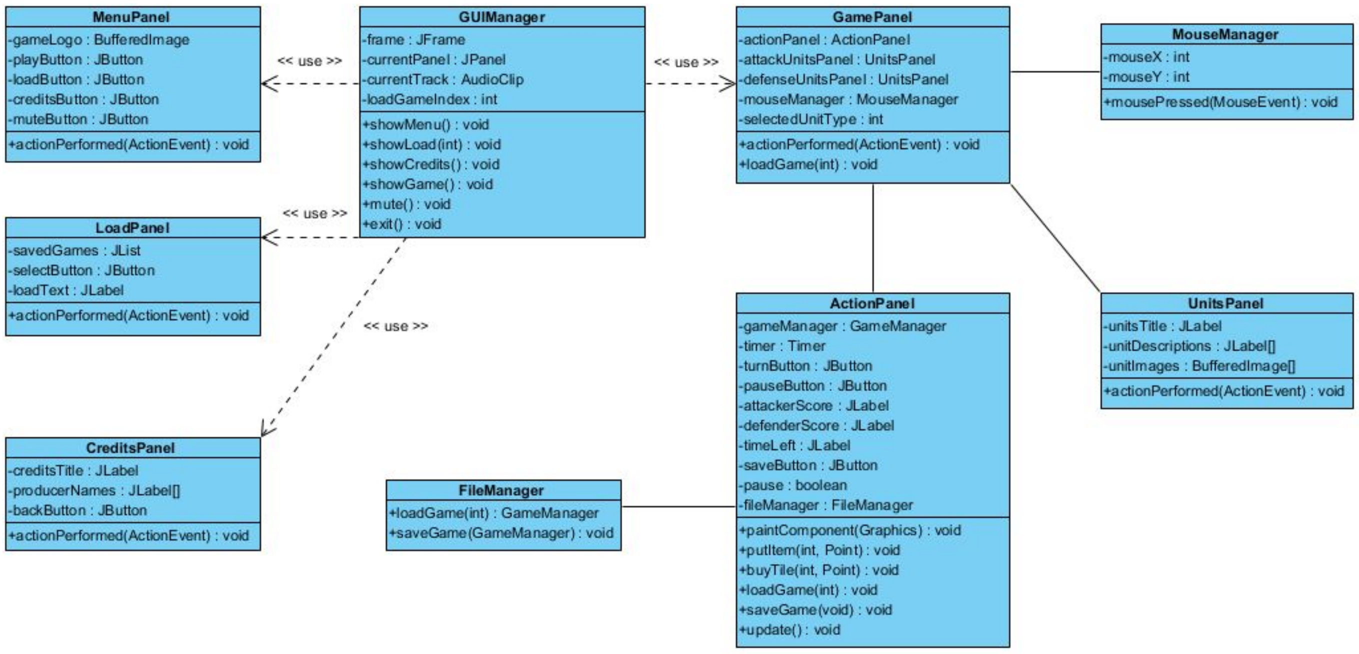
Plot: Alice reopens the game. Clicks to "credits" button. Turns back to the menu and clicks to "load" button. Loads a random saved game.



Description: showCreditsPanel() of the GUIManager creates the CreditsPanel instance and its shown on the screen. Upon clicking to 'back' button, showMenuPanel() method puts the MenuPanel on the frame. showLoadScreen() method creates and puts on LoadPanel when Alice clicks to 'load game' button on the menu. LoadPanel calls FileManagers getSavedGame() method to after Alice clicks to a saved game from the list. The GamePanel instance is created according to the GameObject arraylist that is returned as the result. GUIManager's showGamePanel puts GamePanel on the screen. Note that the GamePanel is a singleton object therefore it will not be recreated by the GUIManager.

