

Requirements and Analysis Document

for



Version: 1.7

Date: 2017-05-29

Author: Philip Nord, Julia Ortheden, Jonas von Essen, Fredrik Viberg

This version overrides all previous versions.

1 Introduction

The purpose of the project is to create an entertaining arcade game in 3D. The game will be taking place in space and is therefore called *Not Enough Space*. It is divided into two separate parts, whereas the focus will be on finishing the first part and if there is time also develop a second part. The aim of the game is to collect, and keep, as many cows as possible which will generate points when the game is finished. Below follows a description of the basic functionality of the game.

1.1 General characteristics of the application

The application is aimed for PC and will be a cross-platform offline game for one player. It is developed for desktop use, standalone, with a graphical user interface and is runnable on Mac, Windows and Linux platforms. It will be made with an arcade game concept. Below follows a basic description of the game.

Part 1: Collect the cows

The player is in control of a spaceship orbiting the planet. On the planet there is cows walking around. By navigating above a cow and activating the tractor beam the cow starts to hover towards the ship where it will be added to the storage. To make it more difficult the player can happen to collect junk such as barns which will waste energy of the beam. It is also possible for the cows to flee from the spaceship.

The player controls the ship with four keys, one for each direction. Two keys for rotation and the blank space is used to activate the tractor beam. It will be a time constraint of 2 minutes for the player on the planet.

Part 2: Transport the cows (to be implemented)

The player is in control of the spaceship and will try to navigate through an asteroid field to reach the home planet without running out of life or losing all the cows. The spaceship can move in four directions and rotate to avoid the asteroids. The same keys as in part one will be used to control the spaceship. When colliding with an asteroid the spaceship will loose some cows, depending on the extent of the crash, and also the life bar will be reduced. If it is a massive crash the ship will explode and the game is over.

The weight of the storage will make the navigation of the ship harder by slowing it down. It is possible to lose weight by shooting out the last collected object from the storage towards the asteroids. It will be a visible storage of the last five objects collected at the side of the screen, and the cows will be included in the last collected objects.

1.2 Objectives and success criteria of the project

The following criterias should be fulfilled for the project to be considered successful:

- The spaceship should be able to navigate around the planet.
- The spaceship should be able to collect cows with the tractor beam, which will generate points.
- The spaceship should be able to collide with satellites and powerups that orbit the planet.

- When the time is up the game is over and a screen with high score will appear and it should be possible to restart or exit the game with the scores saved.
- The application should have a 3D-GUI and 3D navigation.

1.3 Definitions, acronyms and abbreviations

Round - one round inside the game.

Program - the application itself.

Beam - “Tractor beam”, the classic UFO abduction beam.

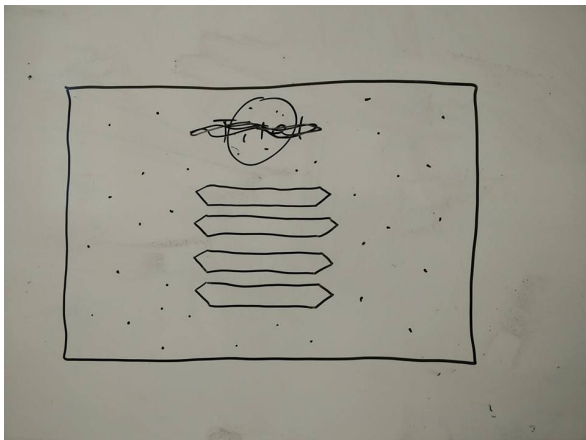
Powerup - Collectible item that charges the ship’s energy or restores health.

Phase/Part - used interchangeably and refer to the two different phases of the game.

