Institutionen för datavetenskap
Department of Computer and Information Science

Final thesis

Exploring the Possibilities of Split Testing on Motion-Based Games

by

Martin Berggren

LIU-IDA/LITH-EX-A--15/047--SE

2015-06-18

Linköpings universitet

Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings universitet
581 83 Linköping

Linköping University
Department of Computer and Information Science

Final Thesis

Exploring the Possibilities of Split Testing on Motion-Based Games

by

Martin Berggren

LIU-IDA/LITH-EX-A--15/047--SE

2015-06-18

Supervisor: Aseel Berglund
Erik Berglund

Examiner: Johan Åberg

Abstract

Split testing is a popular practise to compare and evaluate design elements of web-sites. In this report, it is explored how split testing can be employed to test features of motion-based online games. This is done by applying theory taken from split testing of common web-pages and translating it to work with games. A pilot test is conducted on the newly developed game Phaseformer to evaluate two interaction techniques. With only 105 test participants, the results from the pilot test show with high statistical power that one variant provides the player with a higher level of control over the game than the other.

Contents

Chapter 1

Introduction

Chapter one will contain a brief introduction to split testing and the game Phaseformer which the testing will be applied on. This is followed by a motivation of the work conducted in this thesis as well as the purpose and limitations.

1.1 Background

Over the last couple of years, split testing has seen a huge popularity increase in the field of software engineering. Split testing however, is not a new phenomenon, it has been used in other fields for many decades and can be traced all the way back to the beginning of the twentieth century where a man named William Sealy Gosset used a similar technique as a way to monitor the quality of stout for the Guinness brewery [1].

Split testing has provided software developers with a way of verifying their ideas and guide the product development. Large companies such as Google [2], Amazon [3] and Bing [4] are conducting several hundred test each day on their sites in order to aid them in the development of their products.

While Google's testing could be focused on very tiny design improvements, such as finding the right shade of blue for their links [5], split testing has also seen an upswing in early product development where business critical decisions about a product has to be made. This upswing can partly be traced to the Lean Startup methodology, first introduced in 2011 by Eric Ries. The lean startup uses testing to take data driven decisions (as expressed by Ries "use validated learning") about critical features that defines the product to

shape a product very early on in the development process. [6]

Motion based games using advanced equipment, such as the Nintendo Wiimote [7] or Microsoft Kinect [8], have been very successful in the last decade. This has led the company Therese Kristoffer Publishing AB to start experimenting with more broadly available equipment for motion games such as a standard web camera, since it is available in almost every modern laptop computer. They would now like to employ split testing in their development to test how different interaction techniques work with their games and how users respond to different ways of interaction. It is their belief that a web-camera is an advanced enough tool for interaction to create a successful game.

1.2 Phaseformer

The impelementation and testing described in this thesis will be applied to the game Phaseformer. Phaseformer is a simple 2D platforming game originally developed at Linköping University. The game is currently in an alpha state with critical decisions and much development still to be made before a release can take place. It is therefore a very suiting subject to apply the tests described in this thesis on.

The game can be seen in figure 1.1. At first glance, it looks similar to other platform games with a centralized user maneuvered character with a game-world consisting of multiple platforms. The initial objective of the game was to simply jump around in the game-world and gather stars with no obvious goal. The lack of a clear goal is one of many things that are to be tackled in later sections of this report before any testing can take place.

What is very different with this particular platforming game though is how the player maneuvers the character. Character movement can be produced by applying motion in predefined "motion-zones" on the canvas as the game uses the web-camera for inputs.

1.3 Purpose and Scope

The purpose of this thesis is to define how split testing should be used to test motion based games. This will be done by creating, implementing and performing a test. The implemented module should be easily integrated in the software development process. The objective of the implementation is to allow for a game to be tested early on in its development while not hindering

Figure 1.1: A view of the initial Phaseformer game before any modifications described in this report were implemented.

further development on separate features during the test.

The theory section of this report contains a brief summary of best practises and how split testing should be implemented to properly test software. The main bulk of the theory is focused on testing web-sites that aims to increase its conversion rate and therefore not all of the related work are directly applicable to online games. As follows, the purpose of this thesis is also to elaborate on and suggest alternatives to common best practises of split testing where those are not applicable to split testing of games.

A test will then be conducted as a pilot test for this implementation to test a series of interaction elements for the Phaseformer game as well as analysing this test. This test will be focusing on the layout of the ”motion-zones” to test how players prefers to maneuver the game.

1.4 Motivation

In software development, usability testing has been a very popular practise for a long time [9]. Usability testing lets an observer study the behaviour of users of the product in a controlled environment and may be complemented with a questionnaire and/or an interview.

In contrast, the nature of split testing provides information about how a product or service is actually perceived by the users. Another great benefit of split testing in comparison to usability testing is that split testing does not force users to complete some additional step in order to leave feedback (e.g questionnaire or interview). Instead their actions are monitored while they are simply using the service in their natural environment. [10]

Comparing split testing to usability testing is not easy as both kinds of tests serve their own purpose. Split testing works best when the objective is to compare different ideas and find the best one according to a statistical result, also referred to as quantitative research. Split testing will not tell you "why" users preferred one variant over another. As a rule of thumb, split testing should be used to answer questions about a service that can be phrased as "how many" (ex. "How many users would prefer a larger button") and usability testing should be considered when the question starts with a "why" (ex. "Why do people not like my game") [11]. With this in mind, split testing was chosen as the tool to compare the different interaction properties of the Phaseformer game.

With the introduction of the Lean Startup methodology [6], Eric Ries emphasises how split testing should be used early in the products life cycle, where validation of business concepts rather than design details is of interest. This thesis aims to bring split testing into the development process of motion based games so that testing can be performed with minimal effort as soon as a minimum viable product(MVP) is ready. When developing a motion based game, several very important decisions has to be made. For example: How do users want to interact with the game? How fast should the game be running? If we use split testing to find a statistically significant evidence that people playing the game like some feature over another, then it may not be necessary ask them "why". Split testing could therefore be a powerful tool to test similar properties of games.

Using split testing to provide a data driven underlay for important decisions can take much weight of the developers who can now test ideas and gain feedback almost instantly instead of having to guess what the users will prefer and have the feedback delayed until the time of release [6]. Ron Kohavi et al. [4] underlines this further and suggests that most of the tests conducted results in the suggested modification under testing is not an

improvement and should therefore be scrapped. According to Kohavi, this is because even experienced developers are actually really bad at guessing what the users want.

To the best of our knowledge, not much work can be found about split testing features of motion based games. Therefore, the work of this thesis can be seen as breaking new ground for how split testing can be used with motion based games.

1.5 Problem Definition

This thesis will focus on the usage of split testing in the early forming stages of software, more specifically motion based games, to take critical application defining decisions. Special attention will be focused on how testing can be integrated in the development process as well as how a practical test should be conducted. The problem formulation for this thesis can therefore be summarized as:

- How can split-testing be implemented to evaluate design properties in the development of motion based games?

1.6 Delimitations

In this thesis, some limitations are made, mostly because of limited time but also as a mean of keeping the focus on the most important aspects of the work. The main limitation lies in the actual testing once the test module is implemented. In this thesis, only one A/B test will be carried out because of the time it takes to gather data from real users of the application (an A/B test is a split test where only two variants, A and B, are tested).

Chapter 2

Theory

This chapter will contain related work, theory and background information relevant for this thesis. Descriptions of testing and how to perform them as well as some definitions of terms used in conjunction with split testing.

