Requirements Analysis Document

Group 2

Space Game

Furkan Hüseyin
Berfu Anıl
Islomiddin Sodiqov

Requirements Analysis Document

1. Introduction

1.1 Purpose of the system

The purpose of the system is to offer a game to a range of players to play a space game with an included scenario. Players will play the space game, kill the enemies and collect credit and level up to kill all the enemies on the map. The game will end when the player kills all the enemies.

1.2 Objectives of the project

Richard Bartle suggests that there are four type of players in general which are killers, achievers, socializers and explorers. Killers' and achievers' needs can be satisfied with acting feature of the game. Socializers' and explorers' needs can be satisfied with interacting feature of the game. (*See Figure 1*). This theory is called as Bartle's Taxonomy.

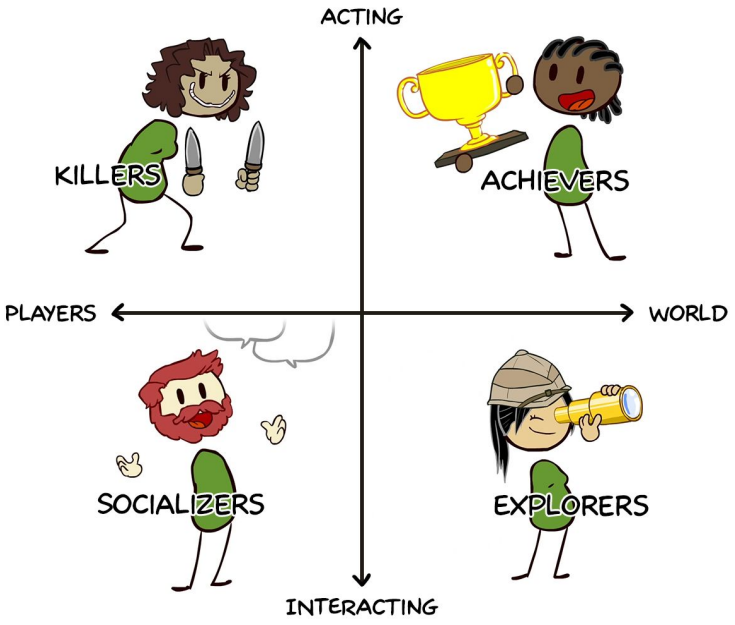


Figure 1

Purpose of the system is to satisfy the needs of achievers' and killers' needs. Since the game is not online, the socializers' needs will not be satisfied very well. On the other hand explorers' needs will be poorly satisfied by the game since the map will not contain large amount of easter eggs and surprises.

B. F. Skinner proposed an idea by illustrating it with a box with a button and a pigeon in it which is called *"Skinner Box"*. Skinner Box is a box where there is a button in it which gives a reward which might be food for a pigeon. If one puts pigeon

into the box and set the button to give a reward every time it is pushed, Skinner suggests that pigeon will gradually lose the attention to the button. However, if the button is set to give the reward at a random pattern which means giving the reward sometimes and not giving it other times even the pigeon pushed the button. Skinner suggests that the pigeon will keep its attention since the button acts unexpectedly.

This Skinner Box structure will be implemented while making reward mechanisms in the game. Gift boxes will not be giving the same amount of credit to keep the attention of the player.

There is also another theory in game design which is called *"Flow Channel"*. (*See Figure 2*). This theory suggests that if the game punishingly challenges the player it will exhaust the player and if the game is not challenging the player or giving too much skills that challenges will not be a deal to the player throughout the game, the game will bore the player.

The game will be designed to keep the player at the Flow Channel which is not too hard or too easy which means playable.