2 Requirements

2.1 User interface

2.1.1 Main Menu



2.1.2 Gameplay



2.1.3 Pause Menu



2.1.4 Highscore View

2.2 Functional requirements

The player should be able to:

1. Start a new game
2. Play the game
 - a. Navigate the ship around the planet.
 - b. Collect cows and barns with the tractor beam.
 - c. Get hit by satellites and hay forks from the farmer.
 - i. Lose the game by reaching zero health.
 - d. Collect powerups that:
 - i. Restore health.
 - ii. Restore energy.
 - e. Open pause menu and choose by following:
 - i. Restart
 - ii. Return to the main menu.
 - iii. Resume game
3. See the high score screen when game is over and choose by following:
 - i. Play again
 - ii. Return to main menu
 - iii. Exit the game

In the main menu, it should be possible to:

1. Start a new round.
2. Exit the program.
3. Go to high score screen.

In the second phase, it should be possible to:

1. *Navigate the ship to avoid the incoming asteroids.*
2. *Hit asteroids and take damage or lose the game.*

List of Use Cases, ordered by priority:

Use cases followed by a number indicates that the use case is for the corresponding phase.

Phase 1

Phase 2

Pause menu

Main menu

1. Navigate
2. Activate beam
3. Deactivate beam
4. Beam (continuous)
5. Finish round
6. Collide with satellite
7. Chase cow
8. Start round
9. Exit program

10. Pause round
11. Resume round
12. Quit round
13. Restart round
14. Get chased by farmer
15. Get attacked with hayfork
16. Pick up powerup
17. Crash
18. Lose round
19. *Navigate 2*
20. *Finish round 2*
21. *Phase transition*
22. *Shoot*
23. *Wormhole*
24. *Boost*

2.3 Non-functional requirements

Usability

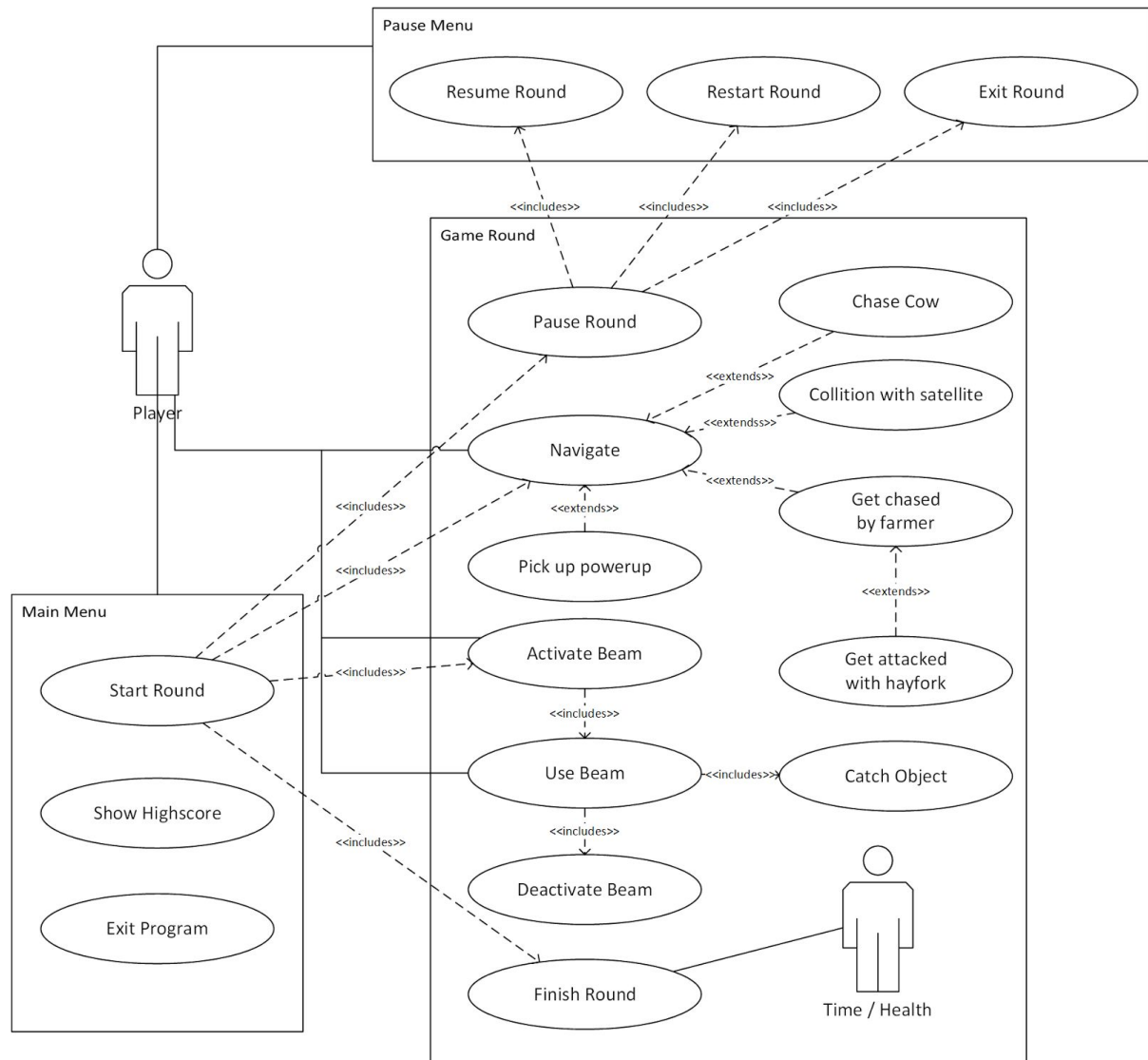
It should be very easy to understand what to do and how to play with just basic computer knowledge. No explicit manual should be necessary, but a non-intrusive instruction display may be considered. The time between running the application and starting a new round should be minimal.