2.1 Split Testing

In software engineering, split testing is one form of controlled experiments, conducted online. The most simple form of split testing is A/B testing where only two variants (A and B) are tested and it will therefore be explained first. The technical term for A/B testing is "two sample hypothesis testing". A/B testing has two variants of some service (could be a web page or a game), A and B, where A is often called the control variant and B is often referred to as the treatment variant. The control variant is usually the service in its current version, included in the test as a reference, hence the "control" name. The treatment variant contains some kind of modification compared to the control variant. Users of the service are then assigned to one of two groups (usually 50/50 but there may be exceptions), where group A receives the control variant when accessing the service and group B receives the treatment variant. When enough data is gathered, the performance of the two variants are compared to each-other by some measurable factor (could be number of clicks on advertisements or time spent on a certain page etc.). In this phase of the experiment, certain statistical algorithms are applied to calculate the performance of the two samples. If the treatment produced more desirable results than the control version with high enough statistical significance, then the treatment version should replace the control version as a new baseline version for the service for all users. [12]

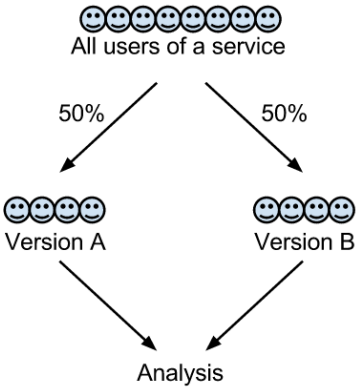

Figure 2.1: This figure illustrates the most basic form of A/B testing. The population is divided into two subgroups of equal size that each receives one variant of the software in testing.

Split testing differs from A/B testing by one important factor. The difference is that split testing is not limited to two variants to be tested so split testing could therefore be seen as an extension of A/B testing. [13]

Multivariate testing is a further extension of this kind of testing. Multivariate testing is when M number of features, each with N number of possible values, are tested together resulting in a split test with N^M number of concurrent running variants of the tested service. Multivariate testing can be seen as a more complex form of split testing suitable for large companies running several tests at the same time. Multivariate testing requires a much higher amount of users observed in the test in order to provide a statistically significant result and should therefore be approached with care if your service does not have a very high amount of visitors. [14]

Worth noting when studying different versions of controlled experiments is that different sources may not be very consistent in their definition of certain tests. Some sources will refer to all forms of the above mentioned tests as A/B tests even when more complex versions such as split testing would be the appropriate wording.

2.2 Why use Split Testing?

The reasons why one would want to employ split testing are many and have a broad range. This section will cover the most commonly mentioned reasons of using controlled experiments online. First and foremost, the objective of split testing is to improve an online service and online experiments have proven to be a powerful tool to do so. In an online era, the gathering of large amounts of data has never been easier. Controlled experiments make use of this data to provide validated information of how end users are interacting with the service. [15]

For Microsoft's search engine Bing, optimizing tiny design elements, such as the placement or size of advertisement, to improve the click-rate on the advertisement by a very small percentage can generate millions of dollars in less than a year. Controlled experiments is a very powerful tool to find these optimizations and can explain why large online actors such as Microsoft, Google, Facebook and many more conducts several hundred experiments each day. [4]

Split testing is not limited to finding minor optimizations of an up and running service, applying testing prior to release of a software to take major feature decisions can also be very beneficial. Split testing a partly developed service on a small set of users can provide insight of if the service will be of any use for intended users or if major course-changing actions are necessary. Testing on this level is closely related to lean development where the goal often is to test and verify a business idea rather than a service. The paths of action to take after such a test has been commonly known as to preserve (i.e to stay on current course) or pivot (drastically change the product or business idea). [6]

Furthermore some argue that giving the developers the opportunity to easily test a new feature, in addition to the rather short feedback-loop to get insight of how that feature performed, will both improve the overall quality of a service as well as motivate and inspire developers to be more productive. [16]

2.3 Definition of Terms

In this work and other literature about split testing, certain terminology may be used that not everyone are familiar with. This section will introduce and define some terms used throughout this report.

2.3.1 Overall Evaluation Criteria

The overall evaluation criteria, often shortened to OEC, refers to the critical metric that the testing process tries to improve [17]. The OEC could be a simple single metric, such as number of clicks on advertisements on a web-page, items put in the shopping cart in a web-shop or search queries executed on a search engine. When the objective of a test is to improve several variables, it is suggested that each single metric is identified and merged together, possibly as a weighted combination of single metrics, as a way of selecting a one dimensional scale of performance [17]. A combination of objectives to a single metric will impose trade-offs between the different objectives but it also serves as a way of uniting the entire organization behind a shared objective.

Alternative words for the OEC found in other literature includes dependant variable and performance metric. When choosing an OEC, it is very important to not select a OEC that is easily improved in the short-term at the expense of long term value [4]. An example of this is if a search engine would select number of search queries executed as their OEC, then a version of the search engine that give less accurate results on average would result in users having to execute more search queries before finding what they were originally looking for. This would result in more queries executed short term whereas in the long run, users would instead switch to using other search engines [18]. An example of a good OEC to choose would instead be one that includes life-time value of a user, at Bing, sessions per user (or number of times a user returns to their page) per month is used as their OEC. As the examples above suggests, lack of caution when deciding the OEC for testing could seriously hurt a company and it is therefore advised to not rush this step. [18]

2.3.2 Variant

Alternative words found in other literature include ”version” and ”treatment”. A unique set of testing parameters resulting in one variant of the service to be tested. For example, in A/B testing, we have two variants, A and B.

2.3.3 Experimental Unit

The experimental unit is easily defined as a monitored user of the system in testing. Metrics for each experimental unit is measured before the analysis can take place. Each unit is assumed to be independent in this kind of

statistics. For readability reasons, experimental unit is often replaced with user or player in this report.

2.3.4 Null Hypothesis and Alternative Hypothesis

Before performing a test, two (or sometimes more) hypothesis should be defined. The null hypothesis should be formulated so that the test aims to disprove it. The alternative hypothesis should provide an alternative that may be true if the null hypothesis is proven wrong. In A/B testing, the null hypothesis is often formulated as $H_0 : (\mu_a == \mu_b)$, where μ_a and μ_b refers to the performance of version a and b. This means that the initial hypothesis is that both versions will perform equally good in the test (which the test then aims to disprove). [19]

It is strongly suggested against formulating the null hypothesis as $(\mu_a > \mu_b)$ since that formulation will give the conductors of the test a favourite version prior to the test which might, in some cases, impact the results [20].

2.3.5 Random Assignment

When users are assigned to one variant of a test it is very important that it is done randomly, and this is called random assignment [19]. If it is not done randomly, or just semi-randomly influenced by some factor, then the result can be biased and the result can not be assumed to apply outside of the tested population.

In some specific cases only a certain sub-population is relevant for a test. In that scenario, only users belonging to this group will be picked for the test. However, it is still important that those selected users are still divided randomly between treatments. One example of when only a sub-population is interesting in an experiment is if we want to split test how users with a specific web-browser react to a change that only affects that specific browser. [19, 20]

2.4 How to perform a split test

Much work has been done to explore the possibilities and find best practises of split testing. In this section, general guidelines will be summarized to provide a brief explanation of the most important steps to consider when performing a split test. Examples will be given for each step for game related

testing.

The first objective when performing a split test will always be to decide a goal of the test. It is very important that the goal is clear so that everyone involved can work united on the project. An example of a goal could be to increase the popularity of an online game. [6, 12, 13, 14, 15, 18]

When a goal is set, it is time to decide a metric to measure the progress towards that goal, explained earlier as the OEC [13, 15]. When testing a web-site, a common OEC is the percentage of users that perform some specific action, for example clicks on an advertisement or buys a product, this is referred to as conversion rate. In game related testing, the OEC could be percentage of returning players or total amount of time played by each user.

Next step in the process is to design the alternative variants of the service(or game) to be tested [13]. Alterations to a game suitable for testing could be alterations to secondary game objectives [21] or different sources of input devices [22]. The possibilities here are basically endless. Sometimes it may be valuable information that the variant currently tested is performing worse than the control variant which in turn validates that some part(s) of the control variant is working well [4, 6]. It is important to remember that all variants included will be presented to users of a service. This means that including a very bad variant may result in users receiving that variant to stop using the service resulting in loss of user life-time value [14]. This is very hard to avoid completely but one way of reducing this issue is to stop an ongoing test as soon as one variant proves to perform better than the other and not prolong the test past that point.

