Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

Requirements and Analysis Document

Galactica: Space Force of Justice

1. Introduction

We have created a 3D game for both desktop and Android. The game is also available for VR through the Android application. The game has design similarities to the movie “Tron Legacy” and is played using a hover based spacecraft confined in a circular course. The spacecraft is controlled using keyboard for desktop, an on screen joystick and button for Andriod, and a separate gamepad when playing in VR. When playing VR, the visual elements will be shown on the phone which is connected to a google cardboard.
The game’s main purpose is to used as a recreational activity and the main target group consists of game enthusiasts with access to the necessary items required to run the application. The aim of the game is to generate as much points as possible by surviving in a course with increasing difficulty.

1.2 Definitions, acronyms and abbreviations

App - The application as a whole.
Game - The actual game that the user will play.
Spacecraft - The spacecraft that the player will control in the game.
Course - The 3d track in the game that the player’s spacecraft will fly in.
Obstacle - Objects that is in the way, when things that you lose life from when you get hit.
Entity - A renderable 3d model that will be visually shown and used in the course.
World - The course,the spacecraft ,entities and all the obstacles.
Camera - A point from which the observers vision is based.
Tron - A sci-fi film we base our design on. Known for its neon blue colors.
HUD (Head Up Display) - A layer with information on top of the game. Displays information such as amount of health, score and current power-up.
Input device: A device that the user uses to control the game (keyboard, gamepad etc).

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

2. Requirements

2.1 User interface

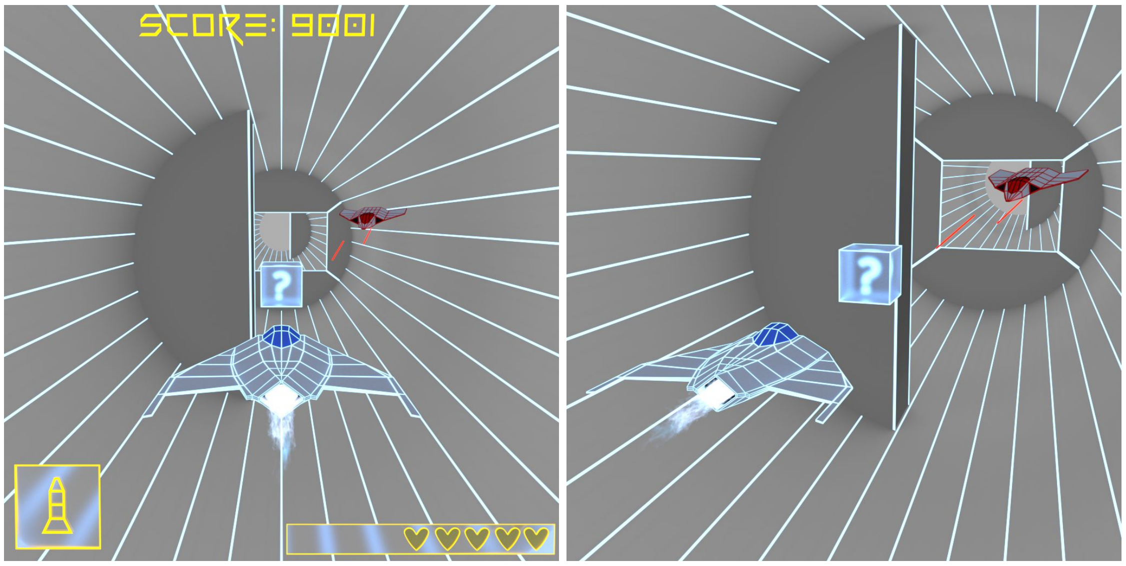


Figure 1 & 2. Initial planning pictures, references used to create the final game.

