Requirements and Analysis Document for NUCLEUS

Version: 2.1
Date: 29/5 - 2016
Authors:
Miranda Bånnsgård, Andréas Erlandsson, Herman Carlström, Erik Jansson, Hannes Lagerroth

This version overrides all previous versions.

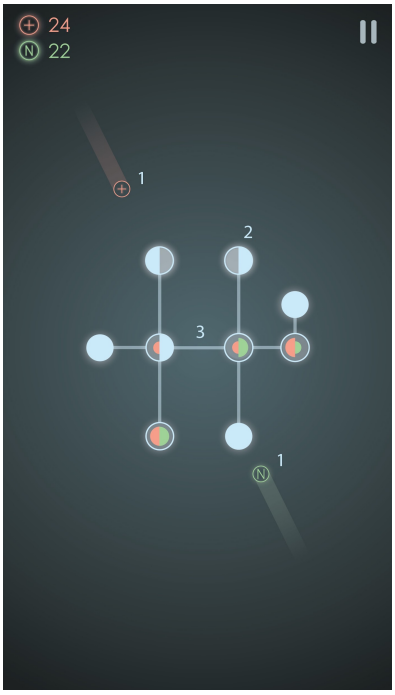
1 Introduction

Here follows a brief overview of the project, including purpose, general characteristics and definitions of the application.

1.1 Purpose of application

The project aims to create a mobile game application for android called Nucleus. The application will utilise a modular design allowing easy modification of gameplay mechanics and level maps.

1.2 General characteristics of application



The project will result in an android game application called Nucleus in which the player, in real time, attempts to catch flying nucleons[1] (protons and neutrons) in order to build atomic elements. A level consists of building a given number of atomic elements by catching an appropriate number of nucleons at a so-called gluon point[2]. These gluon points are fastened in a molecule arrangement[3] which is rotated by the player. Each level consists of a number of objective atoms which are to be built at the gluon points using a finite number of neutrons and protons. These nucleons are fired at the center at random from random points on the screen perimeter. The player loses should he or she overfill a gluon point or run out of protons or neutrons before the appropriate elements have been built.

1.3 Scope of application

The game is a single-player application which is based on level completion. If time allows, the level completion can be expanded to handle different tiers of completion.

1.4 Objectives and success criteria of the project

The main objective of the project is to get a functional android application. It should be possible to play the game with all basic criteria that we have defined. The project will include a desktop and an android application.

1.5 Definitions, acronyms and abbreviations

- **Nucleon** - an umbrella term for protons and neutrons. The building blocks of atomic nuclei.
- **Gluon point** - the attachment point on which the nucleons are attached.
- **Objective atom** - the atom which is to be assembled according to the level objectives
- **Molecule** - the center group of gluon points arranged in a molecule shape. This molecule is rotated to catch nucleons at the appropriate gluon points.
- **Android** - an operating system for smart phones. Uses JAVA for applications.
- **Java** - a platform independent programming language

2 Requirements

In this section we will specify all requirements for the game Nucleus.

2.1 Functional requirements

The player should be able to:

- Open the application on an android phone running at least Android 5.0
- Turn on/off sound effects
- Select level
- Start a game
    - Rotate Molecule
        - Catch nucleons.
        - Let nucleons pass through without catching them.
    - Pause a game
        - Continue a paused game.
        - End/Quit a paused game.
        - Restart a paused game.
    - Win the game
        - If you catch the correct amount of protons (the amount will be specified) and neutrons (also specified) on all of the gluon points and thereby creating the correct molecule the game will be won.
    - Lose the game
        - The amount of nucleons fired towards the molecule is limited, if the player runs out of nucleons before the molecule is built, the game is lost.
        - If you catch too many nucleons (the amount will be specified) on any of the gluon point, the game will be lost.
- End game
- Exit application

2.1.1 Possible future features

- High Score
- Get awards

- Alpha-, Beta- and Gamma-rays functioning as “power-ups” where if you get hit by one of them the difficulty of the game is amplified for a short period of time by for example:
    - Alpha-ray - make the disc not turn and the background with the incoming nucleons on them turn instead for 10 sec
    - Beta-ray - make the incoming nucleons shake and go in a non-linear projection for 10 seconds.
    - Gamma-ray - make all inputs inverted for 10 sec

2.2 Non-functional requirements

2.2.1 Usability

The game should

- reflect its simple game mechanics with a simple and clearly communicated user experience. A large number of in-game tips and tutorials should not be needed though some may be present.
- feel responsive to the player’s actions. Great care should be put into calibrating the rotation mechanic and buttons so that they feel fast.