When the variants are finalized, it is time to add the code to the service in testing [13]. There exists a wide variety of pre-compiled frameworks for testing to aid the developer with this step. There are a few important things to remember when implementing the test in the service. First of all, it is important to have a good randomization process of assigning the users to a group to receive a certain variant. Returning user has to be assigned to the same variant to avoid confusion that may otherwise affect the results. Analyzable data that verifies the OEC has to be gathered to some database. Collecting additional data of some other part of the service should also be considered in this phase. Because of how easy it is to gather data, "Over instrumenting is better than under instrumenting". [12]

The following task is then to upload the service with the test implemented to let users access it. At this point in time, it is not much the test conductor can do before enough data is gathered to perform an analysis on. This part can take a long time depending on how many users that are accessing the service (see *2.4.1 When should a test end* for more details).

When enough data is gathered, analysis can take place by applying the statistical model on the gathered data. The analysis aims to disprove the null hypothesis and in practise also evaluate which variant performed better than the other.

The final step in the testing process is to use the knowledge gathered from the test and perform some action. The action is very often to decide which of the tests variants that should be implemented as a new baseline variant for all users. [4]

2.4.1 When should a test end

Deciding when it is time to end a test is done based on when the analysis of the gathered data can show statistical significant evidence that one variant is performing better than the other(s) [13]. This happens when the confidence interval surrounding the mean performance value for each variant are no longer overlapping with a probability higher than a predefined threshold [13]. This threshold is called the confidence level of the test. Choosing the confidence level should be done with great care. Using a confidence of 90% or 95% are common in split testing but both higher and lower could be used in certain situations. If a confidence level of 90% is used, that means that the test will end when it is less than 10% chance that the currently best performing variant will perform worse than some other variant [13]. However, a high threshold is not always the best as a small increase in confidence could require a large increase in needed test participants in order to arrive at a significant result. It is therefore important to carefully select a balanced threshold level for each test.

In later sections of this report, the parameter p will be used to describe the statistical significance of a test. The p-value is the complement of the confidence level. A p of 0.2 or less means that the test has reached a confidence level of 80%. The parameter Z will also be used to describe the *standard score*. Z is the signed measurement of how many standard deviations an observation is above the mean [23].

Explaining the math behind split testing in more detail is beyond the scope of this thesis but the interested reader is encouraged to study this further, for example in either source [19] or [20].

A great way of reducing the amount of users needed to reach a statistically significant result is to have large differences between the tested variants. A tiny change between two variants will generally require much more participants to find which variant is performing better, if there even is a difference between the performance of the variants.

2.4.2 Choosing an Analysis Method

Which analytical test to use when evaluating the data gathered from a test depends on several variables. The most important thing to remember when searching for a suiting statistical test is that different tests assumes certain facts about the gathered data. It is often both easier and more efficient to use a statistical test that assumes more but these kinds of tests are also seen as "weaker" in comparison to tests that have fewer assumptions.

When the OEC can be measured as either completed or not completed for each user (example, clicking on an advertisement, commonly referred to as conversion rate) then mean percentage of users completing the OEC in each group is likely to be the interest of the analysis. In this scenario, the distribution within each population is often assumed to be normally distributed and in these cases, a standard two sample t-test is commonly used for A/B testing purposes. [15]

When measuring a single true or false variable is not enough, a more complex OEC has to be chosen which in turn may call for another statistical evaluation method. If the data is not normally distributed, a nonparametric method of evaluation has to be used. For example, this could be the case in split testing of games where more complex metrics, such as time played, is often more interesting than what percentage of players that finished the game and that is also the case in this thesis. [19, 21]

2.5 GameFlow

The GameFlow theory tries to define what aspects of a game that makes a game successful. In this thesis, the GameFlow theory is used to understand what aspects of the game that the test tries to improve.

The Gameflow theory is built upon the flow theory which describes the feeling of flow as when a person is performing a task for no additional reward than performing the task itself. For example, a mountain climber does not get paid to climb, the reward is to perform the task itself. [24] Translated to GameFlow that means that a person plays the game for no additional reward than simply playing the game. The aspects that GameFlow is built upon are:

- A task that can be completed - The Game
- Concentration - Ability to concentrate on the task

- Challenge, Player Skills - Perceived skills should match challenges and both must exceed a certain threshold
- Control - Allowed to exercise a sense of control over actions
- Clear goals - The task has clear goals
- Feedback - The task provides immediate feedback
- Immersion - Deep but effortless involvement, reduced concern for self and sense of time
- Social Interaction

Chapter 3

Method

In order to answer the scientific question, a test will be constructed, implemented, performed and analyzed on the Phaseformer game described earlier. The analysis will then cover both the implementation part and also the actual test conducted.

In this chapter, all the work conducted in this report is described. The chapter is divided into several smaller topics. The method is based of the the related work studied in the Theory chapter.

3.1 Pre-study

In order to conduct the test, a testing framework had to be selected. As the number of available frameworks for this task is rather large, a small pre-study was conducted in order to select a suiting framework for this test. This pre-study is rather limited in the amount of frameworks reviewed but should still be seen as a necessary initial step to evaluate certain aspects of the frameworks before selecting one to work with.

In order to find split testing frameworks to evaluate, the list found on the Wikipedia page for A/B testing frameworks was used [25]. That list was then expanded by searching the web. A few examples of search terms used in this step include "A/B testing framework", "A/B testing framework javascript" and "Split testing framework python".

When reviewing the frameworks, a few key aspects was evaluated and ranked against each other. Those aspects are listed below followed by a short de-

scription of each item.

- Cost
- Work to perform a test
- Work to modify the framework
- Trust in the author

The cost factor is merely whether or not the framework was free to use. Paid services for split testing was immediately discarded from further evaluation as there exist several free alternatives of seemingly equal quality.

Work to perform a test factor tries to estimate how easily implemented and integrated a certain framework is with the Phaseformer game. As the game is written in javascript, this favoured certain javascript based frameworks.

Work to modify the framework was a rough estimate of how much additional work that needed to be done with the framework to modify the out-of-the-box version into a more appropriate version for the tests carried out in this thesis. A framework that was not open source was therefore discarded here unless its stock version were found fitting.

Trust in the author of the framework is a rather vague and also subjective factor. It was used to rank the source whether the framework came from a well known source and/or if it has been used by large known companies.

With all of the above things considered, a decision was made to use a testing framework named Abba [26].

3.1.1 Abba

The reasoning about this decision follows the above mentioned factors. Abba is open-source and therefore free to use and modify. Abba is javascript based and the work to conduct a test was, subjectively, much simpler compared to the alternatives.

To start a new test it is required to host the Abba backend service and tie it with a mongo database. After that a script-file named abba.js needs to be included on the website in testing by using something similar to this line of code:

```
<script src="//serverURL:serverPORT/v1/abba.js"></script>
```

When the abba.js is included, the variants needs to be defined. That is done with code looking like this:

```
Abba('My Test')
    .variant('Variant 1', function(){
      //Do something
    })
    .variant('Variant 2', function(){
      //Do something else
    })
    .start();
```

It was highly valued that apart from this small code-snippet, the web-site remains unaltered. In practise, this means that a single version of the code is able to contain both variants which in turn means that the developers can continue to work with other parts of the code without having to update two code-bases during the test.

Finally when a user completed some task, a result should be recorded. This is done with the following line of code:

```
Abba('My Test').complete();
```

This is all the code needed in order to setup a new test. This straight forward way of setting up a new test was a very large factor of why Abba was selected as the framework to work with in this thesis.

The main drawbacks of Abba found in the pre-study were that it did only support the more common conversion rate tests where a test participant can either only complete or not complete the test. This was however also the case for all the other frameworks reviewed in the prestudy so the selected framework would have to be modified in order to fit a more complex OEC. How this modification was done is described later in this chapter.

Finally, regarding trust in the author, Abba fell behind some alternatives from Google and Amazon. However, before going open-source, Abba was initially built for Stripe [27], mobile payment system, which is evidence of one known company using the product.

3.2 Preparation for testing

As mentioned in the introduction chapter, the testing was done on the Phaseformer game in order to find if split testing could aid the developers in find-

ing a better way of interacting with the game through the placement of the motion-zones. The process of conducting this test has roughly followed the process suggested for software related split testing in chapter 2.4. Deciding the rough goal of this test was therefore rather easy since it was the premises of this work to improve the user experience of the Phaseformer game. However, measuring this goal was not a trivial task. What variables are suiting to monitor in order to define how much users appreciate the game? In order to answer that question and decide a suiting OEC for the test, the GameFlow [24] theory was used.