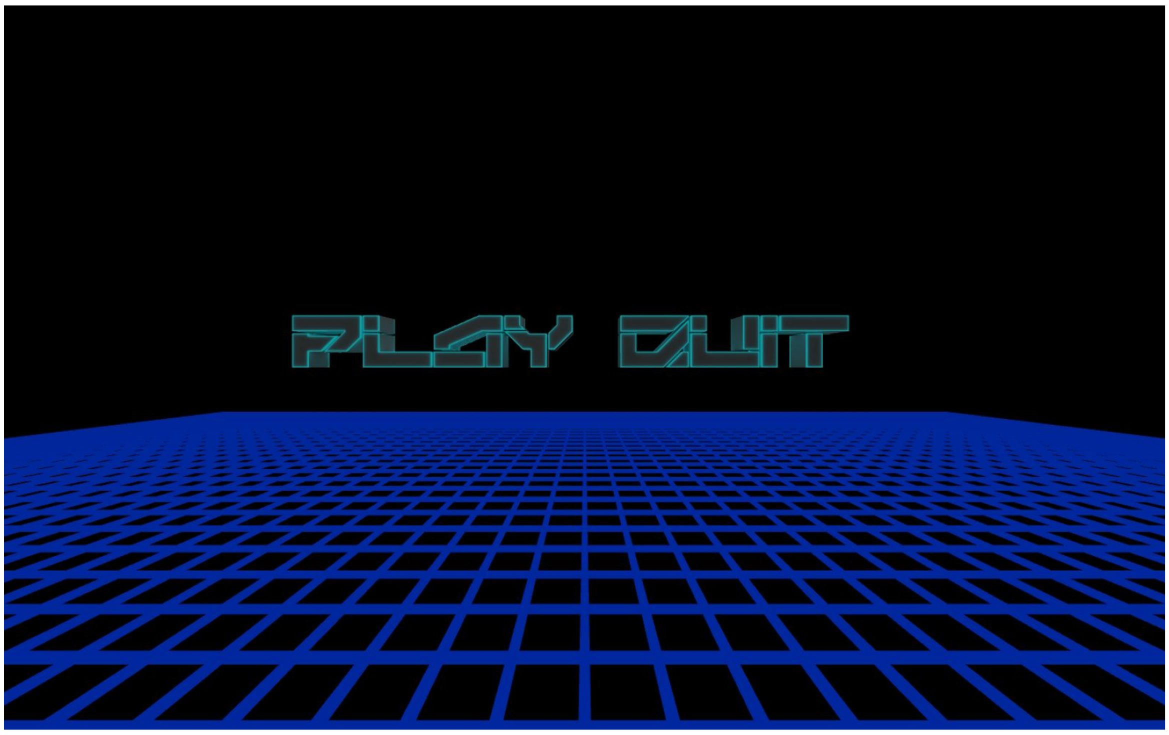


Figure 3. Start menu of the game, where the player can choose either play or quit by shooting a cannon shot towards the text.

