Castle Crush

Requirements

Nina Marie Wahl
Erik Kjernlie
Ludvig Lilleby Johansson
Truls Elgaaen
Jørgen Mortensen

COTS: **Android SDK**, **Google Play Services SDK** and **LibGDX**

Primary quality attribute: **Modifiability**
Secondary quality attribute: **Usability, Availability**

26th of February

1. Introduction

The goal of the project is to learn to design, evaluate, implement and test a software architecture through game development. We are supposed to develop a simple multiplayer smartphone game. We are supposed to focus on some specific quality attributes, and we have chosen modifiability (which all groups are supposed to), usability and availability.

In the requirement phase we are supposed to specify all the requirements for the project as clearly as possible. Both functional and quality requirements need to be in place, to be able to decide the architecture for the game. After this phase, all the group members should be aware of the requirement of the game.

This document contains a description of our functional requirements for the game, quality requirements, different quality scenarios as well as the COTS and the technical constraints. The document will be updated when changes are made, through the whole project.

The game is a turn-based multiplayer game. Although it is turn-based, the game will be running as real-time in online multiplayer mode. You can choose between several game modes, either offline against the computer or online versus friends. Each player has a castle of boxes protecting a game winning object, along with a shooting cannon.

When it is the current players turn, there is a moving pointer indicating the angle of the shot. The player touches the screen too choose an angle. The player touches the moving pointer again to decide the power of the shot. The moving pointer makes the shooting less precise, preventing repeating shots. The speed of the angle and the power can be modified to change the difficulty of the game.

If the bullet hits one of the boxes of the opponent's castle, the box get cracks (needs several hits to vanish) or disappears right away. All the boxes have gravity, which means when a box is hit, all the boxes on top of it will fall rectilinear down. The goal of the game is to be the first to hit the opponent's game winning object (in the illustration below, the object is a heart).

Preliminary illustration of how the game may look after implementation is shown below.

Figure 1: Player 1 shoots at player 2.

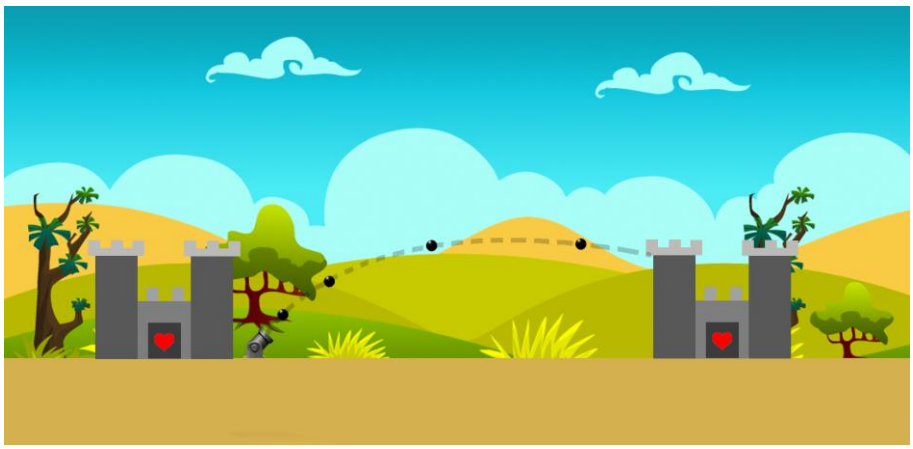
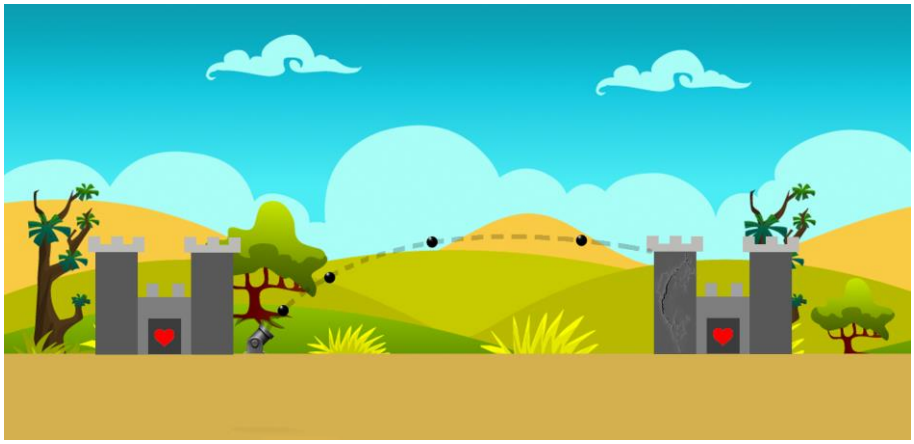


Figure 2: Player 2’s castle is hit, and the boxes cracks.