3.3 Designing the Variants

The details of the test were selected together with the stakeholders of this project. The objective of the test was to compare different interaction properties of the Phaseformer game, or more specifically, the layout of the motion-zones used for input. The two variants used in this test was developed with regards to feedback and suggestions from the stakeholders. The outcome of this process was two rather different motion-zone layouts to be used for the testing. As mentioned in previous chapter, large differences between the variants often require less amount of users to distinguish a winner.

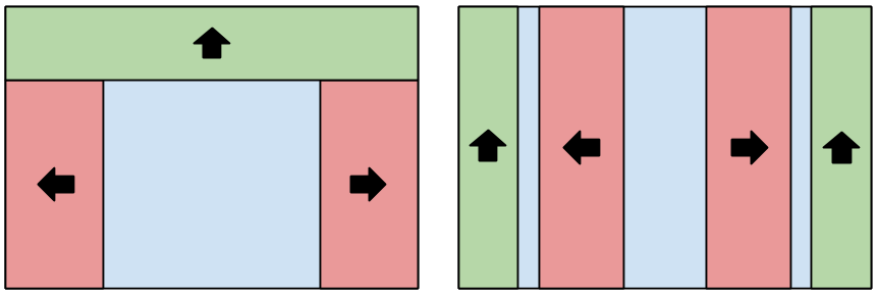

Figure 3.1: The two variants of motion-zones layouts. Variant A to the left and variant B to the right.

The two final versions can be seen in figure 3.1. Motion in the green zones results in the in-game character performing a jump. Motion in the red zones results in the character moving in the direction of the black arrow. Motion in the light-blue area will not trigger any action at all and could be seen as ”safe-zone”.

3.4 Deciding the OEC

The decision of a suiting OEC was possibly the part in the testing process that differ the most between split testing of a website and split testing of a game. This was based on that the goal of the test on a website often results in an OEC. For example, the goal of a split test on a website could be to increase the percentage of users clicking on an advertisement, then the OEC would obviously be that percentage for each variant. In this thesis, the goal of improving the user experiance of the Phaseformer game does not generate an obvious OEC and therefore more work was put on this step than the average website split test.

The process of defining a fitting Overall Evaluation Criterion for this project began with the overall goal in mind of improving the user experience of the Phaseformer game. As the overall user experience of the game is hard to measure, the GameFlow theory was used to break down user enjoyment of the game in the seven GameFlow aspects. Important to remember here is that only the layout of the motion zones will differ between the variants in the test and that all other parts of the game will remain identical between the two variants. One of the aspects from GameFlow that could be affected was the the control difference. Since it will almost certainly affect how well a player can control the game. That does also mean that Challenge/Player Skill could be affected as one variant could result in the game being harder. It is also possible that Immersion may be affected by the test if one variant feels more natural than the other. It could result in that the player will get more involved in the game and play longer.

With the above developed theory in mind, a list of possible OEC's were compiled. All of the suggested OEC's were:

- Total amount of time played
- Total amount of stars gathered (secondary game objective)
- Furthest progress in the level
- Percentage of players completing the level
- Percentage of players restarting the level
- Times completed the level
- Amount of stars gathered divided by time played (Stars/Second)

One thing to keep in mind here is that the test was thought to gather a rather small number of participants, possibly less than a hundred, so the

OEC should be able to gather data from every player since a OEC measuring some action that only a small amount of the users are performing requires a very large amount of users to participate in order to achieve statistical significance of the test [15]. That was therefore an argument against using the percentage of players completing the level as that could likely result in measuring an action completed only by a small fraction of the players.

Total time played per user could be a suiting OEC for this test since that would provide a value for every user that plays the game. E. Andersen et al.[21] also use total time played as their OEC when testing another platforming game. The idea behind using total time played were that if a player enjoy the game more, he or she would spend more time playing it and this OEC would therefore be closely related to the goal of increasing the user experience of the game and also related to the immersion aspect of GameFlow.

Because of the fact that split testing of games lacks best practises and guidelines where the guidelines for traditional split testing of websites are not applicable, it was decided to not limit ourselves to only gather data of total time played. Additionally since the objective of this thesis is also to investigate how split testing should be applied in games, it was decided to gather additional OEC's from the list. The final metrics selected for the test were "Total amount of time played", "Total amount of stars gathered", "Furthest progress in the level" and "Amount of stars gathered divided by time played".

3.5 Implementation of Abba

As seen in the pre-study section, the outcome was that the javascript based framework Abba was used. In order to further verify that this was the right choice, Abba was implemented and tested on a very simple HTML page with only the most basic items required to study the functionality of the framework. This process provided additional insight in how Abba had to be implemented in Phaseformer as well as what parts of Abba that had to be modified to fit in the testing of Phaseformer. It should be seen as a very important step to try a framework in a simplified environment to better understand how it works before implementing it in the actual environment. The main part of the modifications to Abba (which can be found in the next section) was actually made before implementing it in the game since the modifications were easier to test and verify in the separate environment.

3.6 Modification of the test-framework

The main drawback of using the Abba framework was that it did only support tests that could either be completed or not completed out of the box. This was the case for all reviewed pre-compiled testing frameworks as the objective for most A/B tests is to increase the conversion rate (ex. click rate on some advertisement) for a website. Since the OEC's for this test were of more complex nature, a metric spread along a scale rather than a true or false value, some modifications had to be made to the framework in order to support this. These modifications include modifications to the clientside javascript file as well as the server-side back-end and database. This should be seen as the most important step when setting up a test as it is of top-most importance that the test-framework does what the tester wants it to do.

The Phaseformer game is written in javascript using the Phaser game engine. As the clientside parts of Abba were also written in javascript, this made the integration easier. Modifying the clientside of Abba to be able to forward values in order to record data for chosen OEC's was done by modifying the Abba coffee-script files that generates the clientside javascript. Preserving the simplicity of setting up a test, that initially were one of Abba's greatest strengths compared to its competitors, had topmost priority throughout the modification process of Abba.

In order to receive and make use of the additional values sent from the client, the Abba back-end and database had to be modified accordingly. The Abba back-end is written in Ruby and make use of several gems. This is a decently sized application and the complete absence of code-descriptions and comments made this process somewhat tricky. Additional research on both the Ruby language and some gems (Sinatra and Mongo-helper to mention the most important ones) was done before these parts of the code could be fully understood and thereafter modified. The modifications were made so that the back-end could receive the extra values sent from the client and save those values in the database.

3.7 Modification of Phaseformer

Incorporating the test-framework in the software targeted for testing is not always a simple process. As described in the previous section, large modifications may have to be done to the test-framework. But even after the test-framework is ready for implementation, the targeted software is not necessarily created with split-testing of certain features in mind and this software may also have to be modified. This was the case when it was time

to implement Abba in Phaseformer. In this step, Phaseformer were extended with some missing functionality, for example, a played timer, a checkpointing system and a progressive level. The result of these modifications can be found in the results chapter.

3.8 Hosting of the test

To perform the test, both the game and the test-framework back-end had to be hosted on the internet to be available for the public. As a request from the stakeholders, the OpenShift platform for cloud-hosting were selected for this task. Linköping University is hosting their own OpenShift server that are already used for hosting purposes for some web-design courses and this server were therefore seen as a great candidate for hosting said software.

3.9 Testing

In order to perform the actual test, data had to be gathered from a large group of test-participants. Split-testing has the positive feature of not requiring participants to perform any other task than using the tested software the way they normally would. Finding test participants is therefore not harder than spreading the URL-link to where the software is hosted. However, as the Phaseformer game was not previously released and had no established player-base, this task required manual effort to spread the word of the game. A plan was developed to use social media to spread the game to friends and family, they were in turn encouraged to further spread the game in order to gain enough test participants. Additionally, the plan was to post the game on some on-line forums, mainly game development oriented.

3.10 Analysis

Once the data was gathered, an analysis was performed. The data was be extracted from the mongoDB database using the mongo export tool. Several sources [19, 20, 21] recommends using the non-parametric Wilcoxon ranksum test for comparing populations of different sizes where normal distribution can not be assumed, this test was therefore chosen for the analysis of the gathered test-data. Statistical significance of 0.2 was chosen (a confidence level of 80%). This means that we require the p-value from the ranksum test to be less than 0.2 in order to reject the null-hypothesis. The p-value represents the probability of recording a more extreme result than measured

if there is no difference between the two compared populations. The result of this analysis can be found in the next chapter.

