Implementation document

Group 11 - Mandroid Studio

Team members:

Shivam Verma

Martin Halvor Korstveit

Kjetil Vaagen

Dag Erik Gjørvad

Kristian Huse

Thomas Skarshaug

Chosen COTS (Components and technical Constraints):

Android devices

Android Studio

Google Play Game Services

Primary quality attribute - Modifiability

Secondary quality attribute - Usability

Table of contents

1 Introduction

1.1 Description of the project

The educational goal of the project was to “Learn to design, evaluate, implement and test a software architecture through game development.”. We have learned a lot during this process and understood how we can better plan and implement software architecture. All members on the team have the same background, so we have worked together to understand the concepts in this learning experience.

In this phase we have implemented the architecture described in our architectural description document. We have worked as planned, but done a few changes throughout the project as expected.

1.2 Description of the game concept

We wanted to make a game based on one or more players moving in free-fall to avoid obstacles in the path with special focus on modifiability and usability. It should be a multiplayer platform, where you can challenge and play with friends. We wanted to add power-ups and character modification to achieve the modifiability requirement.

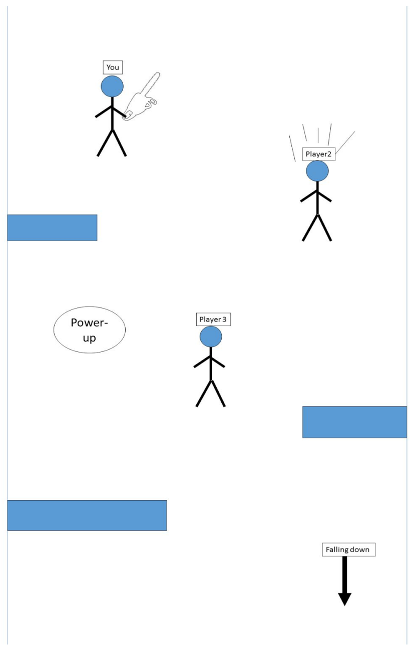


Fig 1: Illustration of game concept

The goal of the game is to reach the ground level before the other players. To achieve this you must avoid obstacles that comes in different forms. You can also position yourself in the way of other player to get an advantage.

1.3 Structure of the document

In this document we will go through the design details, test reports and the relationship between these elements in the architectural description.

- Introduction
- Design details / Implementation details
- User Manual
- Test report
- Relationship with architecture
- Problems, issues, and points learned

2 Design details / Implementation details

During the implementation phase of our project we followed our architectural description, including the class diagram, architectural and design patterns. However, we quickly realized we needed additional classes, functions and variables to simplify and organize our project. For instance, our initial class-diagram consisted of only three singleton-classes, but the final product consists of five singletons-classes. Since making a class-diagram containing every variable and function of each class would end up being too big and messy, we decided to only list the public members in every class with the exception of singleton classes.

An updated version of the class-diagram can be found here[1]. (The yellow headers represent new classes)

Our framework folder contains classes that have specific responsibilities or are custom made to make mathematical calculations for the game logic easier.
AudioFreeFailling is essentially the main media player for the game, one that produces sound when appropriate.

BaseGameUtils is a framework which allows the user to connect their Google account and to the Google Play Game Services. GameServiceListener is the communicator between our app and the google play services API. This class contains various listeners for the connection between players, for instance if a connection fails, a peer declines the connection or a new peer joined the lobby. The GridViewAdapter class is to represent the different characters a user can choose from.

ImageItem is a simple framework class that hold information of an image which is a Bitmap and it's title. RectSAT is a very simple framework class which only hold information of the different corners of a rectangle. These corners are described as points.
VectorSAT is a selfmade framework class which implements vector functionality such as getLength, getUnitVector, etc. Hence this class is primarily used for mathematical issues such as our collision system which is an implementation of SAT (Separating Axis Theorem) but is also used in calculating movement direction and magnitude for the character.

1 https://www.draw.io/?lightbox=1&highlight=0000ff&edit=_blank&layers=1&nav=1&title=Copy%20of%20Logical%20View.html#Uhttps%3A%2F%2Fdrive.google.com%2Fuc%3Fid%3D0B8RTiOVxMNdARGo0RDdIMDkwODg%26export%3Ddownload

In our implementation, we have followed the MVC pattern. This divides the classes into models, views and controllers. The models are the central components of the pattern. They control all the logical behavior, data and rules of the application. The views are all the visual elements in our application and the controllers control the views and models. When structuring the classes like this, we allow for effective code reuse and a good, easy to work with structure. In our implementation, all the views and models have one controller, but some of the controllers are reused to compress the classes needed.

The Drawable interface is the highest abstraction of the models. The Collidable class implements the Drawable interface and defines a model. This model is then extended by the models in the Models directory, which contain Character, GameMap, GameMessage, Obstacle, Player and PowerUp. We also have one abstraction level below this, and that is the implementations of the models. These are very specific models that are used in the game. It is on the lowest level, implementation.models, the highest modifiability level is found, and where you can add additional characters, levels, etc.
Our Singleton classes handles crucial system function, like GameThread, which runs the game. It works as the engine behind all the other classes, and updates all the other parameters.