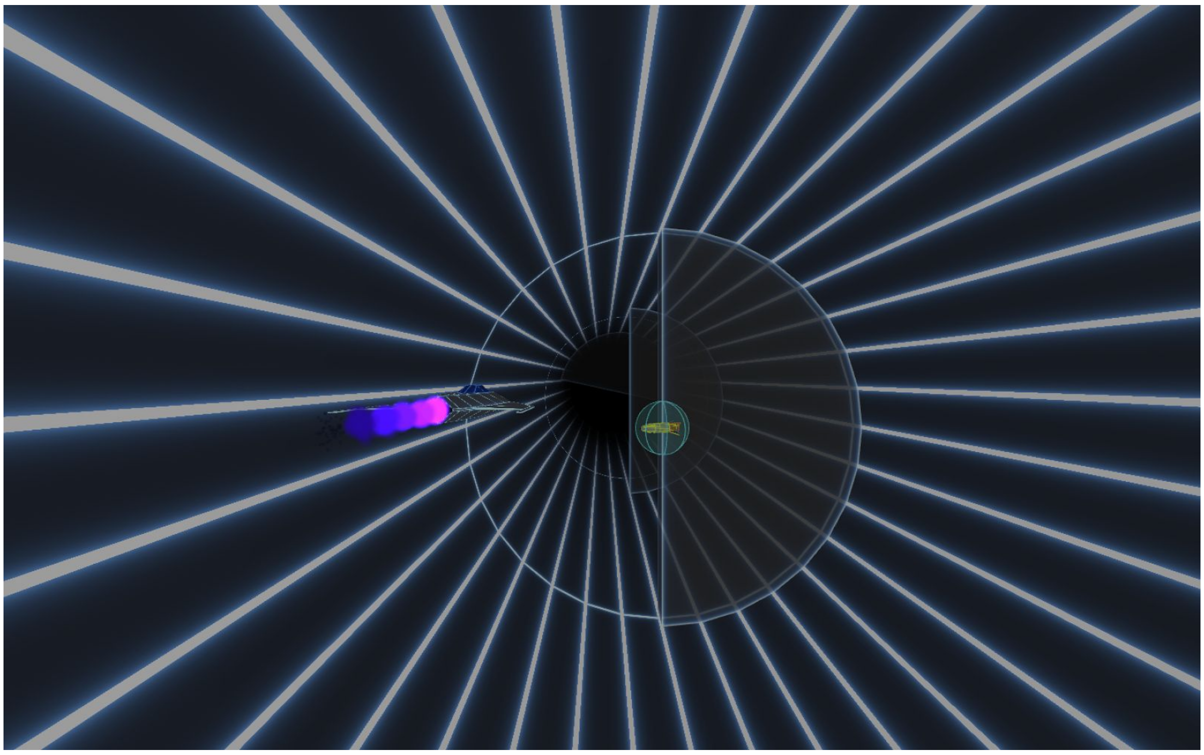


Figure 4. Player phase of the game, also showing a container for power-ups
.

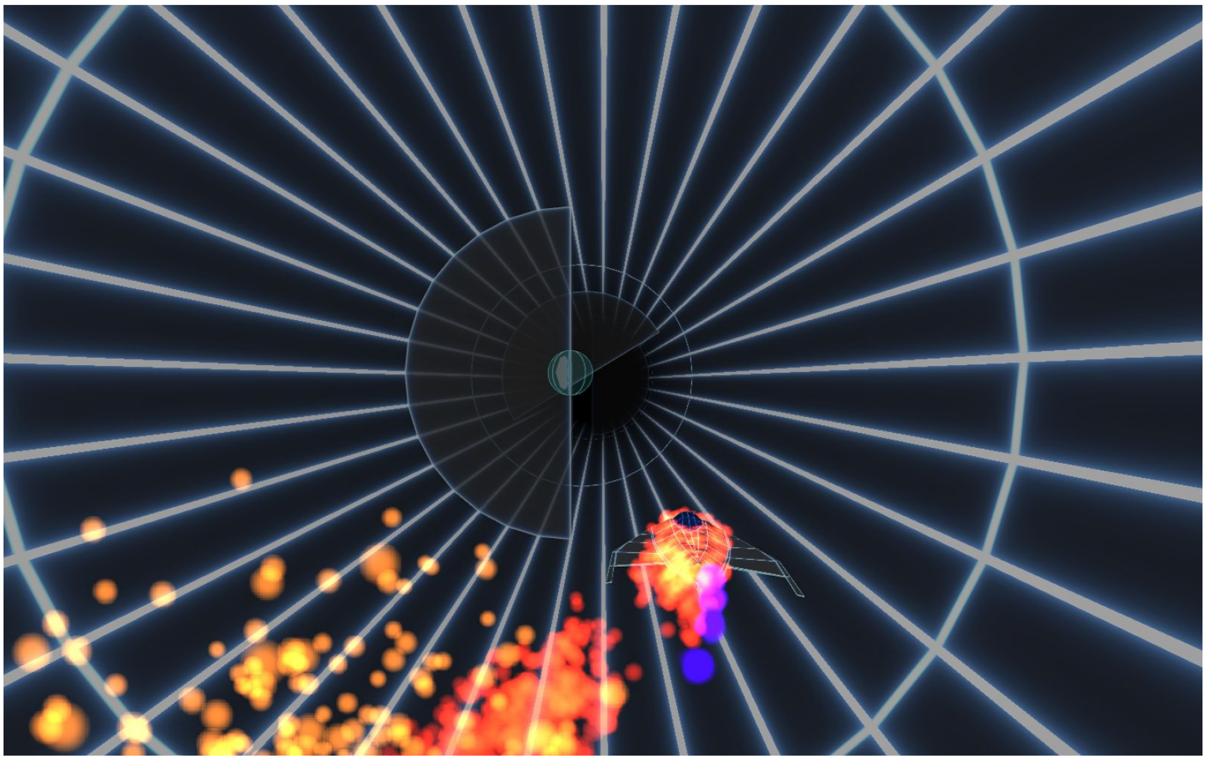


Figure 5. The spacecraft has almost taken fatal damage and has a particle trail indicating that it’s damaged.

