Requirements and Analysis Document for Not enough space, 20.

Version: 1.1
Date: 2017-03-30
Author: Philip Nord, Julia Ortheden, Jonas von Essen, Fredrik Viberg

This version overrides all previous versions.

1 Introduction

The project is a prototype of a computer based arcade game called Not Enough Space. The main purpose of the game will be entertainment and it is created for everyone that needs some more space action in their lives. The game 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 much cows as possible which will generate points when the game is finished. Below follows a description of the basic function 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 over a cow and activating the tractor beam the cow starts to hover towards the ship where it will be added to the storage. As a difficulty the player can happen to collect junk such as barns, trees and stones which will make the spaceship heavier and harder to navigate in the second part of the game. It is also possible for the cows to move away from the spaceship and possibly hide in the barns.

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 about 2 minutes for the player on the planet.

Part 2: Transport the cows
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 harder by slowing it down. It is possible to lose weight by shooting 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.

The application will be made for desktop use, standalone, and single player mode with a graphical user interface for Mac/Windows/Linux platforms. To access further rules, definitions and terms of the game see references below.

Some general characteristics:

- The first part of the game will have a time constraint of about 2 minutes.

- When the time is up the game is over and a screen with high score will appear. *(Or you will automatically be directed to the asteroid field where the second part of the game starts.)*
- *(The application will end when the user has reached the home planet, the life bar is empty or possibly when the game is canceled.)*
- The spaceship will collect the cows by hovering over them and activating the tractor beam.
- The application will have a 3D-GUI and 3D navigation.

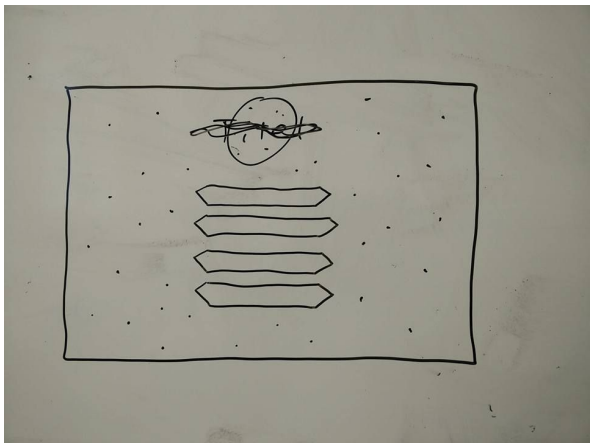
1.2 Definitions, acronyms and abbreviations

Spel - en runda i spelet.
Program - själva programmet.
Fas/steg - används synonymt och syftar på första och andra delen av spelförloppet, under en runda.
Traktorstråle - “Tractor beam” från engelska, alltså en klassisk UFO-dragningsstråle som används för att kidnappa människor eller liknande.

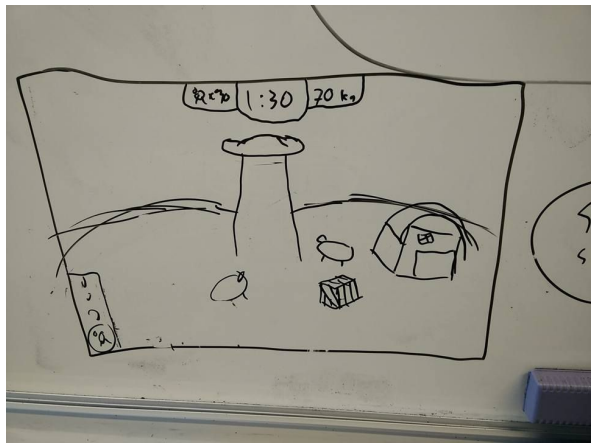
2 Requirements

2.1 User interface

Main Menu



Gameplay



2.2 Functional requirements

In the first phase of the game, it should be possible to:
- Navigate the ship around the planet.
- Beam up cows and junk.
- Pause and restart/exit the round.

In the second phase, it should be possible to:
- Navigate the ship to avoid the incoming asteroids.