ResourceLoader is the singleton that loads all of our resources files (bitmaps etc) from the drawable directory. Config is the singleton that reads and saves one file, the DataHandler. CollisionHandler is the singleton that should do all the collision detection and handling instead of the Collidable class. DataHandler is the singleton that contains all the variables and information about the settings values such as wanted FPS, SFX level, BGM level, screen width and screen height.

3 User Manual

3.1 Functional requirements

1. Android device with api version 15 or later or Android Studio with an emulator that supports google play services.
2. Users are required to have a Google account that's connected to Google games services to enable multiplayer gameplay.
3. The Google account must be allowed as a tester on Google’s Developer Console. (Added by the developers)

3.2 Initialisation: installation, compiling and running

To play the game with an android device:

1. Make sure your phone have an android version that have a higher API than 15.
2. Go to https://goo.gl/mC1ykX
3. Press download. If you get prompted with a question about verification press accept.

To install the game within Android Studio

Note: Emulated devices might have problems running the game, as it does not recognize the drawable folder of the application.

1. Download and open the latest version of Android Studio. Import the project by going to File → Import project
2. Press the green run button on the top to run the project:

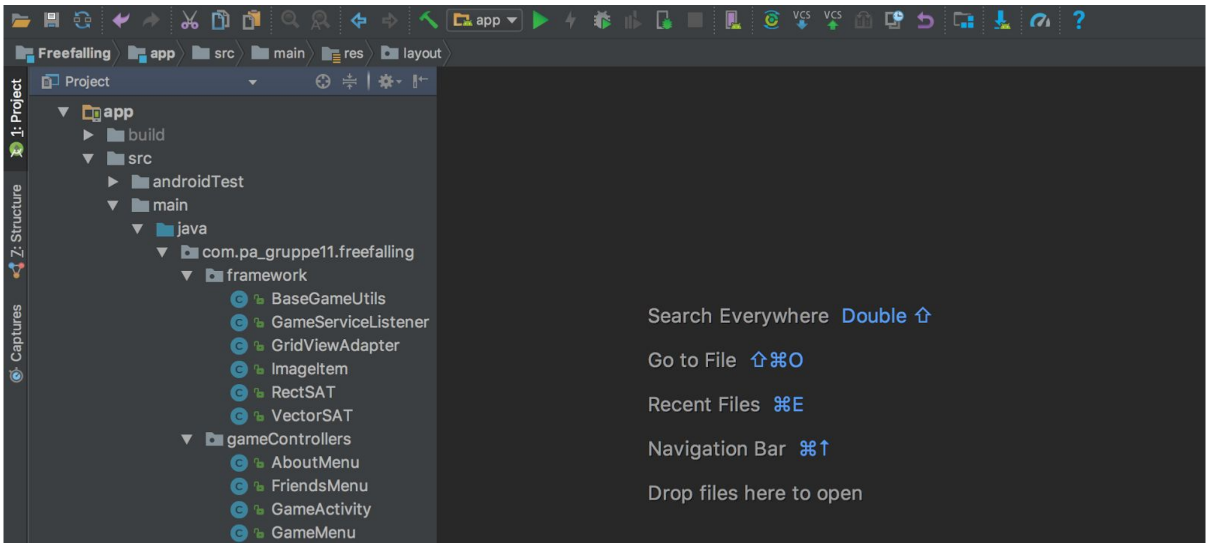
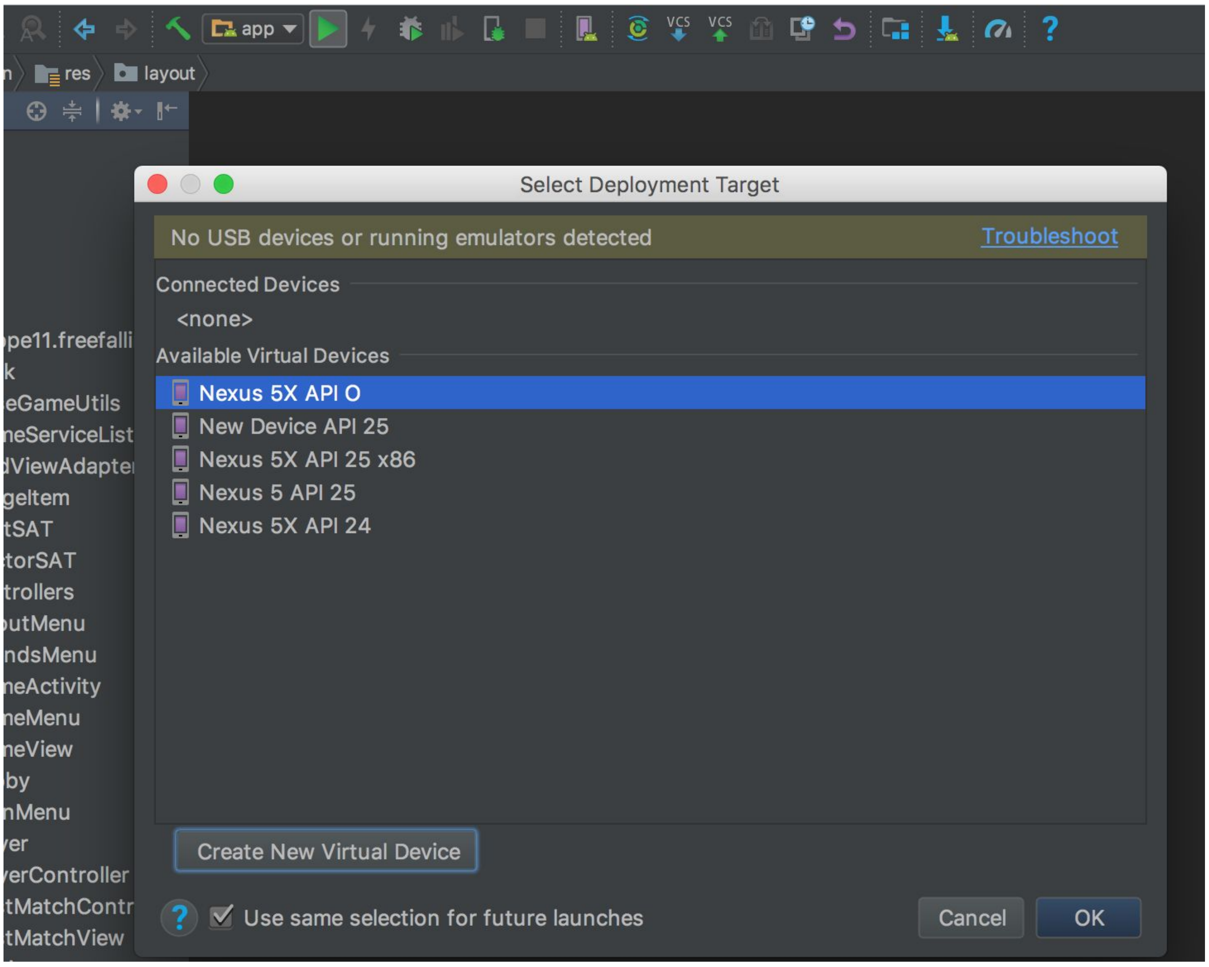


