

Requirements and Analysis Document for “How do I fly this thing”

Table of Contents

Version: 0.5

Date: 2014-05-14

Author: Joakim Thorén, Francine Mäkelä, Mathias Carlsson, Martin Nilsson

This version overrides all previous versions.

Table of Contents

1 Introduction

- 1.1 Purpose of application
- 1.2 General characteristics of application
- 1.3 Scope of application
- 1.4 Objectives and success criteria of the project
- 1.5 Definitions, acronyms and abbreviations

2 Requirements

- 2.1 Functional requirements
- 2.2 Non-functional requirements
 - 2.2.1 Usability
 - 2.2.2 Reliability
 - 2.2.3 Performance
 - 2.2.4 Supportability
 - 2.2.5 Implementation
 - 2.2.6 Packaging and installation
 - 2.2.7 Legal
- 2.3 Application models
 - 2.3.1 Use case model
 - 2.3.2 Use cases priority
 - 2.3.3 Domain model
 - 2.3.4 User interface
- 2.4 References

APPENDIX

Use cases

- Use case: Start game
- Use case: Host
- Use case: Join game
- Use case: Connect to server
- Use Case: Exit (Launcher)
- Use Case: Move ship
- Use Case: Shoot
- Use Case: Collision
- Use Case: Collision with two spaceships
- Use Case: Collision with spaceship and projectile
- Use Case: Collision with spaceship and game world border
- Use Case: Collision with spaceship and pickup
- Use Case: Collision with fixed objects
- Use Case: Ship destroyed
- Use Case: Start Round
- Use Case: End Round
- Use Case: Disconnect

GUI

1 Introduction

This section gives a brief overview of the project.

1.1 Purpose of application

The project's aim is to create a networked multiplayer game in which players control a spaceship with firing capability. The players battle in a restricted zone within space and are supposed to destroy their opponents.

1.2 General characteristics of application

The application will be a desktop, standalone, networked multi-player application with a graphical user interface for the Windows/Mac/Linux platforms/ using only the keyboard for ensuring optimal laptop experience.

The application will be real-time. A user hosts a game, to which another player can connect to with IP. Directly when two players are inside the game a round starts. In this round the goal is to eliminate other players by maneuvering the ship with thrusters, which alters the spaceship's velocity, and shooting projectiles at other players. Upgrades and/or powerups which modify the spaceship can be found. Spaceships can collide with asteroids in the game world, destroying the spaceship. If another player joins the game he will have to wait for the next round to start in order to spawn. Last man standing wins.

1.3 Scope of application

The game won't allow you to save a game. There won't be any server application, only a direct connection to a host which acts as a server. Graphics are entirely 2D and very basic 2D, no special effects except animations. If the host disconnects or shuts down his game session, there will be no host-transfer.

1.4 Objectives and success criteria of the project

Below are features of the game which should be implemented in order to consider the project a success:

- Launcher
 - Host game
 - Join game
 - Options (to adjust video settings and keybindings)
 - Exit
- 2D graphics representing game world
- Audio (sound effects when shooting, getting hit etc)

- Maneuverable spaceship with 3 different thrusters which move the ship and 1 gun which can fire
- Thrusters should move the ship based on where they are located. Thruster on bottom-right rotates ship anti-clockwise and move forward slightly, bottom-left thruster rotates clockwise and move forward slightly, middle thruster move forward fast. The general “feel” of maneuvering the ship should be similar to the game Rakete, see reference for link to Rakete-website¹.
- Spaceship keep their speed if thrusters isn’t used (no gravity)
- Camera (screen) is always moving in a way such that the player is centered in the screen
- Users able to connect to an host via IP-adress
- When 2 or more players are connected a round will start
- Whenever a player enters a host and a new round is about to begin he spawns with a spaceship once the round commences.
- Spaceships have:
 - Hull (hitpoints)
 - Sheild (hitpoints that regenerates. Not as strong as hull)
 - 1 Gun
 - 3 thrusters which are independently controlled with key on keyboard
- Players can get hit by bullets shot by other players
- Players can collide with structures which causes massive amount of damage directly to hull
- Player spaceship explodes if it’s hull is 0.
- Once there is a last man standing a new round commences automatically within couple of seconds
- Users can disconnect the game at any time via ESC which returns the user to the Launcher