- Hit asteroids and take damage or lose the game.

In the main menu, it should be possible to:

- Start a new round.
- Edit options.
- Exit the program.

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 1
2. Activate beam
3. Deactivate beam
4. Beam (continuous)
5. Finish round 1
6. Start round
7. Exit program
8. Edit options
9. Pause round
10. Resume round
11. Exit round
12. Restart round
13. Navigate 2
14. Crash
15. Lose round
16. Finish round 2
17. Phase transition
18. Shoot
19. Wormhole

2.3 Non-functional requirements

Usability

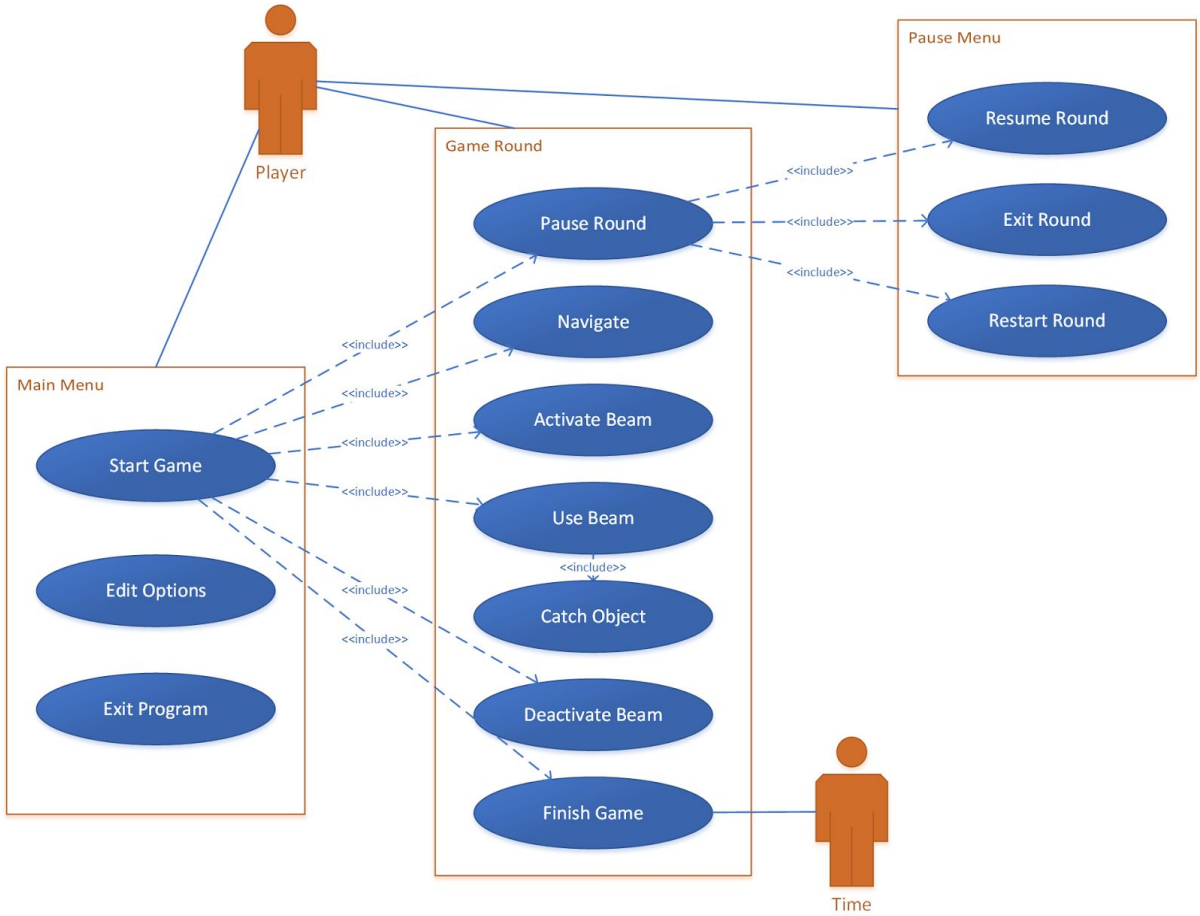
It should be very easy to understand what to do and how to play. No explicit manual should be necessary, but a non-intrusive instruction display may be considered. The time between running the application and starting a new round should be minimal.