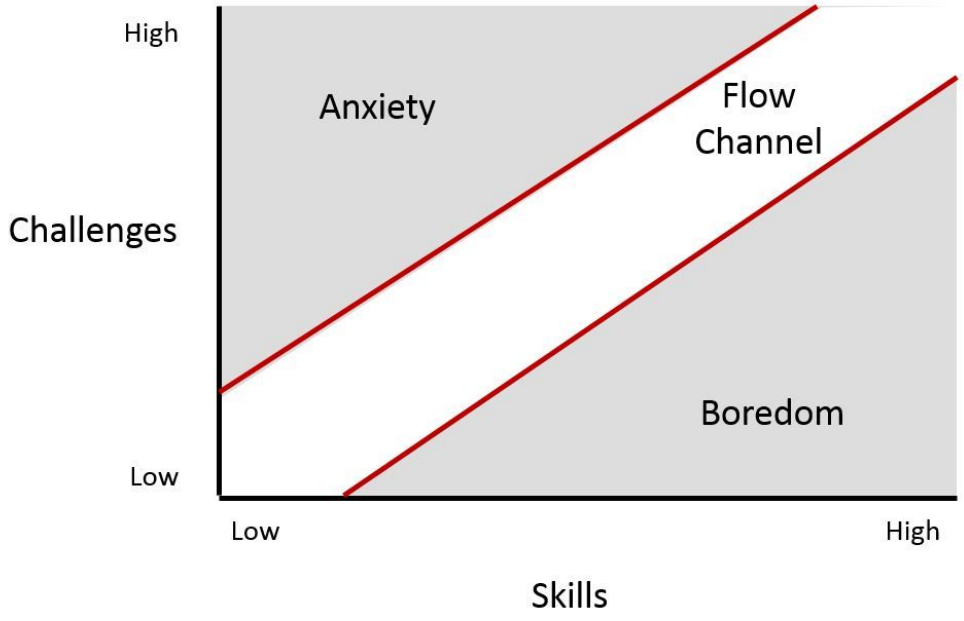


Figure 2

2. Current System (NA)

An existing system does not exist for this project. The whole system will be designed from the scratch.

3. Proposed System

3.1 Overview

The system which is going to be implemented is a two dimensional game which takes place at a space map near a station. The game will consist of enemies in the map and a main character which are all spaceships. The character will level up by playing the game properly which means killing enemies for experience points, collecting gift boxes and completing optional quests. Quests will be there for explaining the story, tutorial and leading the player through the game. The spaceship will be more powerful at each level and game will be over if the player kills all the enemies at the map. The game will end when player kills all the enemies on the map. Enemies will be regenerated after a while when they are killed. This will give the player a specific amount of time to kill all the enemies on the map.

3.2 Functional Requirements

The player must be able to control the game via a keyboard (up, down, left, right, space and enter) and a mouse. All the spaceships must be able to fully rotate which means it does not have four states which are top, bottom, left and right but it must be able to rotate at each angle. Enemies must be able to follow and attack the player if the player attacks them. Enemies must be able to move on the map with their own decisions (wander around or attack to the player).

3.3 Nonfunctional Requirements

Objects which are not in the viewport must not be rendered. If objects are rendered when they enter the viewport, they might look like they appeared suddenly and came there from nowhere. This rendering must start a little away from the viewport to make it like it was rendered at the beginning.

3.4 Constraints

Instead of thread approach for each object on the screen, a frame based approach must be implemented which means the game must have frames and each frame each objects' movement is calculated and rendered. The implementation language must be Java. For the game frame, Swing Library must be used. For the graphics, AWT's Graphics2D library must be used.

3.5 System Model

3.5.1 Scenarios

3.5.1.1 Control Game

Player sees the entrance window and presses any key to play the game. Player presses W key on the keyboard and spaceship move

in the map. Player presses D key on the keyboard and spaceship rotates clockwise. Player presses A key on the keyboard and spaceship rotates counter-clockwise. Player presses space and spaceship shoots bullets in forward direction.

3.5.1.1 Give Quest

Game gives the player a quest to comploete either depending on a time or not. Player does the given tasks in order and completes the quest. If quest has a time limitation or requires not to do another task during the quest, player fails the quest.

3.5.1.3 Move Objects

Enemy moves randomly. Player moves when player controls the game. Player hits the enemy and enemy starts following player while shooting at the player.

3.5.1.4 Attack Enemy

Player shoots around. One of the bullets hits a random enemy. Enemy starts moving. Enemy attacks back.

