



NTNU – Trondheim
Norwegian University of
Science and Technology

Framework for Multi-player GameWall Interaction

Aleksander Aanesl. Elvemo
Vegard Gamnes

Master of Science in Computer Science
Submission date: June 2013
Supervisor: Alf Inge Wang, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Definition

The goal of this project is to design, implement, and evaluate a framework that makes it easy to build multimodal web applications for large displays with multiple users. The framework will use cross platform web technology for creation of simple games or interactive applications. Prototypes will be created in order to evaluate the framework and investigate the potential learning benefits in an educational setting.

Assignment given: 16. January 2013

Supervisor: Alf Inge Wang, IDI NTNU

Abstract

This report describes the experiences and results made by developing a multi-modal web application framework. The framework, called FIGA, is to assist developers in creating web applications where users share a common screen. The applications are platform independent and highly accessible, with an entertainment or educational payload. The report consists of four major parts, namely a prestudy, the FIGA development, the implementation of application prototypes, and an evaluation of these prototypes.

The prestudy introduces the hardware and software aspects of FIGA in terms of expected devices and web development. Developing FIGA is explained both in terms of the software architecture of the framework, in addition to the experiences of creating application prototypes based on FIGA. Each of the developed FIGA applications sheds light on different aspects and requirements demanded by different types of applications, specifically video games and educational applications. The evaluation of the prototypes shows that web applications reaches the look and feel of native applications, as well as documenting the positive responses of effortless connectivity and having no installation procedures. Two experiments gave valuable feedback on the usability and show an increase in perceived learning by using educational applications made with FIGA in a classroom setting.

Preface

This master thesis was written in the time period from January to June 2013 by Aleksander Aanesland Elvemo and Vegard Gannes.

Acknowledgements

First, we would like to thank Alf Inge Wang for this assignment and for his help and guidance throughout this project.

We would also like to thank everyone who participated in the two conducted experiment sessions, and provided us with valuable feedback.

Trondheim, 6. June 2013



Aleksander Aanesland Elvemo



Vegard Gannes

Contents

I	Introduction	1
1	Problem Description	3
1.1	Motivation	3
1.2	Project Definitions	4
1.3	Readers Guide	6
2	Research Questions and Methods	9
2.1	Research Questions	9
2.1.1	RQ1 - Reducing Development Time	10
2.1.2	RQ2 - Supporting Different Application Types	10
2.1.3	RQ3 - Locating Performance Bottlenecks	11
2.1.4	RQ4 - Improving Usability	11
2.1.5	RQ5 - Educational Applications and Learning Benefits	11
2.2	Research Methods	12
2.2.1	The Engineering Approach	13
2.2.2	The Empirical Approach	13
II	Prestudy	15
3	State of the Art	17
3.1	Multimodal Interactions	17
3.2	Crowd Gaming	20
3.3	Impact on FIGA	23
4	Technology	25
4.1	Hardware	25
4.1.1	ServerApp Hardware	26
4.1.2	UserApp Hardware	26

4.1.3	GameWallApp Hardware	26
4.1.4	Development Environment	27
4.2	Software	27
4.2.1	Web Development	28
4.2.2	Distributed Systems	32
4.2.3	Node.js and Socket.IO	34
4.2.4	Browser Support	35
4.3	Application Types	35
4.4	Native Versus Web Applications	36
III	Framework for Interactive GameWall Applications	39
5	Requirements	41
5.1	Functional Requirements	41
5.2	Quality Requirements	43
6	Architecture	45
6.1	Architectural Components	45
6.1.1	ServerApp	47
6.1.2	UserApp	51
6.1.3	GameWallApp	52
6.2	Physical View	54
6.3	Sequence Diagrams	56
6.4	Architectural Limitations	58
6.5	Creating a “Hello World” Application	59
6.5.1	Preliminary Work	59
6.5.2	Developing the ServerApp	60
6.5.3	Developing the UserApp	61
6.5.4	Developing the GameWallApp	62
6.5.5	Starting the Application	63
7	Prototypes	67
7.1	PostIt	67
7.1.1	The UserApp	68
7.1.2	The ServerApp	68
7.1.3	The GameWallApp	69
7.1.4	Lessons Learned	72
7.2	WordCloud	73
7.2.1	The UserApp	73
7.2.2	The ServerApp	74
7.2.3	The GameWallApp	75

7.2.4	Lessons Learned	75
7.3	Categories	76
7.3.1	The UserApp	77
7.3.2	The ServerApp	78
7.3.3	The GameWallApp	79
7.3.4	Lessons Learned	79
7.4	1814	80
7.4.1	The UserApp	81
7.4.2	The ServerApp	83
7.4.3	The GameWallApp	83
7.4.4	Lessons Learned	85
8	User Experiments	87
8.1	Empirical Approach	87
8.1.1	General Information	88
8.1.2	System Usability Scale	88
8.1.3	Technical Considerations	89
8.1.4	Application Comparison	90
8.1.5	Comments Section	92
8.2	The First Experiment	92
8.2.1	Experimental Approach	93
8.2.2	Results	94
8.2.3	User Comments	98
8.2.4	Organizer Experiences	99
8.3	The Second Experiment	100
8.3.1	Experimental Approach	100
8.3.2	Results	105
8.3.3	User Comments	108
8.3.4	Organizer Experiences	108
8.3.5	1814 Results	109
8.3.6	Comments About 1814	113
9	Evaluation	115
9.1	Prototype Summary	115
9.1.1	Development Methodology	115
9.1.2	Application Functionality	116
9.1.3	The Joys of Web Development	116
9.1.4	The Pains of Web Development	117
9.1.5	Scaling the Prototypes	117
9.2	Experiment Summary	117
9.2.1	Experimental Approach	118

9.2.2	Results	118
9.2.3	User Comments	124
9.2.4	Organizer Experiences	125
9.3	Functional and Quality Requirements	125
9.3.1	Functional Requirements	125
9.3.2	Quality Requirements	126
IV	Summary	129
10	Conclusion	131
10.1	RQ1 - Reducing Development Time	132
10.2	RQ2 - Supporting Different Application Types	132
10.3	RQ3 - Locating Performance Bottlenecks	133
10.4	RQ4 - Improving Usability	134
10.5	RQ5 - Educational Applications and Learning Benefits	135
11	Further Work	137
V	Appendix	139
A	Test Environment	141
B	Experiment Evaluation	143
B.1	Technical Considerations Results from the First Experiment	143
B.2	Technical Considerations Results from the Second Experiment . .	144
B.3	Technical Considerations Results from 1814	145
B.4	Gameplay Results from 1814	146
C	Evaluation Survey	147
	References	156

List of Figures

- 3.1 Interaction with MobiToss 18
- 3.2 Flashlight interaction 18
- 3.3 Josh Software’s real-time game 19
- 3.4 Osmus’ game architecture 19
- 3.5 Nintendo Wii U console 20
- 3.6 The audience leans left and right to control the game 21
- 3.7 Multiplayer soccer game 22
- 3.8 MOOSES 22
- 3.9 Lecture Quiz 23

- 4.1 Test environment during application development 27
- 4.2 Basic HTML5 structure 29
- 4.3 A web site with responsive design 30
- 4.4 Paper.js example of bouncing balls 32
- 4.5 One-way communication 33
- 4.6 Two-way communication 33

