BACHELOROPPGAVE:

CAKE DEFENSE GAME FOR THE ANDROID OPERATIVE SYSTEM

FORFATTER(E): ØYVIND BERG
KNUT ØIEN
HENRIK JOTUN

Dato: 23/05-2012

ABSTRACT

Title:	Cake Defense	Date : 23.05.12
Participants	Øyvind Berg	
	Knut Øien	
	Henrik Jotun	
Supervisor(s)	Simon McCallum	
Employer:	Krillbite	
Keywords	Lua, Corona, Tower Defense	
Number of pages: 99	Number of appendix: 30	Availability: Open

Cake Defense is a game project based on the game genre tower defense (TD). With a few disputes in the implementation of this genre we have come up with an interactive TD, resulting in a unique game for the android mobile phones. The player himself must take care of enemies that come at any time, whose mission is to eat the cake in the middle of the player's screen. By touching the screen at the location of the enemy, will hurt the creep, which is how a player can prevent creeps from eating the cake.

The game is made in Corona and is currently only a prototype of what will soon be released, hopefully it will be released in late-June 2012.

SAMMENDRAG

Tittel:	Cake Defense	Dato : 23.05.12
Deltaker(e)	Øyvind Berg	
	Knut Øien	
	Henrik Jotun	
Veileder(e):	Simon McCallum	
Oppdragsgiver:	Krillbite	
Stikkord	Lua, Corona, Tower Defense	

Antall sider: 99	Antall vedlegg: 30	Publisering: Åpen

Cake Defense er et spill prosjekt basert på sjangeren tower defence (TD). Med noen fåtvister i gjennomføringen av denne sjangeren har vi kommet opp med en interaktiv TD,som resulterer i et unikt spill for Android mobiltelefoner. Spilleren selv skal ta seg avfiender som kommer til enhver tid, som er misjon er å spise kaken i midten av spillerensskjerm. Så, bare ved å trykke og / eller berører skjermen på plasseringen av fienden, kan han hindre dem fra å spise kaken.

Spillet er laget i Corona og er per dags dato kun en prototyp av hva som vil snart bli gitt ut, forhåpentligvis blir det gitt ut i slutten av juni 2012.

PREFACE

This report documents the implementation of the project Cake Defense's result and its process. This includes planning, work distribution and handling of problems throughout the process.

The report was written by Øyvind Berg, Knut Øien and Henrik Jotun, which all was students writing their bachelor thesis for the Game Programming course at Gjøvik University College.

Through a partnership with Krillbite the game Cake Defense have been developed as a high fidelity prototype and is ready to get into further development due to this project.

Cake Defense, is a self-made game idea by the participants in cooperation with Krillbite, and Krillbite’s role was to bring design and feedback. This was to make sure the product had a high quality. Cake Defense is planned to be released as soon as possible for android devices.

This report was primarily intended for people who know programming software, and know what is involved in such a process.

Link to the project website:
http://hovedprosjekter.hig.no/v2012/imt/spill/sworddefense/index.html

TABLE OF CONTENTS

1. INTRODUCTION

This chapter will introduce the project and the team behind Cake Defense. For a dictionary of words and expressions we refer you to the appendix. For the resources we refer to [number] throughout the report which matches the bibliography, in chapter 10.

1.1 DEVELOPMENT TEAM

The development team consisted of Øyvind Berg, Knut Øien and Henrik Jotun. All of them were bachelor students taking the game programming course at Gjøvik University College for the last three years, starting out autumn 2009. Two of the members had never seen a line of code before autumn 2009, while one member had some previous experience within the World Wide Web area.

The team has worked together in one previous course, the game programming course in autumn 2011, where they created a game library in C++ with OpenGL and SDL. The team had its flaws back then, but knowing these flaws greatly benefited them within this project. Knowing the pitfalls they might fall into, and knowing what kind of persons they had to work with was a positive thing.

In order to complete this project by the end of May 2012, there was a need for a good vision of labour and an understanding of subjects in several topics. The knowledge distributed in the group was limited when creating a game. We had no designers and no earlier experience in creating a game for the android mobile phone. The whole team was looking forward to get this new experience.

1.2 Partnership

1.2.1Krillbite

When developing our game we wanted to focus mainly on game play and mechanics. However, we wanted to release the game. We realized that most people who buy games for mobile devices care a lot about the visual as well. Eye-candy graphics would then be up to Krillbite, a game development studio in Hamar. They agreed to help us by supplying artwork, as well as suggestions in game design decisions and even some audio to go with it.

Their help throughout the project has been very important. They have given us all the images and sprites we needed for the game. As none of us play Android games regularly, our partner Krillbite's input was also very useful when it came to design decisions, as they had a better understanding of what people who play Android games are looking for.

1.2.2 Contact Person

Our contact person in Krillbite has been Andreas Jordet and we want to thank him very much for making the game what it is today. We have also promised him to finish the game even though it is not possible to accomplish within the end of May 2012.

1.3 Supervisor

Our supervisor for this project has been Simon McCallum.

1.4 Project Description

1.4.1 Reasons for creating a game

For our bachelor thesis we want to make a game from scratch rather than to get a specific task from an external. The reason for this was that we wanted to gain experience in the whole aspect of developing an entire game in general. Not just from specific points in the process.

To improve our skills in programming and our knowledge around system development, a game is superior to delve into versus an application or a web application, although complexity can be found in all. Creating a game will have benefits in areas such as database handling, structuring of code, implementation, GUI-thought processes, testing, multiple threads, various game-algorithms and optimization of code. We considered that delving into a game for the android versus delving into an web application, or creating a game in flash for the web. The game for the android would take us through a higher level of understanding and benefit more in all the mentioned areas.

1.4.2 Implementation Plan

Regular work days were from Monday to Friday. Exceptions would happen due to other courses but Monday, Tuesday and Friday from 11.00 pm. This was the regular schedule unless other agreements were made. We worked together on the scheduled days. The remaining time was individual work, but we always had contact on Skype and everyone was always online.

At these regular work days most of the debates were about what was going on, what needed to be done and eventual criticism on game mechanics. After the meetings were over we worked individually on coding. If some things needed multiple heads on the job, we worked together on the coding to get us through.

There was a lot of work to do, individually and as a group. Delegation of work was done by all of us and some tasks required the group to work together on a specific topic. Other modules where worked on individually by our members. This allowed them to work from wherever they wanted either home or school. All project members have been online through Skype throughout the working days and even, so if anyone was in trouble he would reach the other members on Skype and be able to debate.

We roughly estimated to have used a minimum of 20 hours of work per week, per group member. This makes a total of 22 weeks * 20 * 3 = 1300 hours for this project.

Due to other courses within the project period, some meetings have been extended or cancelled. Everyone would know about this beforehand, so it did not create chaos for the work process.

1.4.3 Regulating Framework

Throughout the project, some subtle parts of the game idea have changed, which also changed the regulating framework from what we started out with in January. This can be found in our pre-project report. The regulating framework we ended up with was:

- Game should be limited to the android phone as of first release.
    - Later releases might be for iPhone and Windows Phone.
- The game should be at least a functional prototype by the end of May 2012.
- Graphics and user interface should be as simple as possible, even for beginners.
    - Eye-candy effect is still required, but limiting graphics to 2D sprite sheets.
- There is only one difficulty mode.
    - Difficulty increases throughout the game, the higher level, the harder it gets.
- Team have little experience developing for android.
    - Might end up somewhere else than what was planned.
    - Choices taken when it comes to simplicity in GUI for an android mobile phone might not end up perfect, due to screen size and our experience working with such a small screen size.
- Team has little to no game-algorithm experience.
    - Might take some time researching algorithms and how to implement various algorithms for our game engine.

    - Our implementation and structure might be suboptimal versus a professional programmer's implementation.
- Structuring code in Lua might be suboptimal due to no experience in Lua.

The development team had limited experience within designing a game from start to finish. Especially within a time frame and document everything that has been thought of, etc. Because of this, we might only get a functional prototype at the very end of the project. We were also aware that the design and ideas that we started out with might (and should) change to something quite different in the final version of the game. This is due to how game designing works. After a game has been implemented, things might do not work out that well, even if you designed a great game. Therefore a game requires a lot of changes to be made no matter how well you designed it.

1.4.4 Organization

1.4.4.1 Group Organization

The group came to the agreement that they would all participate equally on the project and that everyone was responsible for all parts of the product. During the development process we have worked as a team, and everyone has had an equal say in both programming and design decisions. Choices that had a big effect on the product were always decided by voting.

We concluded there was no need for a group leader on the basis on that there are only three members and they are all developers. We saw little benefit to have a leader, and it might instead highlight disputes. A full list of the rules for the group can be found in the appendix at the end of the report.

1.4.4.2 Report Organization

This report is divided into several chapters, were we try to follow a sequence based on how the process has developed over the past few months.

This report begins with defining the essence of the project and naming goals that needs to be reached to reach a good result. After that the report takes you into reading about each phase from design to implementation to result.

The report ends with an evaluation chapter of us brining out the good and bad things that we as a team have experienced over the past few months, which ends up in a conclusion in a chapter of its own.

1.4.5 Project Goals

A working version shall be delivered to Gjøvik University College at 23th of May 2012. The main goal was to create a product that meets the specifications that were made in an early phase of the project. We also wanted to deliver a product that we were all happy with, both us and Krillbite. The product to be delivered will be at minimum a functioning prototype game. Named Cake Defense and written in Lua for the android mobile phone.

Fulfilling this project required some vital functions to be implemented. A list of the useful functions we found reasonable was:

- Character
- Start Game
    - Victory
    - Defeat
- Artificial intelligence
    - Path finding
- Collision detection
- Eye-candy effects

A detailed list can be found in 2.2 Specification.

In the process of creating a game, the project also opens opportunities for us as a group to:

- Plan the process of a product from A to Z.
- Educate ourselves within Lua and tools required for this project.
- Difference between programming a game for the PC than for a mobile device, conserving screen resolution and input data from touch vs. keyboard.
- Get adequate knowledge of various modelling techniques so that we can choose the right model based on needs
    - Using our own modified version of Scrum as a development process.
    - Be able to control the model we chose
    - Be able to plan the system development process based on scientific knowledge

2. PROJECT ANALYZE

2.1 APPROACH TO THE PROBLEM

Games for mobile devices become more and more popular. And still there there's no tower defense that we see as is interactive enough in today's market. Games in the tower defense genre that we've played usually have common steps that loop over and over till the game is either completed or the player is defeated. It is usually played by putting out a tower or two, wait several seconds, then watch the creeps slowly getting killed by the towers. Once creeps are dead you can rebuild, sell, destroy, or place new towers based on gold you received from the creep kills. This, as we see it, was not interactive enough for a android game. We wanted to bring in some features of an RTS game, where you need a lot of actions per minute to defeat the AI. However we figured this to be to much complexity for an android game. It needs to be simpler and less stressful. As a result we wanted to make something in between. Something that does not make the user sit and wait for several seconds without doing anything, but also not having it too interactive so you destroy the phone by touching it too hard.

The problem was thus: How to create an interactive game play in the genre of tower defense with only our own creativity at hand?

2.2 SPECIFICATION

The result of this project shall be a game, fully playable. In shall include a player who can interact with something else than the usual towers in a tower defense game.

The system shall consist of a dynamic level engine with an underlying database that stores information about the player. It consists only of one user, which is the playing player at current time. To access the game the user downloads it from android market and installs it,

then starts the application on his mobile device. From here the user will have a few options that would define the main menu:

- Play
    - Play Tutorial
    - Skip Tutorial
- Settings
    - Takes you to a new screen menu where options as muting sound and reset the database stored are located
- Load
    - Displays a list of previously played characters based on the player names
- Quit

After starting the game and after the player has made his choice, of either playing the tutorial or skipping it, the player gets to fill in a unique player name and pick a character. As of version 1.0, there was no requirement that the system should connect with facebook, but we have in mind that this shall happen at a later date.

Once the player has chosen a character to play he will be left with these options:

- Quit to main menu
- View player profile
- Play a level
    - Creep spawns
    - Completed
        - Store new high score in database
    - Defeat

Once the player is within a level he can interactively decide how to kill the creeps by touching them with various weapons, and based on this, tally up some score.

2.2.1 Design Demands

It was up to Krillbite to choose the main design of the game. The only requirement was that it should suit the product's title. As the development was under way, the development team was responsible for demanding new graphics once they are needed. Once some new design was implemented it should be returned back to Krillbite in a new build, .apk file, for Krillbite to test and give feedback on the implementation.

- The theme shall be consistent
- Everything shall be scaled to fit a mobile device's resolution and size
- Appealing
    - Font and font colour should match the design
        - Font sizes should be viewable and not too large and not too small
    - Buttons should be easy to click, large enough
- The development team will put various sprites into a larger sprite sheet
    - Using GIMP

2.2.2 Code Demands

- Code-files
    - The top comment of the file includes:
        - Authors
        - Description of the file
        - An example of usage if needed
    - Non-obvious functions and/or members shall be commented
    - Have space between functions

- Code in modules
    - Easier to delegate work between the developers
    - Easily maintainable
- Code-problems and or new finds that might benefit the system as a whole; notify all developers
    - Can be a URL
    - Can be a bug you've bumped into and resolved
- Name variables and functions for what they are.
    - Loop indicators is free to choose
        - can be "i" and "ii", as in index and inner index
        - K or J, or PlayerCounter
- Write understanding and clean code
    - Others outside the development team might view our code in a year
- Folders and files should be named properly

2.2.3 Functional and Non-Functional Demands

Function	Description
Menu	
New Player	A user can start to play a new character
Load Player	A user can load a previous created player
Quit	A user can exit the program, which shall not remove data from database
Settings	
Sound	A user can mute or un-mute sound
Vibrate	A user can turn on or off vibrate
Reset Data	A user can clear all information stored in the database

Game	
Show Profile	A user shall view information stored about him
Change Weapons	A user shall be able to change weapons during a level
Start Level	A user can start a level at anytime he or she may want
Interact with Objects	A user can tap or slash on objects in a level during play

Table 1: A list of functional demands.

Function	**Description**
Run smooth	The game should run smoothly in all levels. Starting up or exiting program can delay by some milliseconds.
Simplified User Interface	The game must be easy to navigate through
Defined Tips	The user shall receive clear messages and clear feedback
Error Percent	Approximately equal to 0. We evaluate this as a small product, so close to 0 as possible in version 1.0 is doable. A prototype might have a higher error percentage.

Table 2: A list of non-functional demands.

2.3 DATABASE

The database is used to store information about a player throughout the game. This is useful to save statistics for a player, updating a player's score and unlocking levels when the player completes a lower level difficulty.

Here we had multiple options and ideas:

- Our own server which ran easyPHP and [5]MySQL. Through Gjøvik University College we have our own web area that we can use as long as we are students at Gjøvik University College.

  From this we had another two options:

  1. Create our own website in PHP and list out information about players that have played the game and their score, avatar and nickname.
  2. Connect to facebook and grab statistics from our server to be displayed on a person's facebook account.
- Store database in memory on the android phone. This would lead to increased speed when reading and writing, since the data never reaches the disc, but we would lose all data stored after the game closes or when the battery runs out of electricity.
- Store database on the phone's hard drive, taking up space, slower read and write as the write is done per disc cycle, slowing down inserts greatly.
  - Workaround for increasing insert statements would be to insert multiple inserts per execution of the insert by using Begin and then add the insert statements and then use commit to commit all of the queries.
    - The speed was not a problem up to this moment, due to the fact that there are not many insertions per player, and they happen only once, at beginning of player creation.

Among these options we used to go with the last one, storing on the phone's hard drive. A model of the current database can be viewed below in figure 1.

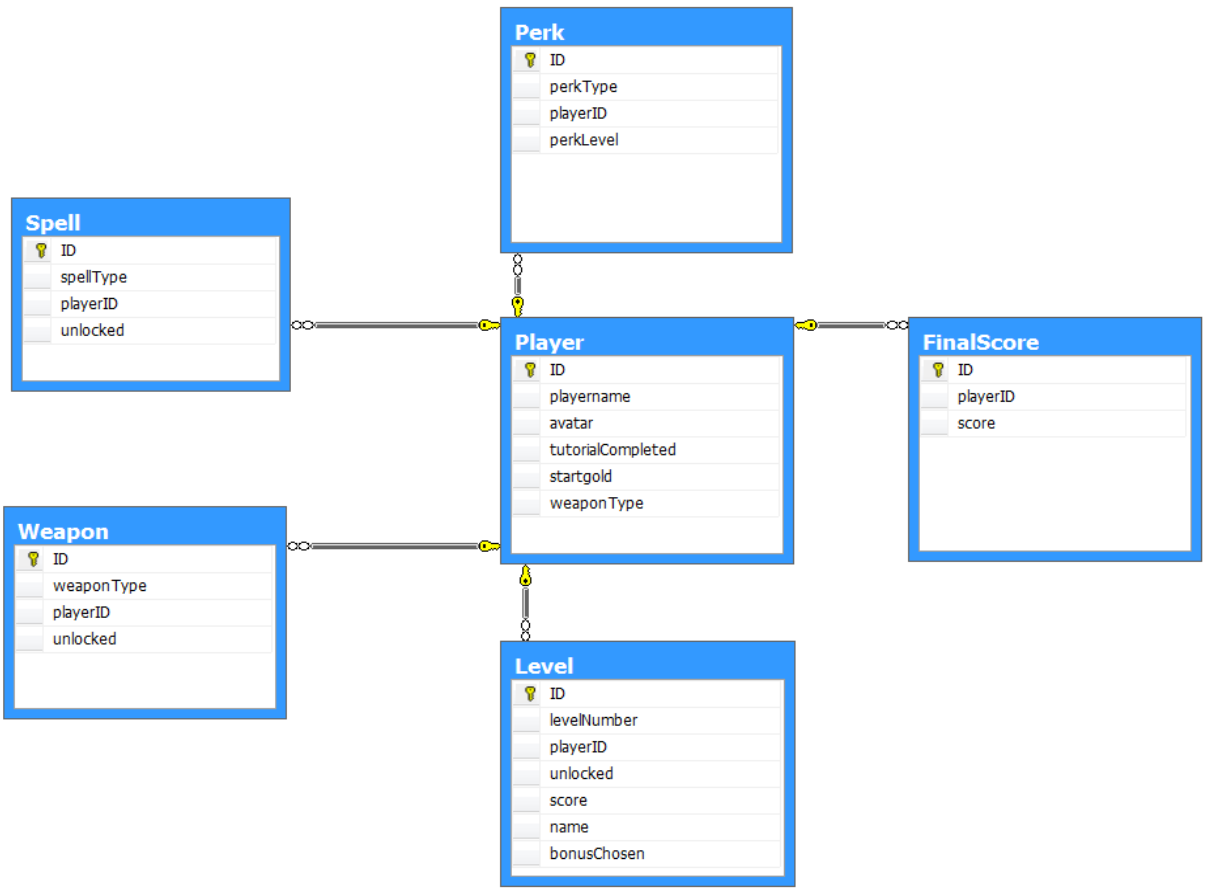


Figure 1: Shows the model of the current database tables. Perk, spell, weapon and level has a many to one relationship. A player can exist only once in the Player table, but can have multiple weapons and therefore multiple entries in the weapon table. FinalScore-table can only have one entry of a player and it is a one-to-one relationship with the player-table.

All tables have changed throughout the game and the model shown in Figure only shows the current version. But after the last feedback from Krillbite the database will change into the model as can be seen in section figure 12, 7.3.2 Database.