Chapter 4

Results

This chapter will contain the results of this thesis. First, a description of how the framework and the game was altered in order to perform the test. After that, the full results from the A/B test of Phaseformer is presented.

4.1 Modification of the Test-Framework

This section will describe the modifications made to the Abba testing framework. There are some additional modifications to Abba not mentioned in this section that are not test-related. Those are described in the "Non Test-related Modifications" section instead.

4.1.1 Modification of Clientside

To support the passing of test-variables as described in section 3.4, the index.coffe file was modified. The index.coffe file is the source of the abba.js file that the target website for testing imports. As a result, the "complete" method that is called from the software in testing, as described in section 3.1.1, were modified to support several optional arguments that were required to record data for the OEC's. After the modification, the complete method could be called with up to eight optional parameters of this structure:

```
Abba('My Test').complete(time, stars, checkpoints, started,
    finished, restarted, alttime);
```

The time parameter is the value in seconds that a user has played the game that has not previously been recorded. The stars parameter is the number of stars the player has gathered since the last complete request were sent to the server. The checkpoint parameter is the number of the furthest checkpoint the player has reached. The started, finished and restarted parameters tracks if the game were started, finished or restarted. Alttime is an alternative time measurement that will be further described in the next section. A more in depth explanation of how the values are stored can be found in the next section. A description of how the values were sent from the game can be found in section 4.2.1 .

As all of the parameters are optional, an example of a complete call from the software in testing that records a time of 10 seconds played would look like this:

```
Abba('My Test').complete(10);
```

Additionally, the complete-method in the index.coffe calls an internal private method called recordComplete which was also slightly modified in order to support the extra variables passed with the complete request.

The final modification of the index.coffe file were the removal of the check that tracked if a test were already completed as this was no longer a necessary feature. This check would initially stop the client from sending several complete-messages to the server as Abba initially only supported tests that could either be completed or not completed (i.e not completed several times).

4.1.2 Modification of Serverside

In order to store the extra data mentioned in the previous section, the server-side of the Abba-framework was also modified. As mentioned in the previous chapter, this part of Abba is written in ruby. The files modified for this task were app.rb and requst.rb, variant.rb and experiment.rb located in the /models/ folder. The files in the models folder represents how the data should be stored in the mongo database. Each experiments have one or more variants. Each variant has a request for each test-participant that received that variant. The modifications done in this step were mainly to the request.rb file that models what values are stored for each variant. Several keys were added to support all of the data from the previous section.

```
key :time, Integer, :default => 0
key :alttime, Integer, :default => 0
key :stars, Integer, :default => 0
```

```
    key :checkpoints, Integer, :default => 0
    key :started, Integer, :default => 0
    key :finished, Integer, :default => 0
    key :restarted, Integer, :default => 0
    key :endtime
```

It was very important that the data received at the server would be stored to the correct test participant's result. To solve this problem, a way of identifying test-participants uniquely was implemented. The solution were the use of the IP-address that the request came with. Abba initially had code that tracked the IP of a user that started the test in request.rb. This was used for the identification process when completed-requests containing test-data to be stored were received. A method named "update" were created in variant.rb that handles a newly received complete-request.

```
def update!(request)
  started = self.started_requests.find_by_ip(request.ip)
  if(started != nil)
    started.increment(:time => Integer(request["time"]))
    started.increment(:alttime => Integer(request["alttime"]))
    started.increment(:stars => Integer(request["stars"]))
    started.increment(:started => Integer(request["started"]))
    started.increment(:finished => Integer(request["finished"]))
    started.increment(:restarted =>
        Integer(request["restarted"]))
    if started.checkpoints < Integer(request["checkpoints"])
        started.set(:checkpoints =>
            Integer(request["checkpoints"]))
    end
  end
end
```

The update method takes a request as an input paramenter, then finds the started_request with the same IP-adress and updates this participants data with the data from the newly received request. Both *time*, *alttime*, *stars*, *started*, *finished*, *restarted* and *checkpoints* are data received in integer format and all of them except *checkpoints* adds the new value to the old value. Example: A player has a time played of 100 seconds, the server then receives an update that the player has played 10 more seconds, the old value of 100 is then incremented by 10 for a new value of 110 that now represents total time played for that player. The value for checkpoints differ slightly from the others. As the OEC defined in section 3.4 is described as "Furthest progress in the level", the received checkpoint value will only replace the old one if it is a higher number.

All of the above modifications is enough to let Abba record data for the OEC's for the test.

4.2 Modification of Phaseformer

Several changes were made to the Phaseformer game in order to make the game ready for a pilot test. This section will describe all of the work done to the Phaseformer game. This section is further broken down into test-related modifications and non-testrelated modifications.

4.2.1 Test-related Modifications

As described in previous section, the testing framework now supports the recording of the data needed for the test. However, the game was also modified to provide the values for the OEC's.

To measure the OEC of the time a player spent playing the game, a timer was created. As it was thought that players often leave games running while not actively playing, the timer had to filter out inactive players. In order to solve this problem, an inactivity-check were created. Two timers were created for this task, the activity-timer and the playtime-timer. The activity-timer started on 60 seconds counting down towards zero but refreshing itself to the full 60 second duration each time the user collected a star or reached a checkpoint. The playtime-timer starts from zero and counts the time a user spent in the game with the condition that the activity-timer having a positive value, else it would pause. The objective of these timers were to filter out data from inactive players that did not perform any meaningful action in the last 60 seconds. As it is not obvious how long a player should be inactive before the timer paused, an alternative timer called "alttime" were also implemented. This timer is basically a duplicate of the first inactivity-check with the only difference being that the activity timer now counts down from 180 seconds.

To measure the OEC of player progress in the level, a checkpoint system was implemented. The checkpoints are completely invisible for the players as they do not affect the game-play in any way. Throughout the level created for the test, there are a total of 15 checkpoints numbered from 1 to 15. A simple integer value stores the number of furthest collected checkpoint in this play section.

The initial version of the game supported counting the number of stars collected and as a result no further development were needed on the game

to support that OEC. Finally, "stars per second" could be generated from star- and time-data and did also not need any additional development.

To send all the test-data to the server, a loop were created in the game that runs once every ten seconds while a player is playing the game. The loop calls the complete method from the abba.js file with the data to be recorded. One iteration in this loop could look like this:

```
Abba('My Test').complete(playedTimer,
    (score-recordedScore), checkpointNr);
playedTimer = 0;
recordedScore = score;
```

Resetting the played-timer was based on how the data was handled on the server side where time is added to the old values to store the total amount of time played. To prevent data from being recorded multiple times, the played timer has to be reset after every complete call. The recorded-score variable solves the same problem of not recording the same data multiple times for the star-data. The extra recordedScore variable was needed as the score variable could not be reset in-between complete-calls as it is used elsewhere in the game.

4.2.2 Non Test-related Modifications

The Phaseformer game were initially very early on in its development process and before any meaningful testing could take place, several other improvements and modifications was done. Below are a small summary and descriptions of additional modifications made to the Phaseformer game.

A new, bigger and improved progressive level was created. Several things were considered when developing this level, for example, both sides of the players body should be used and therefore, a traditional 2D platforming level where the player continuously runs right would not be a very good one. The idea of a vertical level where the objective is to climb to reach the goal was formed. After initial testing of such a level, it was decided that such a level would be too difficult for new players of the game. The level was redesigned to start at the top with the goal at the bottom. This was found to be at a more fitting difficulty level and was chosen as the final design to use when the split test would be carried out. 100 stars were spread around the level for the player to collect along with 15 checkpoints in order to track of player progress in the level. Some difficult jumps were kept in the level that had to be completed by players that wish to gather all 100 stars, however, if the player only wish to reach the finish no difficult jumps or maneuvers are

needed. It was estimated that a first time player would need around 10-20 minutes to gather all 100 stars and reach the finish.

Similarly to the checkpoints, an end-goal was created that would pause the game when a player reached it, tell the player how many stars he/she gathered and finally ask the player if he/she would like to play again.