- 5.1 ISO/IEC 25051 43

- 6.1 Architectural components with their respective framework dependencies 46
- 6.2 ServerApp component 47
- 6.3 UserApp component 51
- 6.4 GameWallApp component 53
- 6.5 A use case scenario of a FIGA application presented with a physical view 55
- 6.6 A second use case scenario of a FIGA application illustrated with a physical view 56
- 6.7 A UserApp connects to the ServerApp 57

6.8	A UserApp sends a message to the ServerApp, which broadcasts it to all GameWallApps	57
6.9	ServerApp using the “Hello World” application	64
6.10	UserApp using the “Hello World” application	64
6.11	GameWallApp using the “Hello World” application	65
7.1	The PostIt UserView on different devices	69
7.2	The PostIt GameWallView filled with post-its	70
7.3	Adding a new PostIt category	70
7.4	The PostIt GameWallView with a highlighted post-it	71
7.5	All post-its sorted based on their category	72
7.6	A word cloud example	73
7.7	The WordCloud UserView	74
7.8	The WordCloud GameWallView	76
7.9	The Categories UserView	77
7.10	The Categories UserView on a tablet	78
7.11	The Categories GameWallView	79
7.12	The original game concept of North & South	81
7.13	The UserView is developed with responsive design techniques	82
7.14	The 1814 configuration screen	84
7.15	The 1814 GameWallView	84
8.1	Brainstorming with PostIt	93
8.2	Brainstorming with WordCloud	94
8.3	Categories in action	95
8.4	Application comparison results	98
8.5	A QR code scaled up on the GameWallView	99
8.6	WordCloud during the second experiment	101
8.7	The WordCloud GameWallView result in the second experiment	102
8.8	The PostIt GameWallView during experiment	102
8.9	The introductory slide for the Categories in the second experiment	103
8.10	Application comparison results from the second experiment	107
8.11	1814 played in a classroom	110
9.1	WordCloud comparison from the two experiments	121
9.2	PostIt comparison from the two experiments	121
9.3	Categories comparison from the two experiments	122
9.4	The second experiment compared to the first experiment with only the participants who participated in both experiments	123
9.5	The second experiment compared to the first experiment with only first-time participants	124

List of Tables

- 5.1 The functional requirements 42
- 5.2 The non-functional requirements 43

- 6.1 FIGAServer class diagram 48
- 6.2 FIGAUser class diagram 52
- 6.3 FIGAGameWall class diagram 53

- 8.1 The SUS questions 89
- 8.2 The technical considerations 90
- 8.3 The application comparison 90
- 8.4 The eight different factors that EGameFlow uses to measure the
users enjoyment of an educational application 91
- 8.5 Eight good characteristics of an educational game 92
- 8.6 SUS score distribution 96
- 8.7 Technical considerations results 97
- 8.8 The gameplay and social interaction questions 104
- 8.9 The SUS score distribution from the second experiment 106
- 8.10 Technical considerations results from the second experiment 107
- 8.11 The technical user experience results 109
- 8.12 The gameplay and social interaction results 112

- 9.1 SUS score comparison 119
- 9.2 Technical considerations comparison 120

- B.1 Technical considerations results 143
- B.2 Technical considerations results from the second experiment 144
- B.3 Technical user experience results from 1814 145
- B.4 Gameplay and social interaction results from 1814 146

Listings

- 4.1 A Node.js example HTTP server 34
- 6.1 ServerApp.js 60
- 6.2 UserApp.html 61
- 6.3 UserApp.js 62
- 6.4 GameWallApp.html 62
- 6.5 GameWallApp.js 63
- 6.6 Start the FIGA application 63

Part I

Introduction

Chapter 1

Problem Description

This chapter introduces the personal motivations for choosing this project, the project definitions, and a readers guide. The motivation section describes the personal interests in this particular field of computer science, and why this project was chosen. The following section introduces the different abbreviations and definitions used throughout the report to avoid ambiguity. The readers guide gives an overview of the remaining parts in the report and will be of particular interest for those with a specific academic field of interest.

1.1 Motivation

Deciding on this particular master thesis was affected by several different personal motivations. The first motivation is preventing the current trend of isolating platforms. Different corporations restrict more and more the interconnectivity between their devices and those of their competitors. This has resulted in technology islands, where devices are isolated from each other with very few ways of communication. In order to communicate, a common standardized format is required. By using web technology to find a common ground between devices will be of benefit for the users in terms of usability and cross platform development. This project will be a part of creating technology that makes even the most competitive devices collaborate.

To amplify interconnectivity this project will turn towards one of the most promising technological trends at the time of writing, namely cloud computing. Shifting the workload to web servers reduces the importance of which device is

used and the location of the device. The only requirement is an Internet connection and a web browser. The evolution of today's web browsers and server technology, combined with clever client-side scripting, masks the previous performance issues with web applications. Investigating the principles of cloud computing was another personal motivation for choosing this project.

Another aspect of this project is how ad-hoc communication of an almost serendipitous nature is possible, with effortless connectivity. The use of web applications means that there is no necessity of installing a native application before interaction. The social aspects of the concept of a shared screen are the most important factor, where technology trends today drifts towards separating persons from each other. Through cooperation and collaboration, technology can bring people together. These aspects of the project give confidence that this project provides a positive contribution to the ever-evolving technological landscape.

1.2 Project Definitions

This report is heavily influenced by a prior research project where technological choices and suggested approaches were made [32]. The problem definition is penned by our supervisor.

Definition 1 (Problem definition). *The goal of this project is to design, implement, and evaluate a framework that makes it easy to build multimodal web applications for large displays with multiple users. The framework will use cross platform web technology for creation of simple games or interactive applications. Prototypes will be created in order to evaluate the framework and investigate the potential learning benefits in an educational setting.*

An abbreviation for the framework is defined, in order to avoid ambiguity.

Definition 2 (FIGA). *FIGA is an acronym for “Framework for Interactive GameWall Applications”. The framework supports the development of multimodal web applications with a large collaborative screen. Note that FIGA will be referred to throughout the report.*

A FIGA application consists of three sub applications, respectively GameWallApp, UserApp, and ServerApp. These sub applications are executed on different devices and communicate with each other using web technology.

Definition 3 (GameWallApp). *The characteristic component of a FIGA application is the GameWallApp. The GameWallApp is a collective term for the web application that is publicly displayed.*

Note that GameWallApp will be referred to throughout the report.

Definition 4 (UserApp). *The UserApp is used to interact with the GameWallApp through communication with the ServerApp. This web application runs in a web browser on a personal device, such as a smartphone or a tablet.*

Note that UserApp will be referred to throughout the report.

Definition 5 (ServerApp). *The ServerApp works as an intermediary between UserApps and GameWallApps. In addition to communication, it performs logic operations and insures that everything is synchronized.*

Note that ServerApp will be referred to throughout the report.

In addition to the naming convention of these three sub applications, two notable graphical user interfaces are defined as follows.

Definition 6 (GameWallView). *The view of the GameWallApp is the graphical user interface displayed publicly. Commonly associated terms are the drawing of graphical primitives, text, animations, and buttons. The GameWallView is of global scope, meaning that everyone in the room will see what is displayed on the GameWallView.*

Note that GameWallView will be referred to throughout the report.