3. PREPARATIONS

3.1 RESEARCH

3.1.1 ANDROID VERSUS IPHONE

Android	iPhone
Simple process for releasing applications means a guarantee that our game will be released into the market.	Applications released in the apple store have to be checked and approved by apple.
Open source. Applications gets free rein assuming permission is given from the user.	Applications are restricted on multiple areas including running in the background, communicating with other applications and more
Android applications can be developed from any kind of computer	iPhone applications can only be developed using a MAC. No group member owns a MAC.
Android phones where available for loan at the school	No easy way to acquire iPhones, as none of us owned one.
Some of the group members had a little experience developing for Android	No experience with Apple development on the team

Table 3: Shows pros and cons for the android against the iPhone.

We decided on using Android as our development platform very early. During the course of this thesis we wanted to make a game that would be completed at the end and ready for release. For a small project with such a short time frame such as this, developing a game for a mobile platform which are populated by easy-to-pick-up, less complex games seemed like a good plan to make sure we could reach this goal.

Another advantage in using Android was that mobile devices are a growing platform, for games in particular. We had a little experience programming for Android but none in making

games for it, or any mobile device for that matter. Having some experience within this field could be very valuable for us in the future when working with game programming.

3.1.2 Unity against Corona

One thing we learned from the Game Programming class was that creating a game completely from scratch without any kind of library to work with was very time consuming. As this project had a relatively short time frame, and we wanted to make a game that was ready for release. We decided to find a good premade library to use for this product. For a library like this, we looked for a couple things that it must have:

- Loading of images, and playing sounds
    - Any image format
- Loading and playing of sounds
- Library that was meant to be for games
- Events for click, touch, tap

After looking around for a while we decided on using [2]Corona. The first reason was that Corona is a fairly cheap software development kit(SDK) compared to the others we considered. The library is free while you are developing, and costs 200 USD per year if you want to release your product. By comparison, Unity3D costs 400 USD and if we choose that we would probably need one each, a student license costs 100 USD, but then you cannot release the game, but you will get the full version of the Unity3D, everything included.

The Corona library was also very easy to use. The simulator has a very clean and simple interface, and the library is the right size for our project in the way that it includes everything

we needed. Another advantage of using Corona was that it can compile your code to either work for Android or iPhone, so if we later wanted to release the game for the other platform, we would not have to do any extra programming.

3.1.3 Integrated Development Environment

3.1.3.1 IntelliJ IDEA versus Notepad ++ versus Sublime Text 2

Before planning ahead a good tool at hand is one of the primary needs to reach the goals by the time in a programming task. We sat up a requirement list that the tools must have at hand without too much fuzz to get things working:

- Lightweight text editor
    - Easy to set up
    - Fast to download and install
        - Reinstalling is also fast if it is needed
- Syntax highlighting
    - Highlighting at least a few key words in programming, or highlighting Lua language
- Panel displaying all files and folders
- Intellisense
- Free
- New
    - None of us should have used it before, learning a new tool is always great

On this basis and after a lot of researching and reading articles up and down, we were interested in developing within one of three different text editors: [3]IntelliJ IDEA, [12]Notepad ++ and [13]Sublime Text2.

Notepad ++ does not deliver a panel where a structure of the root of a project is shown and all files and folders within it, and it was quite a hassle to figure out where all files were, due to the fact that they are all shown in a top bar

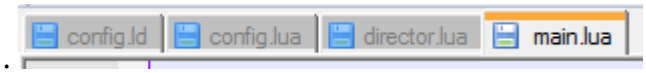


Figure 2: Figure shows the top bar in notepad ++

Sublime Text 2 had a decent look and feel at the first start-up. It had a nice overview of where in the code we were based on a panel showing on the right side. The tabs in the top bar showing which file we are within is also highlighted better than in notepad ++, as can be seen in figure 3 below.

Sublime Text2 also has support for projects which makes it easier to find files in a structured way.

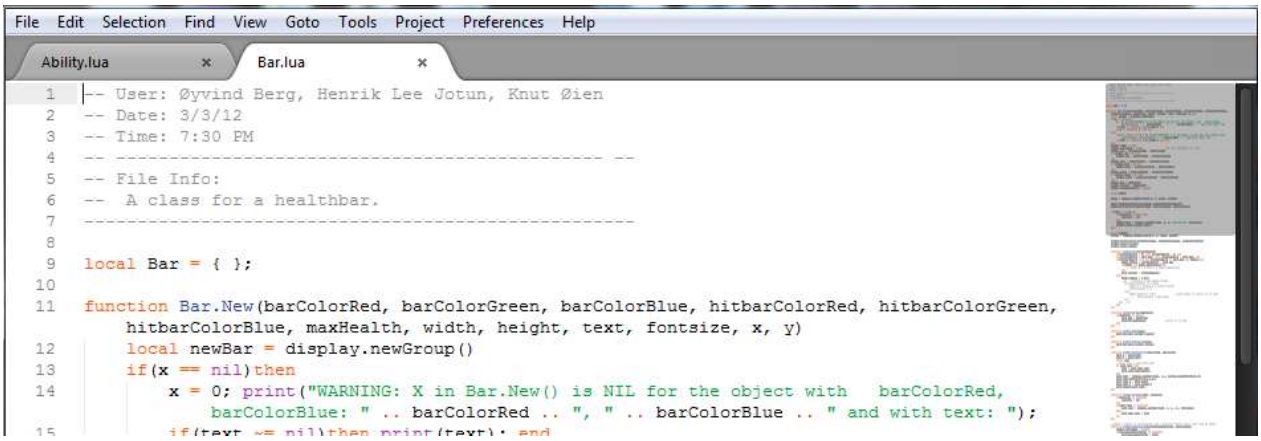


Figure 3: Shows inside Sublime Text 2.

IntelliJ IDEA is a bit larger tool than a simply lightweight text editor. One can see it has a lot of options and settings just by looking at it. But it does have a nice overview of where our files are, which file am I in and it also does Intellisense for the file you work on. An image of the editor can be seen in figure 4 below.

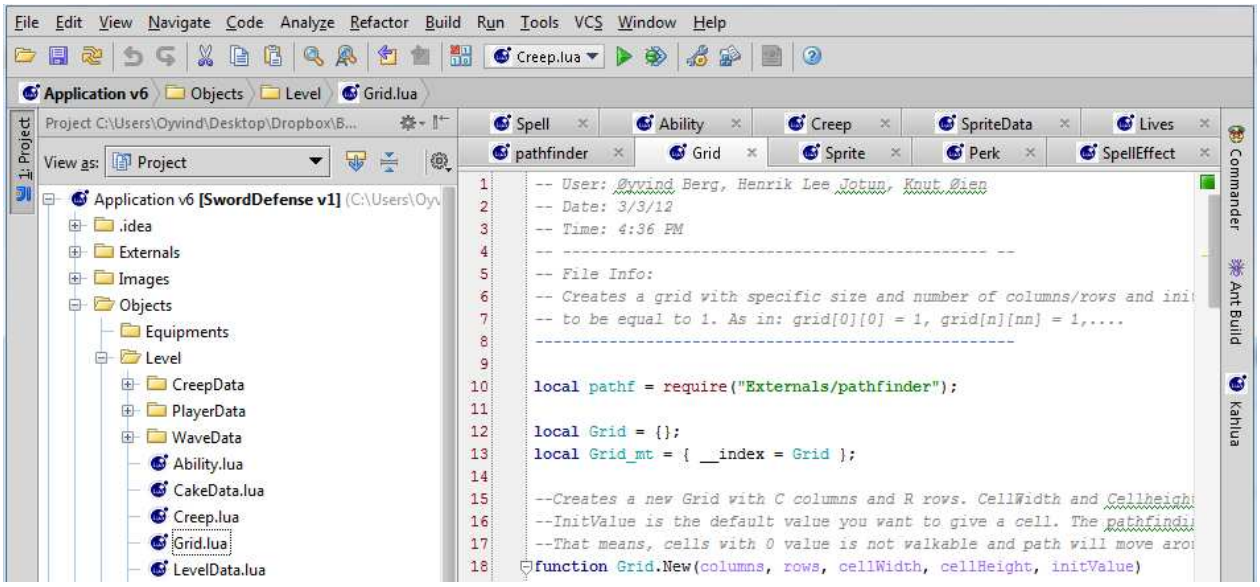


Figure 4: Showing inside of the editor IntelliJ IDEA

3.1.3.2 WHY INTELLIJ IDEA

What we were looking for in particular was an editor that would handle auto complete for Lua and Corona functions, and that could handle multi file projects well. Based on this Notepad++ was out of the question. IntelliJ IDEA was bit too heavy for us, we thought, so we did further research on Sublime Text2 and found a nice article and a 7-step tutorial to get Intellisense for Corona frame work within Sublime Text2.

Tried to look further into Sublime Text2 and what it could do for us. At the same time the group multitasked to test the other two editors as well. Testing [7]IntelliJ IDEA and all their settings and but also keep looking at other tools and scouting for plug-ins for Notepad ++.

Trial and error and suddenly Sublime Text2 would not start, leaving that out of the way. Notepad ++'s plug-in did not quite work as good as expected, so we settled down with IntelliJ IDEA on this very basis. Plus working in a larger tool feels more powerful and IntelliJ IDEA

is not too large for this project. In addition to that, we did discover a plug-in for the IntelliJ IDEA , read more about this in 3.3.2 Additional Tools

3.1.3.3 LUA

Lua is the programming language used with Corona. None of us had any experience with this, so we set out to learn it right after we had decided on Corona as our library. Having a background in C++, it was fairly easy to pick up the basics of [8]Lua, as a lot of the base structure and syntax is very similar. There is also a good bit of information on it on the web, which is mainly how we learned the language. One of the things that worried us early, was that there are no built in classes or structs in Lua. However, upon doing some research we discovered that you can use tables and make them act like a class or struct.

We found many ways to do this. Since none of us had the experience to know which one was best we tested some of them. Of course it a matter of taste which one you pick, since they are very similar.

```
Player = {}                 -- Create the Player table
Player.__index = Player     -- Create meta-table
-- CONSTRUCTOR --
function Player:new( playername )
    local new = {}                      -- new Object
    new.name = playername
    new.score = 0
    return setmetatable(new,Player)     -- make Account handle lookup
end
-- FUNCTIONS --
function Player:setScore( number )
    self.score = number
end
```

Snippet 1 Shows method 1

Method 1

The method shown in snippet 1 works as a struct. It is not a very effective representation because the constructor can be run after the object is created. The reason for this is that it is added to the created object. The object is created when you run the new function and it will return the new object with a meta-table. In this way the new table can get the Player table indexed to itself which in turn give you access to the new tables values. You see an example of this in *snippet 1* where the Player can get "self.score", something that otherwise would not be possible. However by using this method you can avoid defining functions inside the constructor. Adding functions in a way similar to C++.

```
-- The Player function will act as a constructor
function Player( playername )
    -- The new object
    local new = {}
    new.name = playername
    new.score = 0
    -- Add the setScore function to the new table
    function new:setScore( number )
        new.score = number
    end
    -- Return the new table, this will be your object
    return new
end
```

Snippet 2: Shows how to make a struct without a meta-table

Method 2

The method shown in snippet 2 is very similar to the first, method. The difference is that the functions you want included must be inside the body of the Player function. If you do not like the thought of declaring functions inside functions this is not for you. However you can only call the constructor once pr object because it is a function of its own.

```lua
-- The Player function will act as a constructor
function Player ( playername )
    -- PRIVATE --
    local private = {}
    private.name = playername
    private.score = 0

    -- PUBLIC --
    local public = {}
    function public:setScore( number )
        private.score = number
    end
    -- Return the public table, this will be your object
    return public
end
```

Snippet 3: Shows how to create an object and call it's function

Method 3

Method 3, which is shown in Snippet 3 is similar to the second method, but this one will more accurate represent a class. With this method you can keep data private if you wish. If you want to manipulate them you will need get and set functions for them.

```lua
-- Create a Player object
local player1 = Player:new("player1")
-- Call the setScore function
player1:setScore( 100 )
```

Snippet 4: Shows how to create and object and call its function

Even if the method declarations look different the creation of the object can look the same. In the *snippet 4* above you can see how an object is created, which was shown in method 1. The other two methods are created without the ":new" in "Player:new("player1")". If you want them to be the same all you have to do is to add the functions in method 1 and 2 to another table and rename the function name to "tablename:new". In the same way method 1 can be made to work the same way as the other two if you put that inside a function. The way you create them can be the same no matter what style you use.

3.3 Tools

3.3.1 Problems with Corona – Building onto a physical device

Corona has turned out to be a good and easy-to-use framework built on C++ and OpenGL. However, we did encounter some issues when we attempted to [18]install our application on a physical device for testing. These errors have caused us some delay as they often do not give any error messages, and it's therefore hard to track down their cause.

The free version of Corona does not support folders when building to a device, although it does support it for the simulator. When building to an actual phone everything has to be in the root folder, which also means that every path for every file and picture, as written in the code, has to be changed. Also, the simulator is not case sensitive in regards to file names, however, the phone is, and having a file name wrong in terms of case sensitivity causes the phone to report the apk-file as corrupt.

One of the other things that we found would crash the application once it was installed on the phone, but had no effect when running it on the simulator. Was having empty folders in the root folder when you build the application and including files types that Corona did not recognize (even if these where not in use). Also, small errors in the “build.settings”-file would still run on the simulator, but would cause the phone to restart itself every time you tried to install the application.

3.3.2 Additional Tools

Additional to IntelliJ IDEA and Corona we did further research on other plug-ins existed, we still had requirements for the plug-ins to bother to use them:

- Useful for this project in the long run
- Not complex

- Easy to download
- Easy to install

On this basis we found [11]Lime, also created by the Ansca Mobile, that is an expansion from Corona found at this url: http://justaddli.me. We would like to quote from their site:

"Before purchasing you must be fully aware that currently there are performance issues when running Lime on hardware. These issues are getting looked at by myself and Ansca so that they can get sorted as soon as possible. ".

Because of that simple quote from their site and the two facts below we did not delve into it further:

1. We could not find a usage for it
2. It is not free

What we did find out was that Corona has a free open "Share Your Code" site which works like anyone can upload any code snippets and share them with the world. This can be found at the URL: http://developer.anscamobile.com/code/. Here we found Ricardo Rauber's director module free to use, more about this can be read 5.2.1 Scenes.

Additionally we needed a tool to transfer our builds to the android device and we first found HTC sync which installed with ease, so there was no need to investigate further into other options.

A member on the group have sent an email to the Corona team asking if they ever is going to implement so that Corona automatically builds the apk-files both to the computer, but also transfers the apk-file to the phone. They answered that they are looking into it, but should not expect it this year.

Word versus Latex

For documenting ideas and report writing we have looked briefly at Latex, but come quickly to the decision that it is no need to learn latex when we are comfortable with Word and other text editors.

Code Documentation

To get an outline of what a code file does it is important to print out commentaries and functions that are defined within a file, also class and struct. Read more about our code documentation in chapter 5.3 Code Documentation.

3.3.3 Build Tools

Compilation requires only corona to build APK files. To transport the APK files to an android device we have used a free tool named HTC sync. By inserting a usb cable from the PC to the mobile device we get access to transfer files from our desktop to the phone. A flow chart is shown in figure n

Figure 5: Shows the flow process from computer. On the computer we have our project folder, which is created and edited by the development team in IntelliJ IDEA. Building the project is done in Corona Simulator, which outputs an .APK file. HTC sync is also installed on the computer and can transfer this file to the android device.

3.4 Backup – Dropbox vs. GIT vs. SVN

3.4.1 Introduction

When working on a larger project with a team, it is important to be able to work on the same file at the same time without deleting others code. This allows a single person to be able to work on any file or piece of code without fear of losing it. The benefit of this is that the programmer gets more choices on what to do. If you are consistent with "TODO" markings in your code no one will be uncertain on what to do.

3.4.2 GitHub

GitHub gives you a free account for open source projects. If you do not want your code open to the public you need to buy an upgrade. The amount of repositories and collaborators you can have private depends on how much you pay each month. If you are a student you might get a free private repository for as long as you remain a student of the school. Bandwidth consumption is limited due to the fact that is used by many. If you consume a significant amount over the average GitHub user your account might be disabled. This would probably never affect us since we worked on a small project.

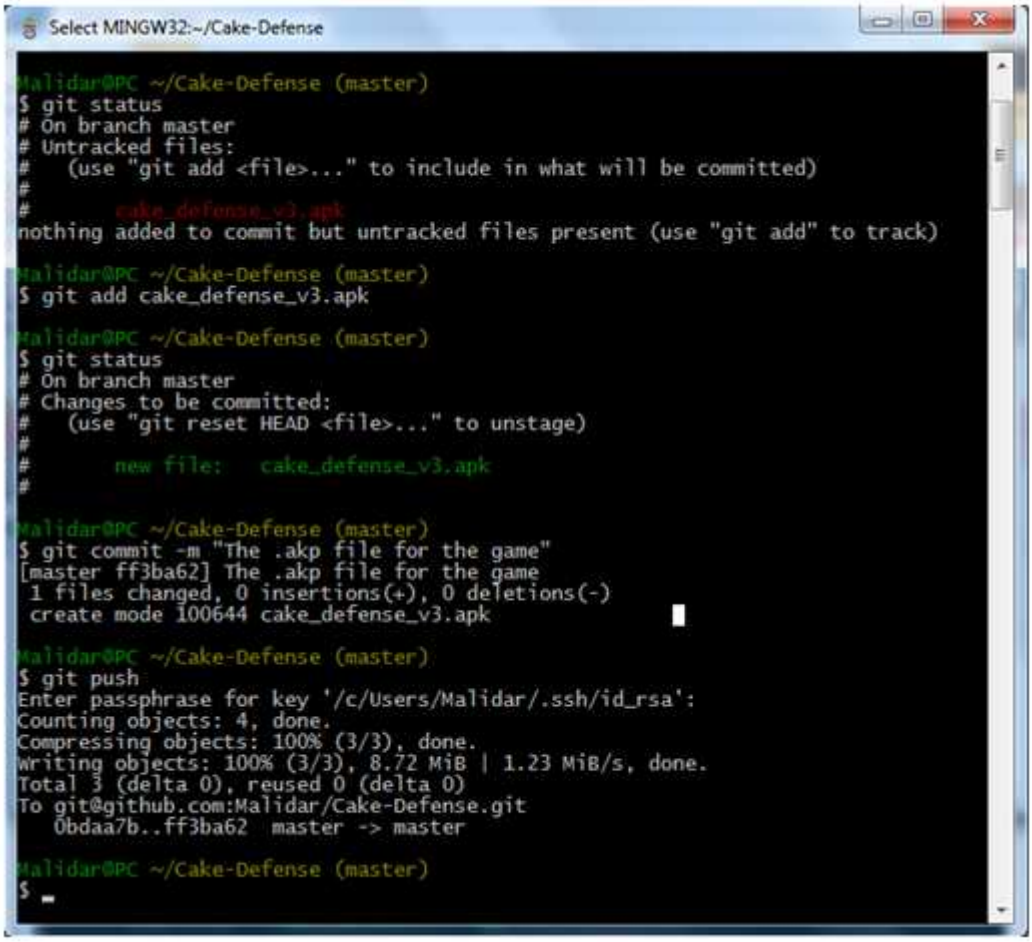


Image 1: Adding, committing and pushing commands.

Everyone on our team has previous experience with Git/GitHub in our previous semester. It ended up being more trouble than it was worth. Since it was a lot to learn before you can use it properly. For example when your version in the repository was out of date, and you got a merging error. You do not really know how to resolve it without resetting it to a previous version or delete your own version.