Fig 2: Run project

3. Once you press run a menu will pop up asking you to choose either to run the program on a connected android device or to choose an emulator (make sure the emulator supports google play services). If you haven’t made an emulator yet, click “Create new device”. Once you have chosen a device, press “Ok” and the game will start.



Select Deployment Target

No USB devices or running emulators detected Troubleshoot

Connected Devices

<none>

Available Virtual Devices

- Nexus 5X API O
- New Device API 25
- Nexus 5X API 25 x86
- Nexus 5 API 25
- Nexus 5X API 24

Create New Virtual Device

[✓] Use same selection for future launches

Cancel OK

Fig.3: Device selection

3.3 Play online

As of the delivery of the application, there are one fully implemented way to play online; Automatchmaking, which queues the user automatically with another user waiting for a match. By pressing the “Quick Match”-button in MainMenu, and selecting “Automatch”, the system queues the player for a match. The game automatically starts when there are 2 players in the same room, as displayed with usernames.
For this online functionality to work, the player must be connected to Google Game Services. If the user’s Google account is not connected to Game Services, an account must be connected. This can be achieved by the player pressing the “Sign-in”-button in MainMenu, where the user will be prompted by an account-creation-screen. When this process is done, the account will be able to play any and all games connected to Google Game Services.
In addition for Game Services to accept the login, the Google-account must be added to our Developer Console, a requirement set by Google to assure only those allowed by the developer gains access (while the game is not published).