Technical

To maintain platform independence and to minimize backend programming, the application will be written in Java using the 3D game engine JMonkeyEngine. The final product will be shipped as a .jar containing the class files, resources and necessary .dll-files.

The game mechanics are designed for a 3D view. This means that, while the model should be separate from the view, no effort will be made to keep the model usable in other environments such as 2D or text-based views.

3 Use cases



3.1 Implemented use case listing

Use Case: Navigate

Summary: The actor is navigating her ship in phase 1.

Priority: high

Extends: Start Round

Includes: -

Participants: Actor, System

NOTE: The keys called “the arrow keys” below don’t have to be the actual arrow keys of the keyboard, but will correspond to the usual purpose of these.

Normal flow of events

Actor moves the ship sideways/back- and forwards with the arrow keys. (The movement described is relative to the surface of the planet, so each movement is actually along a sphere outside of the planet's surface.)

	Actor	System
1	Presses one or more of the arrow keys.	
2		Ship object is moved in the (possibly combined) direction of the arrow(s) that is(/are) pressed. It accelerates quickly up to top speed. The ship continues to move as long as the user keeps the key(s) pressed.

Normal flow of events 2

Actor stops moving the ship by releasing the arrow key(s).

	Actor	System
1	Releases all the arrow keys.	
2		Ship object slows down to a complete halt.

Normal flow of events 3

Actor rotates the ship (around a local y-axis) with the rotation keys.

	Actor	System
1	Presses one of the rotation keys (LEFT or RIGHT).	
2		Ship object is rotated in the direction corresponding to the users key press. The rotation speed has a very quick acceleration phase until it reaches a constant level. The ship continues to rotate as long as the user keeps the key pressed.

Normal flow of events 4

Actor stops rotation of the ship by releasing the currently pressed rotation key.

	Actor	System
1	Releases the rotation key.	

2		The ship object's rotation decelerates quickly until it comes to a complete stop.
---	--	---

Use Case: Activate beam

Priority: High

Extends: Start round

Includes: Use beam

Participants: Actor, System

Normal flow of events

	Player	System
1	Presses spacebar	
2		Starts playing "sonic" sound and beam becomes visible
		See Use beam

Use Case: Deactivate beam

Priority: High

Extends: Start round

Includes: Use beam

Participants: Actor, System

Normal flow of events

Empty beam is deactivated

	Player	System
1	Releases spacebar	
2		Stops playing "sonic" sound and beam turns invisible.
3		Checks for any object still hanging in mid-air.
		No objects in mid-air.
4		Beam energy level starts to increase with constant speed. The on-screen energy meter is adjusted accordingly.

Alternate flow:

Beam carrying objects is deactivated

	Player	System
1	Releases spacebar	
2		Stops playing "sonic" sound and beam turns invisible.
3		Checks for any object still hanging in mid-air.