The game was also zoomed in to get a better view of the character and the area close to him. This was done by simply reducing the resolution of the game which also helped counter some frame-rate issues present on less powerful computers.

Some minor tweaks and additions were also introduced on the web-site the game was hosted on. This includes a new background texture and a small help-section. The help-section have the objective of explaining for new players that they need to enable their web-camera and also how they move their character by producing motion in the motion-zones. On a request from the stakeholders, a register-functionality were also created on the website. The idea were to let users sign up if they wanted to participate in the testing of future games. To implement the registration-functionality, the Abba framework were extended with a function called "mail" that took an e-mail address as input and sent it to the test-backend server for storage in the same mongo database.

4.3 Hosting of the Test

When both the test-framework and Phaseformer were ready for the test, both parts were hosted on-line to be widely available for the public. As mentioned in 3.8, the Openshift platform were selected for this task.

For the A/B test performed, Abba was hosted on:

```
abtestactivelab-eribe.openshift.ida.liu.se/
```

The abtestactivelab-domain was set-up with ruby 1.9 and mongoDB 2.4, both required by Abba. The Phaseformer game was hosted on:

```
myflaskapp-eribe.openshift.ida.liu.se/
```

On this domain, a python web-server were used to host the game. This domain also had an alias to simplify the address. This alias were:

```
activelab.ida.liu.se/
```

4.4 Test Results

When both the game and the test-framework were successfully hosted on the web, the actual test begun. For almost three weeks, data was gathered from all players of the game. During the duration of the test, most of the effort was put into spreading the game to get more people to try it and in turn gather data from. Many channels were used to spread the word of the game and all participants were encouraged to show the game to their friends and relatives. The rest of this chapter will present the gathered data from the test and the analysis that was done on this data.

4.4.1 Gathered Data

During the A/B test, 243 people visited the Phaseformer website. 138 of those had a total time played of 0 seconds, this means that they never got the game to start and they are therefore not included in the analysis. This leaves 105 valid data entries with a time played greater than 0 seconds. 49 of the 105 participants received variant A and 56 received variant B.

A total of 14 participants reached the end of the level. 9 of those used variant A and 5 used variant B. This gives that 18.3% of variant A participants completed the level while 10.7% of variant B participants completed the level. Out of the 14 that completed the level, 9 used the button to replay the level, 6 from variant A and 3 from variant B. That means 12.2% of variant A participants wants to restart the level but only 5.3% of variant B participants.

As mentioned in the Method chapter, the Wilcoxon ranksum test was used to analyse the data. In order to draw any conclusions, the result had reach a statistical significance of $p = 0.2$. The computer software Matlab [28] has built in functionality to calculate the Wilcoxon ranksum test and Matlab was used for all calculations mentioned in the remainder of this chapter.

First up in figure 4.1 is data for "Total amount of time played". X-axis displays the time played in seconds and Y-axis holds the percentage of players that played for at least the amount of time shown on the X-axis. For variant A, the mean time played were 282 seconds, the median time played were 80 seconds and the highest time played were 2660 seconds. For variant B, the mean time were 204 seconds, median time were 85 seconds and the highest time were 1900 seconds. This means that the median player of version B played 6% longer than the median player of version A, however this effect was not statistically significant, $Z = -0.4376$, $p = 0.6617$.

Figure 4.2 shows data for "Total amount of stars gathered". X-axis shows

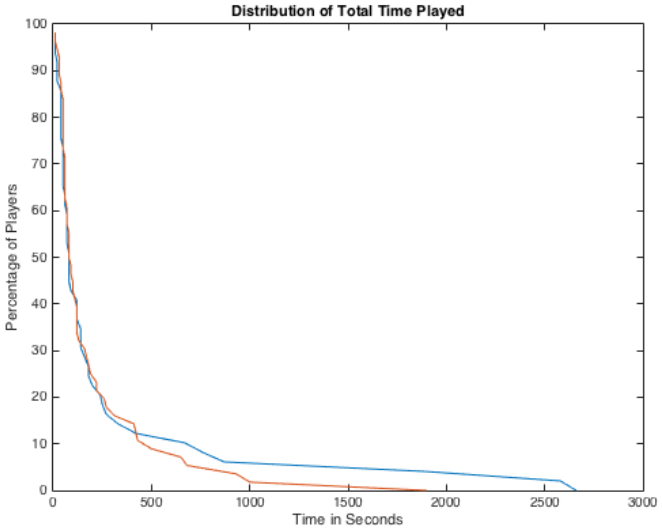

Figure 4.1: This graph shows data for total amount of time played for variants A and B. Variant A is displayed in blue and variant B in red.

the number of stars and X-axis display the percentage of players. For variant A, the mean amount of stars gathered were 33 with a median of 9 and a highest value of 299. For variant B, the mean amount of stars gathered were 16, the median were 8 and the maximum value were 171. The median player of version A therefore gathers 12.5% more stars than the median player of version B, although as in the case with the time, this effect is also far from statistically significant, $Z = 0.5116$, $p = 0.6089$.

Figure 4.3 displays the data for number of checkpoints collected. As described earlier, only the number of the furthest checkpoint collected is stored and this is used to measure how far the players progressed in the level. There are a total of 15 checkpoints in the level and 15 is therefore the maximum achievable value in this graph which means that the player reached the goal. For variant A, the mean number of furthest checkpoint collected were 5.1 and the median number were 3. For variant B, the mean number of the furthest checkpoint collected were 4.0 and the median number were 3. Both variant A and B had players reaching the final checkpoint and therefore does both variants have a highest number of 15. The progress of the median player of either variant did not significantly differ, median = 3 for both variants, $Z = 0.4893$, $p = 0.6246$.

Figure 4.4 shows the combination of star and time data by dividing the total

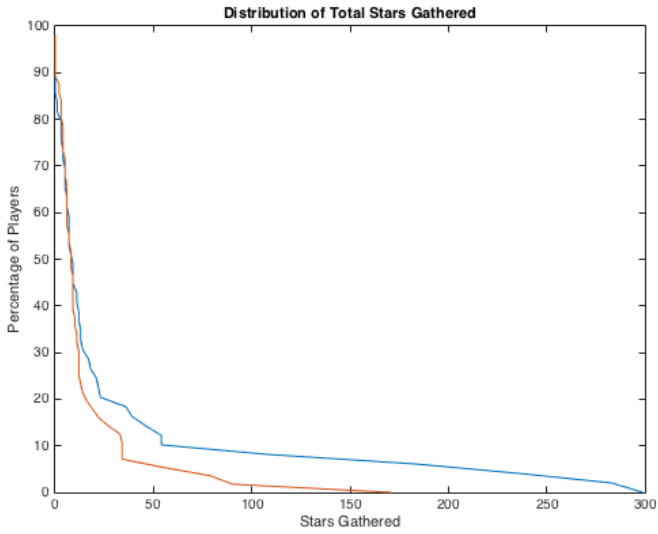

Figure 4.2: This graph shows data for total amount of stars gathered for variants A and B. Variant A is displayed in blue and variant B in red.

amount of stars gathered with the total time played for each test-participant. This gives the "stars gathered per second" metric used to measure the control factor discussed in the GameFlow theory. For variant A, the mean number of stars gathered per second were 0.105, the median number were 0.106 and the highest number were 0.300 stars/second. For variant B, the mean number of stars gathered per second were 0.089, the median number were 0.081 and the highest number were 0.400 stars/second. Players of variant A gathered significantly more (30.8%) stars per second than players of variant B, $Z = 1.7172$, $p = 0.0859$. This is the only result that with statistical significance proves that one variant performs better than the other and more specifically that variant A provides better control over the game than variant B.