2. Functional requirements

Functional requirements describe what the system should be able to do, but not how it is to be implemented or how well it should be performed.

This section contains the functional requirements of the system/game we are developing.

Priorities:
High - Functionality that is essential for the game to have
Medium - Functionality that significantly improves the game
Low - Functionality that is nice to have

ID	Name	Description	Priority
FR1	Touch	The system shall be controlled by touch input	High
FR2	Main menu	The system will have a main menu	High
FR3	Multiple game-modes	The game should have at least two game-modes	Medium
FR3.1	Single player	The game can be played by a single player	Medium
FR3.2	Local Multiplayer	The game can be played by two players locally on the same phone.	High
FR3.3	Online multiplayer	The application shall	High

		be able to connect two players using a gameID. They will be connected through internet.	
FR4	Tutorial	The player should be able to go through a tutorial if preferable	Low
FR5	End of game	The player wins if he shoots the opponents game-ending object	High
FR6	Sound	The application should be able to play sound and sound effects during the game.	Low
FR7	Forfeit game	A player should be able to forfeit and exit the game. The opponent wins automatically.	Medium
FR8	Defence	If a player's defence is hit, it should be damaged or disappear.	High
FR9	Time limit	The players should have a time limit on their turn when playing online multiplayer	Medium
FR10	Rematch	The player should be able to offer a rematch after the game is finished	Medium

3. Quality requirements

3.1 Modifiability

We have chosen modifiability as our primary quality attribute.

Table 1: Changing general graphics/look of the game
There should be easy to change the general look of the game depending on seasons, holidays etc.

ID:	M1
Source:	Developer
Stimulus:	Wish to change graphics
Artifacts:	Graphical resources
Environment	Design time
Response	The graphics of the game are changed
Response measure	The developer should be able to do achieve this in less than 1 hour, without affecting the rest of the game

Table 2: Add different in-game objects
*There should be possible to **add more/new** objects, such as armour and weapon types, each with different properties.*

ID:	M2 - Adding sprites of in-game objects
Source:	Developer
Stimulus:	Wish to add new in-game objects
Artifacts:	Graphical resources
Environment:	Design time
Response:	Create and add additional objects and new sprites in-game
Response measure:	In 3 hours, sprites will be integrated in the system without adapting other modules

Table 3: Modify in-game objects
*There should be possible to **modify** the already existing objects in the game.*

ID:	M3
Source:	Developer
Stimulus:	Wish to modify new in game objects
Artifacts:	Graphical resources
Environment:	Design time
Response:	Make changes, unit testing, possibly user testing

Response measure:	In 30 minutes, without changing the interface of the class

Table 4: Add different game levels/designs
*The developer should be able to **add new** game levels within a certain time limit.*

ID:	M4
Source:	Developer
Stimulus:	Wish to add different level designs
Artifacts:	Game world class
Environment:	Design time
Response:	Make changes, unit testing, possibly user testing
Response measure:	In 3 hours, levels will be integrated in the system without adapting other modules

Table 5: Modify existing level designs
*The developer should be able to **modify** the design of the existing levels within a certain time limit.*

ID:	M5
Source:	Developer
Stimulus:	Wish to modify existing level designs
Artifacts:	Game world class
Environment:	Design time
Response:	Make changes, unit testing, possibly user testing
Response measure:	In 1.5 hours, without changing the interface of the class.

3.2 Usability

Table 6: Starting a new game
Starting a new game can be done within a few clicks when you open the app.

ID:	U1
Source:	User
Stimulus:	The user wants to start a new game
Artifacts:	System

Environment:	Runtime
Response:	Intuitive and straightforward GUI menu
Response measure:	User is able to start a new game with less than 3 clicks from the main menu

Table 7: Understanding the game
Understanding how the game works should be intuitive.

ID:	U2
Source:	User
Stimulus:	The user wants to play the game
Artifacts:	System
Environment:	Runtime
Response:	Understanding the rules and the goal with the game through a quick tutorial.
Response measure:	The user completes the tutorial and understands the basic knowledge of the game in less than 2 minutes.