Definition 7 (UserView). *The UserView displays the graphical user interface of the UserApp. While the UserView has the same graphical capacity of the GameWallView, it will in almost every case be less complex than that of the GameWallView because of hardware constraints. The UserView is of local scope, meaning that what is displayed on the UserView is only visible to the user.*

Note that UserView will be referred to throughout the report.

Naming Conventions

The naming convention will be used in this thesis to reference the different parts of a FIGA application without confusion. Deciding on the naming convention was surprisingly difficult in order to choose names that was memorable and unambiguous. Some material presented in this thesis will not use the exact same naming convention, specifically the forms presented in Chapter 8 and Appendix C. This is the result of determining the final naming convention after the forms were produced and used in the experiments.

1.3 Readers Guide

Different readers might have different areas of interests. Here are the different parts and chapters of the report summarized for the benefit of the reader.

Part I - Introduction

The first part defines the goals of the project and is of interest for those wanting an overview of the project. In order to fully understand the context and scope of the project one should read these chapters.

Chapter 1 presents the motivation and explains the projects definitions, as well as providing the readers guide.

Chapter 2 presents the research questions and methods used during the project.

Part II - Prestudy

The second part will be of particular interest for those interested in distributed interactive web applications, in terms of how they are developed and what kind of software and hardware to expect.

Chapter 3 describes existing solutions with similar traits to this project.

Chapter 4 explains the hardware and software aspects of the project, the different application types that can be developed using FIGA, and a discussion of native applications versus web applications.

Part III - Framework for Interactive GameWall Applications

The third part describes the projects requirements, the technical implementations and software architecture, the developed prototypes, the user experiments conducted during the project, and an evaluation of these different parts.

Chapter 5 presents both functional and quality requirements.

Chapter 6 covers the FIGA architecture.

Chapter 7 explains the developed prototypes in detail.

Chapter 8 describes the conducted user experiments, as well as presenting its results.

Chapter 9 evaluates the prototypes, experiments, and requirements.

Part IV - Summary

The fourth and final part contains the summary and conclusion of what have been accomplished during the project, in addition to proposed further work. These chapters are of interest of those wanting an overall view of the results and further recommendations.

Chapter 10 presents the summary and conclusion.

Chapter 11 presents the further work that can be accomplished in the future.

Chapter 2

Research Questions and Methods

This chapter describes the research questions and methods used in the project. These questions and methods helped direct the project in a clear and concise path. The questions were formed by drawing inspiration from previous work, from own curiosity and motivation, and input from our supervisor.

This project entails creating a framework with value. Value is meant by the amount of reduced implementation time, and supplying a robust technological platform. In order to further define and measure what is meant with value, several technical research questions are penned.

2.1 Research Questions

This section presents five major research questions, each with their own respective sub questions. They will be further described in detail, with explanation of why they were included and how they are planned to be answered.

RQ1 How much can web application development time be reduced by supplying a suited framework?

RQ2 What types of applications can be made with FIGA?

RQ3 What is the performance of applications made with FIGA?

RQ4 What level of usability do applications made with FIGA achieve?

RQ5 Are there an increase in perceived learning by using educational applications made with FIGA?

2.1.1 RQ1 - Reducing Development Time

The first research question will give answers regarding how a suited framework can assist developers reduce the implementation time. If the framework reduces implementation time, then it provides value to the developer. RQ1 has the following sub question:

RQ1.1 What functionality should be provided with FIGA?

In addition to time saved, providing necessary functionality is another way of providing value to the developer. This means saving time not reinventing the wheel, and the existing framework functionality becomes more stable and proven in a variety of projects.

2.1.2 RQ2 - Supporting Different Application Types

The second research question regards what kind of application types that can be made with FIGA. Event-driven and real-time applications are two genres regarding application types, but application types can also concern educational applications and video games. RQ2 has the two following sub questions:

RQ2.1 How easy is it to create an event-driven application with FIGA, and what are the benefits of using FIGA?

RQ2.2 How easy is it to create a real-time application with FIGA, and what are the benefits of using FIGA?

Three educational applications and a video game will be developed in order to answer these questions. Several pedagogical applications will be developed based on the premise that FIGA is well suited for collaborative applications. Additionally, our supervisor has experience in development of educational multimodal application concepts. The video game will push FIGA to its limits, both in terms of network and computational load.

2.1.3 RQ3 - Locating Performance Bottlenecks

The third research question concerns the performance of applications made with FIGA. FIGA should be able to support both event-driven and real-time applications. A major concern regarding distributed systems is network latency. Distributed applications often require a very low network latency, especially video games. One of the tasks will be to investigate if FIGA is suited for handling the network traffic without degrading user satisfaction. RQ3 has the following sub questions:

RQ3.1 How many users can applications made with FIGA handle?

RQ3.2 How will network latency affect the user experience?

Answers to these questions will be found by testing the applications in experiments where qualitative methods will be used to document user satisfaction.

2.1.4 RQ4 - Improving Usability

The fourth research question examines the usability aspects of applications developed with FIGA. This research question will give answers on how user friendly these applications can be made and what developers must be aware of during development. RQ4 has the following sub questions:

RQ4.1 Can FIGA applications reach the look and feel of native applications?

RQ4.2 How easy is it possible to make the setup and start using applications developed with FIGA?

Feedback from the users by usability feedback forms will give insight to the user experience and perspective. To find out how easy it is to setup and use FIGA applications, two experiments in two different lectures will be performed and evaluated.

2.1.5 RQ5 - Educational Applications and Learning Benefits

The fifth and final research question will give answers on how the use of FIGA is beneficial in developing educational applications with learning goals. Adding a layer of social interaction brings new ways of interacting with each other. RQ5 has the following sub questions:

RQ5.1 Which FIGA educational application concept returns highest perceived learning benefits?

RQ5.2 Which FIGA educational application concept cause increased social interaction?

RQ5.3 Which FIGA educational application concept aids a teacher in getting feedback from the students?

The different sub questions all relate to the educational benefits of using a FIGA application. The two first sub questions are viewed from the perspective of the student, and the final one from the perspective of the teacher. These questions will be answered based on feedback from experiments held in lectures, using feedback forms to get the opinions of students. Interviews with the teacher after the lectures will answer the final question.

2.2 Research Methods

This master thesis is heavily based upon a prior research project [32]. The prior project was initiated to prepare for the master thesis, by researching existing similar solutions to the one proposed and finding the most suited technology components.

In literature study, four main approaches are discussed by Victor Basili [28]. These four approaches are the scientific, the engineering, the empirical, and the mathematical method. Each of these uses different approaches to research existing material. Of these four methods, the first three will be used in this project.

The scientific method is a well-suited approach when trying to understand the software process, product, people, and environment. The first step is to observe the world in which the technology is to exist in. After observing the world one propose a model or a theory of behaviour, which is measured and analyzed further. A validation of the model or theory hypotheses is the last step. These steps are repeated if possible.

The engineering method is a variation of the scientific approach, which assumes already existing models of the software process, product, people, and environment. In order to improve the product being studied one modifies the model or aspects of the model. Incrementally, one observes existing solutions and proposes better solutions. This is done by building, developing, measuring, and analyzing. This process is then repeated until no more improvements appear possible.

The empirical method is another alternative of the scientific approach. The first step is to propose a new model that does not have to be based upon an existing model. The new model will provide new effects of the process or product, effects that are analyzed and used in order to improve the model further. Statistical and qualitative methods are then developed, and these methods are applied to case studies before measured and analyzed. The last step is to validate the model and if possible repeat the procedure.

