NTNU – Trondheim
Norwegian University of
Science and Technology

TDT4240 Software Architecture

Group 2 - Requirements
Penguins in Space
Android

Christiansen, Sander Aker
Dahl, Lars-Kristian
Klaussen, Stein-Aage Nibe
Lundenes, Morten
Sjøen, Henry Skorpe
Zhou, Shelley Xianyu

Primary Quality Attribute: Modifiability
Secondary Quality Attribute: Usability, Performance and Interoperability

April 23, 2017

Contents

1 Introduction

1.1 Description of the project and this phase

This document will document a set of functional requirements and architecturally significant quality attributes for a multiplayer game targeted for Android. All documentation will be kept up to date, hence a list of changes is added to this document.
The first phase is about generating a set of requirements, from which architecturally significant requirements (ASRs) can be discovered. Once these are known, the architecture can be designed to support them as well as the chosen quality attributes.

1.2 Description of the game concept

The game is an Asteroids like game, with multiplayer functionality. Figure 2 shows a screenshot of the retro arcade game Asteroids. A lot of the game dynamics are borrowed from this game.

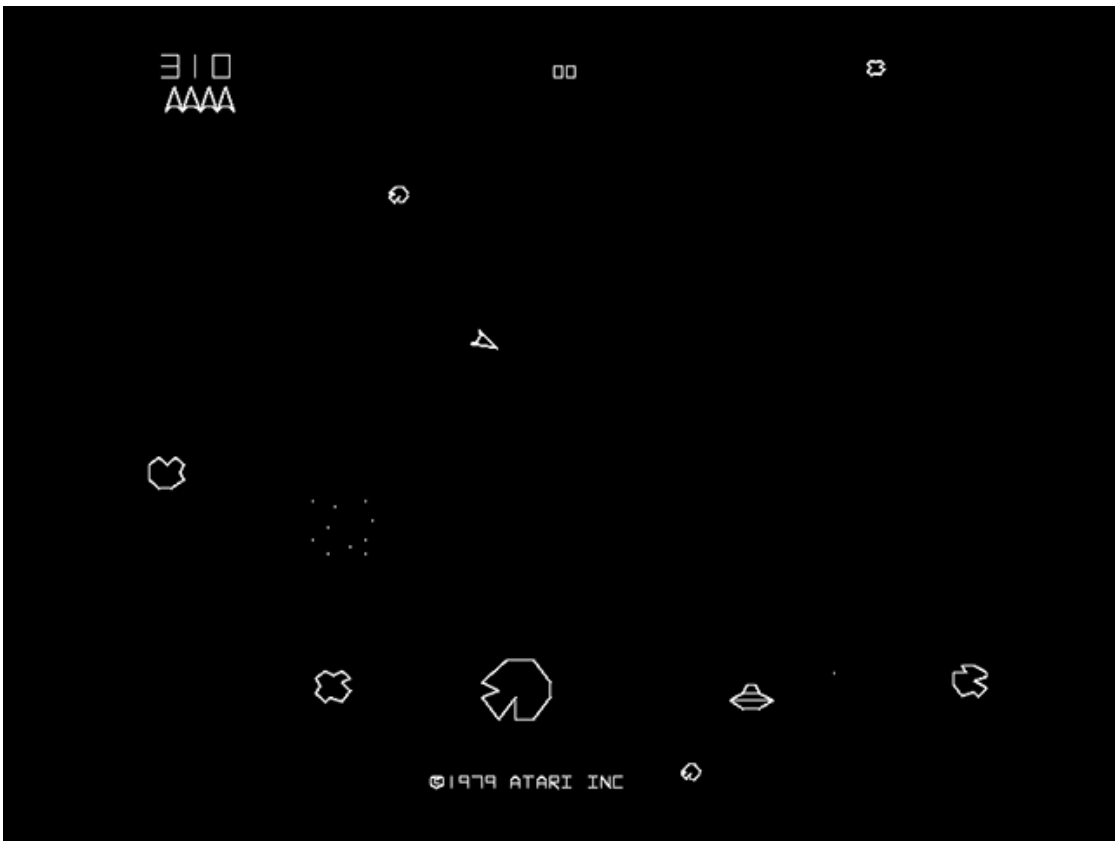

Figure 1: Asteroids, an arcade space shooter released in November 1979 by Atari, Inc.

The name of the game is "Penguins in Space".

You are a penguin floating in space. There are angry snowballs all around spawning randomly. The goal of the game is to stay alive for as long as you can, and get the highest score possible by either shooting snowballs in single-player mode or other penguins in multiplayer mode.