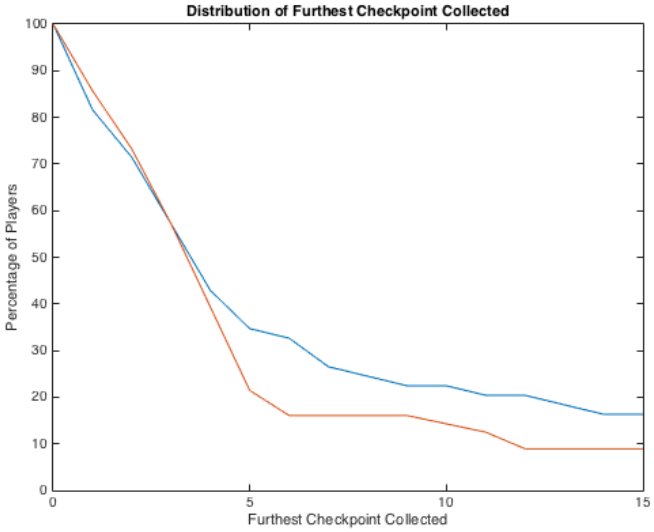

Figure 4.3: This graph shows data for the level progress for variants A and B. Variant A is displayed in blue and variant B in red.

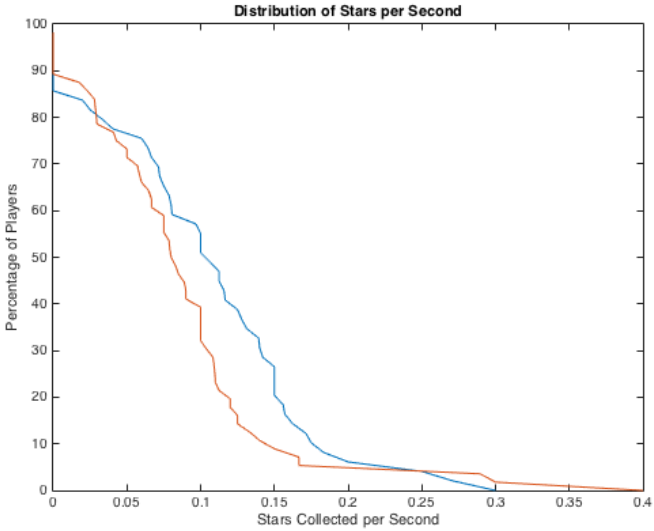

Figure 4.4: This graph shows the combined data of stars divided by time (stars/second) for variants A and B. Variant A is displayed in blue and variant B in red. Y-axis shows the stars per second metric.

Chapter 5

Discussion

This chapter will contain a more thorough analysis of the results from previous chapter as well as the method.

5.1 Test Results

The results from the A/B test conducted shows that one of the measured evaluation criteria (stars gathered per second) was higher for users using variant A with very high statistical significance. The significance level of this measurement reached as low as $p = 0.0859$ which is much lower than the initial goal of $p = 0.2$ despite only attracting 105 players to the test. Since the mean total time played is slightly less than 90 seconds for both variants, one important factor to reach a high value for stars per second is that the motionzone-layout would be intuitive and easy to understand quickly. The results can also be validated by the feedback from some test-participants. The general opinion seemed to be that users who received variant A would rather quickly understand how to maneuver the game and therefore faster move on to focus on gathering stars than users who received variant B. This is of course a rather subjective conclusion but I believe that it could still be used as a complement to explain the results from the test.

Another important thing to remember about the result is that because of the fact that the mean total time played is less than 90 seconds, the result can not be assumed to hold up for long term effects. The result that variant A gives better control than variant B might therefore not be true if both populations kept playing for a much longer time and sufficiently learned both input variants. As discussed by Marli Mesibov [11], split testing will

not tell us *why* users gathered more stars per second with variant A. I can only guess that it is because may be more intuitive, possibly even a more natural way of maneuvering the game than variant B. This guess is based on the fact that variant A has the jump zone at the top and that it could feel more natural to wave upwards to jump.

For this pilot test, a confidence level of 80% were used. As noted in [12, 18, 19] this means that one out of five times, a test will produce enough evidence to disprove the null hypothesis by random chance. Furthermore, as four metrics were used in this test, this means that there is a very high chance that one of them will be statistically significant by chance. Even though the stars per second metric from the test reached a confidence of over 91%, using multiple metrics in conjunction with a confidence level as low as 80% should still be regarded as lowering the validity of the test [12]. For very important decisions, the experiment could be redone with a higher confidence level to test if the result holds up before any action is selected [4].

5.2 Method

Looking back at the method for performing this work, not all of the steps were perfect. This section will discuss improvements to the method and suggestions for further work.

Split testing of games still lacks some solid scientific foundation. Most of the theory used for this thesis is aimed at split testing of general websites and while some steps of the testing process were directly applicable to game testing, other parts required additional effort and research to fit game testing. In the initial literature study process for this thesis, much effort was put into finding well documented split tests of games, however this generated very few results [21, 22]. As a result, the general testing process was well planned going into this project while game specific tasks, such as deciding a suiting OEC for the motion-zone test, had to be estimated with the available knowledge at hand.

As mentioned in 2.3.1, deciding a good OEC for a test is a critical factor that can solely decide if the test will result in any valuable information or not [18]. The OEC decision were probably the hardest step in the testing process of the motion-zones described in this report. In the end, it was settled to measure four OEC's that was thought to be affected by the change of motion-zone layout. In source [12], Ron Kohavi et al. argues that the use of surrogate OEC's can drastically hurt the validity of a test and that focus should be put on the metric that the test actually tried to improve. Only one (stars/second) out of the four measured OEC's were statistically significantly

affected by the layout of the motion-zones in the pilot-test conducted. As stars per second is the metric most closely related to the control factor from the GameFlow [24] theory, I strongly agree with Kohavi about focusing the OEC on the critical metric that the test tries to improve.

Regarding the sources used for this thesis, there exists many contradictions. Online split testing is still an evolving practise and the best practises are still developing. As split testing is also a popular practise, there exists many blog-posts with web-developers giving their opinions of how to conduct a test. Deciding what sources that should be trusted was therefore not a trivial task. Verifying key concepts in several sources has been a valuable practice to tackle this task.

5.2.1 Test-module Development

The construction of the testing-module described in this thesis has not always been a straight forward task. The idea of modifying an existing testing framework to support the features needed for split testing of games has however been a great success. By extending Abba instead of creating a new module from scratch, some basic functionality were already implemented an more focus could be directed at modifying parts to fit game testing. Some parts of the resulting module could however be improved, the main improvements are listed below in this section.

One part of the module regarding how the test-participants were tracked could definitely be improved. In my implementation of Abba, users are tracked in the database by their IP-address which is not an optimal solution as some users may have dynamic IP or some users may have several computers connecting from the same IP-address. The impact of this effect is estimated to be very small but additional measures to properly track the users could increase the reliability of this test.

Another aspect of the method that could have gained more attention is the security aspect. There will always be malicious users on the web that tries to break through the software and find security holes. Abba currently sends test data from the client to the database with common HTTP-get requests and by modifying the payload of these requests a malicious user could write almost anything to the database. This risk was not forgotten in the test and was mitigated with daily backups of the database, however additional counter-measures would be strongly recommended for large-scale testing.

Finally, the data analysis process of extracting data through mongo export and then importing it to Matlab to finally run several calculations before arriving at a result is currently a rather time consuming process. With

more time, this process could have been more automated, possibly with the addition of a web-interface to view the data and automatically perform the calculations.

All of the above mentioned improvements are things I would suggest to consider for further development of the modified Abba testing module.

5.3 Wider Context

The most sensitive aspect of split testing seems to be if the testers using the service should be told that they are part of a test or not. It could be argued that participants could feel monitored and possibly use the game differently if they knew that they are part of a test. On the other hand, there is the question of however it is ethically correct to gather data about people without their consent. However, even if users are affected by knowing that they are part of a test, the upside of split testing is that all variants should be equally affected by this. In turn, that will leave the result unaffected in most cases.

For the test described in this report, it was clearly explained to the test participants that the game was undergoing an alpha-test. This made some participants very worried when they were asked to activate their web camera to play the game as some thought they would be recorded while playing. With the same reasoning as above, this is not believed to have a large impact on the results. However, additional information about that data was gathered anonymously and that no camera data is used could possibly have got a few more people to try the game.

Chapter 6

Conclusions

This chapter will summarize the conclusions from the work described in this report. The chapter will be divided in two sections. The first will discuss the conclusions about how a split test can be applied to test games and present an answer to the question *How can split-testing be implemented to evaluate design properties in the development of motion based games?* The last section will present the conclusions from the motion-zone test.