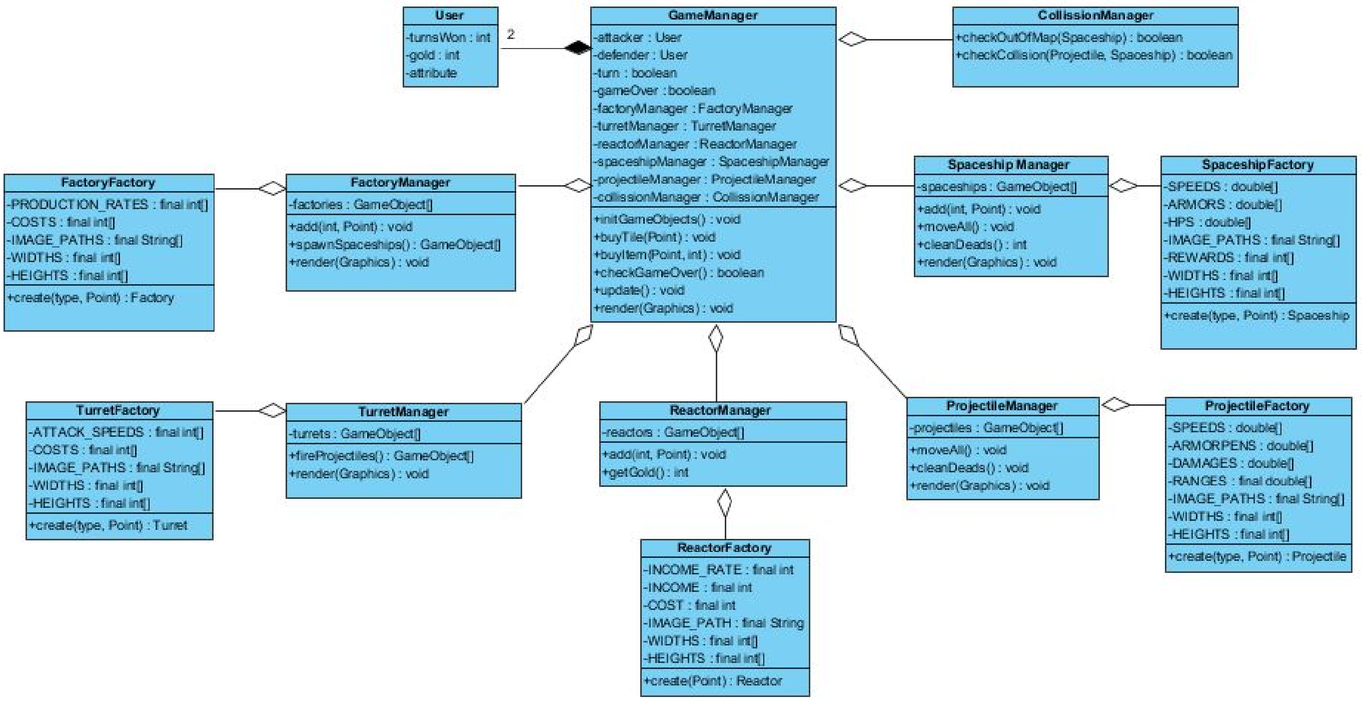
4.3 Object and Class Model

GUI Classes



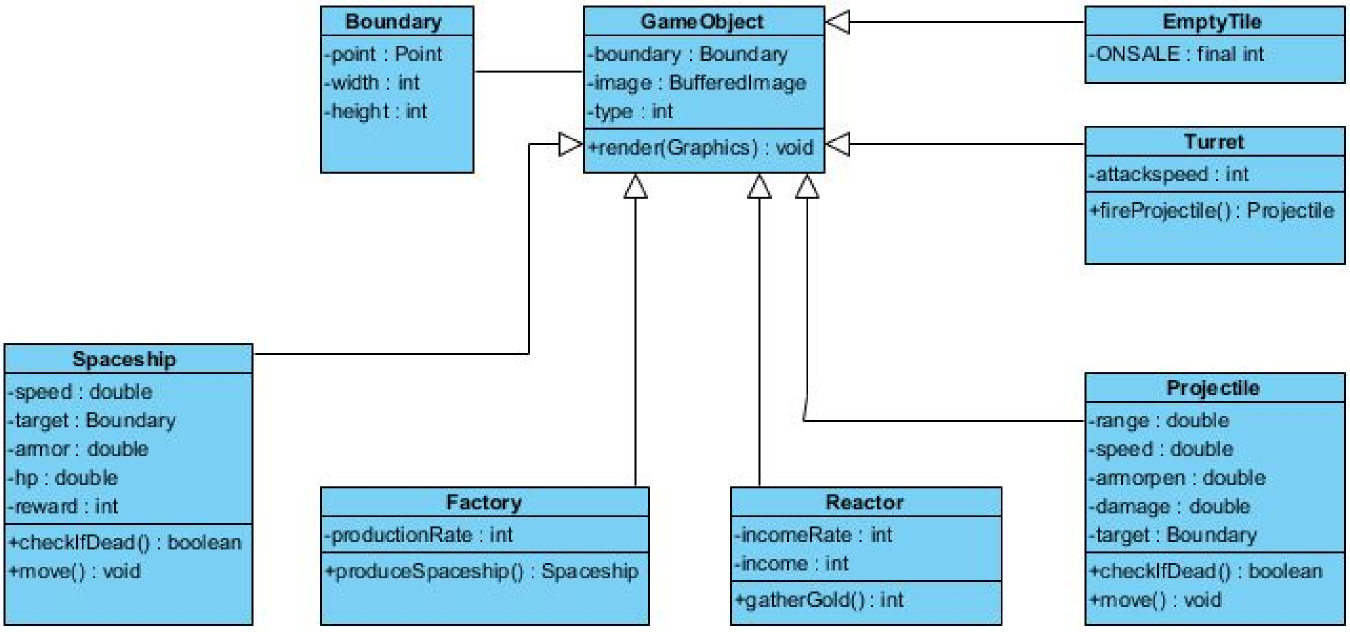
These are the boundary objects that the user will interact with. GUIManager is a Facade class that manages what's on the screen. The other classes are differents screens of the game.