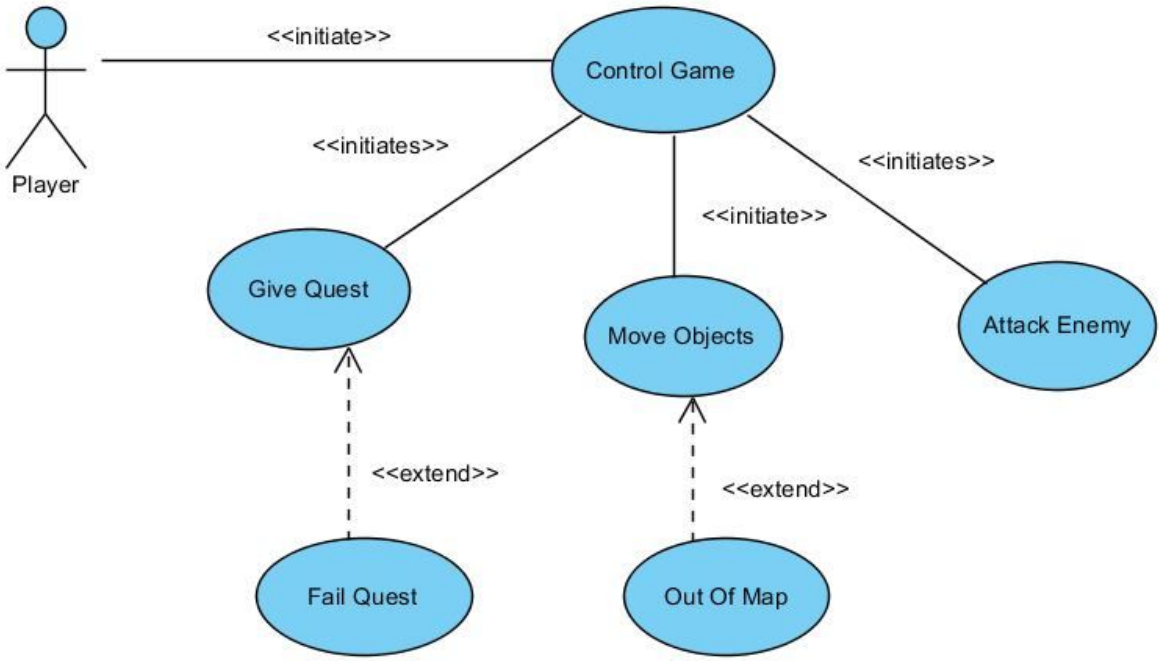
3.5.1.5 Fail Quest

Quest is not done in time or another task is done while quest is active. Quest returns back to quest list for player to take it again.

3.5.1.6 Out of Map

Player or enemies comes to the edge of the map. They try to move more to get out of the map. They cannot move further.

3.5.2 Use Case Model



Use Case Name	ControlGame
Participating Actors	Initiated by Player
Flow of events	1. Player presses any key. 2. The game appears. 3. Player presses W. 4. Spaceship moves forward. 5. Player presses D. 6. Spaceship rotates clockwise. 7. Player presses A. 8. Spaceship rotates counter-clockwise.
Entry Condition	-Player opens the game.
Exit Conditions	-Player finishes the game. -Player closes the game.
Quality Requirements	-The control events does not delay.

Use Case Name	GiveQuest
Participating Actors	Initiated by ControlGame
Flow of events	1. Player requests for a quest. 2. Game gives player a collection of tasks to complete. 3. Player does them in order in a given time if there exist a time limitation. 4. The quest is done if player does all the tasks correctly.
Entry Condition	-Player requests for a game throught controlling game.
Exit Conditions	-Player succesfully finishes all the tasks. -Player fails to success all the tasks.
Quality Requirements	-Quests must not affect eachother.

Use Case Name	MoveObjects
Participating Actors	Initiated by ControlGame
Flow of events	1. Enemy moves randomly. 2. Player control the game to move. 3. Spaceship moves. 4. Player shoots and hits the enemy. 5. Enemy follows player and shoots.
Entry Condition	-Player controls the game. -Enemy generated on the screen.
Exit Conditions	-Game finishes. -Player does not control the game.
Quality Requirements	-Objects must not delay.