In singleplayer mode, power-ups will spawn throughout the game and give you abilities like dropping bombs or becoming invulnerable. The goal here is to survive and destroy as many snowballs as possible. Destroying a snowball increases ones score by one. The levels increase with your score, increasing the speed of the angry snowballs which makes it harder and harder. The goal is to achieve the highest score. In multiplayer mode there are several other players spawning that can attack each other. They will fight in a free-for-all until the last penguin is left standing.

The player controls the space-penguin using an analog stick in the lower left corner and shoots by pressing the button in the lower right corner. Below is Figure 2, a screenshot of the game in single-player mode.

Figure 2: Illustration of our game - Penguin in Space

1.3 Structure of the document

This document briefly describes this project phase and the game concept in Chapter 1, followed by high level functional requirements in Chapter 2. In Chapter 3, it describes

the quality requirements which are also reflected in game programming. COTS components and technical constraints are briefly described in Chapter 4. Issues and challenges faced during the project are described in Chapter 5 and log of changes made to this documentation are listed in Chapter 6. Finally a list of references related to the project are listed on the last page.
Notations used in this document, eg. QA scenario, follow recommendations in Reference [2].

2 Functional requirements

This section describes the main functions of our game.

Table 1: Functional requirements

ID	Requirement	Priority
FR1	The user should be able to control game with touch gestures	High
FR2	The user should be able to control its character with virtual joystick and virtual buttons	High
FR3	The user should be able to choose different game modes from the menu.	Medium
FR4	The user should be able to turn off sound and sound effects.	Low
FR5	The user should be able to quit to main menu while in-game.	Medium
FR6	The user should be able to quit the game from main menu.	Medium
FR7	The user should be able to pick different characters	Medium
FR8	The user should be able to play online multiplayer or singleplayer.	High
FR09	If the player gets shot by another player or enemy the player should lose health.	High
FR10	There should spawn different powerups during the game.	Medium
FR11	If the player collides with a powerup, the player should get that powerup.	High
FR12	The user should have an in-game menu.	Medium
FR12.1	The game should not pause when the in-game menu is chosen while playing in multiplayer mode.	High
FR12.2	The game should pause when the in-game menu is chosen while playing in singleplayer mode.	Medium
FR13	The game should show the player a score, based on a performance criteria.	Medium
FR14	The player has the option to go through tutorial when starting the game.	Medium
FR15	If the user collides with an angry snowball he loses hitpoints.	High
FR16	If the user loses his last hitpoint he dies.	High

3 Quality requirements

We have chosen modifiability as the primary software architecture quality attribute, performance, usability and interoperability as secondary quality attributes. Following

subclauses address the quality attributes individually through scenarios.

3.1 Modifiability

Table 2: Add a new power-up, playable character or enemy.

ID:	M1
Source:	Developer
Stimulus:	Wishes to add a new power-up, playable character or enemy
Artifacts:	Code
Environment:	Design Time
Response:	Changes made and Unit Tested, also play-tested in multiplayer by developers.
Response Measure:	In two hours.

Table 3: Modify existing graphics

ID:	M2
Source:	Developer
Stimulus:	Wishes to modify existing graphics
Artifacts:	Graphical resources
Environment:	Design Time
Response:	Graphics updated.
Response Measure:	The graphic is updated and seamlessly integrated into the game in half an hour.

Table 4: Modify a powerup

ID:	M3
Source:	Developer
Stimulus:	Wishes to modify an existing powerup
Artifacts:	Code
Environment:	Design Time
Response:	Changes made and Unit Tested, potentially play-tested in multiplayer by developers if relevant.
Response Measure:	In half an hour, integrated into the game without specific adjustments to other modules.

Table 5: Modify logic of a component

ID:	M4
Source:	Developer
Stimulus:	Wishes to modify an existing component's implementation
Artifacts:	Code
Environment:	Design Time
Response:	Changes made and Unit Tested, potentially play-tested in multiplayer by developers if relevant.
Response Measure:	In half an hour, without affecting the component's interface or other code.