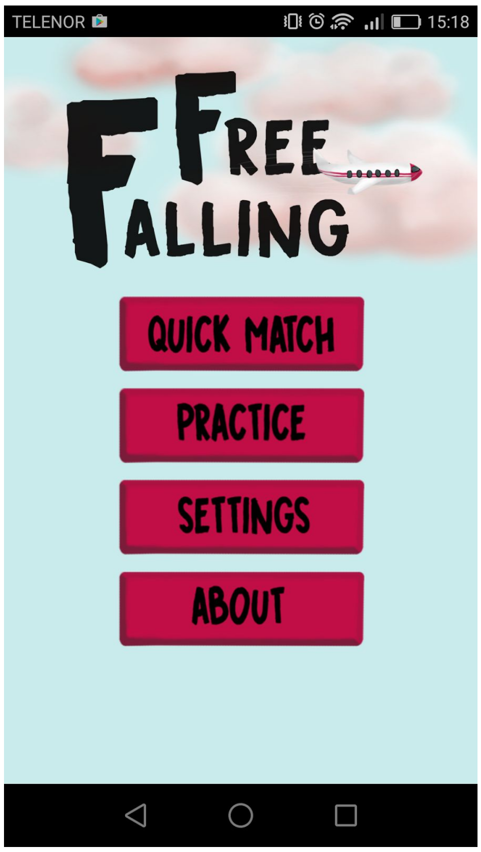


Fig 4. Main menu

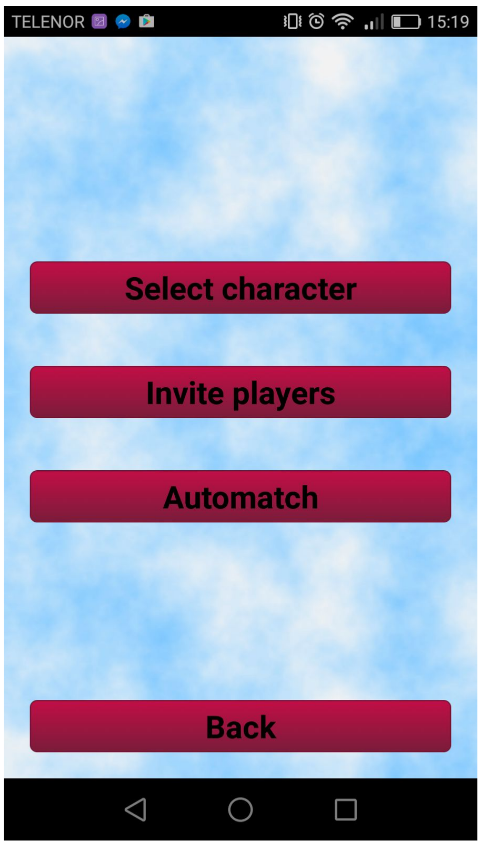
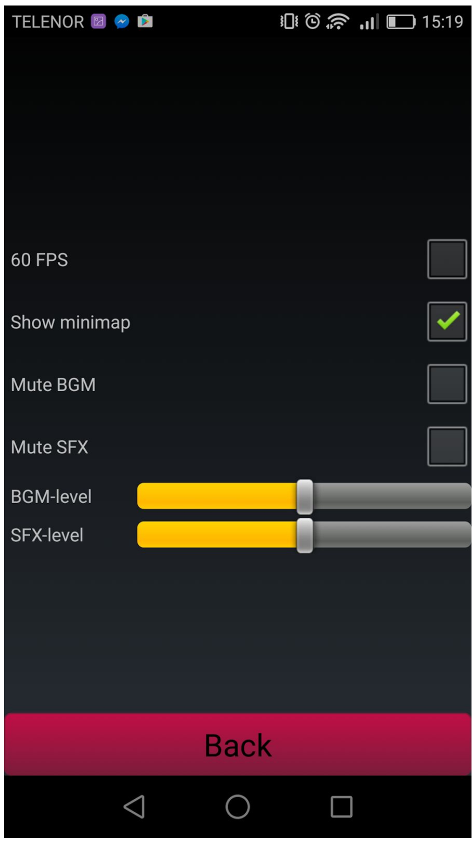


Fig 5. Multiplayer Menu



60 FPS ☐

Show minimap ✓

Mute BGM ☐

Mute SFX ☐

BGM-level

SFX-level

Fig 6. Settings Menu

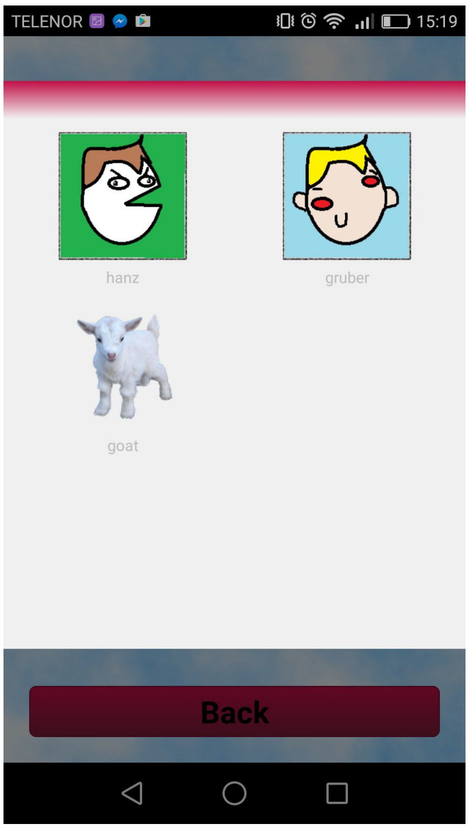
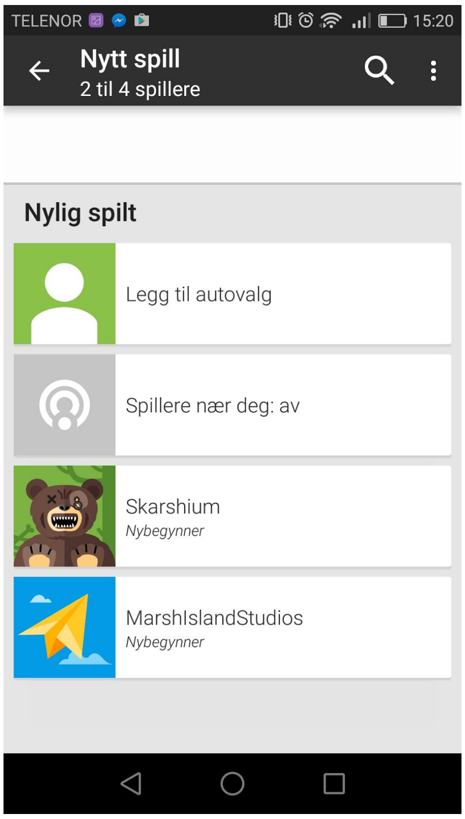