Use Case Name	AttackEnemy
Participating Actors	Initiated by ControlGame
Flow of events	1. Player shoots a bullet. 2. Bullet hits the enemy. 3. Enemy moves.
Entry Condition	-Player control the game and shoots. -Bullet hits the enemy.

Exit Conditions	-Enemy dies. -Player dies. -Enemy is too far away.
Quality Requirements	-Attack must not affect not rendered objects.

Use Case Name	FailQuest
Participating Actors	Extends MoveObject
Flow of events	1. Player makes a mistake while doing the quest. 2. Quest returns back to the quest list and it becomes like it was not taken before.
Entry Condition	-Player fails to complete the quest in time -Player makes a mistake
Exit Conditions	-Quest returns back to the quest list.
Quality Requirements	-Quest must succesfully re

Use Case Name	OutOfMap
Participating Actors	Extends GiveQuest
Flow of events	1. Player or enemy comes to the edge of the map. 2. Player or enemy tries to move further. 3. Player does not move.
Entry Condition	-Player moves out of the map. -Enemy moves out of the map.
Exit Conditions	-Player moves inside the map.
Quality Requirements	-Player must not bounce from the edge of the map.

3.5.3 Object Model

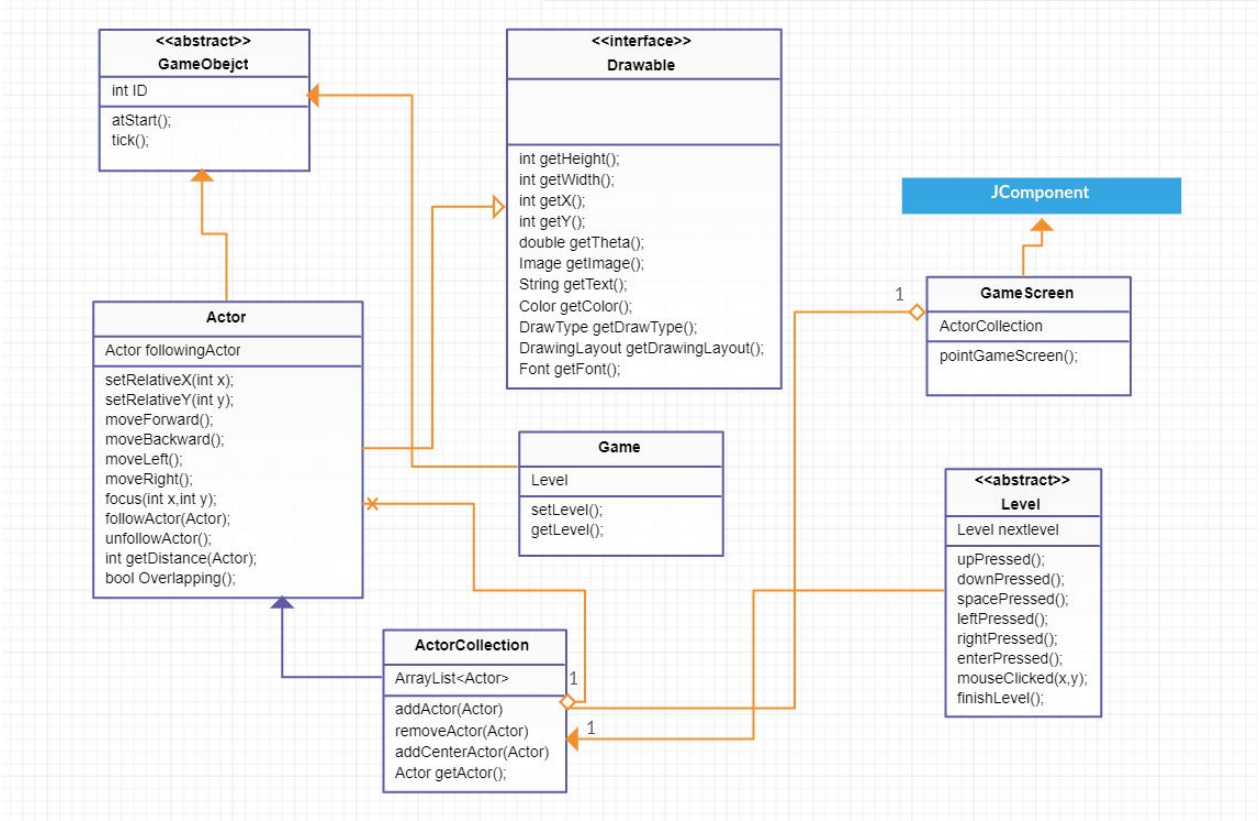
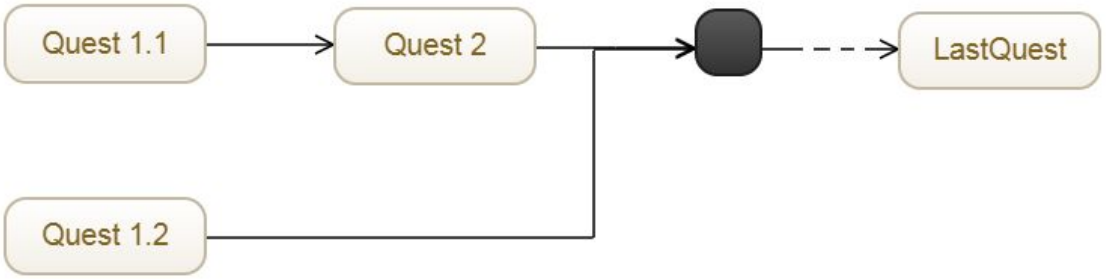
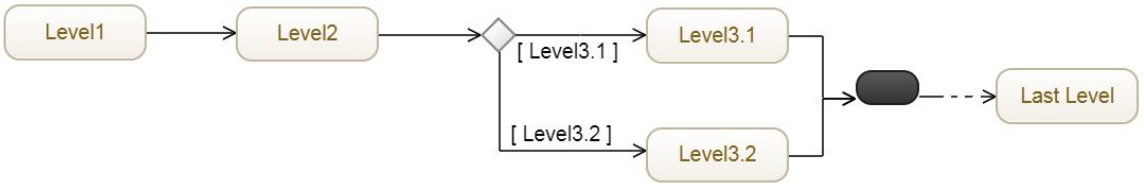
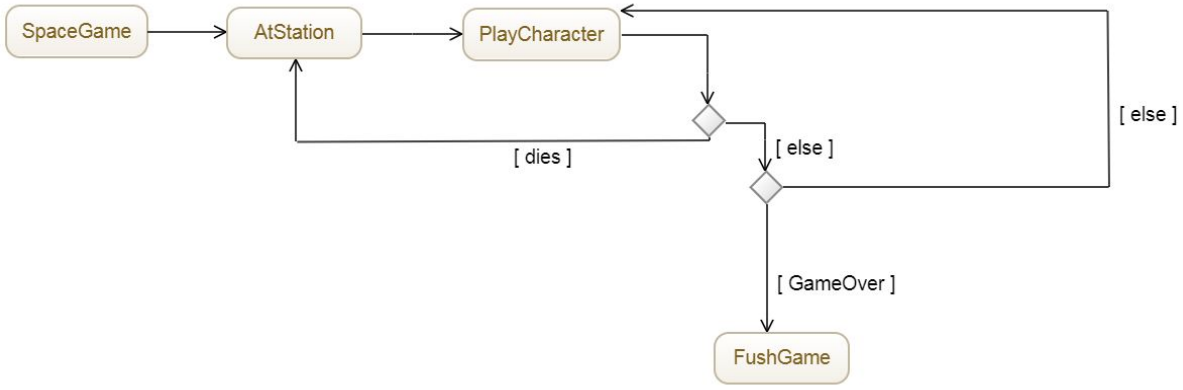