		Objects in mid air is released and fall to ground with realistic gravity. As it falls it gradually regains its original size.
4		Beam energy level starts to increase with constant speed. The on-screen energy meter is adjusted accordingly.

Use Case: Use Beam (continuous)

Summary: The user presses spacebar to beam cows below the spaceship

Priority: High

Extends: Start Round

Includes: Activate beam, Deactivate beam

Participants: Actor, System

Normal flow of events:

The beam is active while no object is in range and beam energy does not run out.

	Actor	System
1	Activate beam	
2		Check for objects below spaceship within reach of beam.
3		No object found. Beam still active.
4		Beam energy is reduced by an appropriate amount. The on-screen energy meter is adjusted accordingly. Energy is not yet empty.

Alternate flow:

The beam is active while no object is in range and beam energy runs out.

	Actor	System
1	Activate beam	
2		Check for objects below spaceship within reach of beam
3		No object found. Beam still active.
4		Beam energy is reduced by an appropriate amount and reaches zero. The on-screen energy meter is adjusted accordingly.
5		Beam is deactivated regardless of whether player is holding down the spacebar or not. See Deactivate beam

Alternate flow:

The beam is active while object is in range, object is picked up. Beam energy does not run out.

	Actor	System
1	Activate beam	
2		Check for objects below spaceship within reach of beam
3		Object within range of beam is slowly lifted towards spaceship. Its size lessens relative to its height above the planet's surface.
4		Object reaches spaceship and is added to "storage". HUD is updated with new values of items and total weight. Beam still active.
5		Beam energy is reduced by an appropriate amount. The on-screen energy meter is adjusted accordingly. Energy is not yet empty.

Alternate flow:

The beam is active while object is in range, object is picked up. Beam energy runs out.

	Actor	System
1	Activate beam	
2		Check for objects below spaceship within reach of beam
3		Object within range of beam is slowly lifted towards spaceship. Its size lessens relative to its height above the planet's surface.
4		Object reaches spaceship and is added to "storage". HUD is updated with new values of items and total weight. A "ship swallowing animation" is shown and a rewarding sound played. Beam still active.
5		Beam energy is reduced by an appropriate amount and reaches zero. The on-screen energy meter is adjusted accordingly.
6		Beam is deactivated regardless of whether player is holding down the spacebar or not. See Deactivate beam

Use Case: Collision with satellite

Summary: The user's ship collides with a satellite.

Priority: low

Extends: Navigate

Includes: -

Participants: System

Normal flow of events

	Actor	System
1		The satellite is colliding with the ship.
2		The satellite explodes. It disappears from the world and an explosion animation is played where it previously was. An explosion sound is played as well.
3		The ship's health meter is reduced by a certain amount.

Use Case: Pick up powerup

Summary: The user's ship picks up with a powerup.

Priority: low

Extends: Navigate

Includes: -

Participants: System

Normal flow of events

	Actor	System
1		The powerup is colliding with the ship.
2		The powerup disappears from the world and a powerup sound is played.
3		The ship's health or energy meter, corresponding to the powerup type, is increased by a certain amount.

Use Case: Chase cow

Summary: The user's ship get's close to a cow which makes it react in a scared manner.

Priority: mid

Extends: Navigate

Includes: -

Participants: Actor, System

Normal flow of events

(The cow's stamina is above a threshold level and the ship chases it until the level reaches zero.)

	Actor	System
1	Steers the ship close to a cow	

2		A moo sound is played and the cow speeds up, running in a way that takes it further from the ship.
3	Continues to steer the ship close to the cow	The cows stamina is constantly going down (this is not visible in the game).
4		After a certain amount of time the cow's stamina reaches zero and the cow slows down to normal speed again and walks as it did before, not avoiding the ship.
5		The cow's stamina starts rising again. When it gets above the threshold level the use case starts over.

Alternate flow of events 1

(The cow's stamina is above a threshold level and the ship chases it briefly.)

	Actor	System
1	Steers the ship close to a cow	
2		A moo sound is played and the cow speeds up, running in a way that takes it further from the ship.
3	Continues to steer the ship close to the cow	The cows stamina is constantly going down (this is not visible in the game).
4	Steers the ship away from the cow or slows down so that the cow is further from the ship again	
5		The cow slows down to normal speed again and walks as it did before.