The first step in the research phase of this project is to collect previous knowledge and experiences made by others in the field. A literature search will be performed in order to find this data. There are several search engines that allows for finding specific publications with valuable information and inspiration to this project, like Google Scholar [7]. In addition, our supervisor recommends relevant articles with similar traits to this project. The articles abstract will quickly give an indication of the articles relevance, and the collected material will then go through a literature study for further reading. The literature study will give insight into different relevant technologies and to learn the experiences of previous work. The study of other projects will give inspirations of what is possible to achieve with the technology in terms of application concepts, in addition to technical advice.

2.2.1 The Engineering Approach

The engineering method is used to build on existing solutions and learn from their past experiences. The iterative nature of the engineering method is found by rapid prototyping. Rapid prototyping is well suited to develop application prototypes and discover required framework functionality. The prototypes in this project will be developed with a specific use case in mind. Three prototypes will be created to test FIGA applications in a classroom setting. In addition to the three educational prototypes a video game will be developed to strain the system to its limits, both in terms of graphical computations and network load.

2.2.2 The Empirical Approach

The empirical method is used in proposing a new model to be further evaluated. The concept of using educational web applications with a shared screen has unique properties that will be evaluated through user experiences.

Quantitative Methods

Two experiments will be organized to gather valuable data regarding the user experience of the different applications and the underlying framework. The experiments will be held at lectures taught by our supervisor and supplemented the lectures with educational activities. After the different prototypes have been tested, forms will be delivered to the participants. Each form consists of a System Usability Scale (SUS), a technical consideration, and an application comparison where the participant can give feedback on the different prototypes. The output from these experiments determines the success of FIGA applications from a user experience perspective.

Qualitative Methods

To collect the experiences from the view of the organizer of a class augmented with educational FIGA applications, interviews will be planned to be held post-experiment. These interviews will show how useful the organizer finds the applications, in addition to the level of preliminary work that is required to use the applications. The organizer will be asked about the different exercises and content for the educational applications, and how they were planned. In addition to the practical experiences, the organizer will be asked about the perceived classroom response of using these applications. Finally, the organizer will be asked about any recommended changes to the applications, in order to improve FIGA and its applications.

Part II

Prestudy

Chapter 3

State of the Art

There is a wealth of existing concepts and prototypes with similar traits to FIGA. The process of finding multimodal systems with a collaborative large screen is done either through literature search or based on recommendations from our supervisor, as described in Section 2.2. This chapter goes more into detail on systems relevant to this project, before a summary on how they impacted the project.

3.1 Multimodal Interactions

MobiToss: a novel gesture based interface for creating and sharing mobile multimedia art on large public displays is a system that combines standard interaction devices, interaction styles, and multimedia processing techniques into a novel system for creating and sharing mobile multimedia art [45]. Interaction devices are mobile phones, large public displays, and devices capable of displaying a web page. In order to interact with the system, the user either uses an interface on the device or moves the device in gestures. The user takes a photo or captures a video with the phone and then “throws it” onto a large display by using a throwing gesture, as illustrated in Figure 3.1. The image or video will then be transferred onto the large display, where it can be manipulated by tilting the phone in different directions.

The author of *Flashlight interaction: a study on mobile phone interaction techniques with large displays* created a light-based interaction between mobile phones and large screens by using the light from the mobile phones camera flash [46].

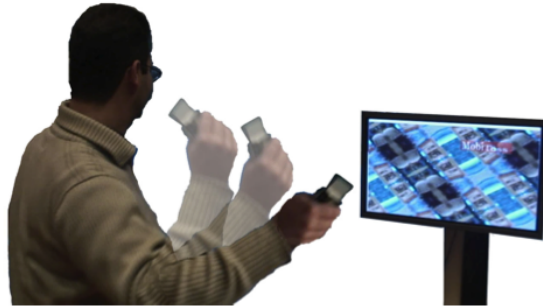


Figure 3.1: *Interaction with MobiToss*

By placing a camera below the display to track the flashlight movements, the system achieves communication with the display without any wireless connectivity from the users. Figure 3.2 shows an illustration of the flashlight interaction system.



Figure 3.2: *Flashlight interaction*

Josh Software posted an article on their web page that addresses how to create a real-time game using HTML5, WebSocket, Node.js, and Socket.IO [18]. They wanted to launch a game on a common display in a web browser, control the game from different browsers, ensure that no installation was required and measure the latency. The common and user display is illustrated in Figure 3.3. The results for latency and concurrency were acceptable. The worst-case scenario with latency was found with Edge, with a latency of 200 ms. The game could easily scale to hundreds of simultaneous users regarding concurrency.

Figure 3.4 shows the game architecture of a game called *osmus* [4]. *osmus* is a browser-based multiplayer game created with the usage of HTML5 and Node.js. It is a clone of another game called *Osmos*, but with multiplayer support [8]. *osmus* is created with distinct, loosely coupled components. The game uses a

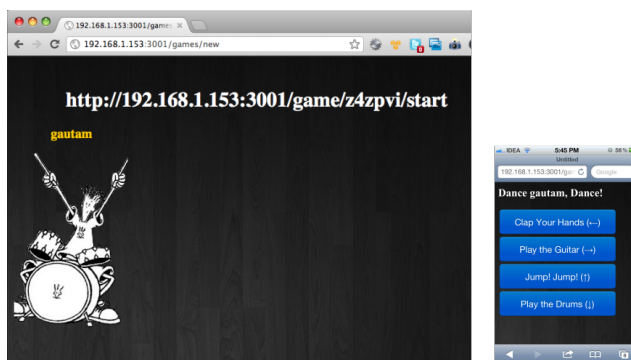


Figure 3.3: Josh Software's real-time game

shared game engine, which is a simple state machine whose primary function is to compute the next game state.

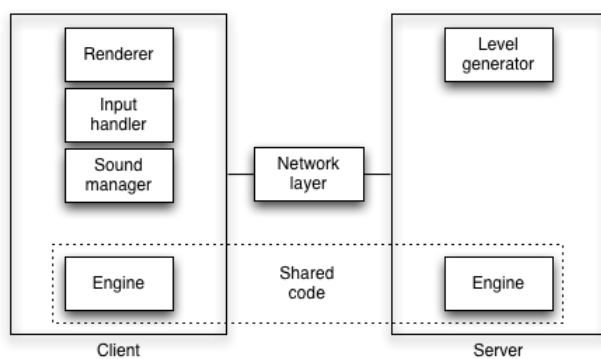


Figure 3.4: Osmus' game architecture

The *Wii U* is a video game console from *Nintendo*, released in November 2012 [14]. Its primary controller, the *Wii U GamePad*, features an embedded touchscreen that can be used to supplement the main gameplay shown on the television, as illustrated in Figure 3.5. While many of the other described systems are prototypes or proposed concepts, the *Wii U* is a major investment by *Nintendo*. Games played with the *Wii U* can be played only on the television, as a combination using both the embedded touchscreen and the television, or only on the embedded touchscreen without the use of an external television.