Logic Classes



Above is the basic outline of logic classes. GameManager holds object manager classes such as TurretManager, ProjectileManager etc which carry their own type of entities. Each object manager class also an hold instance corresponding object factory class in order to create new instances of game entities. GameManager like a master, controls all manager classes and keeps the game going.

Entity Classes



These are the different types of objects in our game. They all extend GameObject class. We hope the inheritance will be useful while keeping different types of objects in arrays and applying generic methods to them.

4.4 GUI

In our project, the interface covers a few sections: Main Menu, Game Interface, Credits, Load and Help. Here, we aim that the game should be visually attractive for the user, contents should be easily accessible for the best game experience and user-friendly to enjoy the game with a delightful in game progress.

Main Menu interface, we have several buttons to interact: New Game, Sound On/Off, Credits, Load and Help. New Game's main purpose is to lead the user to Game Interface.
From here, the user can advance through the screen to play the game. Sound On/Off button is an available feature for the user to enhance the game excitement with supportive background sound effects and musics. The design mock-up of Main Menu interface is shown below.

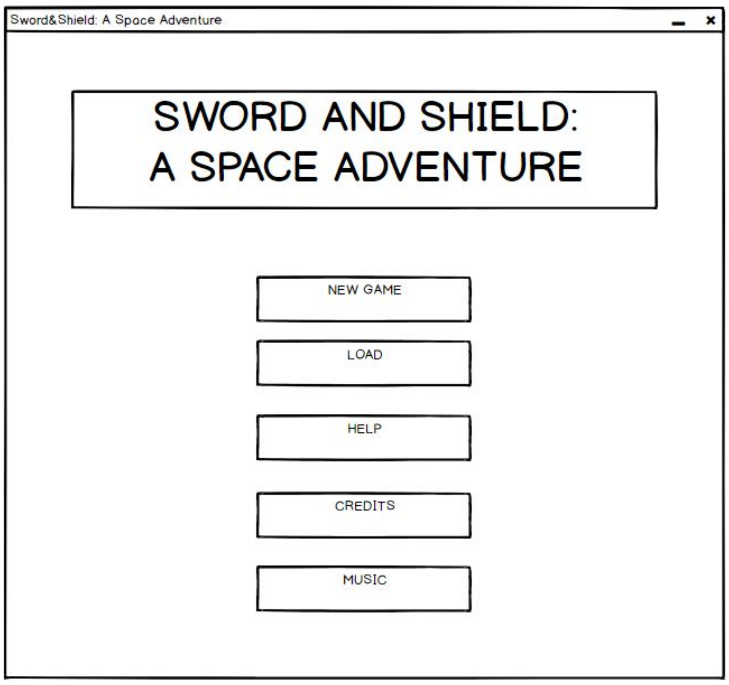
Game Interface is the major frame where the game occurs. When the user comes here, as we will give information about game rules, he/she will use the play/pause button that we are planning to include and the game will progress. The panels on left and right sides of the screen will be for in-game use for the both of player1 and player2. There will also be additional informative indicators on the screen such as timer and player currency. The design mock-up of the game is as below.