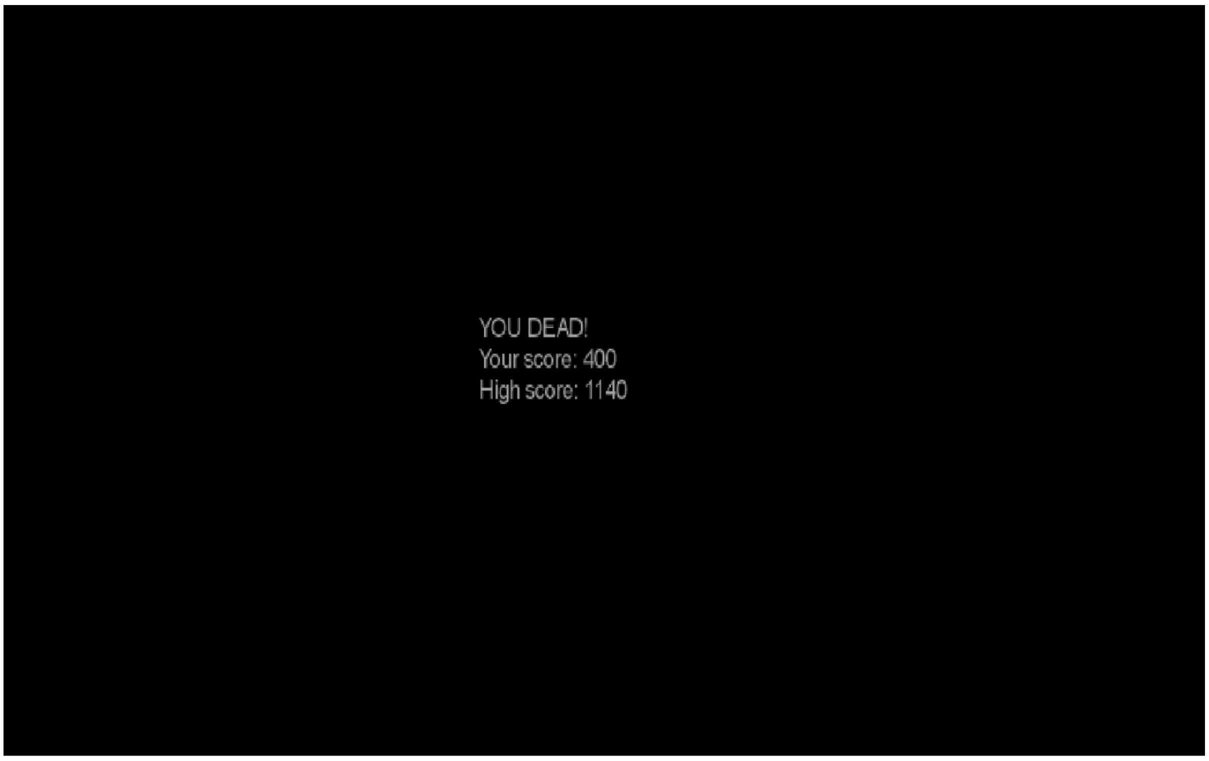

YOU DEAD!
Your score: 400
High score: 1140

Figure 6. The game has ended and a screen with score is shown before the game starts over at the same screen as figure 3.

2.2 Functional requirements

The functionality for the application includes:

1. Start a new game with interactive main menu.
2. Pause and unpause the game. You can look around in VR mode.
3. During a game session a player can.
    a. Move the ship around the course.
    b. Use different Power-ups(cannon, missile or get a shield) that helps the player.
    c. Collide with different objects and depending on the object different outcomes occur.
    d. Toggle a HUD placed by the side of the ship.
    e. Get a score depending on how far the player has gotten.
4. Exit the application.

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

Uses cases ranked by priority

- Game Start
- Move
- Collide
- Crash
- Look around
- Pick up power-ups
- Use power up
- Cannon
- Missile
- Shield
- Toggle HUD
- Start menu
- Game Over
- Pause
- App start

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

2.3 Non-functional requirements

“Galactica - Space Force of Justice” is a computer/mobile game with VR functionality that mainly focuses on: extensibility and performance.

Usability.

The application should be simple enough for an average user to use without any written instructions. The controllers will be very conventional in order to simplify the learning curve.

Extensibility

The application should be working on multiple platforms. The game is playable on desktop, Android and in Google Cardboard, this means that it needs to support multiple screens, different performance limitations and many input devices. The program needs to support the standard gamepads from Android, Xbox and Playstation.

Performance.

The application must run smoothly on all of the platforms. A heavy emphasis has been placed on keeping the program small and with low performance cost. Playing games on a phone strains its resources, especially when using VR. This is a problem, especially since low frame rates in VR can cause so called *simulator sickness.* Therefore you can not simply focus on the computer implementation and expect it to work on a phone, the focus must be reversed.