Figure 3.5: Nintendo Wii U console

3.2 Crowd Gaming

Techniques for Interactive Audience Participation presents a set of techniques that enables members of an audience to participate in shared entertainment experiences, either cooperatively or competitively [31]. The article discusses three different techniques for participation, respectively leaning left and right in their seats, batting a beach ball while its shadow is used as a pointing device, and aiming laser pointers at the screen. One of the benefits of these techniques is that they can all be implemented with inexpensive, off the shelf hardware. Figure 3.6 shows the interaction techniques where the audience leans left and right in their seats for controlling the game.

The inspiration behind *Techniques for Interactive Audience Participation* was the *Cinematrix Interactive Entertainment System* [40]. The *Cinematrix Interactive Entertainment System* is an audience response system. Each audience member is provided with a reflective device. This reflective device has a different color on

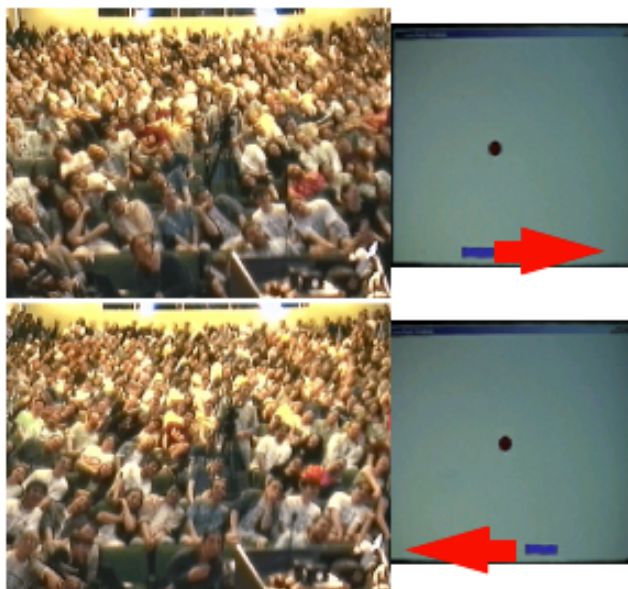


Figure 3.6: *The audience leans left and right to control the game*

each side, green on one side and red on the other. The system emits light from a spotlight that is reflected from the reflectors and recorded by a conventional video camera. A computer receives the processed image, forms a map of the audience and distinguishes the red from the green reflections. This gives the audience members an opportunity to participate interactively in different kinds of activities by displaying the red or the green side of their paddles.

Multi-player soccer and wireless embedded systems describes experiences from developing a multiplayer video game on a large screen [29]. The system had wireless communication between the embedded devices, and they had to devise a communication protocol to handle the communication between players and the game coordinator. The game coordinator polls each player for its move and waits a given time interval for a response before proceeding to the next. A screenshot from the developed game is illustrated in Figure 3.7.

MOOSES is developed at NTNU in collaboration with the company *TellU*, and it is an acronym for “Multiplayer On One Screen Entertainment System” [24]. *MOOSES* is an entertainment system with focus on social interaction, and lets several players play against each other by sharing a big screen in the same room. This is illustrated in Figure 3.8. The user has to download a specific application



Figure 3.7: *Multiplayer soccer game*

on a mobile phone to be able to interact with the system, and has to connect to the system through wireless technologies such as either Bluetooth or Wi-Fi. The mobile phone can then be used as a controller that gives the user feedback through vibration, sound, and information on the mobile phone display.



Figure 3.8: *MOOSES*

Lecture Quiz is a multiplayer game concept that utilizes the infrastructure and equipment available in auditoriums at universities, such as the teachers PC, a projector, and the students' PCs or mobile phones [26]. The game boasts two different game modes that gives the students a series of questions they have to

answer to the best of their abilities on their own PC or mobile phone. The results of the quiz are displayed using the projector and is visible for everyone in the auditorium, as illustrated in Figure 3.9. Lecture Quiz received good reviews from the students that tried the game, claiming it to increase learning and concentration during the lecture. The game is also claimed to be easy to integrate with traditional lecture methods, as lecturers already use a PC combined with a projector to show presentations.

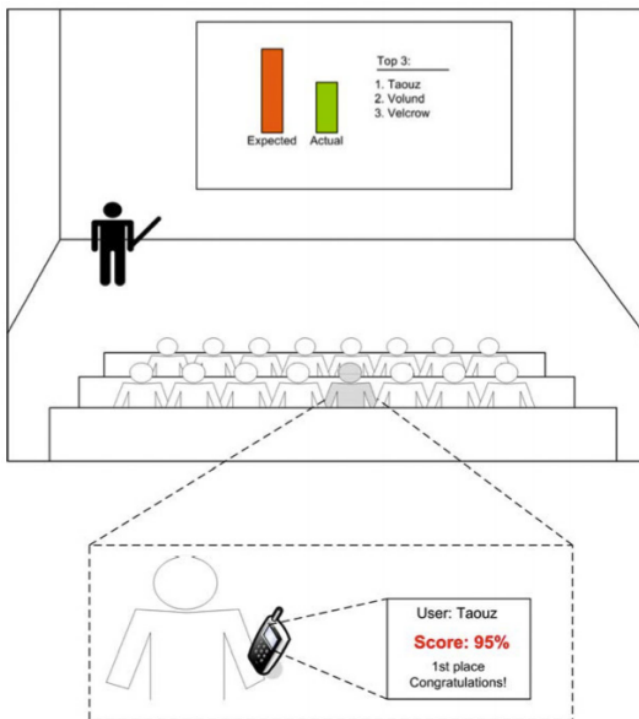


Figure 3.9: *Lecture Quiz*

3.3 Impact on FIGA

FIGA has several requirements regarding platform independency, usability, and connectivity. Josh Software's real-time game, MOOSES, and Lecture Quiz were the most relevant and insightful among the reviewed systems. The three systems use mobile phones as controllers to interact with a common display. MOOSES

and Lecture Quiz require that the user has installed specific software on the user device, while Josh Software ensures that no installation is required. FIGA makes the connection phase as easy as possible, without complicated installations and configurations. Since FIGA is to be used to create different kinds of applications, for instance video games and more educational applications, the project gained inspiration from Lecture Quiz with regards to educational applications that can improve learning.

In addition to these three systems, the Wii U shows that even the larger gaming companies see the potential of the concept of sharing a screen while having a private screen. Most of the existing systems used other types of network technologies than Wi-Fi for communication between the different devices. The majority used Bluetooth, while some tried new approaches like light-based interaction. FIGA will use Wi-Fi, 3G, and Edge, or a network technology that allows HTTP transfer through a web browser.

FIGA will draw inspiration from osmus' game architecture, with distinct, loosely coupled components and its overall architectural structure. This will particularly be useful regarding real-time applications, using software architecture patterns like model-view-controller and client-server. Using several software architecture patterns will help FIGA become a more reliable framework.

Chapter 4

Technology

This chapter contains important information regarding hardware and software relevant to FIGA. *First*, a summary of the most relevant hardware components expected to be used with applications developed with FIGA. *Second*, a web development explanation and the software components utilized during the project. *Third*, a section that describes the application areas that will benefit from using FIGA when creating interactive applications. *Fourth* and final, is a discussion on the subject of native applications versus web applications.

4.1 Hardware

The three sub applications of a FIGA application have different hardware demands. If it is a ServerApp instance, then the device is either a laptop or desktop computer. If it is a UserApp instance, then it is a laptop, tablet, or smartphone. If it is a GameWallApp instance, then it is either run on the ServerApp instance running device or a separate device with good graphical and audio capabilities such as a laptop or PC. While these are the most common occurrences, other combinations of supported hardware are possible.