Fig 7. Character Selection



TELENOR 15:20

Nytt spill
2 til 4 spillere

Nylig spilt

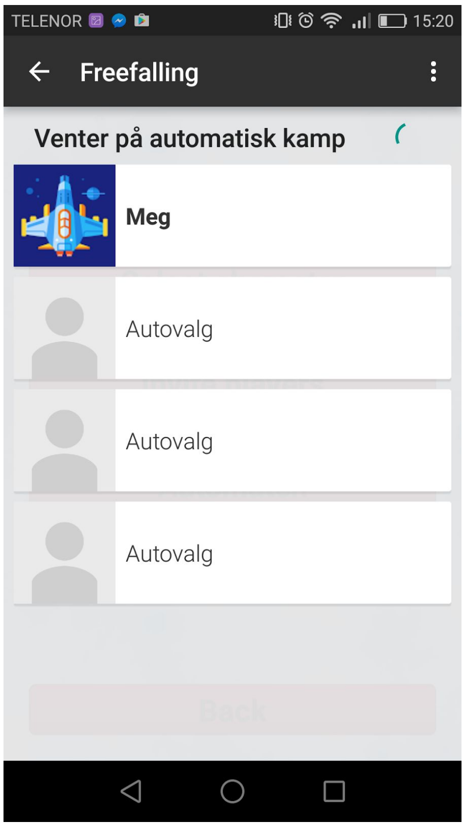
Legg til autovalg

Spillere nær deg: av

Skarshium
Nybegynner

MarshIslandStudios
Nybegynner

Fig 8. Game lobby



TELENOR 15:20

Freefalling

Venter på automatisk kamp

Meg

Autovalg

Autovalg

Autovalg

Fig 9. Waiting for players

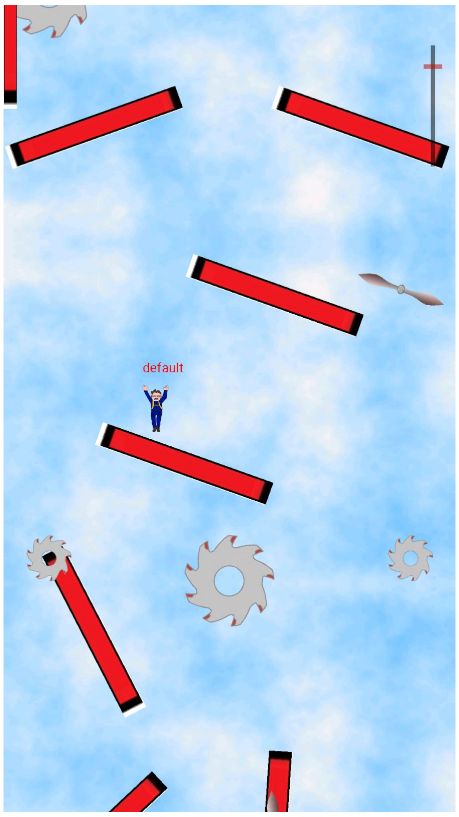


Fig.10 Ingame

3.4 How to play

1. Press and hold the screen to make the player move towards your finger. Change direction by moving your finger.
2. Avoid obstacles.
3. First one to reach the bottom wins.

4 Test report

We did also run extensive testing in the last phase of the project, where we tested both functional and quality requirements. The tests had varying results, mostly due to the project taking more time than we expected.

Functional requirements	
FA1: Having available internet connection.	
Executor:	Thomas Skarshaug
Date:	21.04.17
Time used:	5 sec
Evaluation:	Success
Comment:	The user was able to connect to the internet.
FA2: Able to connect to Google Play Game Services through the app.	
Executor:	Thomas Skarshaug
Date:	21.04.17
Time used:	2 min
Evaluation:	Success
Comment:	The user was able to log in to the Google Play Game Services.
FA3: When touching the screen some acknowledgement output is given.	
Executor:	Thomas Skarshaug
Date:	21.04.17
Time used:	2 min
Evaluation:	Success
Comment:	The user got feedback in the form of visual text after touching the screen
FA3.1: Through touch input, the user can control the character.	
Executor:	Thomas Skarshaug
Date:	21.04.17
Time used:	2 min
Evaluation:	Success
Comment:	The user got feedback in the form of visual movement of the character on the screen.

