Requirements and Analysis Document for “HowDoIFlyThisThing”

Table of Contents

Version: 0.7

Date: 2014-05-21

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 diagram
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
Use Case: Exit
Use Case: Move
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: HealthPickup
Use Case: MissilePickup
Use Case: CookieCrackerPickup
Use Case: Collision with asteroids
Use Case: Spaceship destroyed
Use Case: Start Round
Use Case: End Round

Use Case: Disconnect
Use Case: Destroy asteroid
Use Case: Settings
Use Case: Set fullscreen
Use Case: Set keybindings

1 Introduction

This section gives a brief overview of the project.

1.1 Purpose of application

The projects aims 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 player 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 spaceships velocity, and shooting projectiles at other players. Upgrades and/or powerups which modifies the spaceship can be found. Spaceships can collide with asteroids in game world, destroying the spaceship. If another players joins the game he will have to wait for 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 dedicated server application, only direct connection to host which acts as a server. Graphics are 2D with basic 2D animations. If host disconnects or shutdown his game session, there will be no host-transfer and all players will get disconnected and returned to launcher application.

1.4 Objectives and success criteria of the project

Bellow 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-website[1].
- Spaceship keep their speed if thrusters isn't used
- 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 thrusers 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 the spaceship to explode
- 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.

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

- 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”

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 immideatly.

2.2.4 Supportability

The application must be implemented with support for Windows / OSX / Linux operating

system. The implementation divides the application into a client/server-architecture for net based games.

2.2.5 Implementation

To achieve platform-independence, 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.

2.3 Application models

2.3.1 Use case diagram

See appendix for UML.

2.3.2 Use cases priority

High:

- Start game
- Exit
- Move ship
- Shoot
- Spaceship destroyed
- Collision
- Collision with asteroid
- Collision with two spaceship
- Collision with spaceship and game world border
- Collision with projectile and spaceship

Medium:

- Host
- Join
- Disconnect
- Start Round
- End Round

Low:

- Collision with spaceship and pickup
- HealthPickup
- MissilePickup
- CookieCrackerPickup
- Set fullscreen
- Settings
- Set keybindings
- Destroy asteroid