Testability

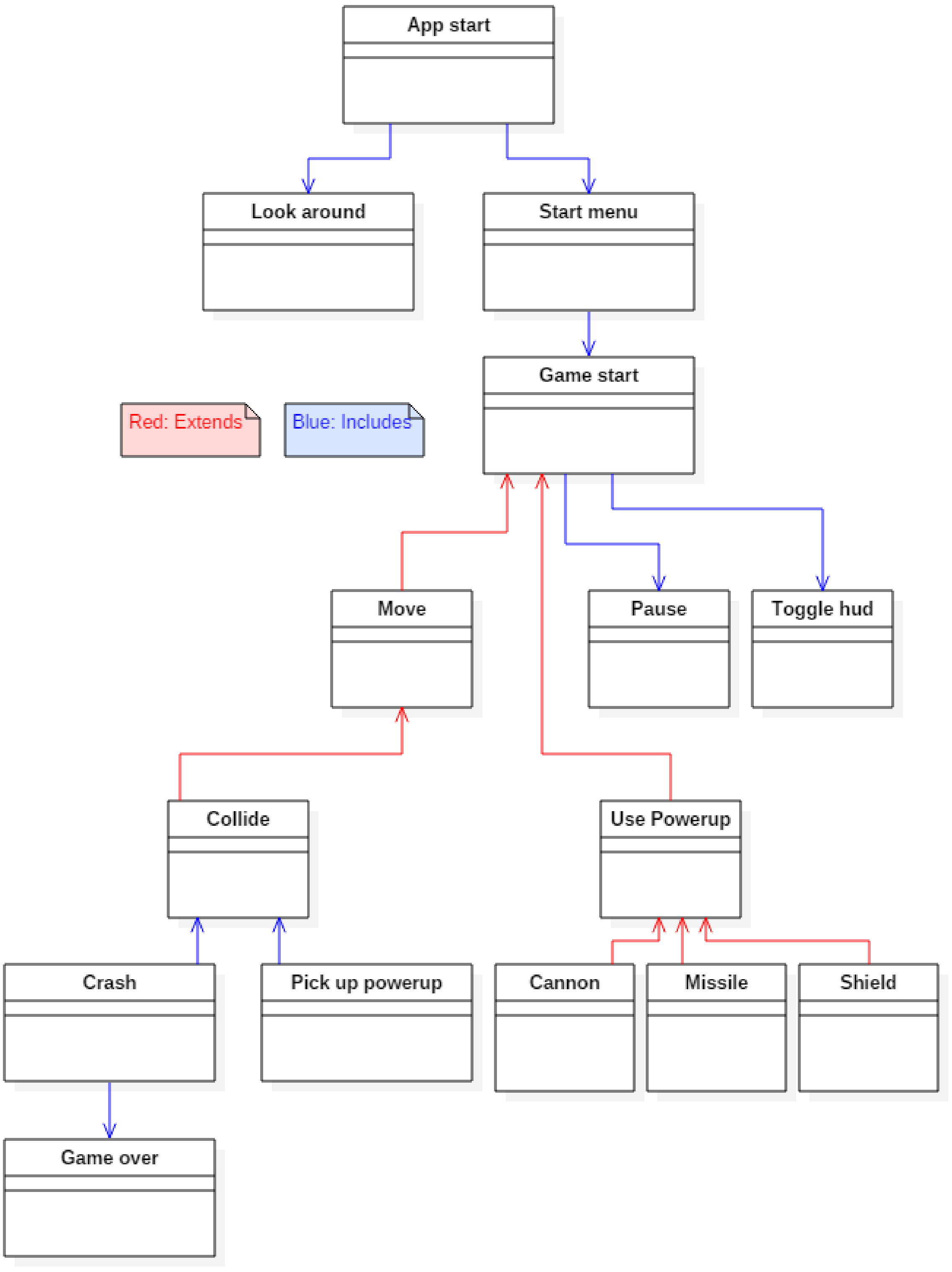
The application has a couple of tests, making it possible to easily find the exact location of bugs or program breaking behavior. This insures that the program runs as intended.

Legality

The application will only use self made 3D models and free to use music. This places us as the full owners of the application and can distribute the game safely. The game can be released on Google Play.