1.5 Definitions, acronyms and abbreviations

- GUI, graphical user interface.
- Java, platform independent programming language.
- JRE, the Java Run time Environment. Additional software needed to run an Java application.
- Host, a computer where the game will run.
- Session, one complete game, ending when the host disconnects.
- Round, a part of a session ending when only one spaceship remains.
- FPS, Frames Per Second in application - how many images are shown every second, higher the better.
- Latency-spike, whenever the connected players temporarily recieves high ping due to server-issues
- Launcher, a window with buttons such as “Host”, “Join”, “Settings”

¹ <http://www.mariov.ch/portfolio/project/rakete>, Rakete by Mario von Rickenbach

2 Requirements

2.1 Functional requirements

The user should be able to:

1. Host a game
2. Join a game
3. Move his spaceship
4. Fire bullets from his spaceship
5. Get hit by other spaceships, causing damage to self
6. Collide his spaceship with:
 - a. Other spaceships
 - b. Asteroids
 - d. Pickups (gaining whatever is inside)
7. Start new round once there's only a last man standing
8. Disconnect from server
9. Exit game

2.2 Non-functional requirements

2.2.1 Usability

Usability is a high priority. Users should be able to execute every use-case without any confusion nor problems.

2.2.2 Reliability

Users should not get disconnected every other second once connected to host

2.2.3 Performance

Atleast stable 30++FPS, without FPS-drop. Disconnecting due to latency-spike shouldn't happen immedeatly.

2.2.4 Supportability

The application must be implemented for support in Windows operating system. The implementation divides the application into a client/server-architecture for net based games.

2.2.5 Implementation

To achieve Windows supportability , and possibly other platforms, the application will use the Java environment. All hosts must have the JRE installed and configured. The application needs to be installed on all hosts where it will run (possibly downloaded).

2.2.6 Packaging and installation

The application will be delivered as a zip-file containing

1. A folder containing

1.1 A file for the application (*.jar)

1.2 Raw-resources

2.2.7 Legal

Not covered here.

2.3 Application models

2.3.1 Use case model

See appendix for UML.

2.3.2 Use cases priority

Use Case: Collision with fixed objects
Use Case: Collision with spaceship and pickup
Collision with spaceship and game world border
Collision with spaceship and projectile
Collision with two spaceships

High:

- Start game
- Exit
- Move ship
- Shoot
- Collision
- Collision with fixed objects
- Collision with two spaceship
- Collision with spaceship and game world border
- Collision with projectile and spaceship

Medium:

- Host
- Join
- Join game

Low:

- Collision with pickups

2.3.3 Domain model

See appendix for Domain model diagram.

2.3.4 User interface

Se appendix for image.

//Text to motivate a picture.