3.4.3 Dropbox

Dropbox is a very easy backup tool. It provides you with 2GB free cloud space. This can be increased to 100GB if you pay. You do not have to commit manually since it will synchronize all files in the cloud automatically. The term "commit" will still be used to describe a person's changes. You can manually set the maximum bandwidth use since the syncing might happen.

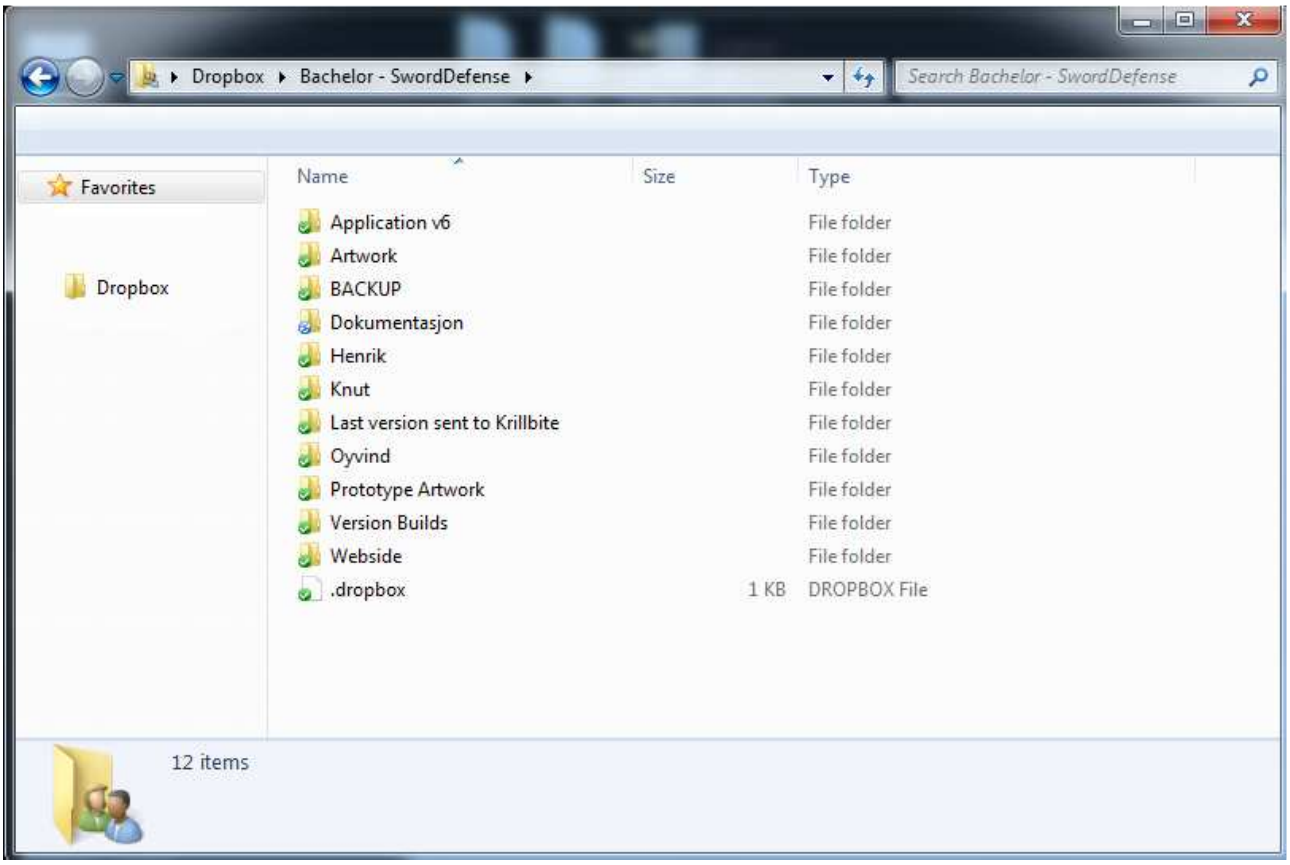


Image 2: Shows our Dropbox folder and folders at root

It does have some version control, since it keeps a backup of your commits for the past month. When a file is changed it will only add/merge the changes instead of the entire file. This makes it a lighter load on the bandwidth. However if there are multiple persons working

on the same file the one making the last commit will overwrite all the others. There is a safety- net (with big holes) to try and solve this problem. If someone changes the file you work on and you are currently using the file. The other persons commit will be saved as a separate file (a conflicted version).

3.4.4 SVN

SVN or Subversion is an Open Source Software versioning and revision control system. Its primary function is to assign unique version numbers to software and keep a full revision history.

It is very similar to Git, not GitHub since it lacks the social network.

SVN requires that we have our own server to commit to, our own repository. The school does offer one, but committing and merging does require a bit of trial and failing, just as GitHub does.

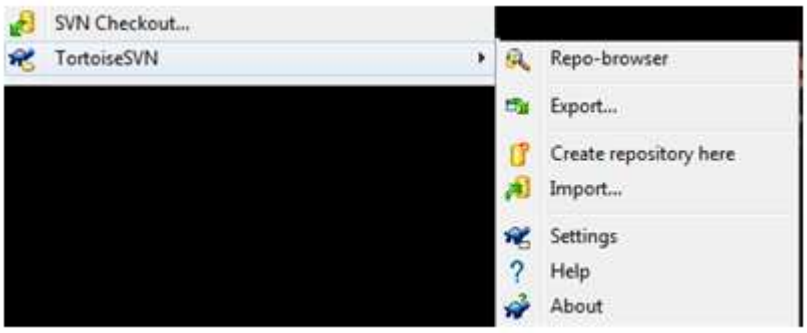


Image 3: options when right clicking a folder that is not in your repo.

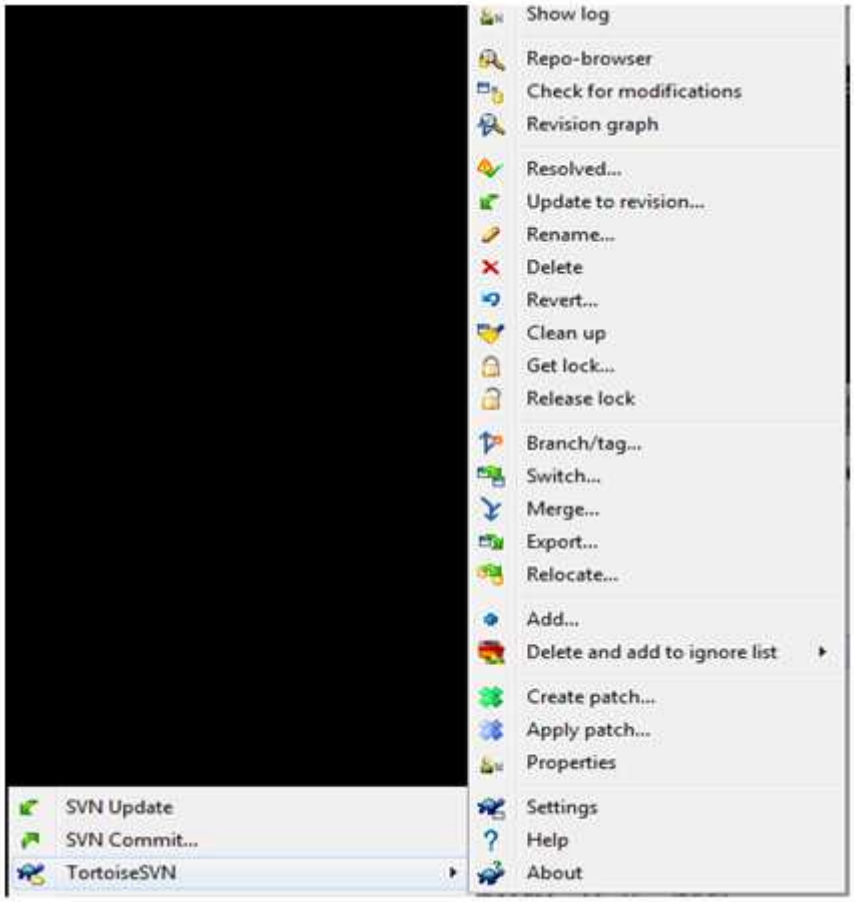


Image 4: Options of a folder that is inside a repo.

Backup tool	Pros	Cons
GIT	• Being able to work on the same file at the same time • Others will not accidentally delete your work • Can create branches	• Takes time getting used to • Merging errors might be hard to fix without loss of data • Our code would be open for everyone to

	from a repo, and merge when ready	see, unless we paid for it
SVN	• Easily viewable what's gone into the commit, if giving it a clear and good comment when committing. • The SVN tortoise comes with a nice shell. Which simplifies committing by just right clicking on the folder and choose "commit", of any other option? • Atomic operating: If an error occurs; abort whole operation, else continue operating	• Committing will give you errors that can be time consuming to fix. • Not as easy as Dropbox
Dropbox	• Very simple to use • It has a backup with a month of your last commits on dropbox.com. • Free cost	• Cannot merge files. Instead it overwrites the oldest file with the newest file. • Not able to work on the same file at the same time. • Overwriting others code is easy to do.

Table 4: Shows a list of pros and cons for various versioning control systems.

The reason we chose Dropbox, was because we are a small team on a small project. We could split the work among us in such a way that each of us had a standalone piece of code to work

on. After finishing the code we added it to the Dropbox version, which is basically putting the file in the correct folder. We also had our own folder in Dropbox with our own code. It was very simple to use, but in turn we had to be extra careful when merging our code, to not overwrite or remove files based on who's the newest, which can be read more about in chapter 8.2 Backup.

3.5 Development process

Group followed the [1]scrum development process with these regulations:

- Each sprint is a week
    - The time frame of this project is limited; we considered that having more short sprints would benefit and also making us skip or find solutions to problem earlier.
    - Project is built upon small modules, which makes it logic to have a short sprint period
- Daily scrum-meeting every work day. Discussion of problems and solving them will be reported and done here. Only issues would be logged.
- No scrum master

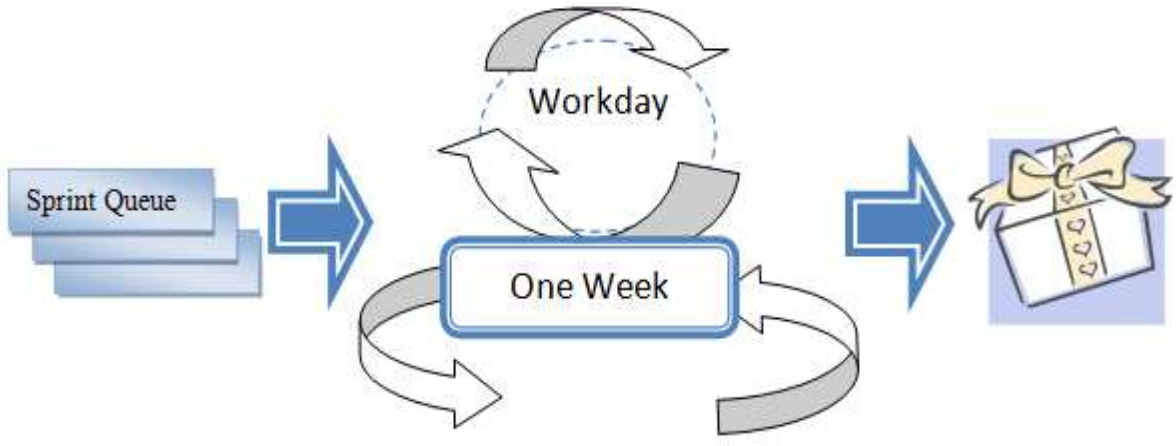


Figure 6: Shows our own model of sprint

Our own model of sprint is seen in figure 6, where each workday is a daily meeting and by the end of each week a new packet arrives that is complete, and then a new box from the sprint queue is taken and this continues on and on throughout the project.

In addition to the smaller scrum meeting we had each week day, we had a meeting each Friday where we discussed what we had done the last sprint. In the meeting we discussed problems we had encountered and made sure we were up to date. In these meeting we also determined what work needed to be done the coming week and divided it between us. A description of what we did during each sprint can be found in appendix – Sprint, page 17.

In terms of design we looked at previous successful mobile platform games and tried to figure out what made them popular. Common themes seemed to be that the games were easy to understand but had a high skill cap. A part or level of the game could be completed in a short amount of time and the game had replay value by having a scoring system. This way you could continuously challenge yourself. In ways of graphics it seemed like clear bright colours was popular and we also noticed that the games had a lot of flashy feedback and big numbers popping up on the screen when the player did something right.

When we were ready to start coding the game we discussed which classes/objects we would need and started coding them one by one, trying to code the ones that had to work closely together at similar times. The work was divided in such a way that each person in the group had their own files to work on. We figured out what functions each class needed and the person responsible for that class' file made it. Splitting the programming up like this solved the issue of different coding styles and also allowed us to use a very light kind of version control, as there where never more than one person committing code to a single file.

4. DESIGN

4.1 THEME – SWORD DEFENSE

This is the theme we first thought of after we had developed the idea. With slashing the creeps being a big part of the game, it was natural to think of swords, and then the medieval age. This includes castles, towers and appropriate enemies to fit that era. This is also the setting in which most tower defenses are set. Part of the reason for this is that tower defense games where populated largely by the RTS game Warcraft 3, which also revolves around the same theme.

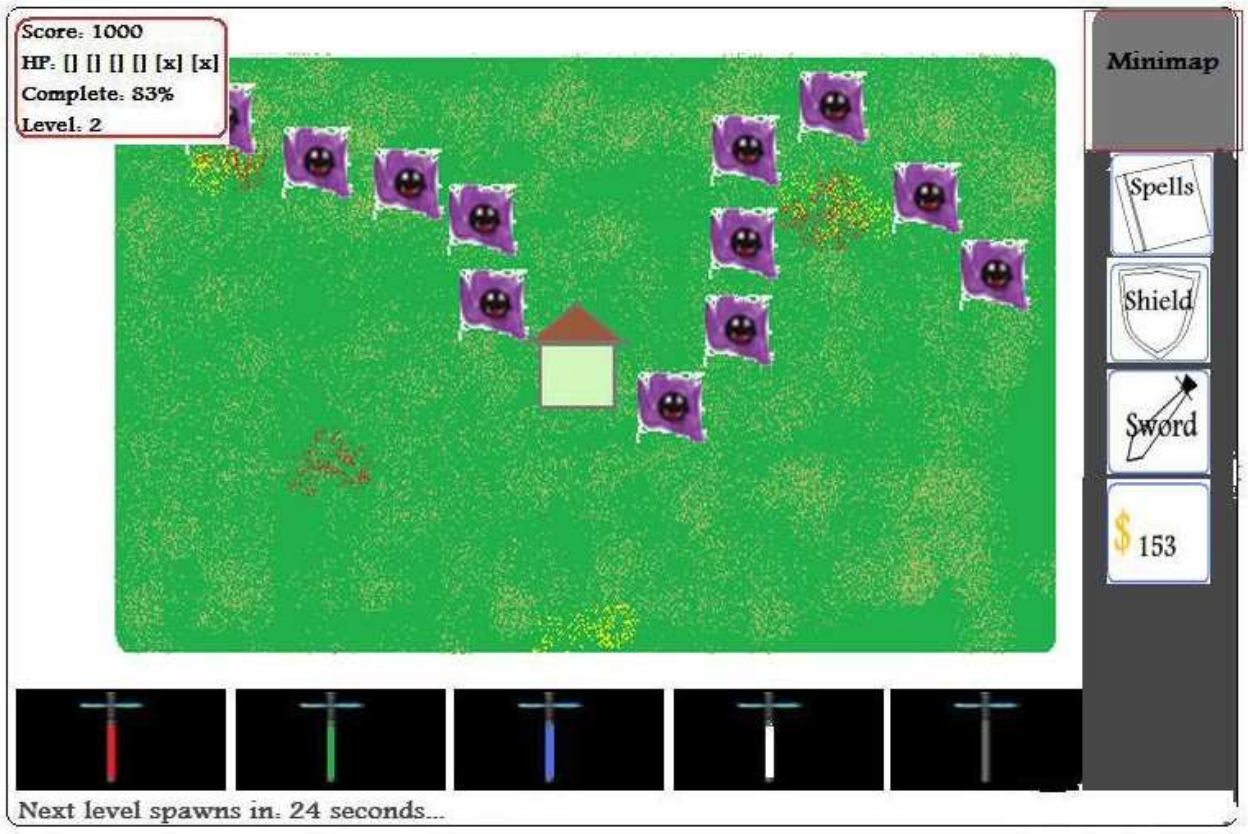


Image 5: Shows a sketch from and for what ideas we were going to implement.

4.1.1 Theme needed to change

When we discussed it within the group we agreed that our idea might be a little too narrow. We thought an idea like this might appeal mainly to teenage boys. As we wanted to hit a much wider demographic and make a game for “everyone” we asked Krillbite for some ideas regarding changing the theme of the game, without having to make any huge changes to the game play. See 5.2 New Theme and New Name for information about why and what we ended up with.

4.2.2 Planned Game Mechanics

- You swipe at the creeps to kill them.
- Your swipe does damage based on the level of your sword and the amount of power you have at the time
- When you swipe you are hitting with a sword
- You can change swords and the swords have different effects
- Swords can be upgraded in the shop
- Your score will be determined throughout the game.
- You will get more score the less money you use
- You can replay levels to increase your score
- Level score and total score are separate values
- Spells will be cast by drawing shapes on the screen
- Levels shall be selectable one by one and be able to play same level multiple times

- The player can choose an avatar which can have a minor effect on the game

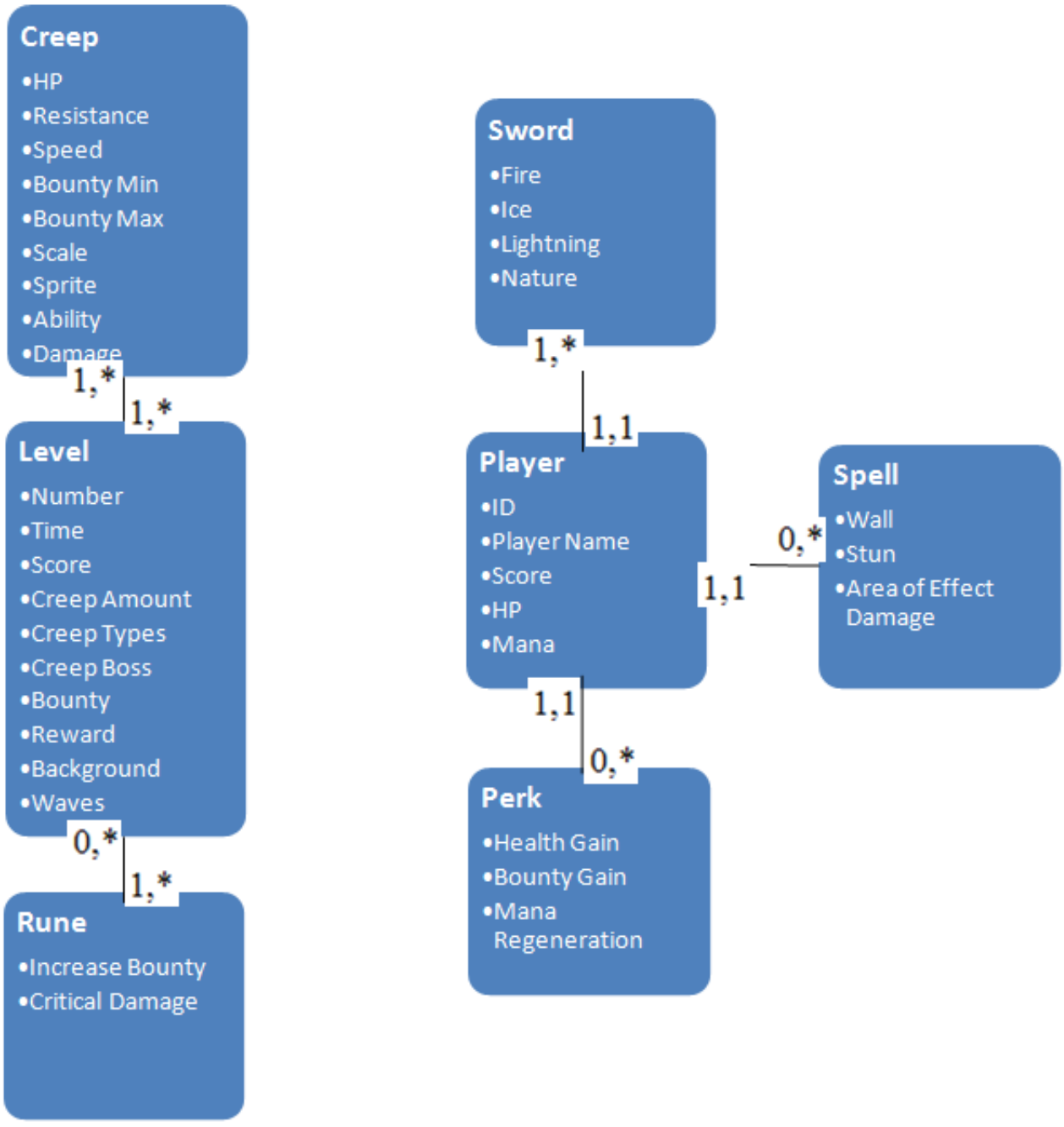


Figure 7: Shows a class diagramme of the classes thought to be implemented at the design phase.