2.2.2 Reliability

The game should

- crash a maximum of one time per ten players and hour.

2.2.3 Performance

The game should

- run at least 50 fps on most android phones compatible with Android 5.0.
- load all screens in at most 2 seconds.

2.2.4 Supportability

The game should

- handle aspect ratios compatible with all commonly found phone devices.
- utilize automated testing to control the quality of the game model and game logic.

2.2.5 Implementation

Java will be used to implement this application in order to achieve platform independence.

The game should

- be implemented in such a way that the game engine library used should be swappable to an arbitrary, comparable library.
- be implemented in such a way that adding additional levels is done not through hard-coding but through, for example, a text file, which the game uses to build the level at load time.

2.2.6 Packaging and installation

- Installed using Gradle. Currently no APK available.
- The finished prototype will be best adapted for desktop demoing and is built with gradle from the command line.
- A README-file contains information in how to run the application.

2.2.7 Legal

The library libGDX that we will be utilizing is under an Apache License.

2.3 Application models

2.3.1 Use case model

See APPENDIX

2.3.2 Use cases priority

1. Nucleus
2. PlayGame
3. BuildMolecule
4. CatchNucleon
5. MissNucleon
6. LoseGame
7. WinGame
8. PauseGame
9. RestartLevel
10. SetOptions
11. Exit
12. ToggleSoud
13. ReversRotation