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 Use case listing

Use Case: Navigate 1

Summary: The actor is navigating her ship in phase 1.
Priority: high
Extends: -
Includes: -
Participators: 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. The ship is also tilted in this direction. 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 buzz 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 tilts upright and 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. It continues to rotate as long as the user keeps the key pressed. While rotating a buzz 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.

Use Case: Activate beam

Priority: High
Extends: Start round
Includes: Use beam
Participators: Actual player

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
Participators: Actual player

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.

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

Use Case: Beam (continuous)

Summary: The user presses spacebar to beam cows below the spaceships
Priority: High
Extends: Start Round
Includes: Activate beam, Deactivate beam
Participators: Actual player

Normal flow of events:
The beam is activated while no object is in range.

	Actor	System
1	Activate beam	
2		Check for objects below spaceship within reach of beam
3		No object found. Beam still active.

Alternate flow:
The beam is active while object is in range, object is picked up.

	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

4		Object reaches spaceship and is added to “storage”. HUD is updated with flashing new values of items and total weight. Beam still active.

Use Case: Finish round 1

Priority: Mid
Extends: Start round
Includes: -
Participators: System only

Normal flow of events

	Player	**System**
1		Time reaches zero.
2		Navigation and beam is deactivated.
3		Play “homeworld” animation where cows are piled on alien planet.
		Score is displayed on High-Score list and endgame GUI is presented. (Replay, options, exit)

Use Case: Start round

Summary: The user clicks the “Start” button in the main menu.
Priority: mid
Extends: -
Includes: -
Participators: User

Normal flow of events

	Actor	**System**
1	Clicks “Start” button.	
2		Main menu is removed and stage 1 is displayed.
3		Countdown?
4		Timer starts and actor gains control of the ship.

Use Case: Exit program

Summary: The user clicks the “Exit” button in the main menu.
Priority: mid
Extends: -
Includes: -
Participators: User

Normal flow of events

	Actor	**System**
1	Clicks “Exit” button.	
2		Program shuts down.

Use case: Edit options

Summary: The player chooses to change the settings in the game menu
Priority: low
Extends: Start round
Includes: All alternative use cases for game settings
Participators: System, Player

Normal flow of events:

	Actor	**System**
1	Player presses “Change settings”	
2		A menu with the alternative: “Change resolution”, “Change language”, “Turn off sound” etc. appears
3	The player presses “Return”	
4		Start menu screen re-appear

Use Case: Pause round

Summary: The user is in-game and presses the pause key (usually Esc).
Priority: mid
Extends: -
Includes: -
Participators: User

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 in the middle of the screen. Screen around the pause menu is dimmed.

Use case: Resume round

Summary: The player chooses to resume the game after pausing
Priority: middle
Extends: Start round, Pause round
Includes: -
Participators: System, Player

Normal flow of events:

	Actor	System
1	Player presses “resume game”	
2		The game screen where the player paused the game appears
3		Countdown: “3,2,1…”
4		The play 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: -
Participators: Player, System

Normal flow of events:

	Actor	System
1	The player presses exit or the cross in the top right corner	

2		A menu becomes visible with the alternative “exit, resume game, restart game”.
3	Player presses exit	
4		Dialogue saying “Are you sure you want to exit?”
5	Player presses “yes”	
6		Game exits and the last screen with high score and option to restart appears

Alternate flow:

	Actor	**System**
1	The player presses exit or the cross in the top right corner	
2		A menu becomes visible with the alternative “exit, resume game, restart game”.
3	Player presses exit	
4		Dialogue saying “Are you sure you want to exit?”
	Player presses “no”	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: -
Participators: System, Player

Normal flow of events:

	Actor	**System**
1	Player presses “Restart”	