The class diagramme (in figure 7) shows what modules that would first be needed to implement to get the basis of the Sword Defense idea. This later changed into Cake Defense and this class diagramme have changed in the implementation phase.

See chapter 5 Implementation for a read on what has been implemented into the product.

4.2.2.1 Story

As of now we have no specific story in mind for our game. We could make a small story based on the art direction that is to be decided later. The story would not be more than a small motive for your actions. Examples of possible stories:

- Protect princess in tower from other heroes/dragons/etc. Medieval theme
- Prisoners are trying to escape the jail, kill them before they can get to freedom
- Criminals trying to steal gold from a vault/goldmine

5. IMPLEMENTATION

5.1 INTRODUCTION

This chapter takes you through some of the struggle with code we have had and how we have implemented various stuff like spells, runes, creeps and level selections.

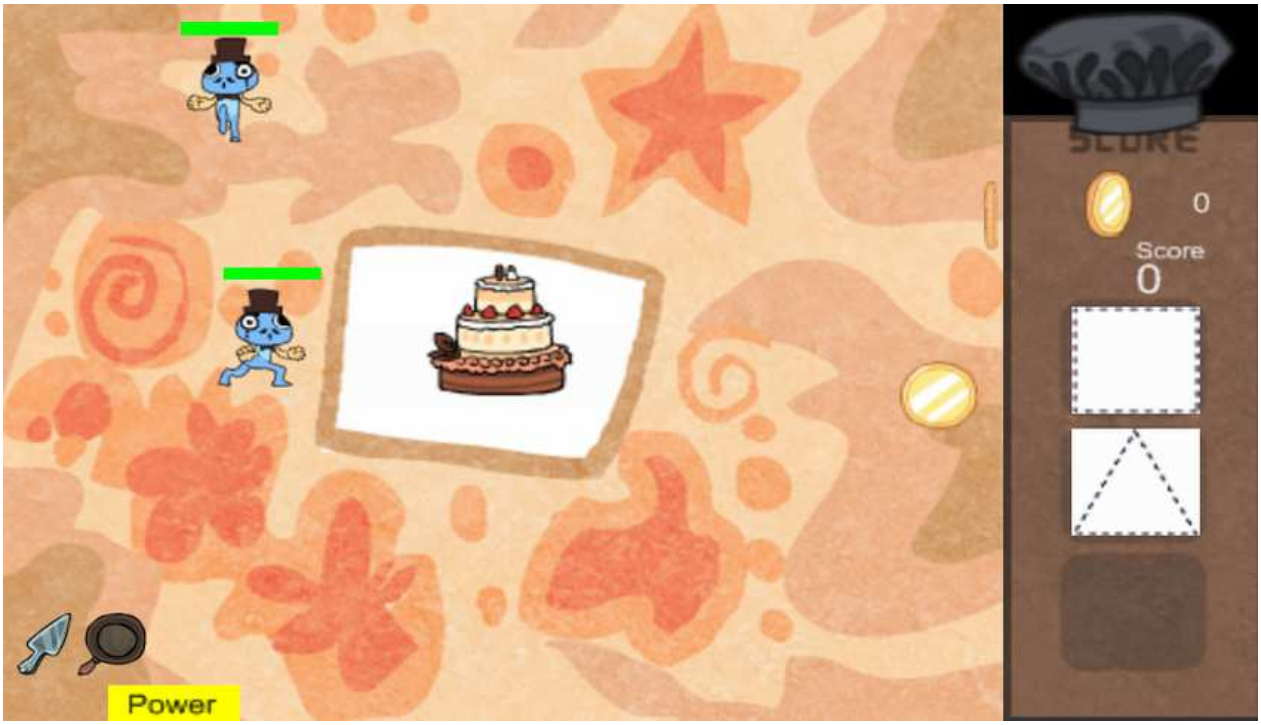


Image 6: Shows the current version of the game. Creeps are running to the cake. As one creep eats a cake, the height of the cake is decreased, once there is no more cake, the level is defeated, otherwise the player manages to kill all creeps, he have finished the level and another one unlocks.

5.1.1 Flow of the Implementation

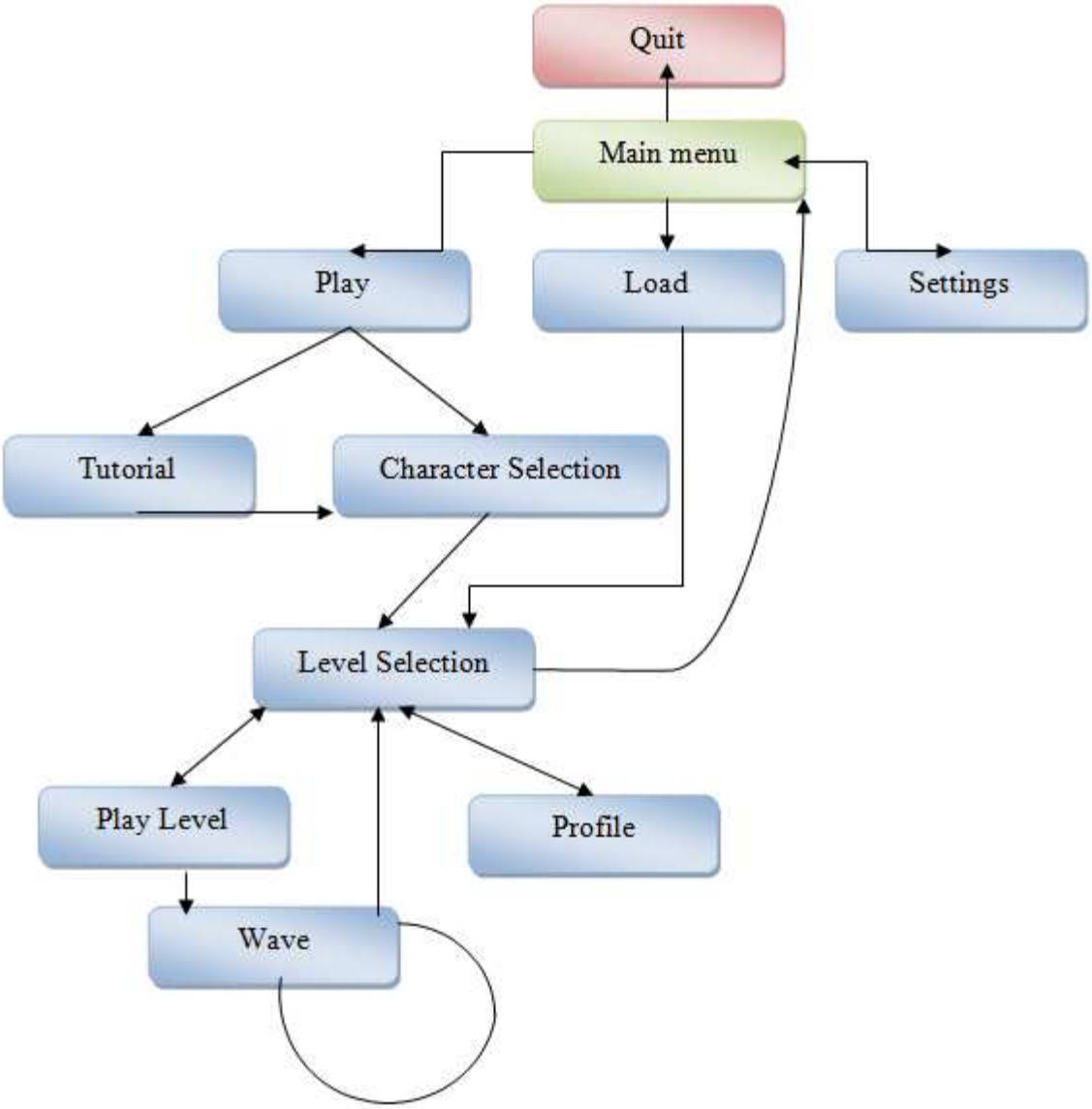


Figure 8: Shows the flow of the current version.

As seen in figure 8 the flow starts in the green cell and is complete in the red cell. By implementing this flow as a priority we would ensure that we would reach the goal of having

a functional prototype by the end of May. By following the philosophy of implementing simple, but useful features first to get this prototype working as a tower defense game. While refining and streamlining happened afterwards, we got a prototype up and running by the first month.

This was a parachute for us to fall back on if changes would come or something unforeseen would appear to slow down or require more time than estimated.

5.2 New Theme and Name

Cake Defense

After talking things over with Krillbite and letting them know that we were looking for a new theme, they came up with the idea of Cake defense. The game now revolves around defending a cake which the monsters are trying to get to so they can eat it.

Krillbite sent us some preliminary artwork to show off the idea and we immediately liked it. We were pretty sure this art direction will let the game hit a much wider audience. The result is a clean and simple look with bright colours and a light-hearted playing environment. The sprites for the creeps, which we received later, looks funny and goofy, which makes them fit the game perfectly as it is now.

Image 7: Various creep types

Image 8: A creeps death animation

5.2 Implemented Game Mechanics

This section is about game mechanics implemented into the product as of today.

5.2.1 Scenes

When we started using Corona we needed a easy way to create and change screens for our app. The first thing we found by searching the web was [9]Ricardo Rauber's Director Class. This is a very popular 3rd-party open source library. It helps with scene management and also allows you to transition between scenes with or without effects. There is also a library that comes with Corona. The Storyboard API is a relatively new addition to Corona and it does everything director does. However it is far more complex compared to director. This is one of the reasons why the director class is still a popular choice for developers. On the other hand if you need more options on how to manage your scenes Storyboard is for you.

Storyboard lets you do some things director do not. This is only a few of the things the storyboard can do:

- Preload a scene before you transition to it.
- There is a function that will effectively create a pop-up screen.
- There is a function that lets you print Lua memory and texture memory usage information in the terminal.
- Purge all scenes except for the one showing.
    - Change scene to the previous or the next

Of course you can get the same effects in director, but you have to create them manually.

We early decided to use director, because it is very simple to use and understand. It provides all the functionality we needed to manage [10]scenes. The storyboard is also a good choice, but it is such a new addition to the Corona SDK a lot of its functions need a fairly new version of Corona. We have an older version of Corona. The reason for this is because if we updated to a newer version we needed to redo a lot of the code we had written.

There are a bunch of new features in the director class as well. We do not have much experience with these new features, mainly because the old features cover all of our needs.

5.2.2 Weapons

The reason we called the game sword defence in the beginning was the fact that the sword was the only intended weapon in the game. The player would have four different swords to kill the creeps. Each sword would be "infused" with a natural element (Fire, Nature/Earth, Lightning/Air, Ice/Water). In addition to the swords you have, you would also be able to equip two swords at the same time, one in each hand. With two swords equipped you would be able to execute combos to do more damage to the creeps and get a higher score. The way this worked was that each sword had a certain effect it inflicted to the creeps hit. Related to the element of the sword the effects were:

- Fire: Inflicted a burn on the creeps to create a damage over time(dot)
- Ice: Slowed down creeps freezing them for a while
- Lightning: Stunning the creeps for a brief duration
- Nature: Inflict a poison to the creeps (weaker combination of Fire and Ice)

These effects had varied effectiveness depending on the resistance of the creep. Example. A water creep would not take damage from fire.

Weapon slot	Fire	Ice	Lightning	Nature
Fire	Extended Effect	Remove effect	Push back	Small explosion
Ice	Remove effect	Extended Effect	More damage	Remove effect
Lightning	Push back	More damage	Extended Effect	Paralyse
Nature	Small explosion	Remove effect	Paralyse	Extended Effect

Table 5: Shows various weapons and their effects and combo effect that could be triggered by slashing with one weapon first, then another weapon right after.

In addition to the effects each weapon deals a base damage in their respective element. This is the damage done by swiping on the creeps. We early discovered that if we did not put a restriction on the damage done, you would sweep / tap the creeps as fast as possible to kill them. This was not the play style we wanted. The swiping should be more strategic, the solution we come up with was to introduce the power bar. The longer you touch the screen the less damage you will do. This prevents wild swiping and results in shorter and more accurate slashes by the player.

The base damage of the weapon was upgraded in a shop with the gold accumulated by killing creeps. The shop was in the HUD, and could be accessed by pressing the dollar button next to the gold count.

Concept from Krillbite

After sending our second version of the game to Krillbite, they wanted a new concept for the game. In this concept the creeps are hungry for cake and you have to defend it. To make the weapons fit the new concept of the game, we removed the swords and replaced them with a cake shovel a frying pan and cake sprinkles. This was also to fit the new concept they had

created for the creeps. They wanted fewer creeps, which you could interact with in different ways.

For example, to overcome the flying creep you would have to smack it down to the ground with a frying pan before you could kill him with the cake shovel.

The weapons ended up being these three:

- Shovel: works the same way as the swords. Swiping with your fingers.
- Pan: tap with your finger to deal a smaller amount of damage over a small circular area.
- Sprinkles: swipe around the creeps you want do deal damage to. Has limited uses.

After these weapons were introduced the ability to upgrade your weapons in the shop was removed. The cause was the fact that the new creeps did no longer drop any gold bounty.

Swipe and Slash

The swords needed an effect to make you feel like you cut the creeps in half. The effect we wanted was the same as the swipe feature in the popular mobile phone game Fruit Ninja. The swipe effect that creates a line where you touch the screen and fades away to make it look like you cut something.

To implement this we searched up a tutorial [16]. This tutorial gave us a simple way of crating the wanted effect in Corona. The swipe was implemented to change colour based on the weapon currently equipped.

5.2.3 Creeps

5.2.3.1 Information

In the beginning the creeps were to have different resistances and spawn in waves. Just like your standard "tower defence" game. The last wave of a level would spawn a boss creep. This creep would be bigger and stronger than everyone else with higher resistance levels.

Levels would spawn creeps in waves and each level had a set amount of waves you needed to clear in order to win. The next wave that would spawn was displayed in a info box in the HUD. This would let you plan for weapon changes in head of time. And to show how far you have gotten trough the level. The creeps' health would increase with each wave. To be able to keep up the creeps they give the player a gold bounty for killing them, thus allowing the player to upgrade their weapons. If a creep reached the "castle" it would deal a certain amount of damage to the player. Some creeps did more damage to your health than others. The player's health was represented by a health bar in the HUD. There was also an early idea of giving some creeps abilities like teleportation***. This was implemented as an experiment. But it was later forgotten.***

One of the things Krillbite would provide for us was the sprites. The sprites Krillbite made for our game was created to fit their new concept for the game. The sprites were great, but using them the way Krillbite intended also meant changing the game play. This new concept included removing the resistances and elemental damage in the game. Replacing it with a interactive way to kill each creep. The players health bar was also replaced with a set number of lives. The lives you had left were shown by how many pieces of the cake you have left. If the cake gets eaten you would lose the game.

The creeps no longer drop gold to upgrade your weapons. Instead they drop gold coins the player must pick up quickly to increase their score.

- Red creep: The basic creep with no special abilities.
- Green creep: A flying creep that needs to be smacked out of the air with the frying pan before it can be damaged regularly.
- Blue creep: A sneaky creep with an invisibility ability that lets him fade away. To make him appear you need to smack him out. Just like the Green creep. The tricky part is remembering where he might be.
- Yellow creep: A fat creep with an immense hunger. This creep can eat up your lives faster than all the other creeps. It is also very tough with a lot of health.

5.2.3.2 SPRITES

Sprites are used in this game to animate figures like cake, runes and creeps. The sprite implementation is done partially by the corona framework. As the implementation of the sprite processed a few points were to notice:

- To create a new sprite is hazardous
- To add a new animation by name of within a sprite sheet was not implemented.
- To get an animation of a sprite sheet was tedious

On this basis a new module arrived: Sprite.lua. This module covered the above points for easy usage of sprite and animating them. All sprites in the application folder are found inside “Images/Sprites/”.

Image 9: Shows the sprite sheet of gold coin

For this to work with our new sprite module we basically had to:

- Add the sprite sheet to all sprites
- Add animation name, which takes a frame start number and amount of frames
- Use Sprite:Get() function to receive a set of frames based on a animation
- Call on the object's play function, object:play();

  See snippet 5 and 6 for some code view over the sprite functions.
- name

```
--Returns the proper sprite object ready to be played.
--See "spriteInstance" in the documentation for Corona on what kind of object that is returned
-- http://developer.anscamobile.com/reference/index/spriteinstanceanimating
function Sprite:Get(spriteName, animation)
    local instance;
    for i = 1, Sprite.amount do
        if(Sprite[i].name == spriteName)then
            instance = sprite.newSprite(Sprite[i].set);
            break;
        end
    end
    if(instance == nil or spriteName == nil or animation == nil)then
        if(spriteName)then
            print("Sprite.lua 85: Could not get the sprite with name: " .. spriteName);
        end
        print("Sprite.lua 87: Please use Sprite:Add() before calling Get() or recheck the spritename.");
        print("Also make sure animation paramter is given a value");
        return nil;
    else
        instance:prepare(animation);
    end
    return instance;
end
```

Snippet 5: Shows the Get function that takes a spriteName, for instance "goldcoin" as seen

```lua
--Purely an example, takes nothing, returns nothing
local function Example()
    --Paramters: Filename, sprite name, width, height, total frames
    Sprite:New("Images\Sprites\GoldCoin.png", "goldcoin", 32, 32, 4);
    --Parametrs: Sprite name, animation name, start frame, amount of frames in animation
    -- duration each frame is shown, forward or backward looping of the frames
    Sprite:AddAnimation("goldcoin", "walk", 1, 4, 100, "backward");
    --Parameters: sprite name, animation name
    local obj = Sprite:Get("goldcoin", "walk");
    obj:play();
    obj:cancel();               --Cancels animating, stops at current frame
    obj:removeSelf();           --Clears out the object, prevent memory leak
    obj = nil;
    obj = Sprite:Get("goldcoin", "explode");
end
```

Snippet 6: Shows an example of usage of the Sprite module.