2.3.3 Domain model

See APPENDIX

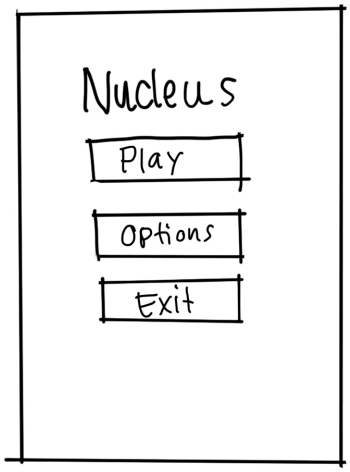
2.3.4 User interface

See APPENDIX

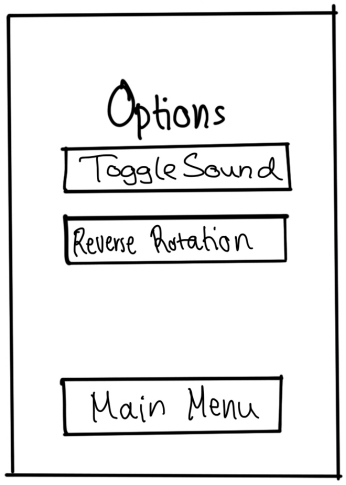
APPENDIX

GUI

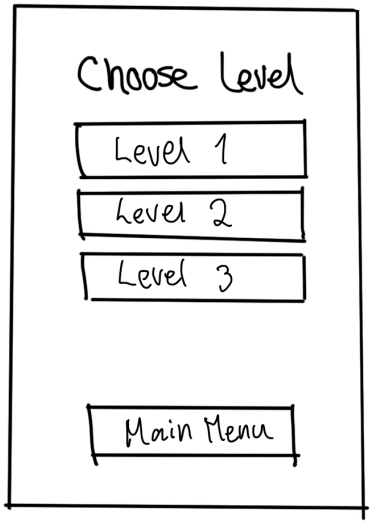
Main menu



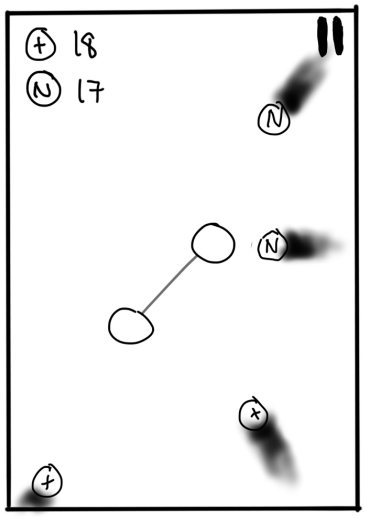
Options screen



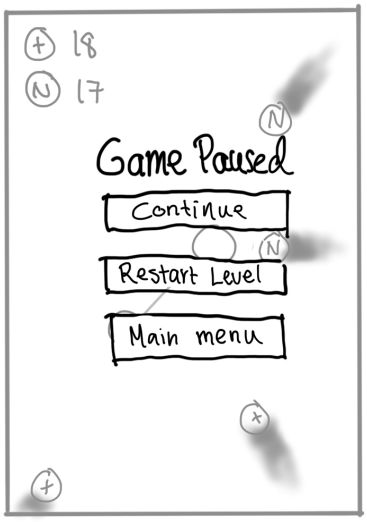
Choose Level menu



Game view



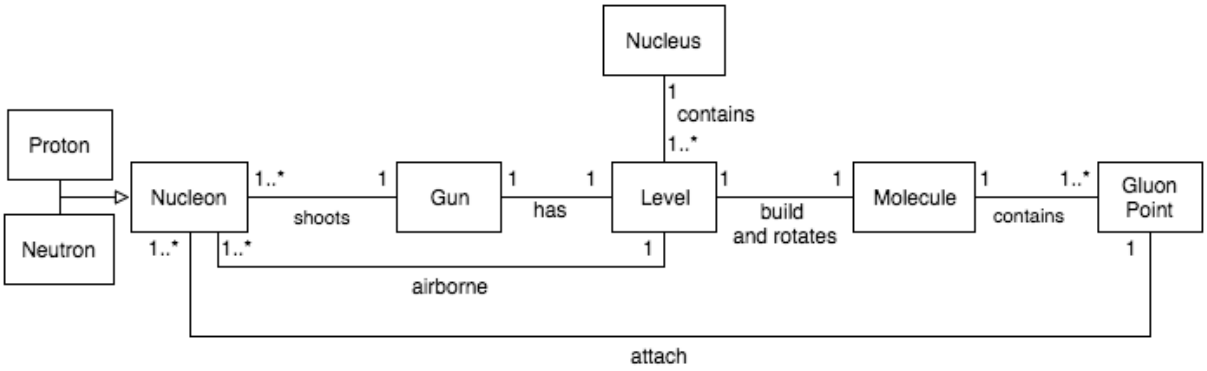
Game view with pause pop-up



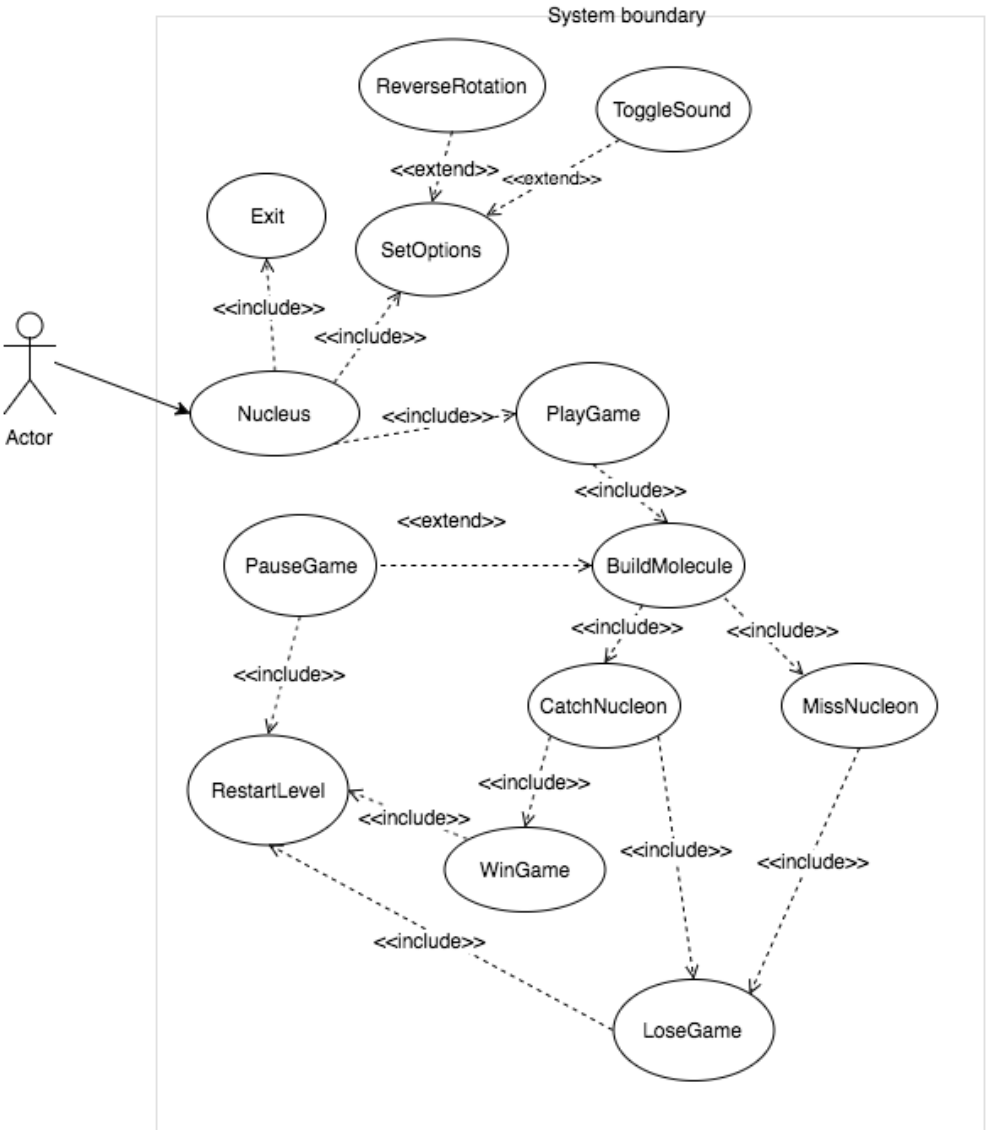
Game view with loss pop-up

Game view with win pop-up

Domain model



Use case model



1. Use Case: Nucleus

Summary: User opens the application Nucleus and the main menu is displayed.
Priority: High
Extends: NA
Included: SetOptions, Exit, PlayGame
Participator: Player

Main flow

The first step in order to play the game Nucleus

	Actor	System
1	Clicks on application icon	
2		Opens application and displays the main menu.

2. Use Case: Exit

Summary: The player chooses to exit the game.
Priority: Medium
Extends: NA
Included: NA
Participator: Player

Main flow

The player exits the game.

	Actor	System
1	Chooses to exit	
2		Application window is closed.

3. Use Case: SetOptions

Summary: Player wants to change the game settings and goes to options menu. Player returns then to main menu.
Priority: Medium
Extends: NA
Included: NA
Participator: Player

Main flow

Player chooses options and then returns to main menu.

	Actor	System
1	Chooses to go to options menu.	
2		Closes main menu.
3		Opens options menu.
3.1 Player reverse rotation	See Use Case: ReverseRotation	
3.2 Player toggle sound	See Use Case: ToggleSound	
4	Chooses to go back to main menu.	
5		Closes options menu.
6		Opens main menu.

4. Use Case: ReverseRotation

Summary: If the player would like to increase difficulty of the game the player can go to options and reverse the rotation of the game.
Priority: Low
Extends: SetOptions 3.1
Included: NA
Participator: Player

Main flow

Player chooses options and then to reverse the rotation in the game.

	Actor	System
1	Chooses to reverse rotation	
2		Reverses the rotation.

5. Use Case: ToggleSound