2.3.3 Domain model

Not embedded here. See project/documentation/DomainModel.pdf

2.3.4 User interface

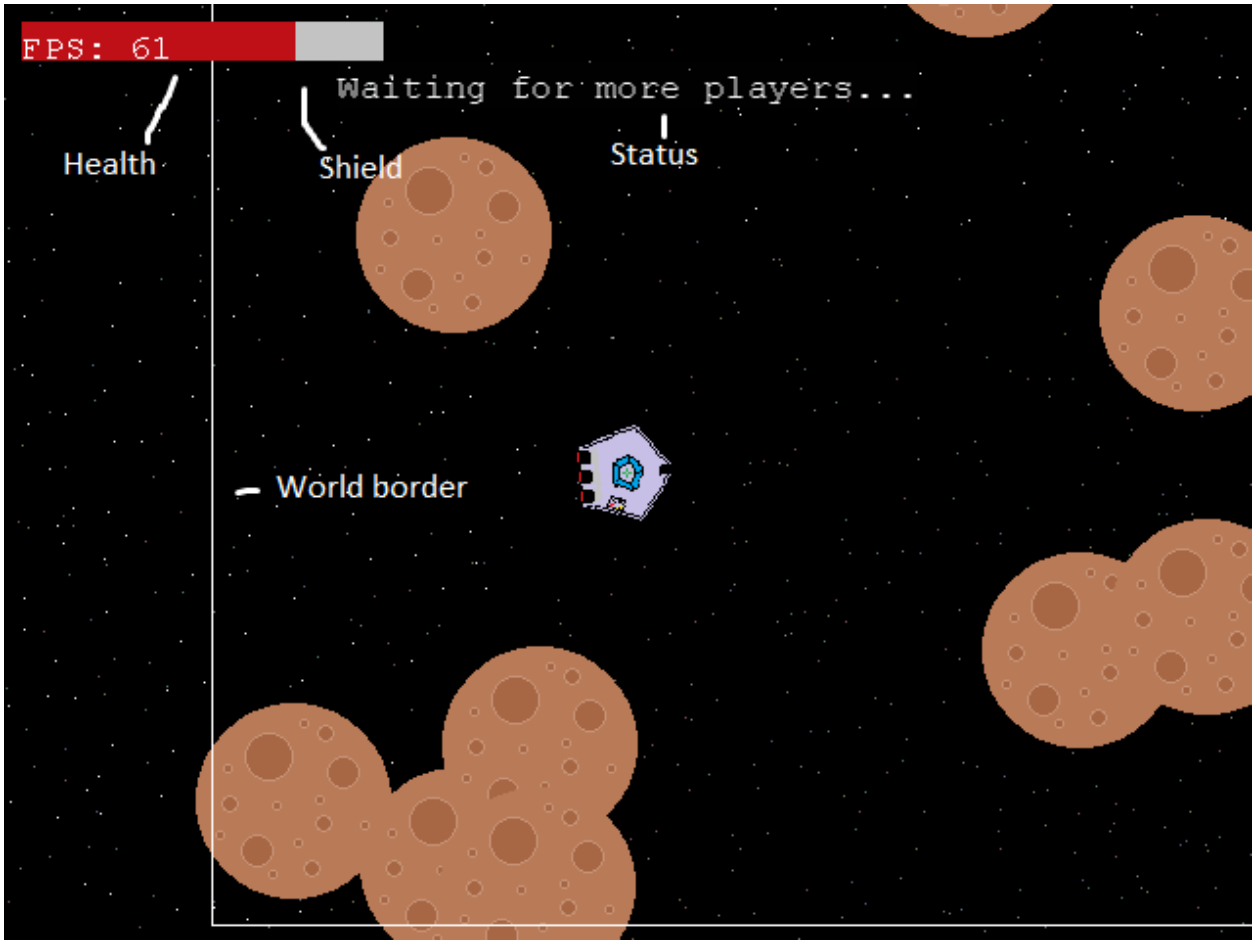


Fig 1

The player controls the orange spaceship. The users shields and hull is reprecented as bars in

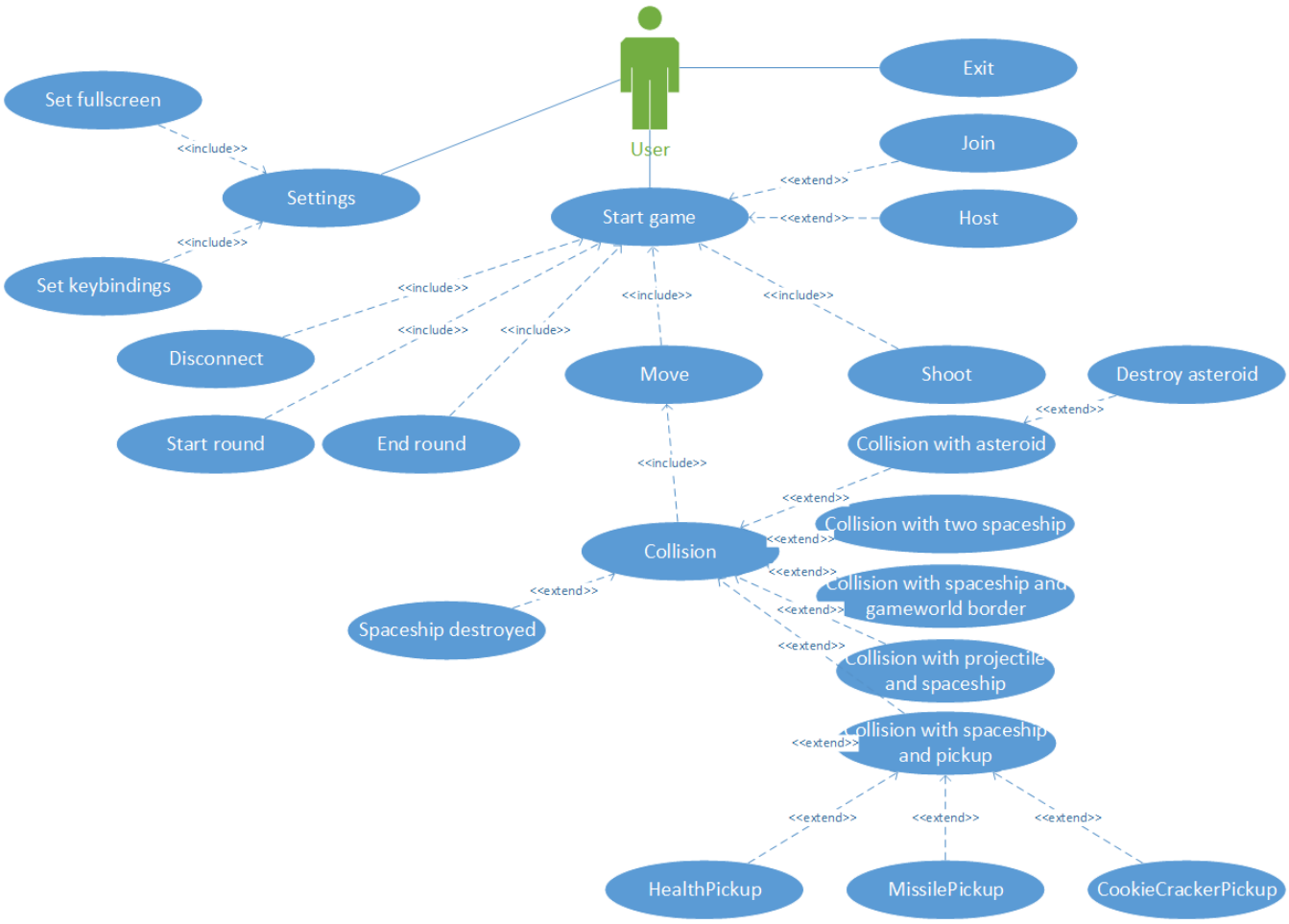
the top left corner of the screen. The hull is the red bar and the shield is the grey bar. A status message indicates what state the round currently is in - if it's active, inactive, commencing or waiting for players. There's also a white border surrounding the playable area. Without this players wouldn't knew when they were heading out of map.