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

3. Use cases listing



Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

3.1 Use cases

1. Use case: App starts

Summary: Launches app and chooses device to run in.
Priority: low
Extends: none
Includes: Start menu,Look around
Participants: Actual player

Normal flow of events:
Player launches app on Desktop

	Actor	System
1	User launches app	
2		See Start menu

Alternate flow:
Player launches app on phone and chooses Android.

	Actor	System
1	User launches app	
2		System shows game mode selector screen
3	User chooses game mode	.
4	User chooses Android	
		See Start menu

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

Alternate flow:
Player launches app on phone and chooses Cardboard

	Actor	System
1	User launches app	
2		System shows game mode selector screen
3	User chooses game mode	.
4	User chooses Cardboard	
		See Start menu

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

2. Use-case: Look around

Summary: The player plays on a device that has access to a gyroscope and which allows the player to look around in the game.
Priority: Medium
Extends:
Includes:
Participants: Player

Normal flow of events

	Actor	System
1	Moves head	
2		The view will rotate towards the user's head direction.

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

3. Use case: Start menu

Summary: The user decides between playing and exiting the game.
Priority: Mid
Extends: None
Includes: Game Start
Participants: Actual player

Normal flow of events:
The player starts the game

	Actor	System
1		System shows targetable buttons for playing and exiting the game.
3	User clicks screen, firing a projectile that travels into the distance.	
4		Checks if projectile collides with a button
5		Collides with start button
6		see “Game start”

Alternate flows:
The player Exits the game

	Actor	System
1		System shows targetable buttons for play and exit game.
3	User clicks screen, firing a projectile that travels into the distance.	
4		Checks if projectile collides with a button
5		Collides with exit button
6		Game closes

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

Alternate flows:
The player misses the buttons.

	Actor	System
1		System shows targetable buttons for play and exit game.
3	User clicks screen, firing a projectile that travels into the distance.	
4		Checks if projectile collides with a button
5		Projectile misses buttons
6		Projectile gets removed

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

4. Use-case: Game Start

Summary: The player is now piloting the spacecraft.
Priority: high
Extends:
Includes: Pause,Toggle HUD
Participants: User

Normal flow of events:

	Actor	System
1		System initializes the game and displays it for the user.
2	The user now controls the ship and decides what will happen next in the game.	

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

5.Use case: Move

Summary: The user drags the joystick or keyboard arrow keys to any direction, and the ship moves in that direction
Priority: high
Extends: Game start
Includes:
Participants: Actual player

Normal flow of events:
The player moves the ship.

	Actor	System
1	Drags the joystick in a direction or presses any arrow key.	
2		Spacecraft pans in specified direction
3		Checks if ship can fly in that direction
4		Ship moves in that direction and stops when the user stops moving the joystick/keys, or hits a wall.

Alternative flow of events:
The player can't move the ship.

	Actor	System
1	Drags the joystick in a direction or presses any arrow key.	
2		Spacecraft pans in the specified direction.
3		Checks if ship can fly in that direction
4		Stays where it is

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

6. Use case: Collide

Summary: The Player collides with an object.
Priority: Mid
Extends: Move
Includes: Crash, Power-up
Participants: Actual player

Normal flow of events:
The player collides with an obstacle.

	Actor	System
1	The user moves the ship into an an obstacle.	
2		See Crash

Alternative flow of events:
The player collides with a power-up that requires activation.

	Actor	System
1	The user moves the ship into an a power-up	
2		See Pick up power-up

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

7. Use-Case: Crash

Summary: Player's ship crashes into an obstacle.
Priority: High
Extends:
Includes: Game Over
Participants: User

Normal flow of events
The player crashes with remaining lives left.

	Actor	System
1		A life is removed from the ship's lives.
2		Crash sound plays

Alternate flow of events
The player crashes with no lives remaining.

	Actor	System
1		Game stops updating.
2.		see Game over

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

8. Use case: Toggle HUD

Summary: The player toggles between showing the HUD and not showing the HUD.
Priority: Mid
Extends:
Includes:
Participants: Actual player

Normal flow of events:
The player wants to see the HUD.

	Actor	System
1	Player presses HUD button	
2		System shows the HUD which displays the spacecraft's current life, power up and score.

Normal flow of events:
The player sees the HUD and wants to hide it.