A simplified example of using this module with the gold coin picture, that has four frames and a width and height per frame of 32x32, is coded in snippet 6.

During the development of this sprite module, Corona saw their need to improve the speed and totally reinvented their sprite implementation to be more like ours. They have now in their latest version brought to life so you can name an animation by a name.

Today we still use our own implementation of the sprite module to load sprite sheets and to retrieve animations from a sprite set, due to updating to Corona's latest version would be a time consuming process in changing a lot of errors that will need to be fixed. There are no issues known using our own module on any android we have tested.

5.2.3.3 PATH FINDING

For creeps to move properly through the levels, which have no explicit path to move along, the project includes a grid for creeps to move within. The grid covers only the region where creeps can move, hence there is no grid on the HUD-area of the screen and creeps will never move on the HUD or behind it. The grid in action can be seen in image 10 below.

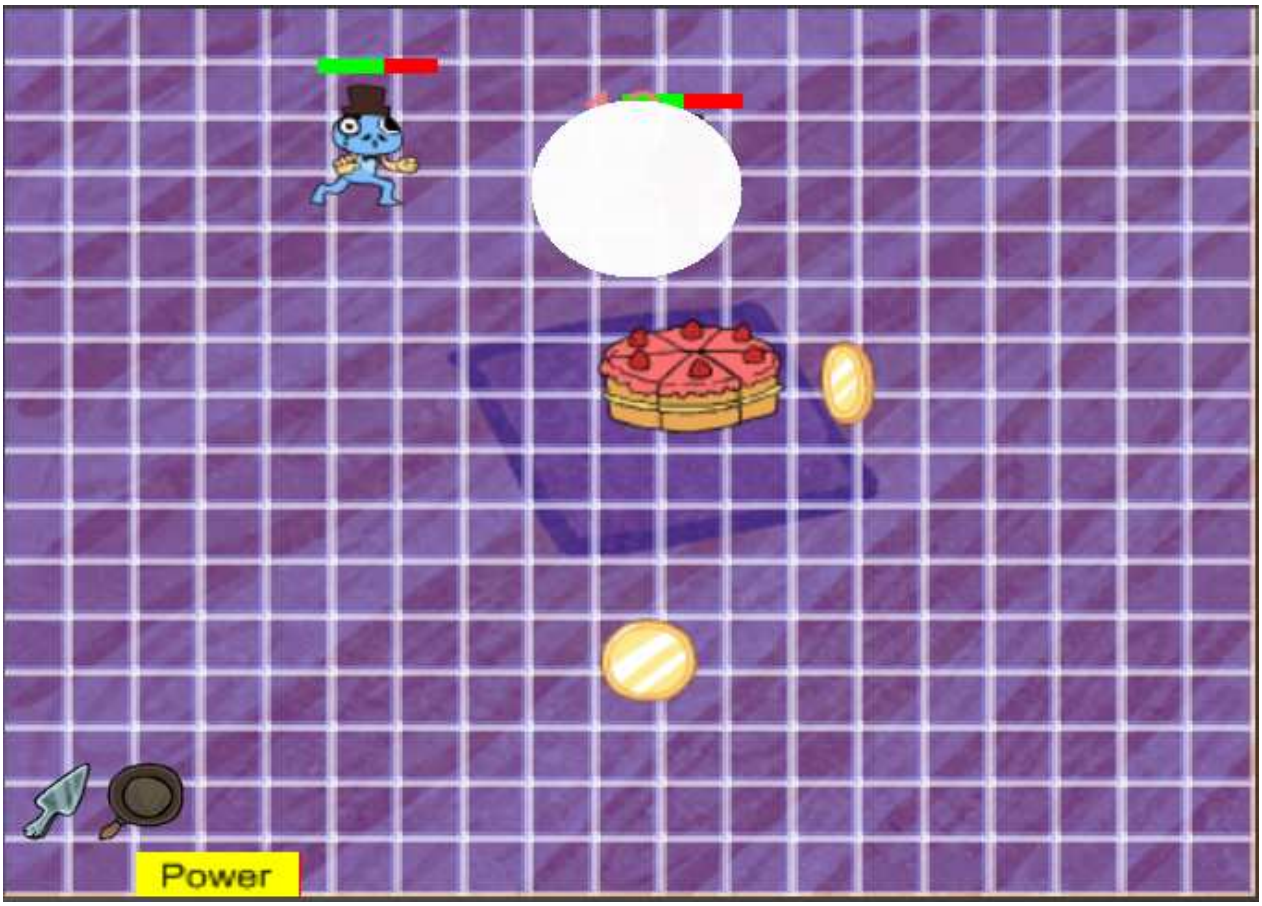


Image 10: Drawing the grid which is upon the level background with an alpha value, so the map is visible through it. White circle is hitting the creep with the Pan.

For path finding we have used the A* algorithm and a grid object. The grid is built up by squares that are equal in width and height as can be seen in figure below.

The grid in Cake Defence is 18 times 15 and each cell is built up by 20 times 20 pixels. Based on this we loop over the grid with the A* algorithm. A code snippet of usage for the creeps, with grid and finding a path can be viewed below.

```
--Returns a table of waypoints, where each waypoint is the center of a certain cell within the grid.
--Else returns false if path could not be found.
function Grid:FindPath(cellStartX, cellStartY, numberOfDirections)
    --  local path = PathFinder.FindPath(self, cellStartX, cellStartY, numberOfDirections);
    local path = pathf.FindPath(self, cellStartX, cellStartY, numberOfDirections);
    if path ~= false then
        local currentCell = { x=cellStartX, y=cellStartY }
        local directionString = "";                                     --Which direction are we heading
        local waypoints = {};
        waypoints.cellWidth = self.cellWidth;
        waypoints.cellHeight = self.cellHeight;
        local startCell = self:GetCoordinatesOfCell(cellStartX, cellStartY);
        local dir = self:GetDirection(Grid.rows, Grid.columns, cellStartX, cellStartY);
        table.insert(waypoints, { x = startCell.x, y = startCell.y, direction = dir});  --Starting Cell
        for pathNumber = 0, #path do
            local cellDirectionX = path[pathNumber].dx
            local cellDirectionY = path[pathNumber].dy
            local count = path[pathNumber].count
            for i = 1, count do                                         --Amount of cells in the same direction
                directionString = path[pathNumber].direction;
                currentCell.x = currentCell.x + cellDirectionX
                currentCell.y = currentCell.y + cellDirectionY
                local coordinates = self:GetCoordinatesOfCell(currentCell.x, currentCell.y); --X and Y values of a cell
                table.insert(waypoints, { x = coordinates.x, y = coordinates.y, direction = directionString });
            end
        end
        return waypoints;
    else
        print("Error Line 181, Grid.lua: ");
        print("Path is false... Make sure goal and start cell are within the grid, and make sure there is a path to the goal!");
        return false;
    end
end
```

Snippet 7: Shows the usage of path finder and grid together.

The code in snippet 7 above builds up a path for creeps to move along. If for example a path is blocked, or the grid changes, obstacles' can roll down through cells, then this function needs to be called again to find a new path to the goal, or in this project, a new path to the cake. Do note: The snippet above does not show the whole context. For this the reader must look through the code. The code above does not show the implementation of A*, nor how a goal cell is added.

A picture of the path from where a creep spawns to where the cake lies can, which is found through the algorithm, can be seen in image 11.

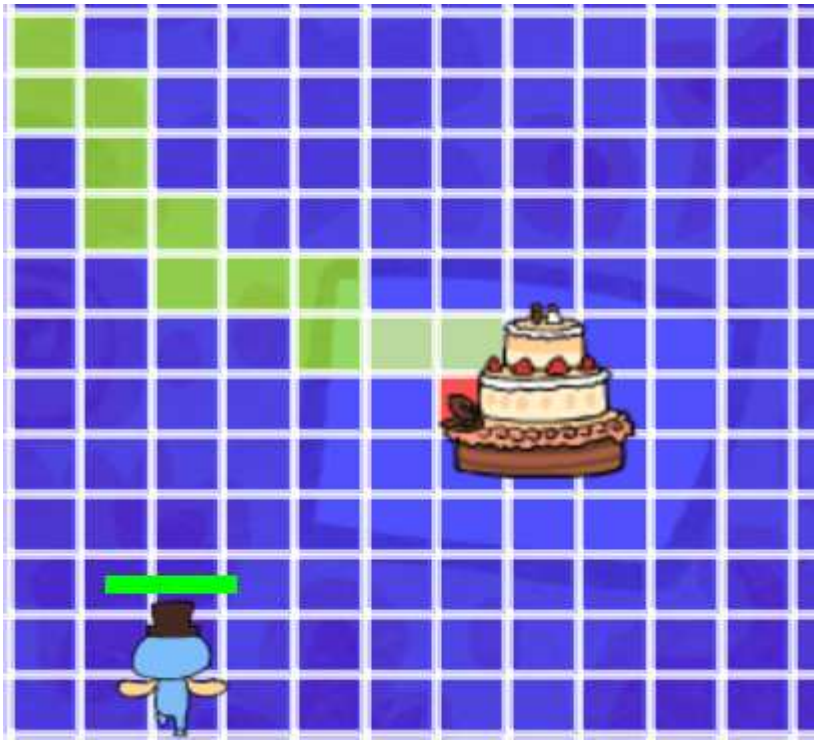


Image 11 : Shows the path calculated. The red square near the cake is the goal cell creeps moves towards when following that path. The cake in the image have been drawn slightly offset to show the red goal node.

5.2.4 Spells

The first spell intended was the "Shield" spell. This was first intended to be a weapon, but it was hard to find a place for it with the weapons. This spell was intended to block creeps in a certain way. This later evolved to the "Wall" spell, where you drew a line where the creeps had to tear down before being able to continue. This spell remained an idea for a long time until it was implemented in the game in the later versions.

The first implementation for spells was to have spells cast from a spell book. To make the casting of spells as interactive as possible we wanted the spells to be cast by drawing a shape on the screen. The spells needed enough mana to be cast. In the early versions we had a mana bar with automatic regeneration. This mana bar was later replaced with a cool down for each spell.

This was both hard to implement in the later version when we wished for simpler game play. This was to fit the new game idea we got from our partners Krillbite. The spells were simplified to be cast by its own respective button, each with their own cool down.

- Wall spell: Creating a wall blocking creeps trying to pass it.
- Earthquake spell: Affecting every creep stunning them and dealing some damage.
- Tornado spell: Drawing a circle around the creeps you wish to deal damage to. The size of the circle would determine the damage dealt. The bigger area, the less damage it will deal.

The last input from Krillbite removed all but one spell. The Tornado spell. The cool down was removed and replaced with limitation to how many times you could cast it.

5.2.5 Runes

Runes are sprites that pop up at random and give you special bonuses. These were introduced very early in the development to create a random element in the game. This will make the levels have varying intensity. To take a rune you need to tap on it, swiping over it will not work.

The runes have different effects depending on their colour and shape, and will only last a short period of time.

The first runes implemented were:

- Critical: a chance to get a damage boost upon hitting creeps
- Gold: Give the player additional gold for killing creeps
- Reset: Resets the cool down of spells

Rune ideas after the weapon change:

- Fire: Adds damage over time to the equipped weapon
- Frost: Adds slow to equipped weapon

Latest ideas:

- Upgrade: Upgrade the equipped weapon by +1
- +1 Spell: Add an extra spell cast.
- Coin gives an increase score automatically, or it can give an increased score throughout the level, for each creep you kill

- Effect-runes gives permanent effect to the weapon until wave is complete or until you pick another rune

5.2.5 Perks

A perk is a bonus given to the player in the form of a small static power-up. In our game, we intended the player to get a choice between a few perks after each level. The effect of the perk would then remain active for that player for the rest of the game. The thought behind this was that the player would get a feeling of progress and building their character, while also adding another minor strategy element. The system for a player gaining perks was finished, but no actual perks where implemented. Perks were later dropped in favour of making the game simpler.

Some perks we had in mind:

- Extra life
- Extra power/mana regeneration
- Critical strike (Chance on hit to deal double damage)
- Extra start gold

5.2.6 Player avatars

A player's avatar is what represents them within the game. At the start of the game the player is given a choice between three characters, who each give a different small bonus. This is mainly to give the player something to recognize as themselves in the game, but also adds a possibility for picking the avatar which best complements you’re playing style. On the other hand, we did not want to make this bonus too strong, as we also wanted the player to be able

to choose the avatar they liked visually, without feeling like they were being punished if it had a different ability from the one they wanted the most.

We considered making it possible to connect the game to facebook. Then you could post your high scores and use your facebook picture as the avatar. Unfortunately this was not implemented due to time constraints.

5.2.7 SHAPE RECOGNITION

5.2.7.1 WHY SHAPE RECOGNITION

Early in the development of the game. Spells was intended to be cast by drawing a shape on the screen. With this method the interface would be very clean. Since you only needed to have one button to represent the cast of a spell. The specific spell would be determined by the shape you drew. The effect of the spells was intended to be global and affect all the creeps to keep it simple.

To make this possible, shape recognition was needed. However, it is not a feature of Corona and was pretty hard to find. So we tried making our own.

Corona already give you the coordinates of all points in the shape by adding each touch event x and y in a table. The real challenge is to find the corners and lines in the shape. What makes it even harder was the fact that a quickly drawn shape has many curves. And the difference between a line and an curve may be very small. Another problem is that in an uneven shape, it might be hard to find the optimal corners. As you can see in th e figure below, the right side of the right hand image is very close to a right line. These slim margins will often appear in a quickly drawn shape.

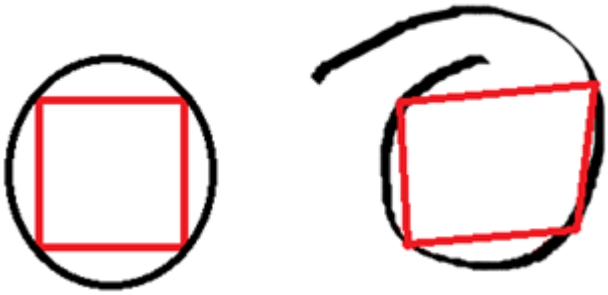


Figure 8: The left image shows what you want, the right image shows what you might get. The red lines are the line between the shapes corners.

I will not go in to full detail of all the functions used, but explain the most important ones. I will also explain my thoughts on the choices that I have made.

5.2.7.2 Finding Corners

The first problem when finding a shape lies in finding the corners. Since making one algorithm to find all corners in every special case is very hard. Many algorithms were made to find the corners in different ways.

5.2.7.2.1 Corner Algorithm 1

This one loops through all the dot products of the shape(p0-p1 dot p1-p2....) and adds the dot product / angle to a sum. If this sum exceeds the threshold angle for the algorithm it will mark that point as a corner. This will be inaccurate if the line is wavy. If it is the corner might be marked to early.

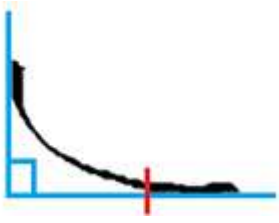


Figure 9: Shows an example of the threshold if it is 90 degrees. Red line marks the found corner

5.2.7.2.2 Corner Algorithm 2

Checking the difference between the dot products / angles by a percentage and manipulating it to create bigger differences. If the percentage is bigger than the threshold value, that point is marked as a corner.

Figure 10: The blue angles are equal therefore the difference is 0. The red angle have a big difference compared to the previous one.

5.2.7.2.3 Corner Algorithm 3

Compare each segment of the shape to see if any combinations exceed the threshold angle. If the threshold is reached the segment breaking it becomes the new segment to compare the rest. The point that breaches the threshold will be marked as a corner.

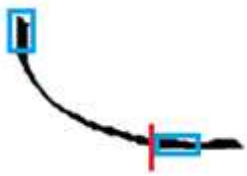


Figure 11: In this example the threshold angle is 90 degrees. The corners compared that exceeds the threshold are marked with blue boxes.

5.2.7.2.4 Corner Algorithm 4

The final algorithm never got implemented, but the idea was to find all points furthest away from the centre of the shape.

To make this work an additional function would have to smooth out the straight lines and sharpen the corners. This can be done by using parts of algorithm 2 and smooth out segments which have the same dot product/angle difference. If the difference is the same and the dot product/angle is below a threshold value it will be evened out. If on the other hand the difference is the same and the dot product/angle is over the threshold value it will be "sharpened". This function creates the "smooth" version of the drawn shape and will be the shape to be tested to find the shape instead of the original. This will make the later tests a lot easier.

Since running all the algorithms will most likely give you to many corners. The last function will remove all corners inside the radius of each corner. The radius (endPointTolerance in snippet 8) is the average length of the longest length segments between all the found corners forms the corner algorithms. This will let you eliminate the excess corners you do not need.

```lua
-- fuse nearby corners into one
local function fuseCorners()
    local i, j
    local x, y
    local mag
    local average, total = 0,0

    -- remove points inside the radius of each corner
    i, j = 1,1
    -- check all corners except the last one
    while( i < #corners ) do
        -- don't check vs the points that are already checked
        j = i + 1
        while( j <= #corners ) do
            x = corners[i].x - corners[j].x
            y = corners[i].y - corners[j].y
            -- find the vector length
            mag = math.sqrt( ( x * x ) + ( y * y ) )
            -- if the length is smaller than the max radius
            if( mag <= endPointTolerance ) then
                table.remove( corners, j )
            else
                j = j + 1
            end
        end
        i = i + 1
    end
end
```

Snippet 8: Removing excess corners

Finding Lines

After finding the corners the next step is to find out if there is a straight line or not between them. There are two functions created. The first one compares the actual length of the segments between the corners with the minimum length. If the lengths are equal the segments make a straight line. However when drawing, it is almost impossible to only make straight lines with your finger on the phone. Therefore, if the total length is below a set value it is considered a line. In order to make this method more accurate it is important that the corners are in the optimal position. The optimal position for a corner is in the middle of the curved lines, or in the sharpest angles. To make the corners a function places the corners you have as

far away from the centre as possible. This place them in a better position to make the line check when comparing the min and max lengths.

The second function checks if **most** of the line is straight by comparing the dot products / angles and mark the end of a line if any of them exceed a threshold (just like algorithm 1). This one also needs to make sure that the longest straight line is as parallel as possible to the actual line between its respective corners. If the angle between the shortest line and the actual line is to great and there is no corners in between. The line is obviously a curved line.

Finding Shapes

After finding the number of lines and corners all that remains was to make specific functions for the shapes you wish to find. The shapes that can be drawn are one of two categories, a closed or not closed shape. A closed shape is determined by the length from the start and end point. There is a check where the start-end length is inside a certain radius of the start point.

The shapes implemented are circle, rectangle, triangle and line.

Circle

This is by far the trickiest one because you will rarely get a perfect circle by drawing.

And in most cases you will overdo the circle when drawing it fast.

By taking the first radius as a reference, and checking the length from the centre to each point. If the check finds that all lengths are within a certain percentage of the reference radius. You can determine that it is a circle. This might not be good enough for high threshold values, because there will be too much room to succeed the check.

Rectangle

By checking the sign of the resulting cross product from each side of the shape vs. the centre. You will find out if the lines lie around the centre. The length of the sides does not need to be compared against each other. This is because the function that fuses the corners will eliminate

the trapezoid shapes. In addition to this the lines found must be equal to the corners of the shape. In other words, if the shape consists of four lines and four corners.

Triangle

Same as the rectangle check.