2		Dialogue: “Are you sure you want to restart?”
3	Player presses “Yes”	
4		The start screen of the game appears
		Countdown: “3,2,1…”
		The game starts

Use Case: Navigate 2

Summary: The actor is navigating her ship in phase 2.
Priority: low
Extends: -
Includes: Crash
Participators: 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**

Use Case: Crash

Summary: The user's ship hits an asteroid in phase 2.
Priority: low
Extends: Navigate 2
Includes: Lose round
Participators: 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: -
Participators: 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: 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: Highscore list
Participators: 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: -
Participators: 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: -
Participators: 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

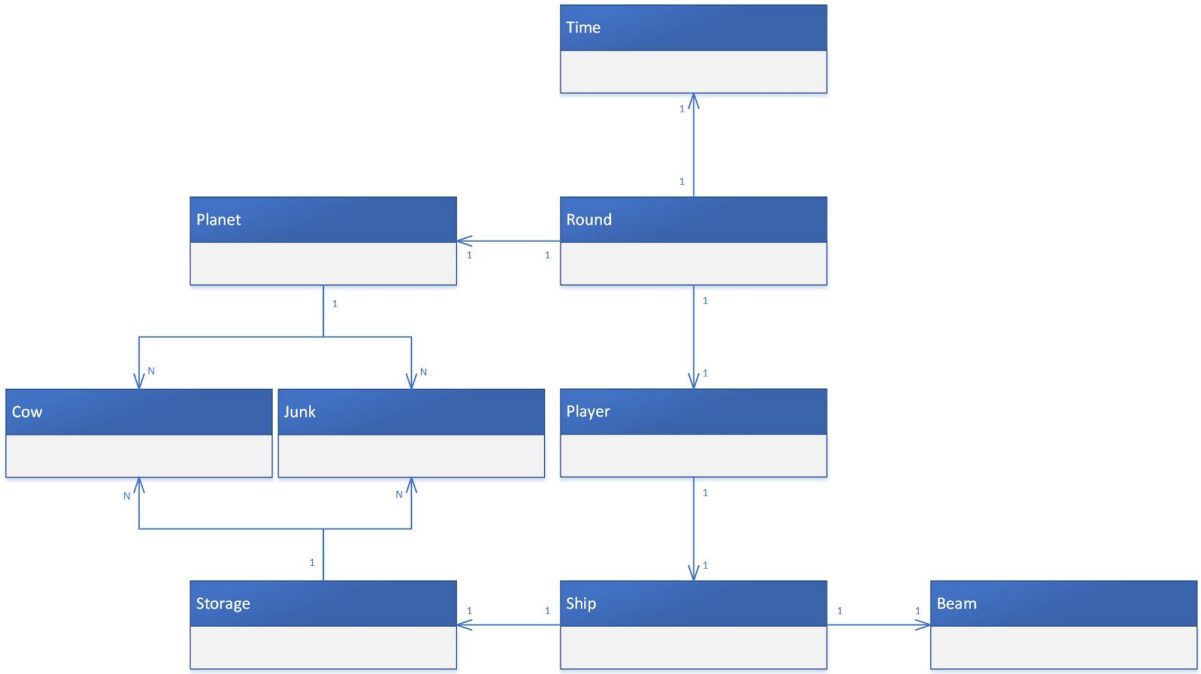
Summary: The user’s ship travels through a wormhole during phase 2.
Priority: low
Extends: Navigate 2
Includes: -
Participators: Actor, System

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**

4 Domain model



4.1 Class responsibilities

Game: The overall representation of the game as a whole.
Round: Represents a game round. Contains the ship, planet, cows etc. that make up the actual game.
Ship: The UFO model that the player controls. Responsible for being moved by the player and activating/deactivating the beam for collecting the cows.
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.
Cow: Cow model responsible for walking randomly around the planet (and maybe also to some extent trying to avoid the ship) and being able to be beamed by the ship.
Junk: Like non-moving cows (but which don't give points).

5 References