	Actor	System
1	Player presses the HUD button.	
2		System hides the HUD

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

9. Use case: Pick up power-up

Summary: The user collides with a power up container and possibly gets a power up.
Priority: Medium
Extends: None
Includes: None
Participants: Actual player

Normal flow of events:
The spacecraft has collided with a container but is already equipped with a different power up

	Actor	System
1		Container gets removed
2		Player already has another power up

Alternative flow of events:
The spacecraft has collided with a container but has the maximum number of that power up in storage.

	Actor	System
1		Container gets removed
2		Checks if the power up type is the same as the equipped power up.
3		It is the same power up as the equipped and nothing happens

Alternative flow of events:
The spacecraft has collided with a container that contains a power up (Cannon)

	Actor	System
1		Container gets removed
2		Player gets a cannon shot.
3		System shows hud with stored cannon shots, now incremented.

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

Alternate flow of events:
The spacecraft has collided with a container that contains a power up (Missie)

	Actor	System
1		Container gets removed
2		Player gets one Missile power up
3		System shows hud with stored missile shots, now incremented.

Alternate flow of events:
The spacecraft has collided with a container that contains a power up (Shield)

	Actor	System
1		Container gets removed
2		Player gets a shield power up
3		System shows a that the player has a shield in the HUD.

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

10. Use case: Use power up

Summary: The user uses stored power up.
Priority: Medium
Extends: Game Start
Includes: None
Participants: Actual player

Normal flow of events:
The player fires a projectile or tries to activate a shield

	Actor	System
1	Clicks the attack button	
2		Fires projectile or activates a shield
3		Removes one projectile/shield from HUD
4		See Cannon, missile, shield

Alternative flow of events:
The player tries to fire a projectile but has no stored power ups

	Actor	System
1	Clicks the attack button	
2		Plays sound for no power up

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

11.1. Use case: Cannon

Summary: The player uses the Cannon Power-up
Priority: medium
Extends: Use Power-up
Includes:
Participants: Actual player

Normal flow of events:
The player attacks and misses the target.

	Actor	System
1	Clicks the attack button	
2		Plays the cannon sound
		Fires projectile in the direction of the ship.
3		Projectile moves past the target into the distance.
4		The projectile is removed from the world.

Alternative flow of events:
The player attacks and hits a wall.

	Actor	System
1	Clicks the attack button	
		Plays the cannon sound.
2		Fires projectile in the direction of the ship.
3		The projectile hits a wall
4		The wall disappears.
5		The projectile is removed from the world

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

11.2. Use case: Missile

Summary: The user uses a Missile power up
Priority: Medium
Extends: Use power-up
Includes:
Participants: Actual player

Normal flow of events:
The player fires projectile and doesn’t hit anything.

	Actor	System
1	Clicks the attack button	
		Plays Missile sound
2		Fires projectile
3		Projectile tracking a static point in the course
4		The projectile is removed from the world.

Alternative flow of events:
The player fires projectile and hits targeted object.

	Actor	System
1	Clicks the attack button	
		Plays Missile sound.
2		Fires projectile
3		Projectile tracking a static point in the course
4		The projectile hits an object
5		The object disappears.
6		The projectile is removed from the world

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

11.3 Use case: Shield

Summary: The user user
Priority: Medium
Extends: Use power-up
Includes:
Participants: Actual player

Normal flow of events:
The player activates the shield power up

	Actor	System
1	Clicks attack button.	
		Shield gets deployed in front of players spacecraft

Alternate flow of events:
The player activates the shield power up and collides with an obstacle.

	Actor	System
	Clicks attack button.	
		Shield gets deployed in front of players spacecraft
1	Drives into an obstacle head on	
		Shield collides with obstacle.
2		Obstacle gets destroyed.
3		Shield gets destroyed

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

12. Use case: Pause

Summary: The user pauses the game.
Priority: Mid
Extends: None
Includes: None
Participants: Actual player

Normal flow of events:
The player pauses the game and then continues

	Actor	System
1	User presses the ”pause” button on an input device.	
2		Game logic stops updating.
4	User presses the “pause” button on the input device.	
5		Game starts again