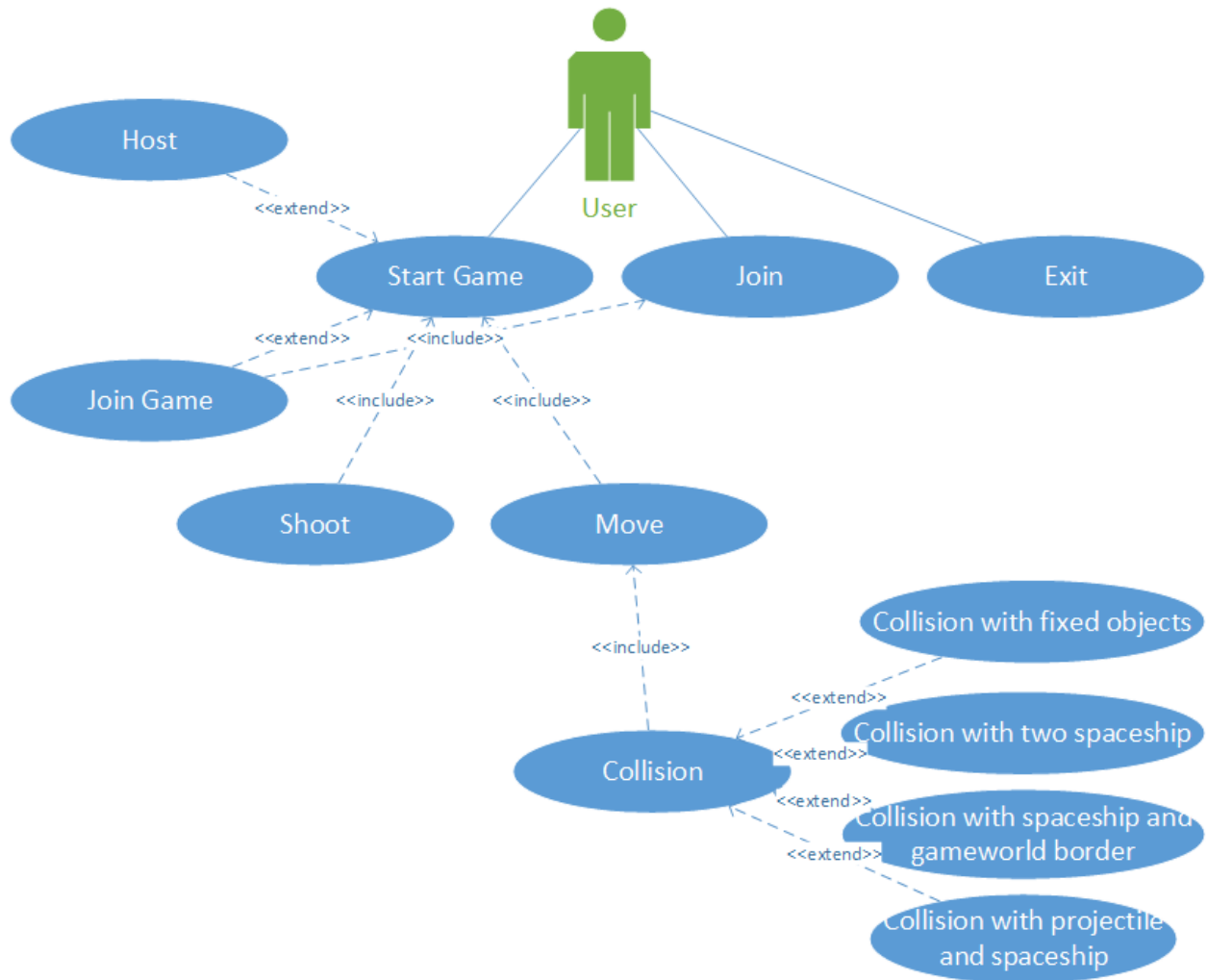
The player controls the orange spaceship. The users shields and hull is represented as bars in the top left corner of the screen.

2.4 References

<http://www.mariov.ch/portfolio/project/rakete>, Rakete by Mario von Rickenbach

APPENDIX

Use cases



Use case: Start game

Summary: A user can start a game which removes Launcher window and displays the world (space) in which he can control his spaceship. UC Host and UC Join game extends this UC.

Priority: High

Extends: -

Includes: -

Participators: -

Normal flow of events

Game world is loaded

	Actor	System
1	Clicks button which loads game world	
2		Load game world
3		Display game world

Use case: Host

Summary: This is how the user host a game for others to join. Top alternative in Launcher which is shown on application startup.

Priority: Medium

Extends: Start game

Includes: -

Participator: User

Normal flow of events

User hosts a game

	Actor	System
1	Clicks the host button	
2		Stop displaying Launcher
3		Start game-loop

Use case: Join game

Summary: This is how an user connects to an already hosted game. Alternative in Launcher which is shown on application startup.