Credits aims to inform the players about the game content, details, contributors, resources and website links. Therefor players who are interested in for future games of our production, they will be able to follow our developer and production team.

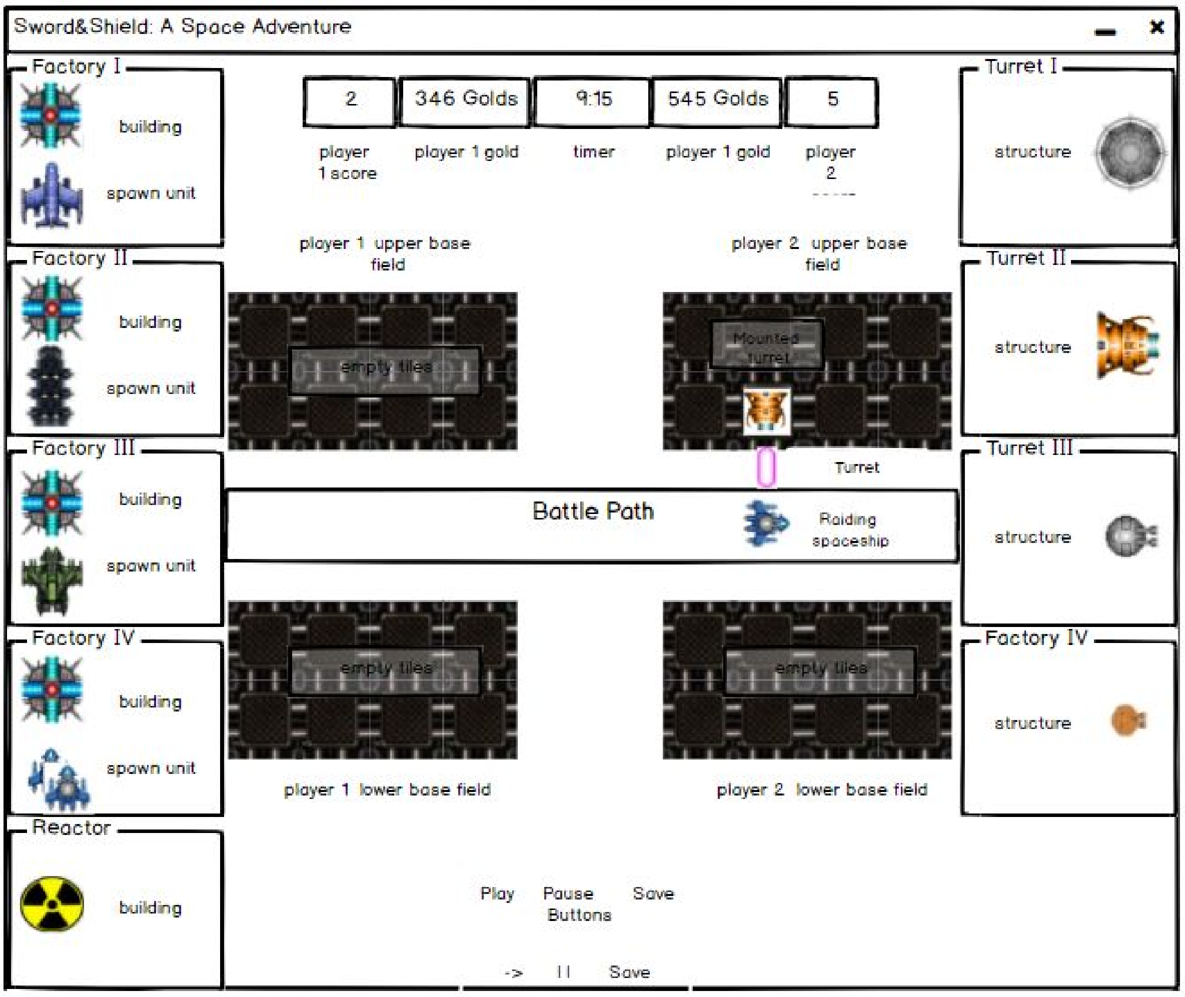
Help menu shows the items that are used during game and how to use them. It also includes images and details about the items. They can be checked by surfing through tabs in the frame.