Line

If the shape is not closed and the length of the line segments are equal the min length between the start and end point. The shape is a line.

```
-- returns the sign of the cross product of two vectors (x0,y0 is the pivot of the lines 01, 02)
local function isSignPosetive( x0, y0, x1, y1, x2, y2 )
    local product = ( ( x2 - x0 ) * ( y1 - y0 ) - ( x1 - x0 ) * ( y2 - y0 ) )
    if( product < 0 ) then
        return false
    else
        return true
    end
end
-- returns true if the draving directions/angles are constant
local function isClosedShape( x, y )
    local currentSign, previousSign

    for i = 1, #corners do
        -- check the sign of the cross product
        currentSign = isSignPosetive( corners[i].x, corners[i].y, corners[i%#corners+1].x, corners[i%#corners+1].y, x, y )
        -- check if the two boolean values are not equal (the first time previousSign is nil)
        if( previousSign ~= nil and currentSign ~= previousSign ) then
            return false
        end
        previousSign = currentSign
    end
    return true
end
```

Snippet 9: Shows how to determine if the sides of the shape lies around its center

WHY WE SCRAPPED IT

During the development the game have had huge game play changes to the point where the casting of spells were cast by clicking a button. The game play was to have a quicker pace with simpler elements. Because of this the shape recognition no longer had a use. This was because it adds more complexity than needed, and the choice of the spell cast was made when you click the respective button.

5.2.8 Score System

The score system was first intended to be increased by killing creeps and doing combos. These combos were not restricted to the weapon combos mentioned earlier in the report. The number of creeps you manage to hit in a slash will also give you a higher combo score.

This means to get the highest score possible you would have to hit as many creeps as possible on finger swipes, and execute lots of combos.

This was replaced with a score system developed by Krillbite. It was created to fit the new game play. The combos of the previous score system are removed.

Point system:

- You gain 10 points per second.
- You gain 125 points by collecting (tap not swipe) a gold coin.
- You gain points and gold coins by killing creeps:
    - Red: 100 points and 1 coin
    - Green:
        - Ground: 100 points and 1 coins
        - Flying: 200 points and 3 coins
    - Yellow: 500 points, 1 coin and 1 cake slice to give you an extra life.
    - Blue: 1000 points and 7 coins

Image 12: Shows various creeps and items that drops from them upon death

5.2.9 Opening screen and level selection

While developing the game our goal has been to make the game user friendly and easy to pick up. We have tried to reflect this in the user interface by making the simple and clean, and as self explanatory as possible.

Image 13: Opening Scene

The opening screen (image 13) is very straight forward. We used icons to express the function of each button rather than words as these icons are more globally known. The arrow pointing

right is for play, the cogwheel is for settings and the "X" is for exit. The players saved profiles will appear under the three buttons.

Image 14: Shows the current level selection map and level nodes

The player presses the orange button for the level he/she wants to play. The buttons that are greyed out represents locked levels.

5.2.10 HUD

The HUD is one element of the game which has gone through a lot of changes during the development process. We wanted it to be as simple and easy to use as possible, while still incorporating everything we needed for the game. Because of the small screen size on mobile devices, it was also important that the interface did not take up a lot of space, which was somewhat of a challenge because of all the elements we wanted to be included.

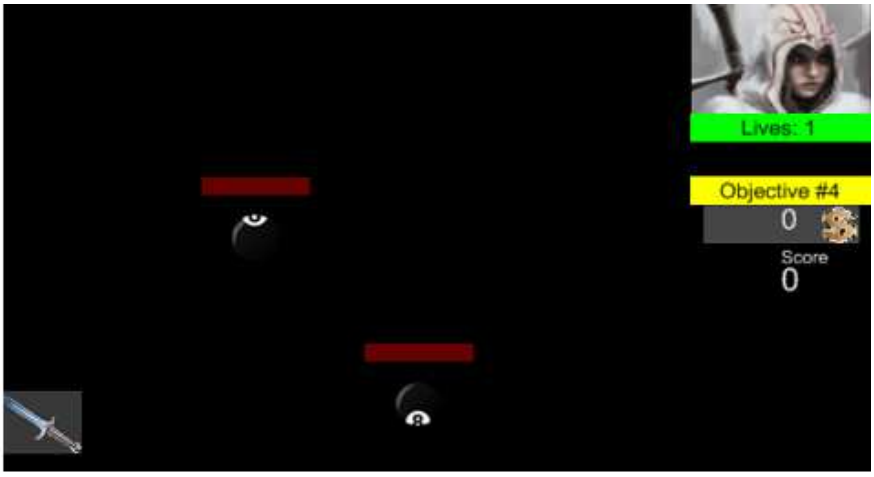


Image 15: Shows an early version of the HUD. The game seen uses developers art.

Figure 3 shows the HUD in an earlier stage of the development. The player avatar is in the top right corner. Below it is the player's lives, creeps left, money and score. In this version the player could open the shop by pressing the gold amount and go into spell casting by pressing the book; these would both open as pop-ups. Changing of weapons would happen by pressing the sword in the bottom right corner; this would open a drop-down menu containing the player's unlocked weapons.

Image 16: Shows the current HUD and game

Image 15 shows the latest user interface in the game which was developed with Krillbite's help. We noticed that the pop-ups for shop and spells broke the game flow in a way that was disturbing for the player. The drop-down menu for changing weapons was also removed after we decided to only have two weapons instead of four.

This version is still under development, weapons should be in the empty squares on the right side, with the last square in the bottom right being for spells which the player can pick up during the level. We also removed the creeps left bar and the life bar. Life is now shown through the number of slices left of the cake in the middle of the playing field.

5.3 Code Documentation

DoxyGen

For this purpose we downloaded first [15]DoxyGen. But due to the fact we haven't commented the code in a DoxyGen-mannered way, we stopped there and researched further.

LuaDoc

We found something named LuaDoc. This was not as easy as it said. Found a few articles which said "this was pretty straightforward to install".

On these two basis above, which we hadn't planned too well to begin with, we thought: Let's just create something similar ourselves, which resulted in a C#, asp.net, web-page that reads on the server side all files in a specified folder and stores comments and functions in a decent mannered way.

Tools used:

- Visual Studio, Web Developer Edition
- C#
- ASP.NET

```
<div>
    Filetype
    <asp:TextBox runat="server" ID="tbFileType">lua</asp:TextBox>
    Commeents:
    <asp:TextBox runat="server" ID="tbComments">--</asp:TextBox><br />
    FunctioN:
    <asp:TextBox runat="server" ID="tbFunctionDeclaration">function </asp:TextBox><br />
    Path folder In
    <asp:TextBox runat="server" ID="tbIn">C:\Users\Oyvind\Desktop\In</asp:TextBox><br />
    Path folder Out
    <asp:TextBox runat="server" ID="tbOut">C:\Users\Oyvind\Desktop\Out</asp:TextBox><br />
    <asp:Button runat="server" ID="btnLoop" Text="Generate Documentation" onclick="btnLoop_Click" />
</div>
<asp:Label runat="server" ID="lblFeedback"></asp:Label>
```

Snippet 10: Shows a simple asp.net syntax that takes inputs through text boxes and a button with an event. The event onClick triggers a C# call and we get our comments and functions printed out on the format HTML. The code and HTML generate files is within the ZIP-file that followed this report.

6. TEST REPORT

6.1 INTERNAL TESTS

We conducted the first tests within the group primarily to weed out the most obvious errors and problems with the prototype; in addition to finding out if it was durable enough to be used for testing. These tests were carried out continuously and informally as we programmed and were executed both on the simulator and also often on an android mobile phone. We were aware that this was not equivalent to have a tester from outside the group. We would get much more feedback from someone that where new to the game, from them telling us about their experience, but also by observing them, we would be able to see what did and did not work.

Basically these internal tests were based on looking at our graphical user interface, and rescaling parts of it to fit the screen, but also to see if our algorithms and other general code for the game were working. Based on these tests we were able to create a functional prototype, which we considered ready for a larger test, a test for people outside the development team.

Test equipment went on a HTC Desire running android 2.1. HTC Desire specifications are:

- 1 GHz
- 576 mb of memory
- 480 * 800 WVGA

6.2 EXTERNAL TEST

Testing within the developing team did not give a realistic impression of how our solution worked, due to everyone within the test group knowing how the prototype functions and how it is coded. We needed feedback from the outside to determine if the prototype at all was

comprehensible to anyone other than ourselves. In addition, there was a possibility that problems with functionality and usability, that could have been improved, had been overlooked in our internal tests. Therefore we wanted to test the prototype with others outside of the development team.

This phase was an opportunity to make use of personas, sex, age and education is gathered, to check for the popularity amongst different persons to match one of our sub-goals, to create a game for everyone.

All test persons were given a consent form, can be seen in appendix page 29 - Consent Form., and explained the task to the test person. The test was based on two steps:

- We installed the .APK file on their mobile device
- We asked them to start Cake Defense and complete the whole game

6.2.1 Test Person 1

Test number	1
Sex	Female
Age	27
Education	Nurse
Amount of hours used on games per week	Two

Feedback during play-through	1. Wants more creeps than two on each level. 2. “Now I do not have any more selections, I do not know what to do”. 3. “Does it get harder after each level? Because it was a bit easy”
Observations made by a developer	1. Tutorial step 4, drawing a circle around creeps. Bit hard, creeps should probably patrol back and forth. Tester played it a second time, after an explanation, and completed it on the first try. 2. Interesting, she tried clicking the houses, park and cops, before she saw the cake lying on the road in the level selection 3. Manages to click on a button behind the popup menu in level selection, “every time”, fix. 4. Drew a spell around one creep and the creep did not lose a life. 5. Game stops after level 6, no more choices within the bonus selection, strange. 6. Tester did not use the shop, apart from the one in the tutorial. 7. Tester never changed weapons
Feedback after finishing the test	• Nice images, but no effort was required to complete the game. • Some buttons were bit tough to select. • Missing indications on which level have I played, and which level is not completed and “what is my score”

Rate Cake Defense **Please have in mind that this is a functional prototype when rating** **From 1-10, where ten is awesome.**	6
What would you be willing to pay for Cake Defense (have in mind this is a prototype) **Examples: 0kr, 1kr, 2kr…**	Could pay 2-3 kroner or 0.

Table 6: Shows data retrieved by observing and having test person number 1 execute the test.

6.2.2 KRILLBITE TESTING

We sent our game over to Krillbite three times and each time we got new feedback and new improved sprites and ideas to implement into the current game.

We sent away a Sword Defense prototype that included a level with a few creeps and a selection map where you selected a level. This was early on and it was to get Krillbite starting on creating sprites and coming up with a theme for the game. We did not need it to be swords, it could be anything, as long as it was based upon having units moving towards a goal, and that the player would interactive the game by hitting and tapping on creeps to kill them.

The three tests Krillbite did, gave us this feedback:

1. Good start, we will come up with a nice theme for the game, making this look good.
2. Feedback after the second test by Krillbite from Krillbite:
    - Cake Defense
    - Creeps moves to centre to eat cake
    - Multiple creep types: Invisible, flying, ground and fat
    - Reduce your menus from two to one
    - Put loading within the main menu
    - Change weapons to cake shovel and pan
3. Feedback after the third test by Krillbite from Krillbite:
    - Remove runes, have only one rune of type gold coin, which adds permanent score
    - Reduce amount of text on screen, shout out we need images instead of text
    - Buttons are too small for the weapons
    - Place everything within one HUD
    - Not all creeps are moving towards centre

6.2.3 Test Result

The selection of test subjects is limited to one, because of the results and feedback given from one test person. We felt we got overwhelmed with feedback, suggestions and fixes that needed to be done before we could test the prototype further.

After this testing we can deduce that the prototype works satisfactorily, but with some flaws that requires fixing. The test panel did well and they did complete the game. Test person 1 thinks the game would be cool if it was further developed and with more to it.

Based on these test results we will make improvements that can be found under chapter 7.3 Future Development.

7. Results and Further Development

7.1 Introduction

As of now May 2012 we have created a functional prototype of the game and we are still in the development process till the game is complete and released.

Release date is set to 10th June 2012.

7.2 Modules Overview

Objects

Module	Information	Status
DB	Module for creating a [4]SQLite file and creates all tables needed	Complete In Use
Globals	Includes modules that are globally needed and initializes the game	Complete In Use
Player	Module for the player object, stores score, player name…	Complete In Use
Slash	Module for drawing a slashing effect on screen	Complete In Use
Smash	Module for drawing a slashing effect on screen	Complete In Use
Spell	Module that adds events for “begins casting a spell” and “finishes casting a spell”	Complete In Use
SpellEffect	Module that creates spell effects upon cast	Complete In Use

Weapon	Module that contains damage and weapon type, etc, for a weapon	Complete In Use

Level

Module	Information	Status
Ability	Built-in abilities for creeps.	Complete Unused
Creep	Module for creating an opponent	Complete In Use
Grid	Module for spawning and moving creeps along a path and around obstacles	Complete In Use
Wave	Module for spawning creeps and runes within a level	Complete In Use
Particle	Module for creating visual effects, sending particles in some built-in shapes	Complete In Use
Rune	Module for spawning a rune of any type (gold, spell rune, …) within a level	Complete In Use
Sprite	Module for loading a sprite sheet and for adding a specific string-name to an animation within that sheet	Complete In Use
Perk	Module that adds special benefits for a player. Player achieves perks upon completing levels	Not Complete Not in use

- **UI**

Module	Information	Status
Bar	Module for creating a bar, for instance: health bar, mana bar…	Complete

		In Use
Button	Standard module for creating button with on click and on release events	Complete In Use
Drag	Module for dragging an object around the screen through touch event	Complete In Use
Dropmenu	Module for populating items as a drop down menu.	Complete In Use
Feedback	Module for creating messages on screen and fading them over time	Complete In Use
FloatingText	Module for creating moving text on screen that fades over time	Complete In Use
HUD	Module that has all HUD-specific data, like score, weapons…	Complete In Use
Timer	Module for triggering an event after a certain amount of time, with displaying a timer on screen	Complete In Use
Splash	Module for creating circular effect and to get an area of around the touched coordinates	Complete In Use
Sound	Module for loading sounds into memory	Complete In Use
PlayerProfile	Module to list information about a player	Complete In Use
ListView	Module to list items that are out of screen bounds. If so, the list becomes drag able up and	Complete

	down	In Use
Lightbeam	Module for adding a light ray effect	Complete Not in use
LevelEndEffect	Module for creating special effect upon level ending, either completed or defeat	Complete In Use

External

Game	Module	Information	Status
	Ability	Built-in abilities for creeps.	Complete Unused
	Creep	Module for creating an opponent	Complete In Use
	Grid	Module for spawning and moving creeps along a path and around obstacles	Complete In Use
	Wave	Module for spawning creeps and runes within a level	Complete In Use
	Particle	Module for creating visual effects, sending particles in some built-in shapes	Complete In Use
	Rune	Module for spawning a rune of any type (gold, spell rune, …) within a level	Complete In Use
	Sprite	Module for loading a sprite sheet and for adding a specific string-name to an animation within	Complete In Use

		that sheet	
Menu	Module	Information	Status
	Main Menu	Module that displays play, quit and settings button	Complete In Use
	Settings	Module where player can turn on and off settings	Complete In Use
Game	Module	Information	Status
	Bonus Selection	Module for selecting a bonus after level is complete.	Complete In Use
	Character Selection	Module for choosing a character if player chooses new game	Complete In Use
	Level	Module that is entered when playing a level, initializes all data for a level	Complete In Use
	LevelSelection	Module where user can chose which level to play	Complete In Use
	Profile	Module to display a users profile	Complete In Use
	Tutorial	Module to teach the user a few general skills required to complete the game	Complete In Use

Table 7: Shows an overview over all modules implemented and not implemented

7.2 Discussion of the results

What we have in the end is something that is very close to a game. All the heavy and difficult aspects of the product are working. Creeps are moving and animating as they should, and they

player can interact with them in the way we intended. Different weapons are implemented and using them as well as changing between them is no problem. The GUI is set up with all the functionality we wanted, both when playing then game and in the menus. We also have three different working spells.

Level selection is working as intended with levels being unlocked as the player completes the previous ones. The database is being updated correctly and they players profile is automatically saved and can be reloaded when restarting the application. The high score is also stored, though at the moment it's only the highest score, not a list of the top ones.

Where the game is currently lacking is mostly graphics and finding the right amount of creeps and their attributes, to make the game appropriately challenging. As it stands a lot of the artwork we revived from Krillbite is still not in the game. At the moment all the levels have equal difficulty and spawn a set amount of creeps. Although we have implemented audio, the sound files are now just dummy sounds, Krillbite have offered to help us record real sound effects for the game.

7.3 Future development

We will continue developing the product after the deadline for the hand in on this report and are aiming for a release in early June. Reaching a game that can be released on the Android market was something we were aiming for from the very start, and something we all still want to accomplish. Krillbite seems equally eager to see the game finished, and we will continue our partnership with them until that is done.

Our goal is to implement all the graphics and to balance out each level. This is something that we will have to do a fair bit of testing to figure out, to make sure we get the difficulty level of the game where we want it.

With the confidence and skill we have gained throughout this project as well as Krillbite's abilities within artwork and game design, we have no doubt that the game will be done in a timely manner and live up to both parts expectations.

7.3.1 CHANGES TO BE MADE

Features / Objects	Changes
Creep	• Eating animation • Jump animation • Victory/Defeat animation on the creeps • Celebrate or crying depends on your score
Weapon	• Cake Sprinkles will be a spell with a limited number of uses.
Spell	• Create a spell failed effect
Rune	• Implement a rune that increment the number of uses of cake sprinkles.
Path Finding	• Path finding for this particular game is required due to objects can get in the way and a need to move around it is essential, that is why the path algorithm A* is implemented. At the moment though; there are no drawn obstacles, but the implementation is done, so it is easily doable to add an obstacle, which is simply done by calling the function seen in Figure N.

	• In later versions, depending on which side of the goal node a creep moves, we might get a larger and more precise collision detection, so that we can play an animation of sprites eating cake one or two seconds then a life is removed, but this will be estimated towards how much time is it going to take to implement versus other implementation methods, just removing the sprite when it comes near instead of when it is in the middle of the cell, which happens to be in the middle of the cake.
HUD	• Figure X shows the HUD as it will be when we are finished with the game. The difference from the current HUD is mostly just changing the position of certain elements. Weapons are moved into their appropriate squares, the player's avatar is in the top right and score is shown below it.

Table 8: Shows changes that are planned for the future development.

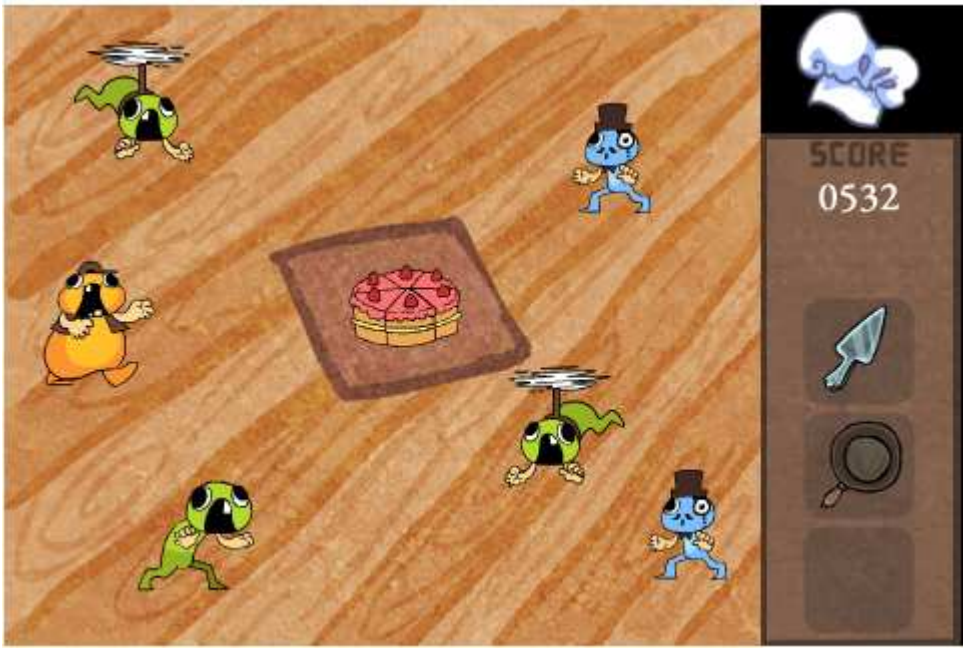


Image 17: Shows the latest version of HUD we are aiming towards, and game.

