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

Table of Contents

Version: 0.3

Date: 2014-03-27

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

This version overrides all previous versions.

Table of Contents

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. There are abandonned space-stations in the space “arena” in which upgrades and/or powerups will be found. Colliding with a space-station will severly damage the hull of the ship. 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

There is no reason to play this game alone, and therefore this will be impossible. The game won’t allow you to save a game. There won’t be any server application, only direct connection to host. Graphics are entierly 2D and very basic 2D, no special effects. No stats are saved permanently (no database). If host disconnects or shutdown his game session, there will be no host-transfer.

1.4 Objectives and success criteria of the project

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

- Main menu
    - Host game
    - Join game
    - Options (to adjust video settings and audio)
    - 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 (no gravity)
- Camera (screen) is adjusting itself to always show what's infront of the ship
- Users able to connect to an host via IP-adress
- When 2 or more players are connected a round will start
- Temporary statboard for each game session
- Procedurally generated abadonned spacestations each round in which upgrades/powerups/equipment can be found
- Whenever a player enters a host and a new round is about to begin he spawns with a spaceship
- Spaceships have:
    - Hull (hitpoints)
    - Sheild (hitpoints that regenerates. Not as strong as hull)
    - Weight (alters movement speed. Increases if gun is a heavy-gun or if ship has much 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 massive amount of damage directly to hull
- Player spaceship explodes if it's hull is 0.
- Once there is a last man standing the temporary staboard is displayed and a new round commences automatically within couple of seconds
- Users can quit the game at any time via ESC -> Quit game
- Users can disconnect from a game at any time via ESC -> Disconnect

1.5 Definitions, acronyms and abbreviations

- GUI, graphical user interface.
- Java, platform indep endent 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.

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

- Latency-spike, whenever the connected players temporarily recieves high ping due to server-issues

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
    c. Spacestation
    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 fairly high priority. In order to get the game to a playable state graphics are needed, and graphics won't be up running at the first couple of weeks hence game won't be in a playable state the first couple of weeks. Having a spaceship moving around and firing bullets with graphics should be up and running approximatly 4 weeks into the project.

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 for support in Windows operating system. The implementation should prepare for the dividing of the application into a client/server-architecture for net based games. It should be easy to partitioning the application into a client-server architecture.

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 congured. 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 file for the application (*.jar)
2. Raw-resources

2.2.7 Legal

N/A

2.3 Application models

2.3.1 Use case model

See appendix for UML.

2.3.2 Use cases priority

High:

- Start game
- Host
- Exit
- Move ship
- Shoot

Medium:

- Join
- Join game

2.3.3 Domain model

See appendix for Domain model diagram.

2.3.4 User interface

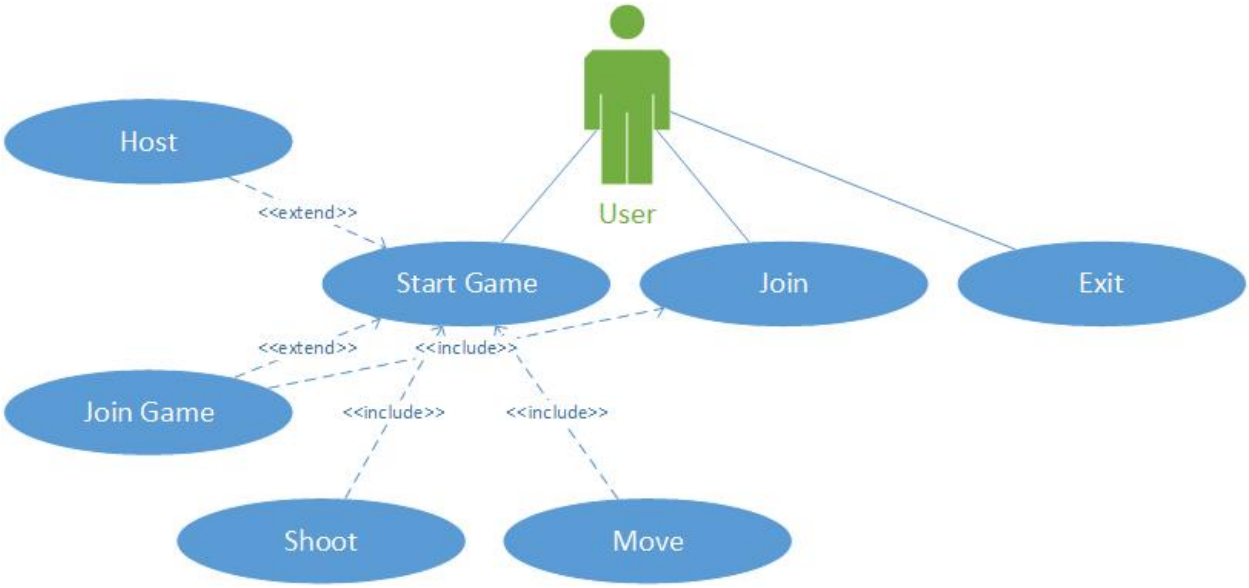
Se appendix for image.
//Text to motivate a picture.
The player controls the orange spaceship. The users shields and hull is reprecented as bars in the top left corner of the screen.