Table 8: Change settings
There should be no problem to figure out where and how you can change a setting (like game volume)

ID:	U3
Source:	User
Stimulus:	The user wants to modify a setting
Artifacts:	System
Environment:	Runtime
Response:	The user finds the setting and modifies it to the desired value
Response measure:	Less than 20 seconds

3.3 Performance

Table 9: Response time to user input

The game should react as fast as possible to user input to make the gameplay smooth.

ID:	P1
Source:	User
Stimulus:	The user wants quick response time when firing a shot
Artifacts:	System
Environment:	Runtime
Response:	The system registers the users touch
Response measure:	The system should register the touch within 50 ms

3.4 Availability

Table 10: Acceptable unplanned downtime

The game needs to have its online game mode available as often as possible.

ID:	A1
Source:	System
Stimulus:	The online multiplayer system should be available no less than 95% of the week.
Artifacts:	Devices
Environment:	Runtime
Response:	If unscheduled downtime should occur, the developers should automatically be notified
Response measure:	The notification should arrive within 1 minutes of failure.

Table 11: Gaps in internet connection

The player needs to be notified if the online opponent suddenly no longer is connected.

ID:	A2
Source:	System
Stimulus:	A player loses internet connection

Artifacts:	Devices
Environment:	Runtime
Response:	The opponent gets notified when the opposite player loses internet connection
Response measure:	Notification received in less than 10 seconds.

4. COTS

COTS - “Commercial off-the-shelf” - are software or hardware products which are tailor-made for specific use and made public for general use. They are supposed to be implemented into already existing systems and are easily installed and user-friendly. Almost all software you normally buy for a computer can be defined as a COTS, it could be operating systems, business packages, antivirus software etc.

4.1 LibGDX

LibGDX is a java game-development framework. It has a lot of built-in behaviour and features for graphic, input and sound. Our code can customize this behaviour letting us build our own game from scratch. An advantage with libGDX is the possibility to make a desktop application, where you can build and test the game. The integrated Android emulator is often slow in comparison.

4.2 Android SDK

Android SDK contains different development tools, an emulator and required libraries to build an Android application and interface with the device system it runs on.

4.3 Google Play Services SDK

Google Play Services provides different APIs to help integrate popular gaming features in your own games.
We want to use the API provided for handling real-time multiplayer game mode[1], which is a peer-to-peer mesh networking API. We will use it to set up multiplayer game rooms where players can invite other players with a gameID.

[1] Reference 1

5. Technical constraints

5.1 LibGDX

Committing to using libGDX constraints us to use the built-in features that libGDX provides, such as object rendering and physics handling. If is not that easy to not make customizations on top of the features made available by libGDX.

5.2 Android SDK

Android applications are built with Java, but not all versions of Java are supported. The supported version are currently Java 7 and a subset of Java 8.
Android as an OS is used by many devices with different attributes, such as screen specifications and the currently installed version of Android. Using an overview of market share of Android versions[2] it was decided to support versions 4.1.x and newer (Jelly Bean and newer/ API levels 16+) of Android OS as this will include more than 99% of all Android devices as compatible devices (per February 5, 2018).
Given that Android device screens come in all types of different sizes and resolutions, graphics that scale with the device screen (at least to some extent) is necessary.

5.3 Google Play Services SDK

Relying on Google Play Services to connect players in a peer-to-peer fashion introduces some constraints. As we don't use a server of our own we cannot make changes to how things are handled, but are reliant on the service and API provided by Google Play Services. When creating or joining a room, interaction between each client and the Google Play servers is required, making a network manager of some sort necessary. A room has a limit of maximum 8 players. This isn't actually a constraint, as we only intend for 2 players to play in the same room.
Using this service may constraint or affect the way general data handling is applied in the game.

6. Issues

The quality requirement availability was added, removed and added again due to insecurity regarding its importance to our project. In the end we decided that it was relevant enough, and that we wanted to pay some attention to the requirement.

7. Changes

Here we will log changes we do after the first delivery.

[2] Reference 2

8. References

Used in the text:

1. Google Play Games Services, “Real-time multiplayer”; Web:https://developers.google.com/games/services/common/concepts/realtimeMultiplayer / Accessed: 25 February 2018
2. Android, “Dashboards”, Web: https://developer.android.com/about/dashboards/index.html / Accessed 25. February 2018

References for general use:

3. Len Bass, Paul Clements, Rick Kazman, “Software Architecture in Practice–Third edition”, Addison Wesley, September 2012
4. Ian Gorton, “Essential Software Architecture - Second Edition”, Springer-Verlag, 2011