FA3.2: Through touch input, the user should be able to interact with the environment	
Executor: Date: Time used: Evaluation: Comment:	Thomas Skarshaug 21.04.17 5 min Success The user got feedback in the form of visual movement collision with the walls and an obstacle.
FA4: Create a new game lobby and configure game settings	
Executor: Date: Time used: Evaluation: Comment:	Thomas Skarshaug 21.04.17 - Failed Game lobby and game settings functionality are not implemented.
FA4.1: It should be possible to put player into the same game based on player ID	
Executor: Date: Time used: Evaluation: Comment:	Thomas Skarshaug 21.04.17 - Failed Functionality that allows players to be invited into the same game is not implemented.
FA4.2: The system should fill available spots automatically if desired.	
Executor: Date: Time used: Evaluation: Comment:	Thomas Skarshaug 21.04.17 5 min Success The user got feedback in the form of visual movement of the character on the screen.
FA4.3: It should be able to play with multiple players (2 - 4).	
Executors: Date: Time used: Evaluation: Comment:	Thomas Skarshaug, Dag Erik Gjørvad 21.04.17 5 min Success The user got feedback in the form of two visual players moving on the screen.

FA5: The system should be able to play sounds when appropriate	
Executor: Date: Time used: Evaluation: Comment:	Kjetil Vaagen 22.04.17 6 min Success The system played sounds appropriately, for instance when a player collision took place.
FA6: Should be possible to configure sound and graphics values in the settings menu	
Executor: Date: Time used: Evaluation: Comment:	Thomas Skarshaug 21.04.17 5 min Success The user was able to configure the different values for the options: BGM, SFX and Graphics(60 FPS or not).

Quality requirements:

Modifiability	
M1: Adding a new playable level to the game after developing the level.	
Executor: Date: Environment: Stimuli: Expected response measure: Observed response measure: Evaluation: Comment:	Shivam Verma 22.04.2017 Run-time A new level is made and need to be added 30 min 3h Failure We have functionality for changing the already existing level map, but not changing between different levels as was planned by this requirement. Implementing this would take about 3h cumulative time, but is not done due to time constraints.
M2: The developmental team should be able to add a new playable character	
Executor: Date: Environment: Stimuli: Expected response measure:	Shivam Verma 22.04.2017 Run-time A new character is created and needs to be added. 20 min

Observed response measure: Evaluation: Comment:	10 min Success Our architecture proved to be sufficient to add a new character in short time due to good planning of superclasses and the use of MVC to separate the logic from the view, making it easy to get an overview of the software.
M3: The developmental team should easily be able to create new obstacles.	
Executor: Date: Environment: Stimuli: Expected response measure: Observed response measure: Evaluation: Comment:	Shivam Verma 22.04.2017 Run-Time New obstacles needs to be added 30 min 10 min Success Our architecture proved to be sufficient to add a new obstacle in short time due to good planning of superclasses and the use of MVC to separate the logic from the view, making it easy to get an overview of the software.
M4: The development team should be able to extend the set of abilities and gadgets that a character can use in the game, for already developed abilities and gadgets.	
Executor: Date: Environment: Stimuli: Expected response measure: Observed response measure: Evaluation: Comment:	Shivam Verma 22.04.2017 Run-time Wanting to change abilities of a character 20 min 2h Failure Abilities like speed and size can now be changed within 10 min, but gadgets like power-ups and weapons is not yet implemented to the extent that was required due to time constraints.

Usability	
U1: It should be intuitive for the end user to start a game with friends.	
Executor: Date: Environment: Stimuli: Expected response measure: Observed response measure: Evaluation: Comment:	Reso Ratnam (external user), Lars Møster (external user) 22.04.2017 Set-up User wanting to start a game with a friend 2 min 1 min Success The user successfully started a multiplayer game with his friend.
U2: It should be intuitive for the end user to configure the settings options of the application	
Executor: Date: Environment: Stimuli: Expected response measure: Observed response measure: Evaluation: Comment:	Reso Ratnam (external user) 22.04.2017 Set-up User configuring the game settings 0 errors 1 errors Failure The user had problems understanding what was meant by SFX(Special effects) and BGM(Background music).
U3: It should be intuitive for the user to understand the mechanics of the game	
Executor: Date: Environment: Stimuli: Expected response measure: Observed response measure: Evaluation: Comment:	Reso Ratnam (external user) 22.04.2017 Run-time User playing the game 2 games 2 games Success After playing similar games for the first time we have experienced that we often have questions about the game mechanics. We therefore expected the user to have the same for us. Every aspect other than knowing the boundaries of the map was intuitive for the for the user to understand after the first game. It was successfully understood during the second game.

Performance	
P1:The system should accommodate devices of different system performance, in such a way that it will not impact the user's game experience.	
Executor: Date: Expected response measure: Observed response measure: Evaluation: Comment:	Kristian Huse 23.04.2017 Game work for android API 15 or newer + between 4'' and 5'' with 1080p Game work for android API 15 or newer + all screen sizes and resolutions Success The software have been designed to accommodate older devices by dynamically placing elements and using compatible code. Have also been tested on different devices throughout the implementation to uncover possible errors early.
P2: Whenever the system changes between menus and in-game, the system should remove unneeded resources from memory, and load required resources.	
Executor: Date: Expected response measure: Observed response measure: Evaluation: Comment:	Kristian Huse 23.04.2017 0 errors 0 errors Success This requirement was intended to make the game switch smoothly between different activities and we have therefore worked towards preventing unnecessary use of resources by closing processes when they are not used.