In educational settings, the required hardware is often readily available. The experiments described later in the report was conducted in the auditoriums at the Norwegian University of Science and Technology (NTNU). In these auditoriums there are projectors, sound speakers, and Wi-Fi access. This makes these auditoriums excellent experiment environments for technologically enhanced lectures. The students that participated in the two experiments were mainly computer

technology students, with everyone possessing smartphones or other hardware capable of running the developed web applications.

4.1.1 ServerApp Hardware

The ServerApp instance is in most cases executed on a laptop or a PC. In the case of educational application it is typically the teacher or professor, or a person in charge, that brings along this device. The device is connected to a local network either through a wire or wirelessly. If the same device is used to handle the GameWallApp, then it should be connected to a larger screen such as a projector or a large TV. The more powerful the hardware is on the device that executes the ServerApp instance the better, as better hardware leads to increased performance.

4.1.2 UserApp Hardware

While any device with a web browser can potentially be a UserApp instance, the focus is on tablets and smartphones. Particularly smartphones, since they are becoming more and more common and always accompany its owner. The limitations of less powerful hardware and a smaller screen means reducing the graphical complexity and the number of elements that is shown on the screen at the same time. Owners of smartphones and tablets have accustomed to using touch gestures to interact with the device. The developed applications should take advantage of this interaction type in order to provide an experience similar to that of native applications.

4.1.3 GameWallApp Hardware

The GameWallApp instance displays the large common screen in a web browser, plays the relevant audio through speakers, and is the main source of attention to everyone in the room. Each user will look and listen towards the GameWallView and interact with it using their device. The device running the GameWallApp instance needs to have the required graphical and audio capabilities in order to provide a smooth audio and visual experience. The GameWallApp running device can be the same device as the one executing the ServerApp, but the accumulated workload of running both these instances should be taken into account.

4.1.4 Development Environment

The development environment was limited and small in scale during the project. The available devices were a 13" MacBook Pro, a 15" MacBook Pro, an iPhone 4S, an iPad 2, and a Samsung Galaxy SII, connected as shown in Figure 4.1. These devices used either wired or Wi-Fi connectivity to communicate with each other during development, using in most cases their native web browser, respectively Safari and Chrome. A wired connection is symbolized using a filled line, while wireless connectivity is symbolized with a stippled line.

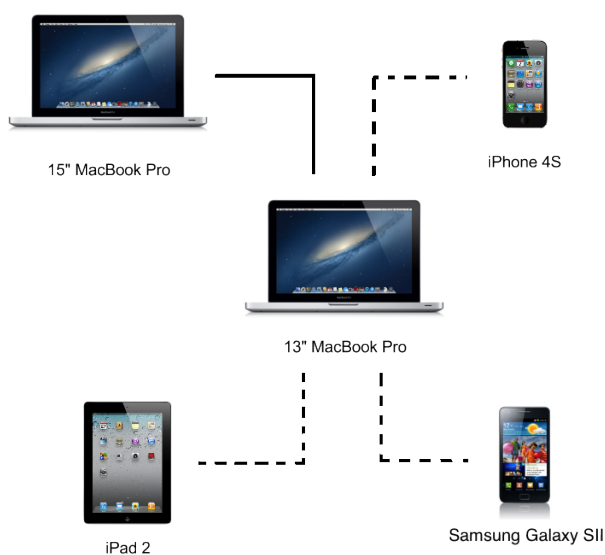


Figure 4.1: *Test environment during application development*

Despite small scale and limitations, the test environment was a good indication of a real case scenario and setting. The specification for these devices can be found in Appendix A.

4.2 Software

The primary success factors of creating a well-functioning framework is understanding its practical use, environment, and interactions with other existing software components. Basing FIGA on web technology has several benefits, such as

avoiding specific software constraints that many technological platforms require. Examples of such constraints are creating applications for the iOS or Android platform which require specific tools and programming languages.

4.2.1 Web Development

Web development is a broad term used for the work involved in developing a web site. Web sites are primarily created for publication on the Internet, but they can also be deployed in a private network. A web application is a web page with layout and functionality resembling that of native applications running in the operating system of the device. The complexity, scale, and scope of a web application can vary from a static page to dynamic and interactive web pages. It is therefore important to make FIGA dynamic and flexible, and not hinder ambitious applications. The following sections describe the different tools of the trade in the field of web development that is relevant for developing FIGA.

HTML5



HTML5 is the fifth revision and latest version of the HTML and XHTML standard, and it is an abbreviation for *Hyper Text Markup Language* [9]. A markup language is used for structuring and representing content, in this case the World Wide Web. HTML5 is also a collection of features, technologies, and APIs. It improves the HTML-standard by including native support of multimedia-standards, partly by adding new syntactical features. Some of the new features are `<video>`, `<audio>`, `<canvas>`, `<section>`, `<article>`, and `<nav>`. These elements make it easy to handle multi-

media and graphical content on the web. A basic HTML5 structure is illustrated in Figure 4.2.

`<video>` and `<audio>` lets the users play video and audio in a web browser, and `<canvas>` is used to draw computer graphics. These improvements, combined with CSS and JavaScript, makes it possible to create large and complex web applications. By adding new syntactical features, HTML5 improves the semantic markup. Semantic markup means that the HTML is written in a way that gives the site content specific meaning. Before HTML5, the semantic markup just told the different browsers where the site content was. Now, it also gives some detailed information about it.



Figure 4.2: *Basic HTML5 structure*

To represent and interact with objects in HTML documents the developer can take advantage of the Document Object Model (DOM) [5]. DOM is a cross-platform and language-neutral interface for addressing and manipulating these objects by using different methods. It can be used to dynamically access and update the content, structure, and style of documents.

The application cache is another major HTML5 improvement that makes it well suited for mobile web applications. Application cache lets the website store content, such as images and videos, locally on the device. This makes the site run faster and smoother if the web page is highly dependent on larger data sets. The combination of the application cache with other HTML5-related technologies, allows a web application to run on the device without Internet connection. The web application can cache pages in advance based on likeliness of the user entering that page, which results in faster load times. If the application is a video game it can cache and store assets that are needed in advance, for example a later level in the game.

HTML5 is designed to be more secure and reliable from both developers and users point of view. Web applications can not read or write files to the user's hard drive. Nor can they read or write data from other web applications or domains. This prohibit that malware is installed on the device.

Cascading Style Sheets

Cascading Style Sheets (CSS) is the most common supplement to HTML [2]. CSS makes it possible to style web pages through simple script-like syntax. This makes customization of the visual representation of the web page easier and more flexible. The core functionality of CSS is supported by all web browsers, with some exceptions regarding newer, experimental features.

Responsive web design is the philosophy of adjusting the user interface based on different viewing devices. It is a web design approach aimed at implementing web sites to provide an optimal viewing experience across a wide range of devices. The combination of HTML and CSS makes it possible to define different styles based on device, display size, and screen resolution. Content that is less important when the display size or screen resolution is reduced can be hidden or shown differently. Figure 4.3 illustrates the same website with responsive design, displayed on different devices.