Alternate flow of events 2

(The cow's stamina is below a threshold level and the ship chases it.)

	Actor	System
1	Steers the ship close to a cow	
2		The cow continues just as before (i.e. does not react to the ship getting close).

Use Case: Finish round

Priority: Mid

Extends: Start round

Includes: -

Participants: System

Normal flow of events

	Actor	System
1		Time reaches zero or ship's health bar reaches zero.
2		Navigation and beam is deactivated. Music stops playing.
3		Score is displayed on High-Score list and endgame GUI is presented. (Replay, quit)

Use Case: Start round

Summary: The user clicks the "Start" button in the main menu.

Priority: mid

Extends: -

Includes: -

Participants: Actor, System

Normal flow of events

	Actor	System
1	Clicks "Start" button.	
2		Main menu is removed and stage 1 is displayed.
3		Catchy music starts playing.
4		Timer starts and actor gains control of the ship.

Use Case: Quit program

Summary: The user clicks the "Quit" button in the main menu.

Priority: mid

Extends: -

Includes: -

Participants: Actor, System

Normal flow of events

	Actor	System
1	Clicks "Quit" button.	

2		Program shuts down.
---	--	---------------------

Use Case: Pause round

Summary: The user is in-game and presses the pause key.

Priority: mid

Extends: -

Includes: -

Participants: Actor, System

Normal flow of events

	Actor	System
1	Presses pause key.	
2		Actor loses control, timer pauses and AI/Physics are stopped.
3		Pause menu is presented.

Use case: Resume round

Summary: The player chooses to resume the game after pausing

Priority: middle

Extends: Start round, Pause round

Includes: -

Participants: Actor, System

Normal flow of events:

	Actor	System
1	Player presses "resume game"	
2		The game screen where the player paused the game appears
3		The game resumes from where it was paused

Use case: Exit round

Summary: The player chooses to exit the game in the middle of a round

Priority: High

Extends: Start round, Pause round

Includes: -

Participants: Actor, System

Normal flow of events:

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

1	The player presses exit key	
2		A menu becomes visible with the alternative “exit, resume game, restart game”.
3	Player presses exit	
4		Game exits and the last screen with high score and option to restart appears

Alternate flow:

	Actor	System
1	The player presses exit key	
2		A menu becomes visible with the alternative “exit, resume game, restart game”
3	Player presses a button that is not the exit button	
	”	See use case for resume game or restart game.

Use case: Restart round

Summary: The player chooses to restart the game from the beginning

Priority: Middle

Extends: Start round, Pause round, Exit round

Includes: -

Participants: Actor, System

Normal flow of events:

	Actor	System
1	Player presses “Restart”	
2		The start screen of the game appears
3		The game starts

Use Case: Get chased by farmer

Summary: The user's ship get's close to the farmer which makes him react by chasing the ship and throwing hayforks at it.

Priority: low

Extends: Navigate

Includes: Get attacked with hayfork

Participants: Actor, System

Normal flow of events

	Actor	System
1	Steers the ship close to the farmer	
2		The farmer starts chasing the ship by constantly running towards it. At random intervals he throws a hayfork at the ship (see Get attacked with hayfork). A constant muttering sound is played. This sound is stronger the closer the farmer is to the ship.
3		When the farmer is very close to the ship he constantly runs in circles below it. This continues as long as the ship is close to the farmer.
4	The actor manages to steer the ship a certain distance away from the farmer.	
5		The farmer stops chasing the ship and goes back to randomly strolling around.

Use Case: Get attacked with hayfork

Summary: The farmer is throwing a hayfork at the user's ship.

Priority: low

Extends: Get chased by farmer

Includes: -

Participants: (Actor), System

Normal flow of events

(The hayfork hits the ship which has enough health left for it not to reach zero.)

	Actor	System
1		The farmer throws a hayfork in the direction of the ship. A

		throwing sound is played.
2		The hayfork collides with the ship. A collision sound is played.
3		The ship's health bar is reduced by a certain amount, but is still above zero.

Alternate flow of events 1

(The hayfork hits the ship and makes its health reach zero.)