Priority: Medium

Extends: -

Includes: -

Participators: User

Normal flow of events

User is prompted for IP to host

	Actor	System
1	Enters IP in text field	
2	Clicks join button	
3		Stop displaying Launcher
4		Connects to host

Use case: Connect to server

Summary: Preceded by UC Join. When IP to host is entered and join game button is pressed, user will join the game.

Priority: Medium

Extends: Start game

Includes: Join

Participators: User

Normal flow of events:

User joins the host

	Actor	System
1	Clicks the connect button	
2		Connect to host
3		Display game world from host
4.		Spawn ship on next round

Use Case: Exit (Launcher)

Summary: On selecting exit alternative in Launcher

Priority: High

Extends: -

Includes: -

Participator: User

Normal flow of events

Application is closed

	Actor	System
1	Clicks the Exit button	
2		Application is closed

Use Case: Move ship

Summary: When user presses any of the thruster keys the ship will activate corresponding thruster and move accordingly to laws of physics.

Priority: High

Extends: -

Includes: UC Start game

Participants: Player

Normal flow of events

Player ship moves

	Actor	System
1	Clicks any thruster key	
2		Move ship according to thruster

Use Case: Shoot

Summary: On pressing the shoot button in game

Priority: High

Extends: -

Includes: -

Participant: User

Normal flow of events

Projectiles fires from the ship, in the proper direction.

	Actor	System
1	Presses shoot button	
2		Ship fires shots.

Use Case: Collision

Summary: On colliding with any solid object

Priority: High

Extends: -

Includes: -

Participator: User

Normal flow of events

Spaceship collides with anything in game world

	Actor	System
1	Moves and collides	
2		Collision reaction of some sort

Use Case: Collision with two spaceships

Summary: Whenever a spaceship collides with another spaceship they both explode (and die) instantly

Priority: High

Extends: Collision

Includes:

Participator: User

Normal flow of events

	Actor	System
1	Whenever a user maneuvers his spaceship and crashes into another spaceship	
2		Both ships instantly explode
3		Both players loses the round and become spectators

Use Case: Collision with spaceship and projectile

Summary: Whenever a projectile collides with a spaceship the spaceship takes damage to its shield or hull corresponding to the projectiles power.

Priority: High

Extends: -

Includes: Collision

Participator: User

Normal flow of events

	Actor	System
1	Projectil collides with a spaceship.	
2		The projectil is removed.
3		The spaceship takes damage.

Use Case: Collision with spaceship and game world border

Summary: Whenever a spaceship collides with the border of the world the spaceship instantly explodes

Priority: High

Extends: Collision

Includes:

Participator: User

Normal flow of events

Spaceship collides with game world border.

	Actor	System
1	Spaceship collides with game world border	
2		Spaceship explodes instantly
3		User becomes a spectator

Use Case: Collision with spaceship and pickup

Summary: Whenever a spaceship collides with any pickup in the world, it will gain the powerups or weapons (depending of content of the pickup)

Priority: Low

Extends: Collision

Includes:

Participator: User

Normal flow of events

	Actor	System
--	-------	--------

1	Spaceship collides with pickup	
2		Spaceship equips the weapon or get the powerup depending on the content of pickup

Use Case: Collision with fixed objects

Summary: Whenever a spaceship collides with another fixed object (asteroid)

Priority: High

Extends: Collision

Includes: -

Participator: User

Normal flow of events

	Actor	System
1	Spaceship collide with fixed object	
2		Spaceship explode instantly
3		User becomes a spectator

Use Case: Ship destroyed

Summary: When a ship has lost all hull (health) it is immediately destroyed.

Priority: Medium

Extends: -

Includes: -

Participator: User

Normal flow of events

User spaceship is destroyed.

	Actor	System
1	Spaceship has no hull.	
2		Spaceship is destroyed.
3		Corresponding player becomes spectator.

Use Case: Start Round

Summary: When the game starts or a round is over a (other) round starts.