3.2 Usability

Table 6: First-time play

ID:	U1
Source:	User
Stimulus:	Wants to play for the first time
Artifacts:	System
Environment:	Runtime
Response:	User is introduced to controls and mechanics within the game through a short and simple textual tutorial
Response Measure:	Finished the tutorial within 2 minutes and has a basic understanding of the game and its controls

Table 7: Modify game through settings

ID:	U2
Source:	User
Stimulus:	Wants to modify a setting under options (like volume of music or effects)
Artifacts:	System
Environment:	Runtime
Response:	User finds the setting and successfully modifies it
Response Measure:	Setting is found without any navigation-errors in 15 seconds or less

3.3 Interoperability

Table 8: Player wants to initiate a multiplayer room

ID:	I1
Source:	Player-client
Stimulus:	A request to set up a new player room
Artifacts:	Player-client and the Google Play services servers
Environment:	The client and servers are discovered at runtime
Response:	The request is accepted and a room specified by parameters is set up
Response Measure:	100% of the sent requests that are in the correct format and order are accepted

3.4 Performance

Table 9: Client receives data from server

ID:	P1
Source:	Server
Stimulus:	Server sends data of other players
Artifacts:	Player-client
Environment:	Normal Operation
Response:	Data is used to update the relevant players locally
Response Measure:	Latency below 30 ms for the update of local state

Table 10: Periodic client update

ID:	P2
Source:	Client
Stimulus:	Periodic update
Artifacts:	Player-client
Environment:	Normal Operation
Response:	All local data is used to update the state
Response Measure:	Throughput of 30 successful updates per second for steady frame rate

4 COTS components and technical constraints

4.1 Android devices and OS versions

Android is an OS for many different devices, mainly smartphones. These devices differ in many ways, among them are screen resolution and OS-version, which makes for technical constraints related to this game.
The difference in screen resolution makes the game have to support scaling the GUI. This also puts constraints on graphics and code related to the size of them.
As for the version of the Android the game has to either be made using only features supported by older versions, or simply not supporting devices with old OS-versions. After taking a look at Reference [1] the choice was made to support 4.1.x and above (JellyBean and up), as this makes up 98% of the user base.
Furthermore Android OS also puts constraints onto which version of Java we have to use. Currently the full Java 7 and a subset of 8.

4.2 LibGDX

LibGDX maintains a stack of "screens", introducing a constraint in terms of how our view will be represented from the MVC. Actually each screen has it's own controller and view by how they are defined in this framework, meaning that MVC will not be implementable on this lower level. Rather than updating a single screen from an underlying model, LibGDX has a set of different screens in a stack, with only the topmost being active. Therefore the architecture has to implement several Screens, all implementing the GDX. Screen-interface, specifically one for the main menu and one for playing the actual game.

4.3 Ashley ECS

Ashley requires Java version 6 or higher, this is not a problem as Android already has put this constraint on the game. It also restricts development to use Entity-Component rather than inheritance, having direct consequences for our architecture in terms of

components. Each entity, for instance a Penguin, consists of several components. Each component adds behavior or functionality to the entities using it. This means that a set of components have to be developed, used by the relevant entities, rather than the entities inheriting from superclasses. See the Architectural Document for more information on this.

4.4 Google Play Game Services SDK

This COTS requires the users of the applications to be connected to the Internet.
Furthermore it forces the game to adopt their p2p-mesh for connecting players to each other. To set up a room, a user has to interact with the Google Play services servers, the same goes for joining one of these rooms. This leads to that a Network Manager must be implemented to allow the server to communicate with each client. The rooms themselves also have a capacity of 8 players.

5 Issues

We decided to drop some of the functional requirements relating to black holes and increasing the size of penguins based on score as the gameplay was fine without them and they did not fit in after all.
We also decided to change the response measure of some of the QAs as we initially had no way of measuring them in the given way.
Getting used to Ashley ECS and LibGDX for the members never having used these before took some time, so most of the memebers had to read up to be able to contribute to the requirements-document, architectural document and actual implementation.

6 Changes

This section lists change history for this document.

Date	Change History	Comments
February 27, 2017	First released version	None
March 13, 2017	Changes to Quality Requirements and COTS/constraints	Updated M1, M2, U1, U2, I2 and P1 based on the first evaluation as well as added specific constraints for COTS
March 17, 2017	Chapter 'Changes' moved towards the end.	Updated based on feedback from teacher.
April 20, 2017	Additions and modifications of QAs related to modifiability and interoperability. Removing dropped FRs.	FRs that were dropped after discovering they did not fit with the gameplay. More QAs added to modifiability as it is our main QA. Changed around priorities of FRs as all cannot be high.

7 Bibliography

[1] Android Developers. Android dashboards - platform versions. `http://developer.android.com/about/dashboards/index.html`, 2016.

[2] Rick Kazman Len Bass, Paul Clements. *Software Architecture in Practice - Third Edition*. Addison-Wesley, 2012.