Summary: The player chooses to turn on/off the sound, depending of if it is on or off from before.
Priority: Medium
Extends: SetOptions 3.2
Included: NA
Participator: Player

Main flow

Player chooses toggle the sound.

	Actor	System
1	Chooses to toggle sound.	
2		Toggles the sound

6. Use Case: PlayGame

Summary: The player chooses to which level to play. Player can not choose a level that has not been unlocked. A level gets unlocked when preceding level is won.
Priority: High
Extends: NA
Included: BuildMolecule
Participator: Player

Main flow

Choose which level to play and the level comes up and is ready to play

	Actor	System
1	Chooses play the game.	
2		Closes main menu.
3		Opens choose level menu.
4	Chooses a level by clicking on a level button	
5		Closes choose level view
6		Opens the game view with chosen level set up.
7	See Use Case: BuildMolecule	

Alternate flows

Flow 3.1 - 5:

	Actor	System
3.1.1	Chooses to return to main menu.	
3.1.2		Closes the choose level view.
3.1.3		Opens the main menu.

Flow 4.1 Player clicks on an locked level

	Actor	System
4.1.1		Notifies that the player picked a locked level.

Flow 6.1 - 7: Entities not loaded

	Actor	System
6.1.1		Fails to load all needed entities for the game view.

7. Use Case: BuildMolecule

Summary: Nucleons (protons and neutrons) are shot from off-screen and player swipes across screen to rotate the molecule in order to catch the nucleons.
Priority: High
Extends: NA
Included: CatchNucleon, MissNucleon
Participator: Player

Main flow

Game is playing and nucleons (protons and neutrons) are shot from off-screen towards the molecule in the middle of screen.

	Actor	System
1		Shoots a nucleon (proton/neutron) from off-screen.
2		Updates the number of neutrons and protons left in level.
3	Swipes across screen.	
4		Rotates molecule in direction of swipe.
5		Nucleon interacts with gluonpoint of molecule
5.1 Nucleon is catched by gluonpoint	See Use Case: MissNucleon	
5.2 Nucleon is missed by gluonpoint	See Use Case: CatchNucleon	

8. Use Case: RestartLevel

Summary: The level is restarted.
Priority: High
Extends: NA
Included: NA
Participator: Player

Main flow

The level is restarted.