6.1 Split Testing of Games

The initial goal of implementing a module to test features of the Phaseformer game via split testing has been achieved. By conducting a pilot test targeting the motion-zones of Phaseformer I have validated that the test-framework works and that statistical significance can be reached with only 105 participating users. The simplicity to set up a test with the Abba testing framework has been maintained throughout the modifications. While performing a split test will always require sufficient planning, the actual implementation of a new split test with the modified Abba would be a rather simple task.

The final modified Abba framework provides many features that can be used to split test on-line games. The plug-and-play nature of the module means that it can easily be integrated in a game to perform a test and also quickly removed once the test is over. The functionality to perform several tests in parallel has also been maintained throughout the modifications of the framework. Additionally, the ability to perform a test without splitting the code into several variant-unique codebases means that the module can be

used to perform a test without hindering further development of separate features during the test. All of the above mentioned factors shows that the purpose of the testing module as described in section 1.3 has been achieved. While there are still some parts of the module that can be improved, the conclusion is that my modified version of Abba can be a great tool to perform split tests of on-line games in the future.

This entire report tries to provide an answer to the initial problem formulation of "How can split-testing be implemented to evaluate design properties in the development of motion based games?". My conclusion is that incorporating my modified version of Abba and following the steps described in this report to plan and perform a test provides one answer to that question. Important to remember is that the work performed for this thesis gives *one* answer to that question but there are alternative ways of conducting a split test. This work should therefore not be seen as a general answer to "How to split test games?" as it is important to properly plan and tailor a test to work under given circumstances.

The confidence level selected for the pilot test in this report was selected to be 80%. In practise, that means that there is up to 20% chance that the wrong decision could be taken with support of the test. Despite only attracting 105 test participants for the motion zone test, a confidence of 91.5% were achieved for the stars per second metric. The conclusion is that future tests should not settle for a confidence level as low as 80% since 90% and even 95% are realistic goals.

One finding from this work is that a surprisingly large part of the work needed to perform the test were not directly related to the actual test. Prerequisites as developing the game and properly setting up the OpenShift hosting platform accounts for several weeks of preparing the test. While these tasks may be seen as not directly test-related, it is still very important to properly setup and plan the test to give it the best possible pre-conditions.

Another very important finding is that getting new test-participants required very much effort. I have estimated that about 40 out of the 105 people that participated in the test did so because I asked them to in person. Posting a short description of the game together with a with a link on Facebook and online forums were also a valuable way of spreading the game as it gathered some additional test participants, however, prior to the test I would have guessed that to be much more effective. Suggestions for future tests is to plan more carefully how to find people to test the software.

6.2 The Pilot test

As mentioned very early on in the introduction, the gathering of data from an on-line service is very easy. The main focus of the work to perform a split test lies in sufficiently planning it to ensure what is thought to be tested is the same that is actually measured. As an example, the very first idea was to only measure total amount of time played with the hypothesis of "if the motion-zones are better then the game is more fun and users would play for longer". Looking at the results of the test, the time played did not significantly differ between the two variants tested and this measurement did therefore not conclude any winner of the test. This initial hypothesis were discarded along the way as it became evident that the motion-zones are closely related to how well a player can control the game and a OEC should therefore try to directly measure the control. The GameFlow theory was used to decide that stars gathered per second were most closely related to control and it was also the only metric that provided a result with statistical significance. The conclusion from the test is therefore to use motion-zone variant A as a new baseline for the Phaseformer game.

Bibliography

[1] Joan Fisher Box. Guinness, gosset, fisher, and small samples. *Statist. Sci.*, 2(1):45–52, 02 1987.

[2] Diane Tang, Ashish Agarwal, Deirdre O'Brien, and Mike Meyer. Overlapping experiment infrastructure: More, better, faster experimentation. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 17–26, New York, NY, USA, 2010. ACM.

[3] Ron Kohavi and Matt Round. Front Line Internet Analytics at Amazon.com, 2004.

[4] Ron Kohavi, Alex Deng, Brian Frasca, Toby Walker, Ya Xu, and Nils Pohlmann. Online controlled experiments at large scale. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '13, pages 1168–1176, New York, NY, USA, 2013. ACM.

[5] Marissa Mayer. 41 shades of blue, August 2011.

[6] Eric Ries. *The lean startup : how today's entrepreneurs use continuous innovation to create radically successful businesses.* Crown Business, New York, 2011.

[7] Wikipedia. Wii remote — wikipedia, the free encyclopedia, 2015. [Online; accessed 18-May-2015].

[8] Wikipedia. Kinect — wikipedia, the free encyclopedia, 2015. [Online; accessed 18-May-2015].

[9] Glenford J. Myers and Corey Sandler. *The Art of Software Testing.* John Wiley & Sons, 2004.

[10] Jürgen Münch, Fabian Fagerholm, Patrik Johnson, Janne Pirttilahti, Juha Torkkel, and Janne Jäarvinen. Creating minimum viable products in industry-academia collaborations. In Brian Fitzgerald, Kieran

Conboy, Ken Power, Ricardo Valerdi, Lorraine Morgan, and Klaas-Jan Stol, editors, *LESS*, volume 167 of *Lecture Notes in Business Information Processing*, pages 137–151. Springer, 2013.

[11] Marli Mesibov. The right test at the right time, September 2011.

[12] Thomas Crook, Brian Frasca, Ron Kohavi, and Roger Longbotham. Seven pitfalls to avoid when running controlled experiments on the web. In *In Proceedings of the 15th ACM SIGKDD*, pages 1105–1114, 2009.

[13] M. Beasley. *Practical Web Analytics for User Experience: How Analytics Can Help You Understand Your Users*. Morgan Kaufmann Publishers, 2013.

[14] A. Margara G. Tamburrelli. *Towards Automated A/B testing*. Springer International Publishing, 2014.

[15] *A/B testing: A promising tool for customer value evaluation*. IEEE, 2014.

[16] Mary Poppendieck and Michael A. Cusumano. Lean software development: A tutorial. *IEEE Software*, 29(5):26–32, 2012.

[17] Ranjit K. Roy. *Design of experiments using the Taguchi approach: 16 steps to product and process improvement*. John Wiley & Sons, New York, 2001.

[18] Ron Kohavi, Alex Deng, Brian Frasca, Roger Longbotham, Toby Walker, and Ya Xu. Trustworthy online controlled experiments: five puzzling outcomes explained. In Qiang Yang 0001, Deepak Agarwal, and Jian Pei, editors, *KDD*, pages 786–794. ACM, 2012.

[19] S. Siegel and N.J. Castellan. *Nonparametric statistics for the behavioral sciences*. McGraw–Hill, Inc., second edition, 1988.

[20] Geoffrey Keppel and Thomas D. Wickens. *Design and analysis : a researcher's handbook*. Pearson Prentice Hall, Upper Saddle River (N.J.), 2004.

[21] Erik Andersen, Yun-En Liu, Rich Snider, Roy Szeto, Seth Cooper, and Zoran Popovic. On the harmfulness of secondary game objectives. In Marc Cavazza, Katherine Isbister, and Charles Rich, editors, *FDG*, pages 30–37. ACM, 2011.

[22] Lennart E. Nacke, Michael Kalyn, Calvin Lough, and Regan L. Mandryk. Biofeedback game design: using direct and indirect physiological control to enhance game interaction. In Desney S. Tan, Saleema Amershi, Bo Begole, Wendy A. Kellogg, and Manas Tungare, editors, *CHI*, pages 103–112. ACM, 2011.

[23] Wikipedia. Standard score — wikipedia, the free encyclopedia, 2015. [Online; accessed 5-June-2015].

[24] Penelope Sweetser and Peta Wyeth. Gameflow: A model for evaluating player enjoyment in games. *Comput. Entertain.*, 3(3):3–3, July 2005.

[25] Wikipedia. A/b testing — wikipedia, the free encyclopedia, 2015. [Online; accessed 2-June-2015].

[26] ALEX MACCAW. Abba - javascript a/b testing, 2015. [Online; accessed 10-May-2015].

[27] Stripe, 2015. [Online; accessed 2-June-2015].

[28] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.