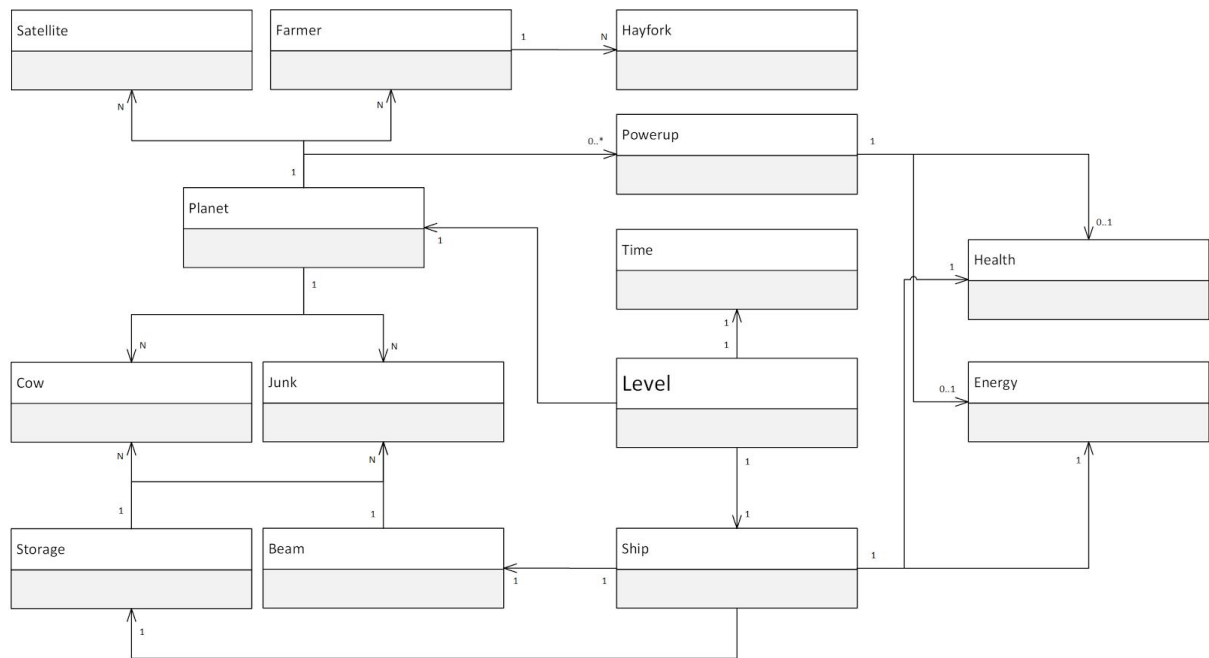
	Actor	System
1		The farmer throws a hayfork in the direction of the ship. A throwing sound is played.
2		The hayfork collides with the ship. A collision sound is played.
3		The ship's health bar is reduced by a certain amount, which makes it reach zero.
4		Finish Round

Alternate flow of events 2

(The hayfork misses the ship.)

	Actor	System
1		The farmer throws a hayfork in the direction of the ship. A throwing sound is played.
2	Steers the ship so that it avoids the hayfork.	
3		The hayfork continues into outer space.

4 Domain model



4.1 Class responsibilities

Level: Represents a game round. Responsible for creating the ship and the planet. Keeps track of score and ends the game when the time is up.

Ship: The UFO model that the player controls. Responsible for being moved by the player and activating/deactivating the beam. Keeps track of its own health and energy levels, as well as a storage.

Storage: The UFOs internal storage. Keeps track of stored objects.

Beam: The UFOs tractor beam. Drags cows and junk towards the ship when they're inside the beam.

Planet: The planet that the ship is navigating around. Contains a list of beamables, i.e. cows and junk for the ship to beam up. Responsible for spawning new beamables and satellites continuously. Also keeps track of the farmer enemy.

Cow: Cow model responsible for walking randomly around the planet and to some extent try to avoid the ship. Cows can be beamed by the ship.

Junk: Like non-moving cows (that don't give points).

Satellite: Satellite model that rotates around the planets atmosphere. Can collide with the ship, dealing damage.