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 extends this UC.

Priority: High

Extends: -

Includes: -

Participators: User

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

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

Priority: Medium

Extends: Start game

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: Exit

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

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

Priority: High

Extends: -

Includes: Start game

Participators: User

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: Start game

Participator: 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: Start game

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: Two users

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: Collision

Includes: -

Participator: User

Normal flow of events

	Actor	System
1	Projectil collides with a spaceship.	
2		The projectile 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). This content is specified by three other Use Cases: HealthPickup, MissilePickup, CookieCrackerPickup

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: HealthPickup

Summary: Whenever a spaceship collides with a healthpickup, it gains extra health (aka hull).

Priority: Low

Extends: Collision with spaceship and pickup

Includes: -

Participator: User

Normal flow of events

	Actor	System
1	Spaceship collides with healthpickup	
2		Give extra hull to spaceship

Use Case: MissilePickup

Summary: Whenever a spaceship collides with a missilepickup, it gains missile as a weapon. Missiles accelerate, have slower rate of fire and deal more damage.

Priority: Low

Extends: Collision with spaceship and pickup

Includes: -

Participator: User

Normal flow of events

	Actor	System
1	Spaceship collides with missilepickup	

2		Spaceship fire missiles instead of weapon

Use Case: CookieCrackerPickup

Summary: Whenever a spaceship collides with a cookiecrackerpickup it gains a cookiecracker as a weapon. Cookiecrackers accelerate, have slower rate of fire and are capable of hurting asteroids (triggering the Destroy Asteroid use case).

Priority: Low

Extends: Collision with spaceship and pickup

Includes: -

Participator: User

Normal flow of events

	Actor	System
1	Spaceship collides with cookiecrackerpickup	
2		Spaceship fire cookiecrackers instead of previous weapon

Use Case: Collision with asteroids

Summary: Whenever a spaceship collides with asteroid

Priority: High

Extends: Collision

Includes: -

Participator: User

Normal flow of events

	Actor	System
1	Spaceship collide with asteroid	
2		Spaceship explode instantly
3		User becomes a spectator
4		Asteroid take damage

Trivia: Asteroids are also known as Cookies

Use Case: Spaceship destroyed

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

Priority: Medium

Extends: Collision

Includes: -

Participator: User

Normal flow of events

User spaceship is destroyed upon collision with either asteroid or a projectile

	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: Medium

Extends: -

Includes: Start game

Participator: User

Normal flow of events

	Actor	System
1	Second player joins or round has ended.	
3		Countdown 5sec, then enable damage
4		Reset the map
5		Respawn all players

Use Case: End Round

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

Priority: Medium

Extends: -

Includes: Start game

Participator: User

Normal flow of events

	Actor	System
1	Is only one left alive	
4		Start new round (UC Start 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: Medium

Extends: -

Includes: Start game

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

Use Case: Destroy asteroid

Summary: Whenever an asteroid reaches 0 health, it will explode. It's health is visualized by cracking it up the closer to death it is. It loses health from spaceships colliding with it or CookieCrackers hitting it.

Priority: Low

Extends: Collision with asteroid

Includes: -

Participator: Asteroid and user or projectile

Normal flow of events

Asteroid get hit by Spaceship or CookieCracker

	Actor	System
1	Spaceship or projectile hit asteroid	
2		Asteroid take damange
3		Crack it if damage threshold is reached
4		If it's health is bellow or equal to 0, it dies

Use Case: Settings

Summary: When user select settings in Launcher

Priority: Low

Extends: -

Includes: -

Participator: User

Normal flow of events

User has started launcher and selects Settings

	Actor	System
1	User selects settings	
2		Settings window is displayed

Use Case: Set fullscreen

Summary: When user check fullscreen setting

Priority: Low

Extends: -

Includes: Settings

Participator: User

Normal flow of events

User click the checkbox “Fullscreen”. On hosting or joining a game afterwards the game will be in fullscreen.

	Actor	System
1	User checks checkbox fullscreen	
2		On hosting or joining game will be fullscreen

Use Case: Set keybindings

Summary: When user sets a key in settings

Priority: Low

Extends: -

Includes: Settings

Participator: User

Normal flow of events

User click the a textfield and hit a key, which will set the action to be binded to that key.

	Actor	System
1	User click a textfield and hit a key	
2		Specified action will be binded to that key