Load menu includes images and information of saved games and they are accessible by pushing saved game image.

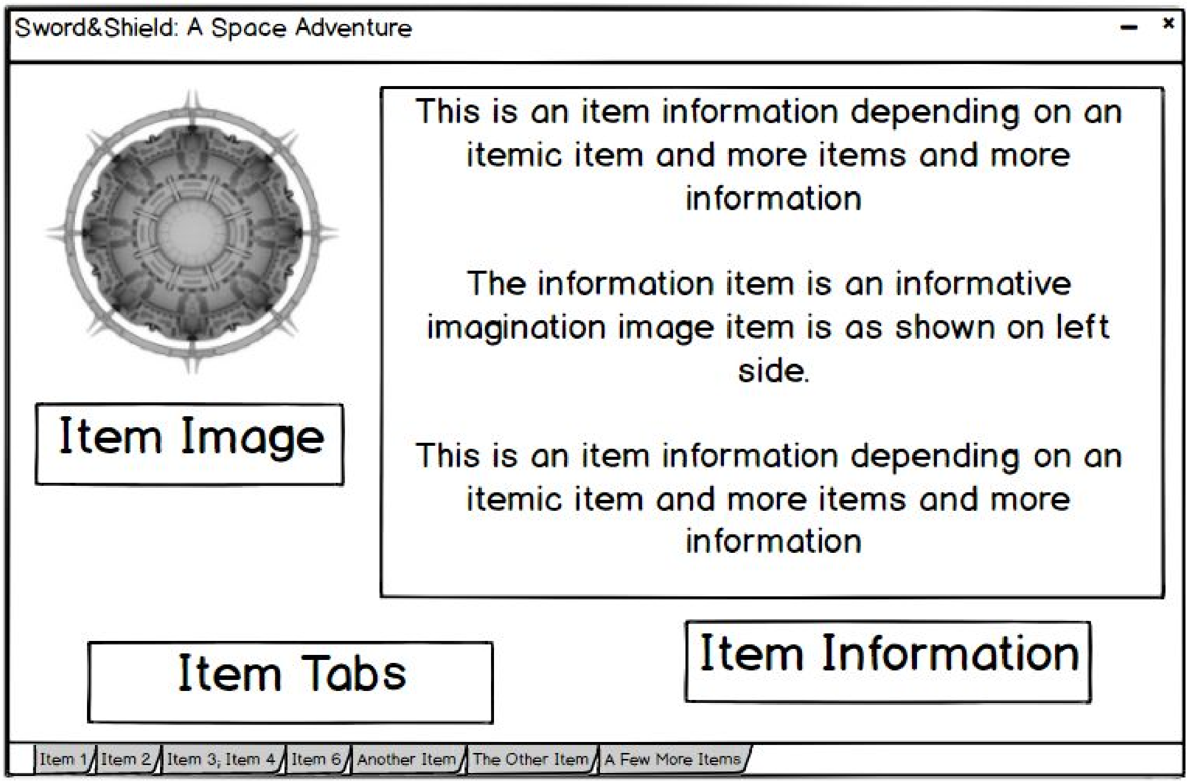
Main Menu Screen


Sword&Shield: A Space Adventure
SWORD AND SHIELD: A SPACE ADVENTURE
NEW GAME
LOAD
HELP
CREDITS
MUSIC