Farmer: Angry farmer model that runs around the planet looking for the ship. If the ship is within a certain range, the farmer throws hayforks at it.

Hayfork: Hayfork model that is thrown by the farmer. Can collide with the ship, dealing damage and sticking to the ship model.

Powerup: Powerups are responsible for keeping track of their location and powerup type, each powerup can be of either health or energy type.

Health: Represents the fitness of the ship, which drops when hit by hayforks and satellites and rises when a health powerup is picked up.

Energy: Represents the resource used to power beam. Is consumed constantly when beam is activated and recharges 2x slower than it is consumed. Energy can also be recharges quickly by picking up a powerup of the energy type.

5 References

TVTropes (2017). *Aliens Steal Cattle*. Accessed from <http://tvtropes.org/pmwiki/pmwiki.php/Main/AliensStealCattle> at 2017-05-26

APPENDIX

Unimplemented Use Case Listing (Phase 2)

Use Case: Navigate 2

Summary: The actor is navigating her ship in phase 2.

Priority: low

Extends: -

Includes: Crash

Participants: Actor, System

NOTE: The keys called “the arrow keys” below don’t have to be the actual arrow keys of the keyboard, but will correspond to the usual purpose of these.

Normal flow of events

Actor moves the ship sideways/up- and downwards with the arrow keys.

	Actor	System
1	Presses one or more of the arrow keys.	
2		Ship object is moved in the (possibly combined) direction of the arrow(s) that is(/are) pressed. It accelerates quickly up to top speed. The ship continues to move as long as the user keeps the key(s) pressed. While moving a gentle swoosh sound is constantly played.

Normal flow of events 2

Actor stops moving the ship by releasing the arrow key(s).

	Actor	System
1	Releases all the arrow keys.	
2		Ship object slows down to a complete halt.

Normal flow of events 3

Actor rotates the ship (around a local z-axis) with the rotation keys.

	Actor	System
1	Presses one of the rotation keys (LEFT or RIGHT).	
2		Ship object is rotated in the direction corresponding to the users key press. It continues to rotate as long as the user keeps the key pressed. While rotating a swoosh sound (different to that playing while moving as described above) plays.

Normal flow of events 4

Actor stops rotation of the ship by releasing the currently pressed rotation key.

	Actor	System
1	Releases the rotation key.	
2		The ship object's rotation decelerates quickly until it comes to a complete stop.

Alternate flow of events

The ship's movement results in it hitting an asteroid.

	Actor	System
1	Moves the ship so that it hits an asteroid.	
2		Crash

Alternate flow of events

The ship's movement results in it hitting a powerup.

	Actor	System
1	Moves the ship so that it hits a powerup.	
2		Activate powerup

Use Case: Crash

Summary: The user's ship hits an asteroid in phase 2.

Priority: low

Extends: Navigate 2

Includes: Lose round

Participants: System

Normal flow of events

Ship collides with the asteroid head-on.

	Actor	System
1		Actor loses control of the ship.
2		Ship object is removed and explosion animation plays.
3		Lose round

Alternate flow of events

Ship collides with the asteroid from the side, takes damage but doesn't run out of health.

	Actor	System
1		Health bar goes down slightly and becomes red. Fades back to normal color over 1 second.
2		Cow counter goes down by a small number, but doesn't go below 0. Number becomes white and fades back to normal color over 1 second.

Alternate flow of events 2

Ship collides with the asteroid from the side, takes damage and runs out of health.

	Actor	System
1		Health bar goes down to 0.
2		Actor loses control of ship.
3		Ship object is removed and explosion animation plays.
4		Lose round

Use case: Lose round

Summary: The player runs out of life and loses the game

Priority: low

Extends: Start game, Crash?

Includes: -

Participants: System

Normal flow of events:

	Actor	System
1		Lifebar turns red and start blinking to signal that the player is about to run out of life
2	The player touches/ crashes into an asteroid	
3		Life bar gets empty
4		Dialogue pops up saying: "Game over"
		The exit-screen appears with the high score and the alternatives: "Restart" and "Exit" game"

Use case: Activate powerup

Summary: Ship collides with a powerup in phase 2.