7.3.2 DATABASE

In the next planned version after the bachelor thesis is due, the database will take a drastic change in terms of size. The final product requires half of the tables to be removed, and final score will be merged into the player table. The game will store a minimum of data to for simplicities sake. Based on feedback from Krillbite this is the path to go, which we all agreed on.

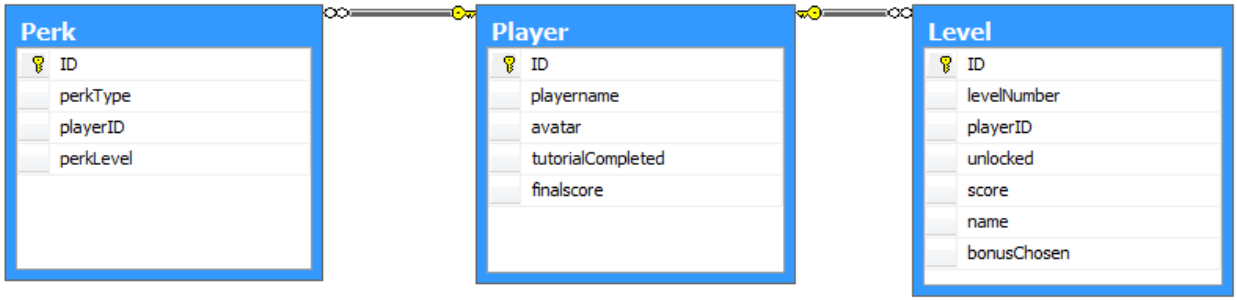


Figure 12: Shows the model of the database we are aiming at. It is small and stores basically just a player and his score.

8. EVALUATION

8.1 INTRODUCTION

In this chapter we try to clarify and evaluate the project as a bachelor thesis, and as a form of work.

8.2 BACKUP

The backup system used as described in Section N where we match several backup systems against each other, concludes with us using Dropbox.

Dropbox was a perfect choice for this project; we have also seen others working with Corona whom also uses Dropbox and IntelliJ IDEA as their main tool while developing games for android phones.

Dropbox have never let us down, it has not been a bottleneck in production, nor in how to get things together due to the brilliant Corona framework, which eases everything out in separate modules for us to combine with ease.

Although, over the last few months we have come to regret just once that we used Dropbox. At least, so we thought when a member accidentally deleted a file and Dropbox synchronized the deletion of that file on all computers.

The solution was simple, log into Drop box's website and there you had backups from one month back in time. We knew this before we started out, but got a little shaky even though you know. Guess the lesson is learned, you cannot have enough backups.

For a larger project it is hard to conclude that one should use one over another. It really depends on the software you develop within, what kind of project is it and how much money can you use on a version control system. A list of thoughts comes to our minds:

- Dropbox enough?
    - 2GB on Free

    - Download and upload rate is slow
- Is GitHub enough?
    - Download rate and upload rate estimations
    - Public or premium to get a closed repository
- SVN
    - Installation and server setup
    - Costs
    - Download and upload rate

On this basis some of these thoughts might change one persons mind. A project related around video production requires a lot more than 2GB, so free Dropbox would then be out of the question.

To conclude this topic we would like to state that Dropbox was and is far superior for this kind of project.

8.3 Group

Together we have worked pretty well, based on little experience together before, especially for a programming task of this magnitude.

If we were to roughly estimate our work load per week, then an estimation cost for the whole project, for us three developers:

22 hours a week * 20 weeks * 3 = 1320 hours total

Based on estimation above the cost of this project is as high as 300.000 NOK, if cost per hour is 250 NOK, Disagree, or agree, that this is a high cost for a small value. We see it a bit deeper than. If we were hired by a firm, first few months usually aren't good for the company, but in the long run, after we have been educated, we can do the same task now, within half the time frame, and get a better result.

We could have had a better flow in the process with these changes as we see it:

- Programming
    - Better planning ahead of what modules to implement
    - Algorithms
        - Think them more through before delving into, it is harder than it seems
- Have a project leader
    - Group roles are needed
    - A leader can end an discussion by taking action
- Teamwork
    - Working together better as a team
    - Working with Krillbite closer to begin with

On these basis we believe we would reach a more complete result, compared to the result today, which would be even closer to what ours and Krillbite's wish.

8.4 IntelliJ IDEA

Looking back at our development environment, we still believe we made the right choice when it comes to using IntelliJ as our text editor. Even for a relatively small project like ours, you end up with a lot of files, and handling multiple files was what IntelliJ IDEA excelled at compared to the other editors we looked at. Working on a project of this size or more would be incredibly frustrating without a proper overview. The Lua plug-in, which we at the time of testing considered to be the best of the three, has also worked well throughout the development.

As we knew going in, IntelliJ IDEA is the heaviest weight of the alternatives. This has resulting in it running slowly on our computers at times when the project started getting big, and slowness was resolved by restarting the IntelliJ IDEA. However we consider this a price worth paying for the advantages we got by using it.

8.5 Group rules

Not all of the rules we put down where followed at all times. In a bachelor thesis, where people often have other courses going on at the same time, things do not always go as planned. Our meeting times ended up being different from the ones in our contract. Some people had their class schedule moved so that they conflicted, people where sometimes out of town, got sick, etc. However, other rules worked very well, like our agreement to end all conflicts regarding the project by voting, quickly resolved some issues that might have consumed a lot of time otherwise.

Even so, we all feel that having rules put down at the start of a project like this is very important. In a real job situation, having rules and procedures to deal with certain difficult situations can save a lot of time and effort.

8.6 Project

Creating a game for the bachelor thesis benefits us all. This is the reason why we all attended this course, to actually programme a game.

Throughout the process, none of us regrets the path we have chosen based on the project. We believe choosing a 2D game, for an android device, with a framework is a rapid development process.

As a conclusion to projects and evaluation of programmers we want to link to this great article, about a programmers burning desire to sit up all night and programme:

http://blog.anscamobile.com/2012/05/the-coders-dilemma-and-happy-belated-birthday-corona/

8.7 Criticism

We are satisfied with our goals and result. We have created a game; we only wish it was more complete and that Cake Defense was introduced to us before. But we guess it is a process that needed to be gone through to get to the ideas and new goals that are set today. We see it as a necessary step to go through sword defense, and implementing that starting idea, to then send it to Krillbite to then change the game play and theme quite a lot. Although: Most of what is implemented, the most complexes algorithms for creeps, path, grid, spells, etc. is not scrapped, and the remained 95% the same from the start, which was our requirement for Krillbite. That the creeps had to use a path, the creeps would spawn somewhere and moving through some cells in the grid, which is the basics of most tower defenses. Key things that took major change is the HUD, what would go in it, what would not. We have tested throughout the last month of various HUD's, but there are a few key points that took a major change, lead to small fixing everywhere which takes time.

One thing we could have done better while learning Lua was to agree on a coding style. When learning a new language like this, and especially when we were not all necessarily learning from the same place, you get into certain habits regarding how you code. As we did not agree beforehand on how certain things should be programmed, it resulted in different looking code based on the person who did it.

For our projects this was not a big problem because, for the most part, each person had their own file to work on. However, when working together on a project of a certain size, it is inevitable that you will eventually have to go into another person's code for some reason. This did result in some minor confusion, but as we had good communication between the group members, it was resolved quickly.

8.8 THANKS TO

Thanks to Krillbite for lending us a hand when it comes to design and art work.

Thanks to Simon for being our supervisor during the project.

Special thanks go out to the people of Ansca Mobile who created Corona and made this project happen and to Google for creating Android.

9. CONCLUSION

This report has concentrated on the systems around developing Cake Defense and implementing algorithms for the product to work. Based on discussions and analyzes in this report and during the development process, we draw the conclusion that we have had a great advantage and benefit of the systems, tools and the partnership with Krillbite. It is our impression that Krillbite and ourselves has still high hopes for the game and that the product is functioning satisfactorily.

As it emerges from the test results from both our one tester and Krillbite, we see that the product still has a long way to go, but the most complex algorithms have been implemented which leaves every partner satisfied.

As the bachelor thesis comes to an end, we feel that we have accomplished much since we started the development of the idea at the start of January. At the same time we do not consider the game to be complete at this point, although what remain to be done are mostly minor details. We think that the game has a lot of potential in the Android market and our partner Krillbite has expressed the same opinion. Therefore, we will continue development together with them for a short while to finish the remaining programming work and get the game properly released.

Working in a group with this kind of project has been challenging and have given us a lot of experiences which will be very valuable for us later. We now have a much better understanding of what it is like to work on a programming project of a bigger scale, starting with nothing and ending up with a finished product.

Solid planning and regular communication is vital to the success of a project this size, and it just becomes more crucial the bigger the project is. We now understand the importance of these things when working in a group, for big projects in general, but more specifically when working on large multi file programming assignments.

In addition to the programming skills we have gained in general, which will certainly prove useful to us, the members of the group now all have experience with programming from scratch in coding language which was unknown to us before hand. We can also claim to have taken part in a complete development process for a game for a mobile device, both in terms of programming and design, which is a growing platform for games today.

The knowledge and expertise that we have gained during these months are certainly relevant within the game programming field and should provide very useful in the coming future. We now feel more confident in our abilities and are happy that our work has resulted in a product we are proud to present.

10. BIBLIOGRAPHY

[1] Scrum. Our development model of choise

http://en.wikipedia.org/wiki/Scrum_(development)

[2] Corona's API

http://www.jetbrains.com/idea/ - IntelliJ IDEA main page. The text editor we used.

[3] IntelliJ IDEA

http://www.jetbrains.com/idea/

[4] SQLight

http://www.sqlite.org/

[5] mySQL

http://www.mysql.com/

[6] Lua

http://www.lua.org/

[7] Tutorial for installing IntelliJ and set it up for use with Lua

http://producerism.com/blog/how-to-develop-in-corona-sdk-on-windows/

[8] Crash course in Lua

http://luatut.com/crash_course.html

[9] Ricardo Rauber Director class

http://developer.anscamobile.com/code/director-class-10

[10] Tutorial for using the Director class

http://techority.com/2010/11/19/how-to-use-scenesscreens-in-corona/

[11] Lime. Additional library for Corona.
http://justaddli.me/

[12] Notepad++
http://notepad-plus-plus.org/

[13] Sublime
http://www.sublimetext.com/

[14] LuaDoc
http://keplerproject.github.com/luadoc/

[15] Doxygen
http://www.stack.nl/~dimitri/doxygen/

[16] Tutorial for creating a fingre swipe in Corona
http://howto.oz-apps.com/2011/11/create-finger-swipe.html

[17] LearningCorona.com – A big list of Corona tutorials
http://www.learningcorona.com/

[18] Tutorial for installing apk files to Android devices
http://www.new-htcphones.com/how-to-install-apk-files-to-htc-legend.html/

APPENDIX CONTENTS

1. DICTIONARY

AI	Artificial intelligence. Simulation of intelligent behavior in computers
Android	Open source operating system from mobile devices
Apk file	Application package file. The file format of application when they are read to be installed on an Android device
Corona	A software development kit designed to build applications for mobile devices
Creep	Within the tower defense genre. A creep is an enemy attempting to walk part your towers
Dropbox	File hosting service offering cloud storage, file synchronization and client software
Drop Down Menu	A menu which expands or retracts when an action is preformed (most often triggered by clicking it)
GIT	A free version control system for managing source code
GIMP	A free software for editing images
HTC sync	A program by HTC which is used for, among other thing, loading application onto Android devices
HUD	Heads-up display. How information is visually represented to the player in a game
InterlliJ IDEA	A source code editor
Krillbite	A game studio located in Hamar. Out partner for this project
Lua	A lightweight programming language and the language used by Corona

MySQL	Open source database
Perk	A small, static, bonus given to the player
Prototype	A very early version of a product to show off or test a concept
RTS	Real time strategy. A genre within gaming
Scrum	A software development method used to manage product or application development
Spell	Feature of something magical happens, usually done by a magician. In this game it is done by the player drawing a specific shape, and then a tornado, or any other event occurs
Sprint	In scrum, a sprint is a work cycle lasting between one week and one month
Sprite	A two dimensional image or animation that is a part of a larger scene
SQLite	A database
SVN	Also known as subversion. A version control system
Tower Defense	A genre within gaming where the objective is to kill enemies by placing tower which fire at them
Unity	Game engine and development toolkit
User Interface	The part of the game which handles interaction between player and game

2. THE PROJECT OUTLINE

Time and date: 21th January

Project Title: Sword Defense

Members of the project:	Øyvind Berg, 091046
	Knut Øien, 091379
	Henrik Jotun, 091044
Partner:	krillbite
	Hamar, Norway
	www.krillbite.com
Contact Person:	Ole Andreas Jordet
	Phone +47 936 80 982
	Mail: ole.jordet@krillbite.com

About krillbite and the project

The company krillbite is a gaming studio located in Hamar, Norway. They do not have much experience in the industry, but it as studio that is up and coming, which has created small games for everyone to enjoy.

Quote from their website:

"We are currently having the time of our lives developing our very first project! It keeps us busy day and night"

Our project is to create a game for the android device. Krillbite will support with design and game play ideas, while the members of the project will stay behind programming and getting everything up and working. The game is a tower defense game, with a few twists

3.SWORD DEFENSE

Pre-project Report

Members:

Øyvind Berg 091046

Knut Øien 091379

Henrik Lee Jotun 091044

Pre-project-report contains the task and the project's goals for our bachelor thesis

Date Written: 11.1.2012

Last Edited: 27.1.2012

Location

Høgskolen i Gjøvik

1. BACKGROUND

We are three students who study game programming at Gjøvik University College. We have little to no experience in creating a full game for the android mobile phones. We all wanted to delve into creating a game for mobile phone for this particular reason. We landed on wanting to create a game for the android system instead of windows phones and iOS phones due to the fact that android market is up and coming. Also because none of us has had experience with an iOS phone before, or apple products in general, but at least knew what the android system looked and felt like.

Our employer is ourselves, where we have come up with this brilliant idea for a game.

Supervisors:

Simon McCallum and Jayson Mackie

Group Members:

Øyvind Berg, Knut Øien and Henrik Lee Jotun

2. REGULATING FRAMEWORK

As a group we have come to a couple of decisions for regulating the framework:

- The game is not thought to be working on any other device than an android phone.
- Users are required to know a bit about the mobile phone device themselves to download and install it from the android market.
- The graphics and user interface will be as simple as possible for beginners.
- We will have two modes one for beginners and one more advanced for better players.
- Our group has limited experience within designing a game from start to finish; therefore our choices might not be the best when designing the game.
- Our group has little to no experience using a android phone, therefore some choices on how to play the game might end up differently than what we planned.
- The group has little game-algorithm experience; therefore some of our algorithms from a professional programmer's perspective might see it as a strange implementation of some of the algorithms.
- Our group has no experiencing using Lua, nor structuring the code in Lua, so from outside of the group, some of the structuring of our code could be questioned.

3. PROJECT GOALS

We want to develop a game for the android mobile device that is fun, exciting and easy to play for both genders. The goal is to create the game that players in all ages will enjoy playing it, and play it hopefully more than a couple of minutes, because of the joy and challenge it brings.

A list of things that we see as useful functions in the game:

- Nickname
- Score
- Start game
- Complete game
- Storing of stats for each player, locally and/or publically.
- AI path movement
- Collision detection

Main goal for the project is to finish a complete game, written in Lua, for the android mobile phone, and also release it to the android market within the time frame. We want to bring a new experience of game play to the mobile phone for the genre Tower Defense through using our creativity.

The goal is also to learn and improve our knowledge of android development, android market, releasing a game on our own, and using other tools outside the normal tools used here at HiG.

In the process of creating a game the project also opens opportunities for us as a group:

- Plan the process from start to finish.
- Educate ourselves within Lua and all the tools required for this project.
- Using Scrum as a development process.

4.PROJECT DESCRIPTION AND ORGANIZATION

a. PROJECT Description

For our bachelor thesis we want to make a game from scratch rather than to get a specific task from an external. The reason for this was that we wanted to gain experience in the whole aspect of developing an entire game in general, not just from specific points in the process. After discussing it we decided on creating a 2D game, on android, using Corona framework.

We decided to make the game 2D for simplicities sake. Given the time frame we thought a 2D game would be more than enough work and that we would have a greater chance of completing it as a whole. The group also felt that being sure we could complete the game would also result in a better learning experience as we would get to go through every stage of the game development.

Our decision on what platform to develop for was basically between Windows and Android. We had about the same experience working with both. After a while we settled on android, it was close between the two, but we all wanted to learn more about developing for mobile devices and the android market also makes it very easy to publish the game once it is done.

Our game will be called Sword Defense and is a tower defense style game based around using the phones touch function as "tower fire" to kill the creeps. The basic idea is that creeps spawn in the middle and attempt to walk to the outside of the screen, either walking freely or through a maze. The player’s goal is to kill all the monsters before any of the reach the outer edges by sweeping across the phone screen and "slashing" the enemies.

We first thought about using unity as our game engine, but decided on corona and eclipse instead. The initial problem with unity was mainly the price. By the end of the bachelor thesis we hope to have a game that we can release into the android market, with unity we would have to buy the pro version which is very expensive, as the student version does not allow releasing and neither does the trial (even if you convert to pro before you release).

Corona is free while you develop in it, and only cost money if you want to sell your game, which you can pay right before the release. Also, looking through the demos of Corona is seems like a pretty user friendly program, which is a plus when you, like us, have not done a lot of full game development before. We choose to code in eclipse because we have a little experience with it

from previous courses and we were quickly able to find Lua plug-ins for it, which is what we need to use with Corona.

At the time of writing, a full Unity license of android costs 400USD and we would probably need one each, we could buy a student version for 100USD each but then we would not be able to release our game. Corona, on the other hand, costs 200USD and we would probably only need one between us as we only need to pay for the release, not while developing.

B. Organization

Our group has come to the conclusion that we all will participate equally on the project, due to the fact that the group is small, three members; we will all be responsible for the project.

We have planned to work a “minimum” of 18 hours each and planned meetings on Tuesday, Thursday and Friday at 11.00 pm.