Game Interface Screen



Help Screen

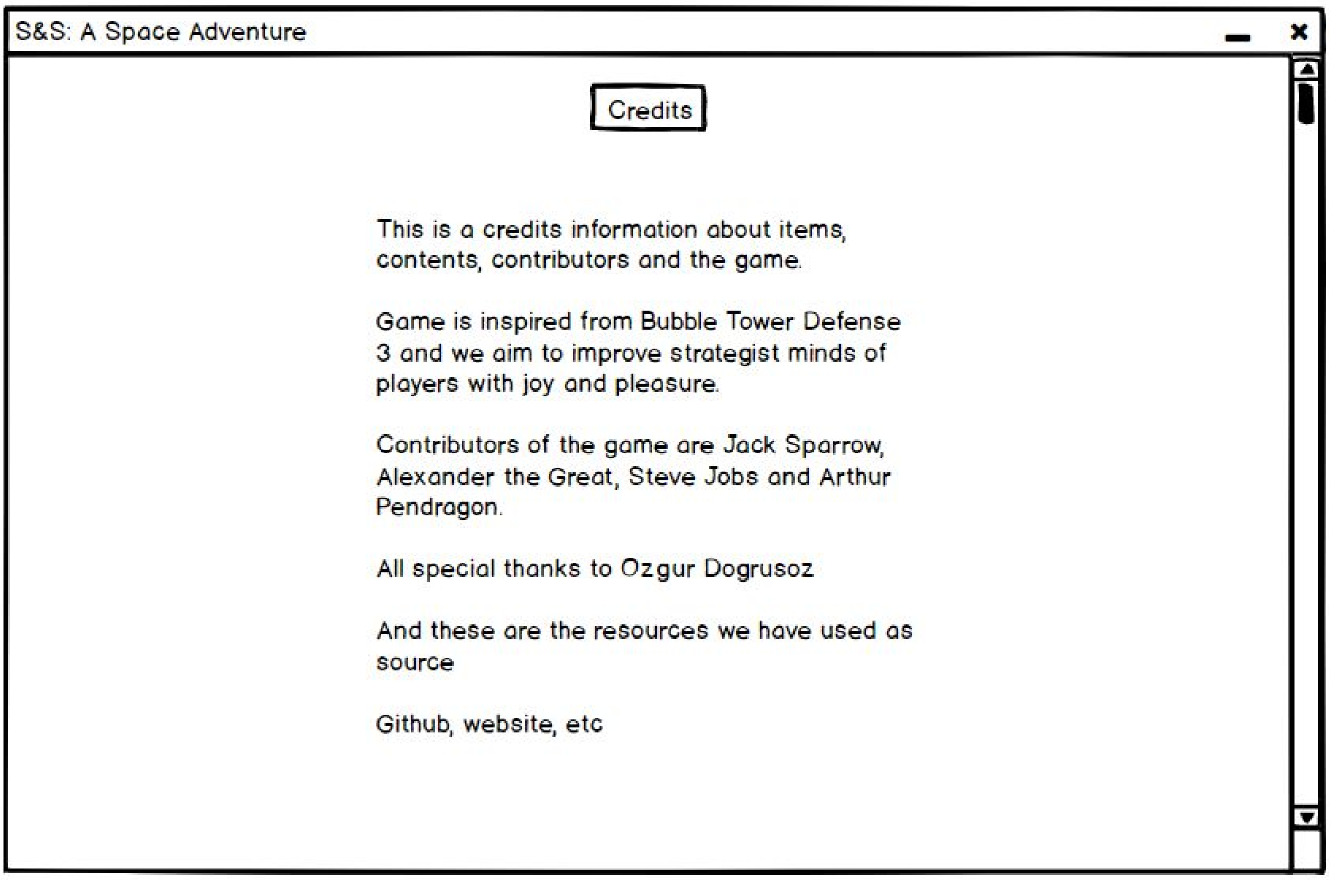

Sword&Shield: A Space Adventure

This is an item information depending on an itemic item and more items and more information

The information item is an informative imagination image item is as shown on left side.

This is an item information depending on an itemic item and more items and more information

Credits Screen


S&S: A Space Adventure

Credits

This is a credits information about items, contents, contributors and the game.

Game is inspired from Bubble Tower Defense 3 and we aim to improve strategist minds of players with joy and pleasure.