Priority: low

Extends: Navigate 2

Includes: -

Participants: System

Normal flow of events:

	Actor	System
1		"Powerup" sound plays.
2		Specific powerup effect is activated.
3		Specific powerup visuals are shown on ship.

Use Case: Finish round 2

Summary: The user's ship successfully gets to the end of phase 2 and releases the cows on its home planet.

Priority: low

Extends: -

Includes: High Score list

Participants: System

Normal flow of events

Ship reaches the end of the asteroid belt with at least one cow and releases it's goods onto its home planet.

	Actor	System
1		Ship reaches the end of the asteroid belt.
2		Winning animation is played (something like: new view where the ship hovers over its home planet, opens its hatch and releases all of its cows onto an already huge pile of cows on the surface).
3		Highscore list

Alternate flow of events

Ship reaches the end of the asteroid belt with no cows left, gets destroyed and loses the game.

	Actor	System
1		Ship reaches the end of the asteroid belt.
2		Losing animation is played (something like: new view where the ship hovers over its home planet, opens its hatch to release some cows but doesn't have anything to release and then gets destroyed by a destroyer beam from the very angry control tower).
3		Losing screen is shown telling the user that she lost and asking her if she wants to try again or go back to the main menu.

Use case: Phase Transition

Summary: Stage 1 timer reaches 0:00

Priority: low

Extends: Start round

Includes: -

Participants: System?

Normal flow of events:

	Actor	System
1		Actor loses control of ship.
2		"Alert" sound plays.
3		Ship moves and rotates in an arc and ends up facing the asteroid field. Camera follows normally.
4		Actor regains control of ship.

Use Case: Shoot

Summary: The user's ship shoots some of it's goods out in phase 2.

Priority: low

Extends: Navigate 2

Includes: -

Participants: Actor, System

Normal flow of events

Ship shoots out it's most recently gathered item and it hits an asteroid..

	Actor	System
1	Presses shoot button.	
2		The item on top of the UI item stack gets bigger and then shrinks to nothing to signal that it no longer is on the stack. All the other items get moved up one step.
3		The item gets shot out from the front of the ship, moving forward with great speed. An appropriate sound (e.g. a "MOOO" if the item is a cow) is played.
4		The item hits an asteroid. Both the item and the asteroid get destroyed. (Visualised as the object disappearing in a small explosion while the asteroid gets divided into many small stones scattering in all directions.)

Alternate flow of events

Ship shoots out it's most recently gathered item and it doesn't hit anything..

	Actor	System
1	Presses shoot button.	
2		The item on top of the UI item stack gets bigger and then shrinks to nothing to signal that it no longer is on the stack. All the other items get moved up one step.
3		The item gets shot out from the front of the ship, moving forward with great speed. An appropriate sound (e.g. a "MOOO" if the item is a cow) is played.
4		No asteroid is in the path of the item, so it just gradually fades away in the distance.

Exceptional flow of events

Ship tries to shoot out it's most recently gathered item but doesn't have any items stored.

	Actor	System
1	Presses shoot button.	
2		A sound indicating an empty storage is played.

Use Case: Wormhole

Summary: The user's ship travels through a wormhole during phase 2.

Priority: low

Extends: Navigate 2

Includes: -

Participants: Actor, System

Normal flow of events

The ship enters a wormhole and gets teleported to the corresponding "out" wormhole.

	Actor	System
1	The actor navigates the ship so that it enters a wormhole.	
2		The ship disappears into the wormhole while a teleporting sound is played.
3		The whole view very quickly moves to the corresponding "out" wormhole and the ship reappears there (coming out of this wormhole).
4		Navigation

Use Case: Boost

Priority: Low

Extends: Navigate 2

Includes: -

Participants: Actor, System

Normal flow of events

	Player	System
1	Presses boost button	

2		Visually reduces the fuel meter by a certain amount. (The level is seen dropping.)
		Plays swoosh sound and shows fire animation (from the ship's rear) while making the ship move forward at a higher speed than normally for a few seconds.

Alternate flow of events (no fuel left)

	Player	System
1	Presses boost button	
2		Shows the fuel meter blinking in red, signalling that there is no fuel left.