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

13. Use case: Game Over

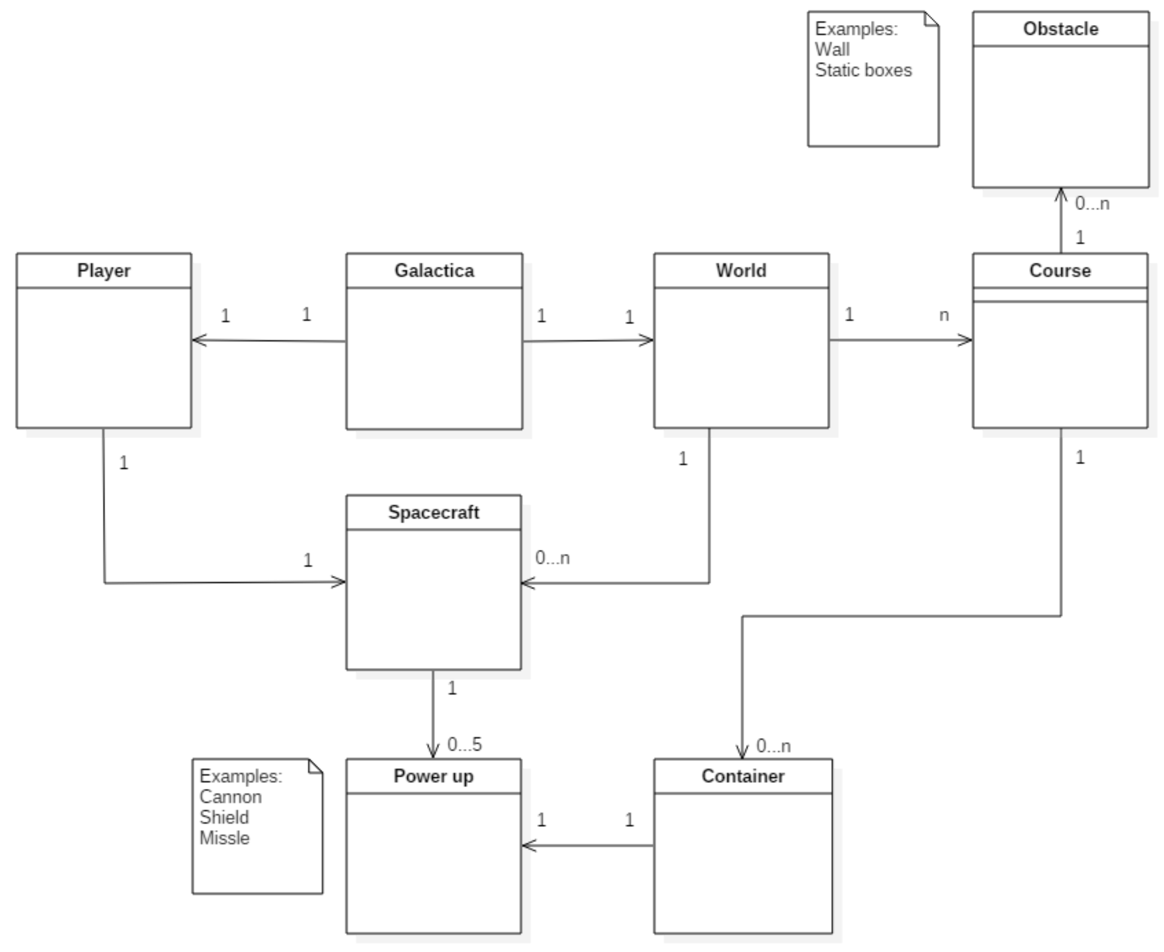
Summary: The user has died.
Priority: Mid
Extends:
Includes:
Participants: Actual player

Normal flow of events:
The player lost a game

	Actor	System
1		System turns screen black
2		System shows the High score list to the user
3		Screen turns black
4		see Start Menu

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

4. Domain model



A domain model with the central elements in the application. The model is representing data, the amount of each element and dependencies between elements.

Galactica is the central application, and is what holds everything together. It’s connected to the world that the player plays in.

Version: 3
Date: 2017-05-28
Author: Rasmus Lindgren, Markus Pettersson, Anthony Kalcic, Ludvig Andersson

4.1 Class responsibilities

“Player” user of the application.
“World” handles the tubular track that the spacecraft can fly in and all of the other objects that exists in it.
“Course” - The infinitely long tubular course that the player is confined to. Also holds obstacles and containers.
“Container” a 3d object that contains a power-up that the spacecraft can pick up.
“Obstacle” is a 3d object that will exist in the course that the spacecraft can collide with and lose health.
“Spacecraft” is a 3d object in the course that will be controlled by the player.
“Power up” is an ability that the player’s spacecraft can use once the spacecraft has picked one up from the containers.

5. References