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 main menu 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 main menu which is shown on application startup.

Priority: High

Extends: Start game

Includes: -

Participator: User

Normal flow of events

User hosts a game

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

Use case: Join game

Summary: This is how an user connects to an already hosted game. Alternative in main menu 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	Clicks the Join button	
2		Stop displaying menu
3		Prompts for IP 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 (main menu)

Summary: On selecting exit alternative in main menu

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 accodringly to laws of physics.

Priority: High

Extends: -

Includes: UC Start game

Participators: 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: -

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 with two spaceships

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

Priority: Medium

Extends: -

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: Medium

Extends: -

Includes:

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: Low

Extends: -

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: -

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 (eg. spacestation or asteroid)

Priority: Low

Extends: -

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 imediately 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 bringing up in-game menu user can chose “disconnect from server” which destroys the players spaceship and disconnects him from server, bringing him back to the main menu

Priority: Low

Extends: -

Includes: -

Participator: User

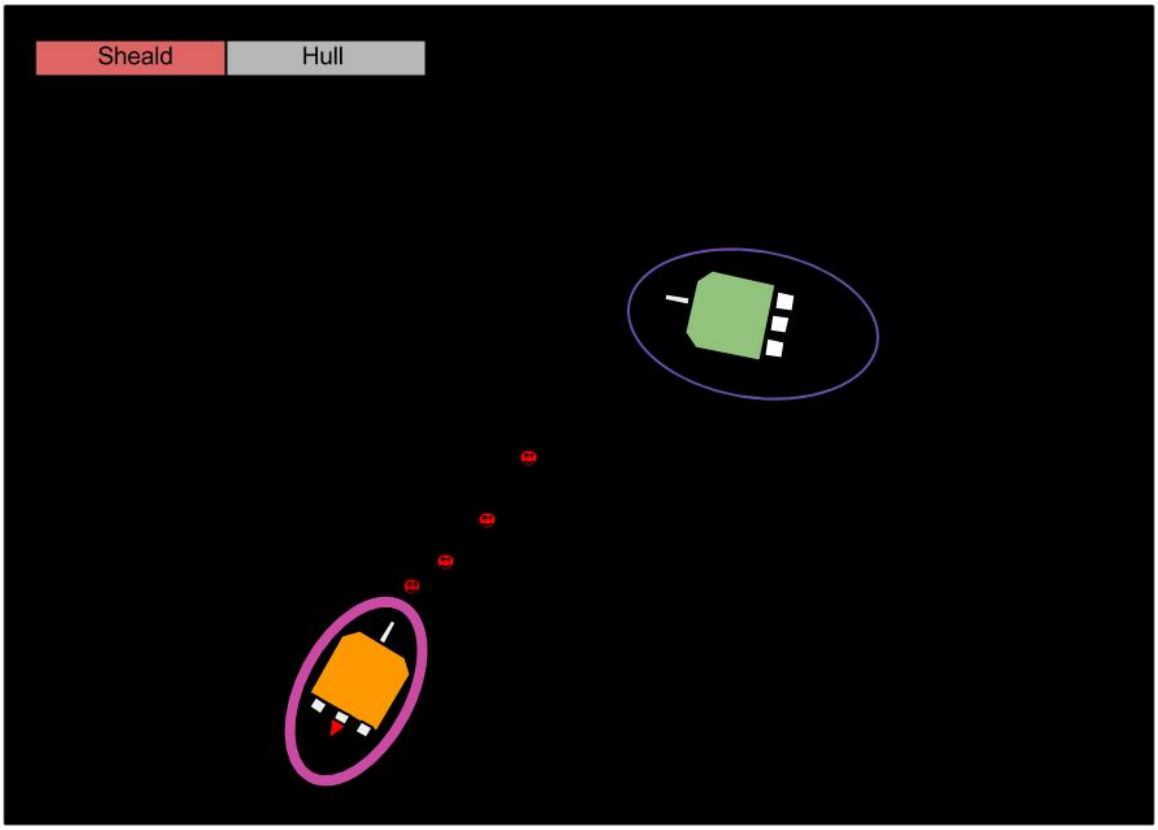
Normal flow of events

User spaceship is destroyed and user is disconnected

	Actor	System
1	Selects Disconnect in in-game menu	
2		User spaceship is destroyed
3		User is disconnected from server
4		User is brought back to main menu

GUI

Preliminary GUI



Domain model


Main menu
Application
In- game menu
Server
Round
1..*
Session
1..1
Status
User
1..1
Spectator
1..*
Game world
*...1
Player
1..1
1..1
Border
Pickup
1..1
1..1
Powerup
1..*
Spacestation
Weapon
1...1
Spaceship
*...1
1..*
Asteroid
1..*
Projectile
1...3
Thruster
1..1
Damage
1..1
Shield
1...1
Hull