Priority: Low

Extends: -

Includes: -

Participator: User

Normal flow of events

	Actor	System
1	Second player joins or round has ended.	
3		Reset the map
4		Respawn all players
5		Enable all players

Use Case: End Round

Summary: When only one player remains alive the round ends.

Priority: Low

Extends: -

Includes: -

Participator: User

Normal flow of events

	Actor	System
1	Is only one left alive	
2		Disable players
3		Display all stats
4		Start new round

Use Case: Disconnect

Summary: Whenever clicking ESC in-game: destroy the players spaceship and disconnects him from server, bringing him back to the Launcher

Priority: Low

Extends: -

Includes: -

Participator: User

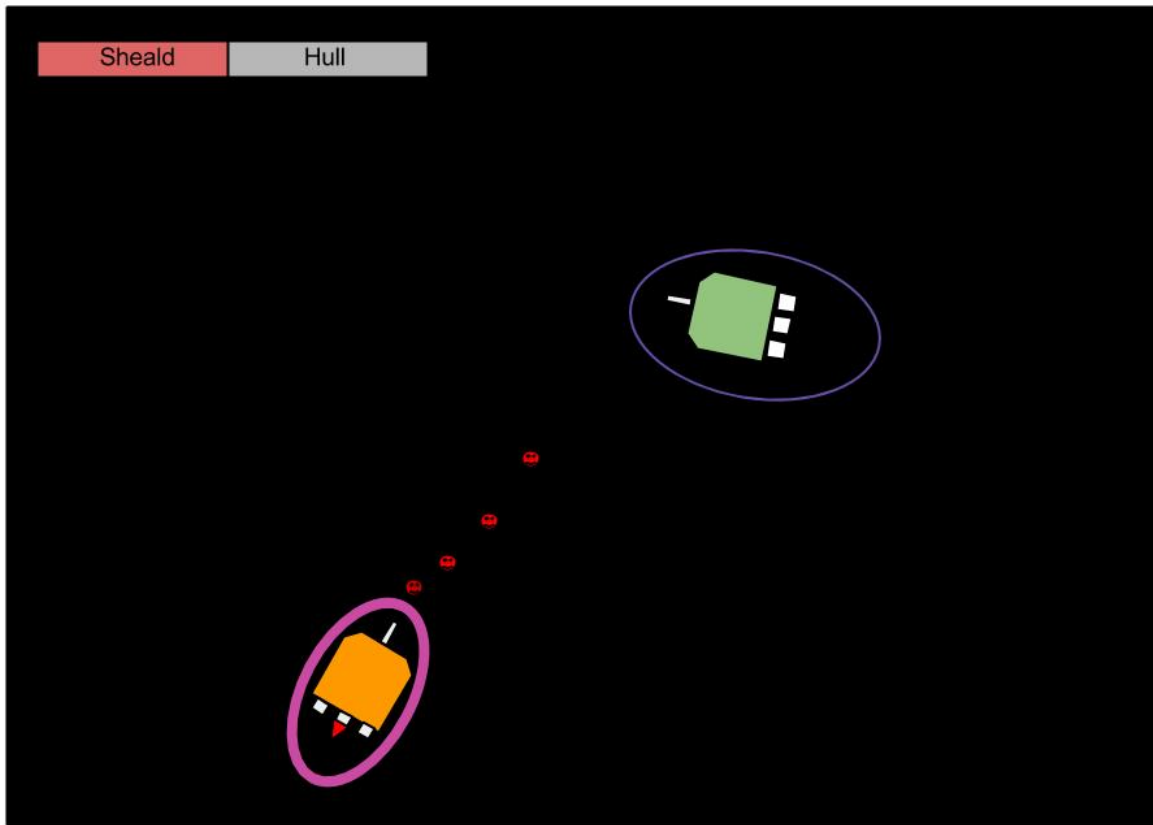
Normal flow of events

User spaceship is destroyed and user is disconnected

	Actor	System
1	Press ESC in-game menu	
2		User spaceship is destroyed
3		User is disconnected from server
4		User is brought back to Launcher

GUI

Preliminary GUI



Domain model