Figure 4.3: A web site with responsive design

JavaScript



JavaScript is an interpreted computer programming language integrated in web browsers [33]. JavaScript is primarily used as a client-side scripting language embedded in web browsers for helping developers with tools necessary to interact with the user, control the browser, alter the document content, create web page animation, or handle user interaction with more control. Examples of JavaScript usage are web pages where lists unfold smoothly, the location of the cursor alter the layout of the web page, or interactive content based on user input.

A major benefit with JavaScript is the large selection of libraries. These libraries typically contain utility functions or functionality tailored towards a specific problem domain. A popular JavaScript library in use today is the jQuery library [12, 23]. jQuery is a fast JavaScript library that simplifies and ease the code traversal of HTML documents, handle events, and performs animations [10]. It is designed to simplify the client-side scripting of HTML and is frequently used for rapid web development.

There exists a various amount of JavaScript rendering libraries for drawing computer graphics. Rendering is the process of drawing computer graphics on the screen through the use of various algorithms. While it is possible to use HTML5 to handle all computer graphics rendering, the JavaScript graphics libraries will help with using best practices. Paper.js and D3.js are two examples of JavaScript rendering libraries.

Paper.js is an open source, vector graphics framework that runs on top of the HTML5 canvas [17]. The library simplifies the rendering process of simple graphical primitives such as lines, squares, and images. Figure 4.4 illustrates an example with use of Paper.js.

D3.js is a data-driven visualization JavaScript library [3]. It allows the developer to bind arbitrary data to a DOM, and then apply data-driven transformations to the document. An example could be to generate an HTML table from an array of numbers, or to draw complicated graphs and charts.

Some JavaScript libraries make it easier to integrate JavaScript with other web development technologies. Several libraries also include code to detect differences between runtime environments to provide tailor made services for different platforms.

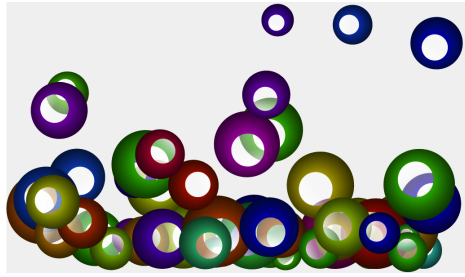


Figure 4.4: *Paper.js* example of bouncing balls

4.2.2 Distributed Systems

Applications created with FIGA are distributed systems. A distributed system consists of multiple devices that communicate by sending messages through a network [30]. This network can either be a local network, or the devices can be geographically separated. Developing distributed systems is a complicated affair with synchronization issues and error prone messaging. These underlying problems quickly impact the stability of the application. If the latency, the time it takes for a message to travel from one device to another, becomes too high then the user can perceive the application as unresponsive. This hurts the overall usability and user experience. While physical limitations cannot be avoided, the user interface can be designed to give the user a clear indication of system status. Applications, such as video games, have strict requirements regarding low latency in order for the system to be perceived as responsive.

Traditional HTTP connections work in the following manner. A user requests a web page through a web browser, by typing the address in the URL field. This sends an HTTP request to a web server that listens for incoming connections, which replies to the client with the respective web page data. This type of communication is called one-way communication. The user requests a service from the server, and the server responds with the desired response. Figure 4.5 illustrates a one-way communication.

There is no concurrent connection over time between the client and the server in one-way communication. Additionally, the server cannot transfer data to a client without a prior request from the client. This means that the client needs to request updates actively from the server in intervals if it is to keep up-to-date with a server that contains fast changing information. This process is called polling, or long polling, and can be implemented by the use of jQuery [41]. There are two main problems with this technique. First, the server cannot send messages

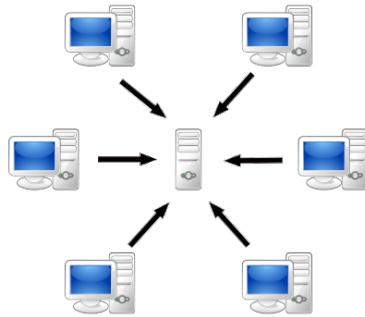


Figure 4.5: *One-way communication*

to the clients since it does not have a reference to them. Second, this technique scales poorly. When the number of connections increases, the server will receive more messages than it can process. This creates delays in responses or in worst case dropped messages.

If there is a need for a two-way communication, traditional polling is not enough. In a two-way communication scheme both server and client can initiate communication. Two-way communication is needed in most applications where the server needs to be able to actively send information to the clients. FIGA must support this communication scheme, and it must work through a web browser. A two-way communication system can be seen in Figure 4.6, where both participants can send and receive data.

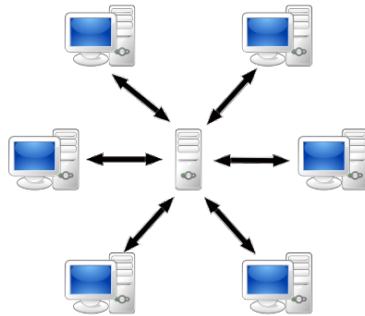


Figure 4.6: *Two-way communication*

Both Node.js and Socket.IO support the two-way communication scheme using TCP for guaranteed delivery. In order to support this communication scheme and supporting every modern web browser, Node.js and Socket.IO was chosen as

the communication modules.

4.2.3 Node.js and Socket.IO

Node.js is a platform for building scalable network applications [15]. It imports a JavaScript file when executed which setups the Node.js server according to the contents of the file. Web applications are well supported by Node.js, where it is commonly used as a web hosting service. Only a few lines of code is required to initialize a web server, as can be seen in Listing 4.1.

```
1 var http = require("http");
2 http.createServer(function (request, response) {
3     request.on("end", function () {
4         response.writeHead(200, {
5             'Content-Type': 'text/plain'
6         });
7         response.end('Hello HTTP!');
8     });
9 }).listen(8080);
```

Listing 4.1: *A Node.js example HTTP server*

Socket.IO gives developers great compatibility and networking control, by a framework which maintains persistent connections [19]. It makes it possible to connect to a Node.js server through a web browser by using JavaScript. Critical features such as setting up the connection, timeouts, and closing down the connection is handled by Socket.IO. It follows the event standard from Node.js, making it easy to integrate them with each other.

The library Socket.IO is used in order to create two-way communication between a Node.js server and a web application on a client device. A socket on both the server-side and the client-side handles the low-level communication. A socket is a controller that parses and sends messages through a network using standardized protocols. Node.js uses TCP for its networking protocol.

Node.js is built around events. An event is a descriptive character string that is connected to a function. If a defined event is received, then the corresponding function is executed. This makes Node.js event-driven in nature, and the events can have a payload. The payload could be, for instance, state information sent from the server to a client.

Another characteristic property of Node.js is channels. Channels are used for categorizing connections to the server application. This makes it easier to send messages to a specific group of connections, all in one function call. This makes

the source code cleaner. The use of channels will be used when differentiating UserApps from GameWallApps.

Several current web services uses Node.js such as Cloud9, nodejitsu, Transloadit, eBay, and LinkedIn [15]. There exists a wealth of APIs developed for Node.js that makes it easy to integrate it with other platforms such as Google services and PayPal.

Socket.IO claims wide support with almost every browser with backward compatibility with older versions. In time of writing the browser support is Internet Explorer 5.5+, Safari 3+, Google Chrome 4+, Firefox 3+, and Opera 10.61+ on the desktop, as well as Safari on iPhone and iPad, Android WebKit and WebOs WebKit on the mobile platform [19]. Socket.IO has good performance, minimal overhead, lean code base and scales well.