5 Relationship with architecture

The updated class diagram can be found in class_diagram.png at the root of the delivery.

We have tried to followed the architectural description to the best of our knowledge, but some inconsistencies has occurred:

We have an inconsistency regarding our planned implementation of the Factory pattern and our implementation of it. We chose not to implement the Factory pattern because of the time constraint we were under and, that we primarily focus on implementing the MVC pattern. It would be difficult for us to foresee this inconsistency since it is an inconsistency based upon how the project evolves and how much of it we were able to implement. Hence we decided not to go with it. That does not mean that we do not think that the Factory pattern would be beneficial in our case, but we just did not have the opportunity to realize it.

Another inconsistency is the size of Collidable class. One of our architectural tactics was to keep the size of the modules down, and the size of this module is too big. The Collidable class which in itself should only be a model class does all of our collision detection and handling. We have created a class CollisionHandler which should take care of all collision related issues such that Collidable should not be concerned about it and stay. This could be discovered during the ATAM session since we and the other group should be able to see that when the collision should be implemented this module would grow too large. Hence we should have come up with the current thought of solution or implemented a higher module i.e GameObject that would hold a Collidable and this collidable would only implement the collision and not everything else as well.

We also have an inconsistency between our planned prototyping and user testing and our actual performed ones. There are many reasons behind this situation. First of we had a lot of issues getting started with the project and the coding itself due to Gradle and Git issues which took a lot of time as well as most of the team had other large projects going in different courses. When we finally started to get the work flow going smoothly we were closing in on the end of March and did not have the time to focus heavily on an iterative designing process which most likely would ensure a better design than we have at the moment. Hence we have only performed a couple of undocumented usability tests during the development of the project as opposed to our goal of doing extensive undocumented testing throughout the project.

We could have partly identified this inconsistency at an earlier point by having a better communication inside the group about our schedules and planned projects during the month of March. And hence could probably get a smoother start to the work of the project by spreading out the work load to the beginning of March as well. About the Gradle and Git issues there is not a lot we could have identified at an earlier point. Only one person in our team had any relevant experience with Gradle, so we all had to learn while developing the project.

Inconsistencies with the 4+1 view models as planned vs implemented

Logical view

We have made a few important changes from the planned implementation to the actual implementation.
Due to the benefits of using xml, we have removed the views in the diagram excluding GameView and replaced them with xml-versions. These xml's are not included in the view, as they in some ways work outside our MVC-pattern, with the actual view being created through Activities.

Additionally, our class diagrams does not contain the actual implementation of models, such as Sawblade, Block, Goat and similar extensions of Obstacle or Character, as the already large diagram would become more complex and confusing.
Mentioned in Architectural Description, we had a Player with variables we were to send via Game Services to other players, but there is a byte limit to sent messages, and performance issues with sending the entire Player class and its components. As such, we created a GameMessage, acting as protocol and dataholder, which can be sent to others. This decision did not impact our architectural choices, but it made peer-to-peer communication much easier.

Regarding the Lobby we planned to have, where the players could invite players and choose GameMap, we had to scrap it due to time constraint. The replacement were automatically finding someone to play against, and no way to select a GameMap. We made it possible however for the player to select a character before they started an automatch, and given enough time, we could have made a similar implementation, where the players voted for a map.

Regarding the inconsistencies with views, we could have identified it at the early development phase, as xml’s were being worked on early, and we could have seen the benefits already then.
Using the player as the message could not be identified before we started sending messages between devices, due to the implementation of Game Services was done midway in the development phase. Detecting the issues with lobby will discussed at the next section, Process view.

Process view
Regarding the process view “Initiate game with friends” there is an inconsistency between the planned process view and its implementation. In the implementation it is not possible to initiate a game with friends since we made multiplayer work with Google Play Game Services. It made us an auto matchmaking functionality for us called automatch in-game. This was also true regarding the “menuSelection” view. Hence these views are inconsistent with our implementation. Detecting this inconsistency would be hard to detect before the actual implementation, since this dependent on how we implemented Google Play Game Services. Of course if we had absolute knowledge of how the API works we could probably have identified this issue earlier.

6 Problems, issues, and points learned

This was a challenging project. In preparation for the first phase we needed to study the theory behind making good architectural decisions and learn how to build systems in android studio. In addition to this, we needed to learn how to use android studio which was going to be our main development tool. As expected we used much more time in this phase than we normally would if everything had been understood.

Since the project was realized as a learning activity we have done some small changes to our work method during the duration of the project. This especially in regard to communication and work division.

We didn't have so many unexpected large problems or issues, but we did have some problems regarding the development process. Even though everyone have the same educational background we have different skills, which made it hard for everyone to contribute in the same way. It was also hard for everyone to have a clear overview over the whole project. With regard to this problem we should have analyzed and delegated each and everyone's workload and areas of expertise better.