	Actor	System
1		Sets up the chosen level.

9. Use Case: PauseGame

Summary: The player chooses to pause the game. From here the player can choose to continue the paused game, restart the level or go back to the main menu.
Priority: Medium
Extends: BuildMolecule
Included: RestartLevel
Participator: Player

Main flow

Player pauses the game and then chooses to continue it.

	Actor	System
1	Player pauses the game	
2		Pauses the game. Shows a pop-up over the paused game.
3		Displays a pause pop-up. There are three options available: 1. Continus 2. Restart game 3. Main Menu
4	Player chooses to continue game	
5		Closes the pause pop-up
6		Continues the game.

Alternative Flow

Flow 3.1 - 6: Player chooses to restart game.

	Actor	System
3.1.1	Chooses to restart the level	
3.1.2		Closes the pause pop-up.
3.1.3	See Use Case: RestartLevel	

Flow 3.2 - 6: Player chooses to go back to the main menu.

	Actor	System
3.2.1	Chooses to go back to the main menu.	
3.2.2		Closes the pause pop-up and game view.
3.2.3		Opens the main menu.

10. Use Case: CatchNucleon

Summary: Gluon point catches nucleon shot from off-screen.
Priority: High
Extends: NA
Included: WinGame, LoseGame
Participator: Player

Main flow

Nucleon is caught by a non-full gluon point

	Actor	System
1		Nucleon collides with a gluonpoint.
2		Gluonpoint is not full. Nucleon becomes part of gluonpoint, this is visually indicated.
3		Continues to shoot nucleons from off-screen.

Alternative Flow

Flow 1.1 - 3: Nucleon is caught by a full gluon point

	Actor	System
1.1.1		Gluon point is full. Nucleon overfills the gluon point.
1.1.2	See Use Case: LoseGame	

Flow 1.2 - 2: Nucleon fills a gluon point

	Actor	System
1.2.1		Atomic element at gluonpoint is completely built. This is visiually indicated

Flow 1.3: Nucleon fills the last gluon point

	Actor	System
1.3.1		Atomic element at last gluonpoint is completely built.
1.3.2	See Use Case: WinGame	

11. Use Case: MissNucleon

Summary: The nucleon misses the gluon point in the molecule.
Priority: High
Extends: NA
Included: LoseGame
Participator: Player

Main flow

A nucleon flies pass the molecule and its gluon points

	Actor	System
1		Nucleon flies off screen

Alternative Flow

Flow 1.1: Last nucleon misses unfinished gluon points

	Actor	System
1		Nucleon flies off screen
2	See Use Case: LoseGame	

12. Use Case: LoseGame

Summary: The game is lost because of overfilled gluon point or that the player run out of nucleons.
Priority: High
Extends: NA
Included: RestartLevel
Participator: Player

Main flow

Player is informed that the level is lost and chooses to restart the level

	Actor	System
1		Displays a pop-up informing the player that the level is lost. Two options are displayed: 1. Play Again 2. Main Menu
2	Chooses to play again	
3		Closes the pop-up window.
4	See Use Case: RestartLevel	

Alternative Flow

Flow 2.1 - 4: Player chooses to go back to the main menu.

	Actor	System
2.1.1	Chooses to go back to the main menu.	
2.1.2		Closes the pop-up window along with the game view.
2.1.3		Opens the main menu

13. Use Case: WinGame

Summary: The game is won when all atomic elements are completed and together have formed a molecule.
Priority: High
Extends: NA
Included: RestartLevel
Participator: Player

Main flow

Player is informed that the level is won chooses to go back to main menu.

	Actor	System
1		Displays a pop-up informing the player that the level is won.
2		Two options are displayed: 1. Play Again 2. Main Menu
3	Chooses to go to main menu.	
4		Closes the pop-up window.
5		Opens the main menu.

Alternative Flow

Flow 3.1 - 5: Player chooses to play the level again.

	Actor	System
3.1.1	Chooses to play the level again.	
3.1.2		Closes the pop-up window.
3.1.3	See Use Case: RestartLevel	

Use Case: ColorblindMode (never implemented)

Summary: In order to see what nucleons that should be caught to complete level, there is a colorblind mode in Options.
Priority: Low
Extends: NA
Included: NA
Participator: Player

Main flow

Player chooses options and and then switches on colorblind mode.

	Actor	System
1	Chooses options.	
2		Chooses to switch to colorblind mode.
3		Changes the theme of the game to colorblind mode.

Sequence diagrams