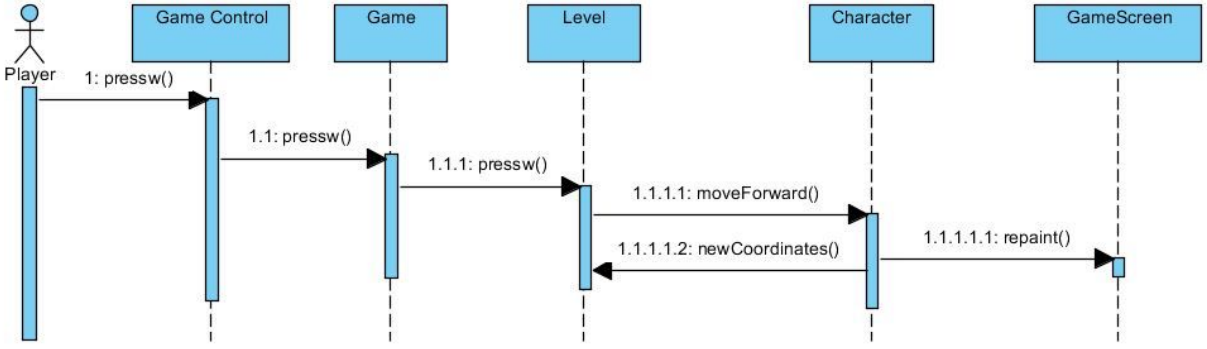
Figure 4

3.5.4 Dynamic Model





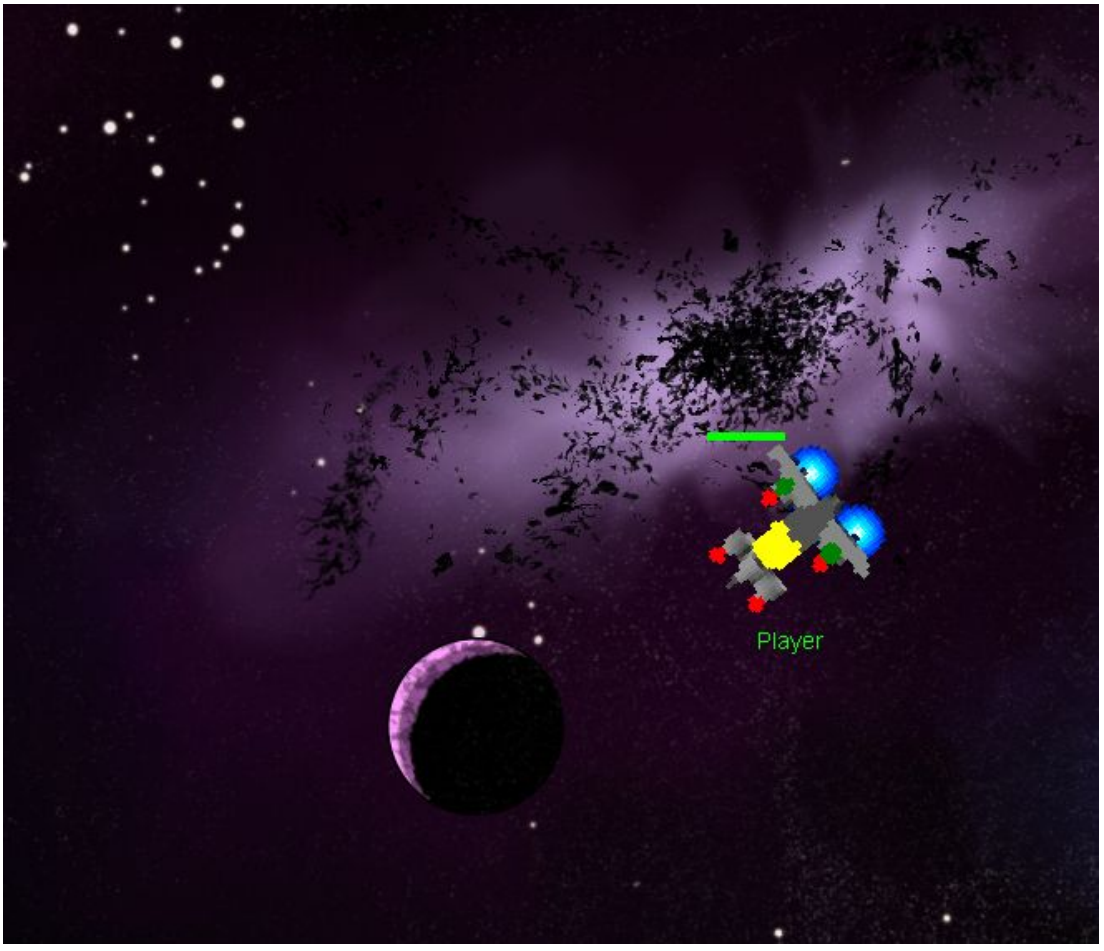


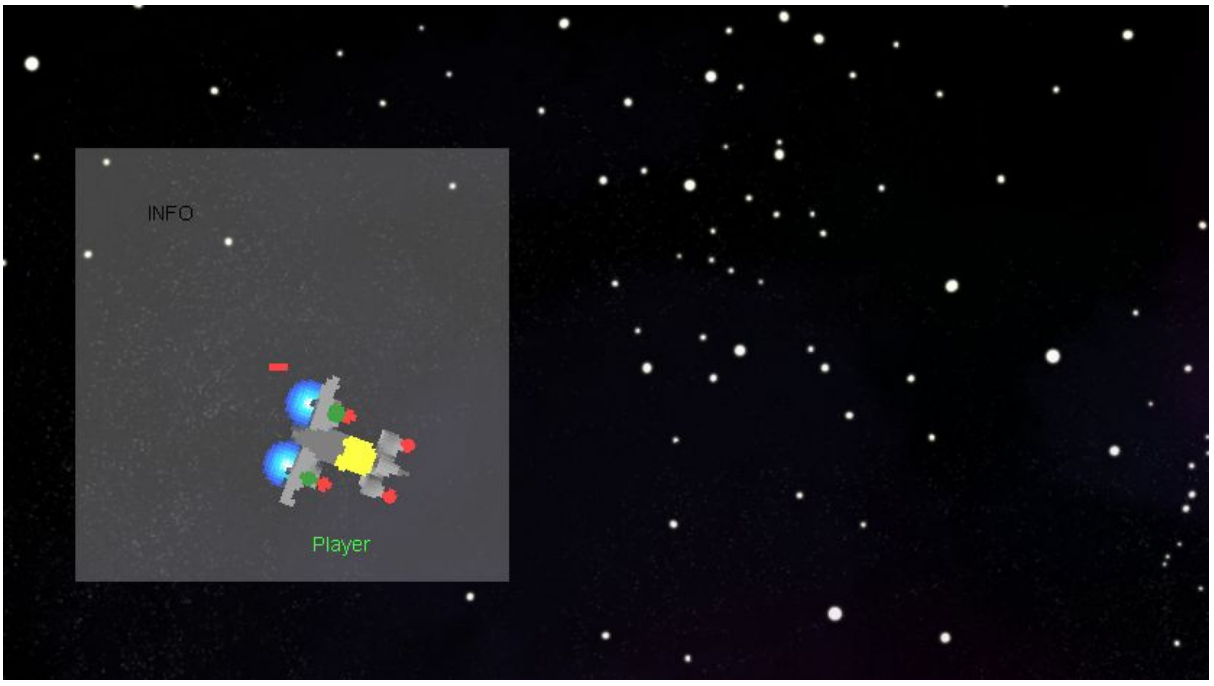


3.5.5 User Interface

A Space Game

powered by Java Game Engine





4. Glossary

Player refers to who plays the game. Eventhough spaceship is named seperately, player sometimes can refer to the spaceship which the player controls. Controlling the game means that player presses keyboard keys.

5. References

1. https://www.youtube.com/watch?v=yxpW2ltDNow&t=43s
2. https://www.youtube.com/watch?v=tWtvrPTbQ_c&t=81s