6.1 Pattern implementation / architectural experiences

6.1.1 MVC

There were slight changes to some of the views, where we decided to exchange Java-views with xml, for easier handling and drawing. PostMatchView were changed with an xml, to keep track of the previous Google-connection, controller and canvas. The only remaining pure java-view, is GameView, which tells drawable models to update their draw-methods through GameActivity, which acts as both controller and a connection for the in-game MVC. This makes the MVC-structure a little off, as the view gets the model through the controller. Additionally, GameView does not directly draw objects, but gives its canvas to models implementing the interface Drawable, as designated by GameController. For this to be considered full-fledged MVC, we must think of the Canvas as the View, and the model updates the Canvas according to the controller's update function(manipulates).
The illustration below depicts this implementation.

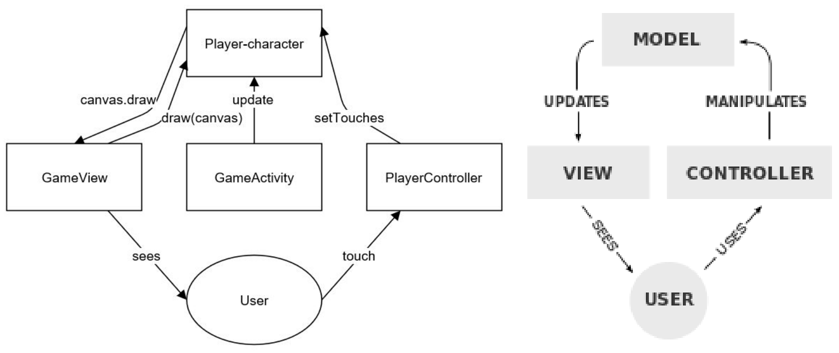


Fig.4: Illustration of our implementation(left) and standard MVC(right).

Instead of MVC having knowledge of the other components, we have GameActivity as a connector(indicated by the curved lines), where the models/Player-character, GameView and PlayerController can access each other. GameActivity can be considered as containing the rules of the game / gamemode, but as we never intended to change game-mode, nor will we, this decision only gives us benefits.

6.1.2 Peer-to-peer pattern

The peer-to-peer pattern is realized through Google Game Services and its available methods, which runs underlying communication using peer to peer. Methods such as sendUnreliableMessage(UDP) and sendReliableMessage(TCP) communicate without a need of a client acting as server, but it is also possible to designate one of the hosts to act as a server where the other hosts can communicate with each other through said host, which we did not need to do in our application.
We know that this API uses some sort of multithreading, as methods runs separate to our GameThread, but the exact implementation of GameServices is not visible, so a more in-depth understanding of the peer-to-peer can not be achieved.

6.2 Gradle and Git

During the initialization phase of the system we had some problems with Gradle in Android studio. We had issues with Gradle building the project since when we initiated the project some of us had different versions of Gradle and Android studio installed. This resulted in different build.gradle and settings.gradle files being pushed and pulled to/from Github and caused all sorts of havoc. And since most of us had almost no experience with Gradle this was a huge source of despair. At last we found a solution to these problems, but we had tried to change Git repository and uncountable amounts of reinstalls of the project before we got it to work nicely for all of us.

6.3 Google Game Services

By using Game Services we spared a lot of work such as setting up a connection and settings for P2P. One backside with this framework is that certain functionalities as keeping a friends list and inviting them are not directly supported in their framework. Although there are some workarounds to achieving this, we decided not to implement these functionalities. Instead we implemented matchmaking with support for randomly selected players or nearby players.
Additionally, it was a hassle to setup the API for usage by the group, as we had to add testers to the Google's Developer Console, and generate an encryption store to be shared with the entire group.

6.4 Time usage

The use of time did also heavily exceeded our expectations. Even though we took into consideration that we would have to learn a lot of new things; analyzing, discussing and formulating the architecture took much more time that we had planned for. Some of this was due to it being our first time considering the architecture this thoroughly, but generally we learnt that this need to take time to get right. This is a valuable experience for future projects since we now can take the architecture development phase into consideration when we estimate how much time it would take.

6.5 Points learned

There are many things we learned from doing this project. By building an android mobile app from scratch we of course enhanced our java programming skills and got familiar with mobile development environment and the way of doing things, but maybe most importantly we learned about software architecture - what it is and what it isn't, and why it exists. It's safe to say that every member of our team have now realised the importance of the software architecture of a system. Although a good architecture alone cannot guarantee the functionality or quality that's required, it acts like a bridge from abstract ideas to specific goals and requirements that can be well documented and tested. Following the architecture while implementing proved to be more difficult than we thought, but is definitely worth doing. We also learned that making the software consistent with the architecture is as important as defining it. It is therefore important to have some knowledge about the feasibility of the implementation already at the architecture definition phase.

6.6 Last words

We had fun realizing our system. We were all excited to create apps and to realize it as a part of our grade, gave us incentive to do our best. Some problems regarding the technology is to be expected, but when everything was sorted out we worked effectively.

We feel confident with our performance in the first phases, planning the architecture and sorting out requirements. Despite none of us had considered program architecture at this scale before, we quickly formed a good and realistic architecture which we realised satisfactorily to our expectations.