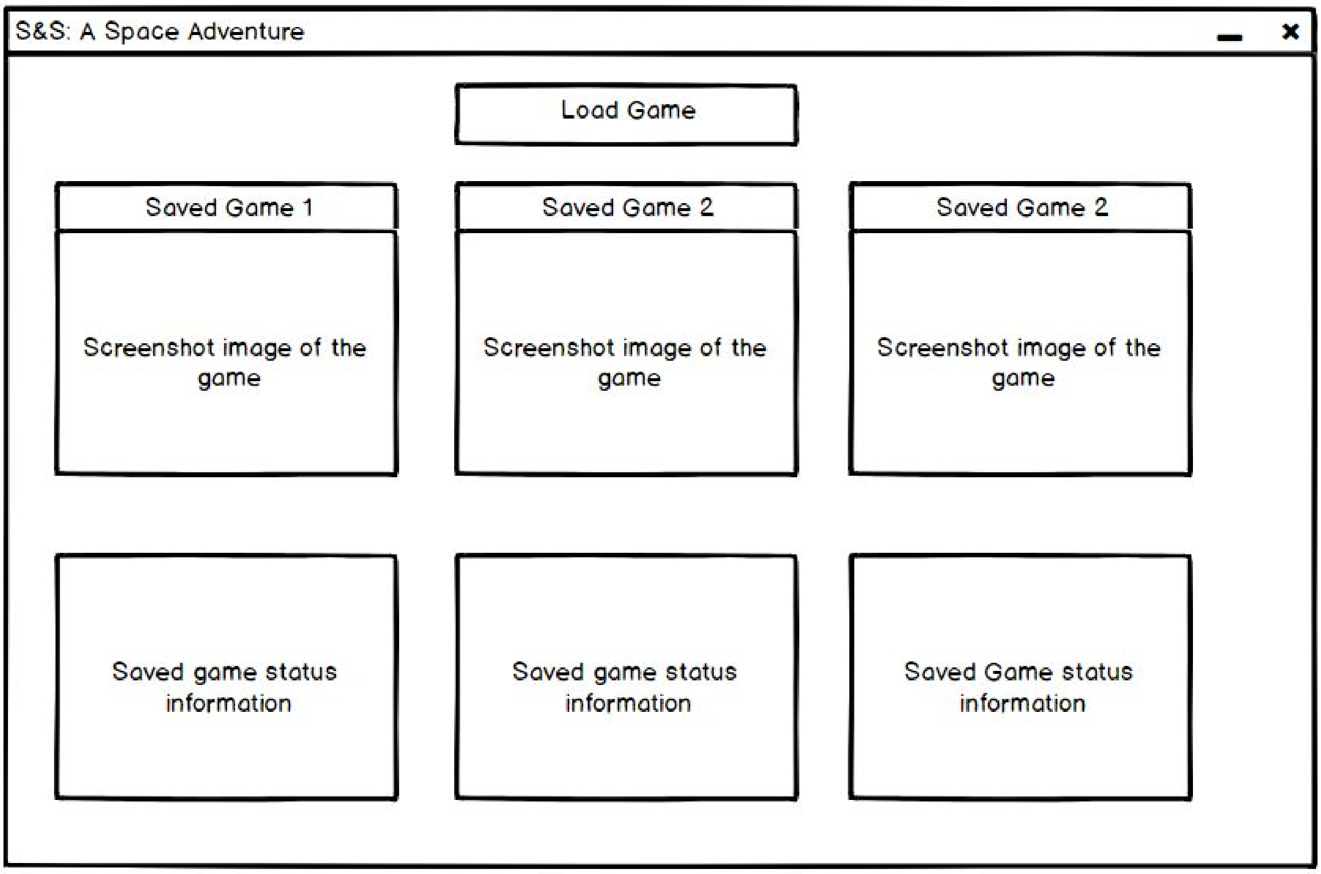
Contributors of the game are Jack Sparrow, Alexander the Great, Steve Jobs and Arthur Pendragon.

All special thanks to Ozgur Dogrusoz

And these are the resources we have used as source

Github, website, etc

Load Game Screen


S&S: A Space Adventure
Load Game
Saved Game 1
Saved Game 2
Saved Game 2
Screenshot image of the game
Screenshot image of the game
Screenshot image of the game
Saved game status information
Saved game status information
Saved Game status information

5. References

inspired from:

https://ninjakiwi.com/Games/Tower-Defense/Play/Bloons-Tower-Defense-5.html#.WoiAgXYUpEY

assets are taken from:
https://opengameart.org/content/pack-spaceships-and-building
https://opengameart.org/content/research-building-for-mobile-strategy-games
https://opengameart.org/content/starbase
https://opengameart.org/content/towers-for-tower-defense
https://opengameart.org/content/sci-fi-shoot-em-up-object-images
https://opengameart.org/content/space-ship-mech-construction-kit-2
https://opengameart.org/content/space-ship-construction-kit
https://opengameart.org/content/space-ship-building-bits-volume-1
http://kenney.nl/assets/tower-defense-top-down
http://hirefreelanceartist.com/free-tower-defense-graphics.html
https://t-allen-studios.itch.io/turret-pack-01
http://ask4asset.com/construct-2-capx-google-play-high-score-and-achievements/