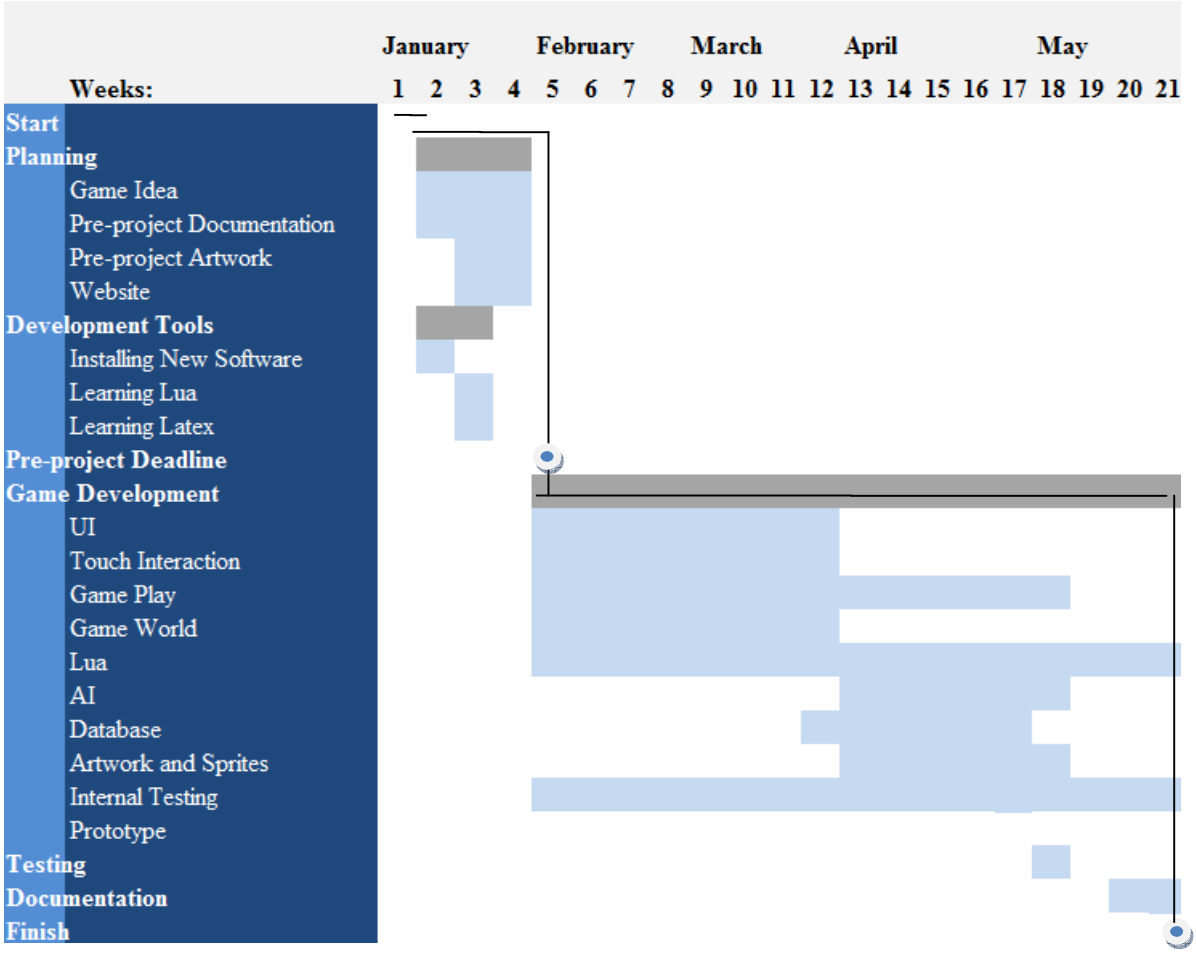
We will follow the scrum development process with some minor regulations as follows:

- Each sprint is a week
    - It is short project, and each module that has to be done is small enough to fit in a week’s sprint.
- Daily scrum-meeting every work day, three times a week or more. This is where problems will be reported and discussed. Only problems will be logged
- No Scrum Master
    - The whole team is a scrum-master; everyone together is responsible for having a meeting when said.

5. SCHEDULE

Our schedule for the project is represented by a Gantt chart. This first version is made out of rough estimations since we do not have enough experience to know how long these things will take. We will update the Gantt chart during the project to make it more accurate. This will be done at the sprint meetings.

The Gantt chart will be in a Excel Sheet including distribution of our resources, roles and organizing our work load.



6. QUALITY SECUREMENT/RISK ANALYSIS

Risk	Likelihood	Damage	How we will deal with it/prevent it
Project is not done in time due to time constraints	Low	Medium	Good planning. Set milestones. Keep the deadlines we have set for ourselves. Have meetings often to ensure that everyone is on time with their work
Part(s) of the project takes more time than anticipated during the planning period	Medium	Medium	Prepaid for it by planning in a way that allows the project to finish, even with small delays in the critical path
Project cannot be completed due to the leave of a group member or a group member has not done his work	Low	High	Good communication between group members to ensure that everyone knows what everyone is doing and that we are all doing the work we are set up to do.
Part(s) of the project cannot be completed because the current technology does not allow us to program what we had envisioned.	Low	Low	Be prepared to simplify some of the game's features.
Loss of source code	Low	High	Take regular backups.
Source code becomes public	Low	High	Use care when dealing with the code on the web

Disputes within the group halts the development process	Medium	Medium	Have clear group rules that state what should be done to resolve disputes quickly
Disputes with the externals halts the development process	Medium	Medium	Write and get approved, a clear and concise specification of what is expected of both parts before starting development
Similar game is released while we are still developing	Low/ Medium	Medium	Can't do much about it?

4. Sprint

Sprint 1

Goal

1. To develop the initial game. Creating ideas for the type of game play we wish to go for. Also the basic theme of the game and the target group must be established early on.

1. Decide on what platform we wish to develop the game for. Then research the IDE and tools available and figure out what we need to learn to create the game within the time we have.

2. Create a small website for the bachelor project was we can upload the information about the game and our group.

3. Write the pre-project report on what we are going to do and how we are going to do it. This will be uploaded to the website as well.

4. Create rules for the group to make sure we work consistently and know what to do in worst case scenarios if a group member does not fulfil his part.

Results

We have decided to create an android game and use Corona SDK as an IDE. It seems to be simple and clean compared to Cocos2D, but it costs money. This is only a small sum of 199$ to be split among the three of us.

We still need a text editor to go along with Corona. We have already tried out eclipse and Sublime text 2, but they do not feature a very good auto complete for Lua or Corona.

We have created a website for the project

link: http://hovedprosjekter.hig.no/v2012/imt/spill/sworddefense/index.html

Sprint 2

Goal

1. To decide to have Krillbite as external or not. If we decide to have them as an external

   We need to write a detailed report to Krillbite about the project.

2. We need to find a better text editor since eclipse and sublime does not meet our requirements.

3. The pre-project report and group rules needs to be uploaded to our website in a pdf-format.

4. To do research on tools for graphics (sprites, 2D images) and figure out what we want the game to look like.

5. We need to learn how to use the Lua programming language and the Corona API / resources.

Results

We have decided to work with Krillbite, and have sent them an email with our project details and ideas

The text editor we have found to work best is IntelliJ with the Kahlua plug-in to get code auto complete for Lua and Corona functions.

The documentation has been uploaded to the website.

The look we want for our game is a medieval theme with swords and castles, as it fits the tower defence genera perfectly. The tools for sprites is included in Corona.

Lua seems to be a simple language. The only thing that will take some time getting used to is the fact that it does not have Structs or classes.

Sprint 3

Goal

- Take action based on response from Krillbite

- Learn how to create and use these functions from the library:
  - Button
  - Text box
  - Background
  - Sound
  - Events
  - Touch/Multi-touch
  - Slide /Scrolling
- Begin the planning of all functions/tables/files required for the project.

Results

We have written a game design document to give Krillbite a understanding of our ideas for the game. It also includes the size of the sprites and screen, story and the destination/goal of the creeps.

We have been able to cover all the points we set as a goal to cover. The text box feature was forgotten and needs to be done in the next sprint.

We have started writing down the required functions and tables. There is still more work needed on this. On the other hand the file hierarchy is somewhat done.

Sprint 4

Goal

- Meet with Krillbite and make an agreement about the project. We also need to get their signature for the project agreement document.
- Learn how to create and use the Text box function from the library.

- Define every file, function, table… that is required for the whole game.
- Creating a diagram (UML) with a tool, rational rose (?). Research needs to be done on the tool we need to do this.
- Test all things created in Sprint 3, on a android phone. To make sure it is working properly.
- Research on how a Corona game handles storing stats on SQLite/external server.

Results

The meeting with Krillbite was postponed, until Monday 6th February.

We have learned to use the Text box. It needs to be mentioned that it is not implemented in the Corona SDK emulator. And needs to be tested on a phone.

A definition of the basic functions/tables and files we expect to make are done. This list is likely to have a lot of changes in the future.

We created a class diagram for the games objects. To do this we used the Violet UML Editor which is free software downloadable from their web page.

All the functions we implemented in sprint 3 seems to work fine on the mobile device. We decided to use Director.lua to handle the screen changes. This is a free class that can be downloaded from the creator's website. Link: http://rauberlabs.blogspot.com/

Sprint 5

Goal

- Meeting with Krillbite, 06.02.2012 – 13:00
- Program a very simple "game" including only a background and a simple sprite which is defined as a creep object. We will program this together so that everyone knows how the basics of this world works, which will make it easier for everyone to program separate parts in the next sprint.

- Design and program the start menu with a text box to write in your nickname.
- Create the database tables.

Results

The meeting took place as planned and they agreed to help us with the design and art for the game.

We each created a program with a creep with a health bar. The creep took damage by touching it.

The start menu is done. Play, load and quit button and a text box to register player name when starting a new game.

The database is created with a few tables to store the score and player profiles.

Sprint 6

Goal

- Get the project contract sorted out with Krillbite

  Sign and send the contract with Krillbite to HiG.
- Create (program) all the classes we need from the UML diagram
- Finish the opening screen UI
- Program the in game HUD/interface
- Copy and paste the website to the new URL from the "IT-avdelingen".

Results

The contract is finished and signed. We have sent in the paperwork required by HIG.

Creating the whole game will take more time. The objects created are tested an working. Note: When testing the new features we discovered Corona does not handle sub folders when building the .apk file for the phone. This makes it a bit messy.

The opening UI is more or less completed.

We did not have time to create the HUD.

SPRINT 7

GOAL

A “complete” game:

- Main menu with nickname/text box and a background.
- Button “Play”
  - Stores the nickname chosen in the SQLite database if not exists from before
  - Changes scene to “map”
    - On the map(a background with couple of new buttons) you can choose a destination. You choose a destination by clicking on a button. This will create a “text pop up”, displaying info about the level. The pop up will have a play and cancel button. Clicking play will start the respective level.
    - Having a few units moving from one side to the other, with a background. Add the HUD and a few buttons displaying the score/sword/shields/…
    - A victory and loss event.
    - Finish swiping/slashing graphics and implement it

RESULTS

Obviously too much work to finish in one week. The victory and loss event is now finished. We will use the next sprint to finish what we have left.

SPRINT 8

GOAL

- Continue with sprint 7
  - Too much is remaining

RESULTS

We have made level with spawning creeps.

The basic HUD is complete and the slash effect is working. However they need some more tweaking to work the way we want.

SPRINT 9

GOAL

- Put our individual code together and send to Krillbite

RESULTS

The first version of the game included the menu and level selection. There is one level with two moving creeps. This is enough to get a feel of the game we are aiming for. We sent this to Krillbite to see if they thought we were on the right path.

SPRINT 10

GOAL

- Create multiple spawn and end points for the creeps.
- Create the sword and player class.
- Implement 4 swords
- Get HUD working. Make the "shop" where you upgrade your weapons.
- Floating combat text

- Creep abilities
- Shape recognition to cast spells
- Level selection

Results

Multiple spawn points for the creeps are now implemented. There is however only one end point.

The sword and player class is completed. The player class needs to be merged together, because of two separate versions.

The four swords implemented do not yet have special effects added to them.

The HUD now have a shop and the needed elements, health, score, wave info is now included.

Floating combat text is now finished but not in the game yet.

Creeps have the ability to teleport.

There is no shape recognition library for Corona and it is hard to find it on Google. We will try creating a simplified one yourself.

Level selection not complete

Sprint 11

Goal

- Finish shape recognition
- Finish a tutorial to introduce the concept of killing creeps and casting spells.
- Fix power bar to decrease the damage of the swords the longer you touch the screen. Regenerating when the finger is not touching the screen.
- Create perks
- Create a perk selection where you can choose a perk after a level is complete.

Results

The shape recognition is very ambitious and will take more time.

The tutorial is almost done, it only needs the shape recognition to be completed.

The power bar now works as intended. Fixed a small bug with our bar class.

The perks can be made, but are empty since we do not know exactly what to do with them yet.

The perk selection is complete.

Sprint 12

Goal

3. Multiple and endpoints for creeps
4. Create spell book. When clicking on the spell book icon in the HUD a new screen/effect needs to be shown. Otherwise you would not know when you can cast a spell.
5. Floating text for showing the damage upon hitting creeps
6. Make the sprites work. And begin fixing the first level of the game.
7. Implement avatars for player profiles and save the progress
8. Shape recognition

Results

The creeps are now able to have different end point destinations.

The spell book is implemented with a cast and cancel button.

The current damage of the equipped weapon will pop up when you hit creeps.

The sprites in Corona seems to be lacking, so we need to make some functions of our own to make it work properly. The first level is now complete. It only spawns two creeps, when you kill them you win.

Avatars have been implemented. When you start a new game you choose your avatar. It also features a small bonus. Examples of starting bonuses: More starting gold, Faster power regen, and an extra life. When you have created a profile it is saved in the database and can be loaded when playing the game again.

Needs more time with the shape recognition, a almost functional version is tested on the android. Needs more tweaking.

SPRINT 13

GOAL

4. Fix the spell book to cast a spell when you finish drawing the shape.
5. Implement shape recognition with the spell book. If you press the spell book a transparent picture will show over the screen where you can draw a shape/cast a spell.
6. Put together a new version to send to Krillbite. This one is to have the tutorial and level unlocking(two levels)

RESULTS

Spell book now works as intended. Removing the old cancel button.

Touch recognition is implemented. You can now cast spells by drawing a shape. However it does not work very well at the moment.

A new version have been put together. This one features a short tutorial and two levels. The second level gets unlocked after playing the first one. The progress is saved. When you start the game you can click on your avatar and load the last session.

SPRINT 14

GOAL

- Corona recently made a update that fixed a lot of problems with the sprites. They now seems to work the way we wanted them to. Should we update and change a lot of our code?
- Save level data in the sqllite database and make levels unlock/show on the level selection map when you have completed a level.

Results

We have decided to continue using the version we have. Some of the new changes it provides is tempting. Like the new fix that lets you build the game with sub folder for the phone. However we already have working code. And we can simply find/create a script to extract all the files from the folders to the root. This will save us time when building the apk file.

Data of levels is now saved and a second level is unlocked after completing level 1.

The win condition seems to be bugged and needs to be fixed.

Sprint 15 [Easter]

Goal

- Implement levels throughout the game, up to 10-11
- Implement so creeps are running to a goal node properly
- Implement sprites
- Get sound working

Sprint 16

Goal

- Krillbite gave us feedback on the last version on the game. They came up with a new game play and style. Implement as much of this as possible before we visit them in Hamar next week.

- Remove swords
- Implement 3 new weapons, cake shovel, frying pan and cake sprinkles

Results

Most of the features suggested from Krillbite are implemented. We will probably finish the rest of them before visiting them next week.

The swords are removed. The sword class is now a “weapon” class to create the new weapons wanted by Krillbite.

The new weapons are implemented. As of now they all act the same way.

Sprint 17

Goal

- Visit Krillbite 19.04
- Get feedback on version
- Start implementing the feedback given at krillbite
- Fix error upon loading and playing sound files

Results

The meeting with Krillbite turned out well. We talked more about the spells, and how they should work. We also did some game play research by checking out some other games.

Sprint 18

Goal

- Remove shop
- Idea use runes for upgrading
- Make the new weapons unique functionally

- Create spells. Three basic ideas Wall, Tornado and Earthquake.

Results

The shop is removed due to the new play style. The creeps no longer get more health. Instead they will increase in number adding difficulty. The other reason is the fact gold as a currency is removed from the game.

Since you no longer upgrade weapons with a small sum of gold. You can upgrade weapons with runes instead. Not implemented.

The cake shovel is now used to slash foes, frying pan to smash (tap) foes and the cake sprinkles we do not really know what to do with yet.

The spells will probably be finished sometime next week.

Sprint 19

Goal

- Implement the new spells
- Sent new version to Krillbite
- Start working on documentation
    - Gather information about types to document
    - Latex VS. Word
- Create a user test and test some persons on the prototype

Results

The three spells are implemented as the “circle”(area damage), “triangle”(earthquake) and “square”(wall). These are situated in the left-hand HUD

The new version we sent to Krillbite have 10 unlock-able levels with a bonus selection at the end of each. In the bonus selection you can choose a new perk, spell or weapon.

SPRINT 20

GOAL

- Preparation for other exams (AI/Math)
- Fix bugs that occurs when playing tutorial level
- Work on documentation
    - Get all topics in a document

RESULTS

Using word to

SPRINT 21

GOAL

- Write documentation
- Pack everything in a zip that needs to be done

RESULTS

5. CONSENT FORM

1 CONSENT FORM FOR CAKE DEFENSE'S USABILITY

The game to be tested is a functional prototype of the game Cake Defense. It runs on the android operating system and will not take more than 15 minutes to complete. The test is anonymous, and we will not use personal information in our report. Remaining data collected from the test will be used by the development team to improve the game, and to be published in the final report. Testing is voluntary and the tester may withdraw at any time during the test.

PURPOSE A:

Let the development team get feedback from persons outside the development team.

PURPOSE B:

To see if the game play elements and user interface is simple enough to understand and use, and to find any bugs that we might have missed.

2 RESULTS

Data collected during and after the test will not include any personal information about the tester. Data collected will only be about what the tester did and did not do, and any feedback the tester may give.

I have read and understood the above text. I hereby give my consent to testing so that the test can be conducted.

Signature **Signature by a developer**

_____ _____

Date th of may 2012.

6. GROUP RULES

Group Contract

Agreements

- Meetings
  - Each group member should meet at school for Scrum meetings each Friday around 14.00. If a group member is unable to participate, he must inform other group members as soon as possible and try to be available on Skype if that is possible.
  - Each work day we will hold a short scrum meeting, around 14.00, where we will report problems that have occurred since last short scrum meeting, also deciding what actions needs to be done. These meetings are done orally, only issues and problems that occur will be logged.
    - Every Tuesday, Thursday and Friday we start at 11.00pm, at HiG.
- Deadlines
  - Each group member is responsible for finishing their work within the given time frame. If a member is unable to complete a given task, the member is responsible to inform other members and based on a short meeting we will take actions there and then. Dividing the work load and setting deadlines will be done at the end of each Sprint (Fridays).
- Work
  - Each group member agrees to work at about 18 hours a week; some weeks might be longer, some shorter, based on agreements within the group.
- Documentation
  - Each group member is responsible for logging his own work.
- Disputes
  - Disputes within the group will be decided by voting between the group members where everyone has an equal vote and a decision will be based on the highest vote.

Members

Øyvind Berg, 091046
Knut Øien, 091379
Henrik Lee Jotun, 091044

Signatures

ØyvindBerg

Knut Oien

Henrik Lee Jotun

7. ZIP FILE ITEMS

The zip-file that followed with this report contains the following items:

- This Document
- Newest source code for Cake Defense
- Code Documentation – Self-made
- Code Documentation Tool Created in C#
- Version Builds
    - Version1, version2 and version3
- Website
    - The project's website
- Usability Test Form
- Prototype Artwork
    - By developing team, not krillbite
- First item sent to krillbite
    - How it all started out
- Contract
    - Self-created between students and krillbite