4.2.4 Browser Support

One of the most frustrating aspects of web development is the lack of standards within web browsers concerning code interpretation. Each web browser has an independent implementation of HTML5, CSS, and JavaScript, meaning that several design considerations must be made. Styling and design is often the main concern, where the look and feel of the web site differs from browser to browser. Other differences becomes evident with computer graphics, where some browsers perform much better than others in terms of how fast they can draw graphical primitives.

These differences mean extra implementation work with corresponding testing. Some functionality might be discarded by taking the middle road, where some functionality support might be missing in some browsers. These differences have been a key factor in deciding the best suited software with platform independency being top priority.

4.3 Application Types

FIGA enables developers to create interactive applications with various underlying mechanisms and target audience. Two of the most central application types that benefit from FIGA are entertainment and educational software.

Entertainment software is primarily associated with video games. A video game is an electronic game with visual feedback on a device along with interaction involvement through a user interface. While video games started with development

teams of small sizes, today the video game industry is a billion dollar business and it is common to find development teams with over a hundred people in the largest game projects.

Examples of popular video game platforms are Sony's PlayStation consoles and Microsoft's Xbox consoles [20, 13]. A video game console is a dedicated hardware device optimized for running game software. Multiple people can play together with their own controller gathered in front of a TV. The similarities between this and the capabilities of FIGA gave valuable input on game expectations. In this case the GameWallApp represents the television, the UserApp represents the controller, and the ServerApp represents the game console. By implementing and developing different views for the UserApp and GameWallApp, and let the ServerApp compute the game logic, one can converge with the same user experience as playing a video game on a dedicated video game console.

Computer software with learning aspects is categorized under the term educational software. Its primary purpose is teaching and self-learning, often using elements from video game such as gratification to reward the player. Gratification is used to show the player that his or her actions are correct by awarding the player points, gold stars, or other measurements of achievement. Educational software can be used as a classroom aid, which is used for helping the students get a better and clearer understanding of the subject and learn more than without the aid.

Recent experiences show that video games and other educational software can be effective and compelling context for learning [37, 43]. This has resulted in video games becoming more and more used within schools over the last couple of years to teach children and students various subjects. Using video games within a classroom can be beneficial for academic achievement, motivation, and classroom dynamic [44].

4.4 Native Versus Web Applications

The idea of creating fully-fledged applications using web technology was a radical notion only a few years ago. The emergence of successful web applications such as Facebook and Twitter, in addition to the cloud computing web technology, the pure web application is now more feasible than ever. One of the major setbacks of developing web applications has been available resources and performance. A advantage with native applications is the ability to access lower level hardware acceleration. Native applications use optimized programming languages that gives better performance, better and larger memory allocation, optimized

input handling, access to integrated hardware such as camera and gyroscopes, and GPU-access. This performance gap is slowly closing with web browser developers catching up with the performance of native application by optimizing their virtual environments for code interpretations and interfaces with device GPUs.

An innate property of web applications is making device location irrelevant. By making it possible to access the same data regardless of device makes it possible to use different devices and be at different locations. Cloud computing makes it possible for devices with reduced computational power to tap into the capabilities of a server farm. Native mobile applications may also utilize this advantage, which *SoundHound* is a good example of. SoundHound uses the power of an external server farm to match a recorded sound against a database to determine which song is currently playing [21]. Both the storage and computational limitations of a mobile device is thereby solved by the use of external power.

Web applications have the benefit of not requiring conventional installation procedures. Where native applications must be downloaded and installed, web applications are executed in the web browser. It still needs to download data, but this is handled automatically by the browser, hidden from the user. This means that the developers need to be aware of what files are to be transferred to a connecting client and the file size. Heavy use of large image files makes the download process long and increases the danger of overloading the server if too many clients connect. The client will always access the newest version of the web application each time the user accesses the web page, instead of requiring the user to manually update the application. Fortunately, there exist several solutions of reducing the web server load. One example is to move the computations from the server to the client, using different scripting languages. However, the developer has to keep in mind that client-side scripting can present security risks and the need for additional scripting libraries.

It is possible to create a hybrid application, using a native application shell around an integrated web view. This makes it possible to access hardware otherwise not accessible. Pure native applications can communicate with a Node.js server. The usage of Node.js and Socket.IO makes it possible to create native applications that can interact with the web server along side with traditional web applications. However, this requires that the developer holds extra knowledge and other programming language experience.

Considering the pros and cons mentioned above, it is easy to understand the motivation for developing a framework that tries to use all of these advantages to their full extent. The applications developed and documented in Chapter 7 describes the experiences of developing web applications that tries to reach the

proress of native applications.

Part III

Framework for Interactive GameWall Applications

Chapter 5

Requirements

Requirements are used in order to evaluate the success of a project. This chapter presents the functional and quality requirements for FIGA. These will be used for measuring how valuable FIGA is to developers.

5.1 Functional Requirements

The functional requirements (FR) of a system is a set of instructions reflecting the functionality which must be implemented. The functional requirements listed in Table 5.1 is prioritized in the following manner; “Requirement” (X, Y). X is a placeholder for the importance of the requirement and is classified as either low, medium, or high. In addition to importance, it is common to define the expected implementation difficulty of the requirements. This is done similarly to importance, using low, medium, and high. Y represents the expected implementation difficulty of the requirement.

In order to provide the reader with clear definitions, this section describes the functional requirements in a closer detail.

FR1 Supporting shared screen applications

- The most important functional requirement is that the intrinsic value of FIGA is as high as possible. In order for the project to be considered a success, FIGA must allow for shared multimodal web applications.

<i>Description</i>	
FR1	FIGA must support development of applications with multiple users sharing a large common screen (H, H)
FR2	FIGA UserApps and GameWallApps must be executed in a web browser (H, M)
FR3	FIGA must be modular and enable integration of external components (M, L)
FR4	FIGA must simplify the low-level networking process for the developer (M, M)
FR5	FIGA must support the development of event-driven and real-time applications (M, M)

Table 5.1: *The functional requirements*

This means that a FIGA application must support one ServerApp, one GameWallApp, and at least one UserApp.

FR2 Web browser support

- The UserApp and GameWallApp are web applications that must be supported in web browsers. There must be no additional required installations of native applications in order to execute the client-side web application.

FR3 Modularity

- The framework components of FIGA must be able to integrate with other web technology. Third-party libraries that are supported in a web browser must be importable in a FIGA application.

FR4 Network simplifications

- The networking process of web applications are controllable on a low-level scale. FIGA must supply a layer of abstraction on top of the low-level, in order to simplify the messaging system.

FR5 Application types

- Two major types of applications must be implementable using FIGA, respectively event-driven and real-time applications.

<i>Description</i>	
NFR1	FIGA should provide persistent connections to at least 85% UserApps and GameWallApps (M)
NFR2	FIGA should provide a password protection system that prevents at least 95% of intrusion attempts (M)
NFR3	At least 50% of users should experience the look and feel of a native application with a FIGA application (L)
NFR4	At least 80% of users should experience FIGA applications as responsive (H)
NFR5	A FIGA ServerApp should be executable on Windows, OSX, and Linux (H)

Table 5.2: *The non-functional requirements*

5.2 Quality Requirements

Quality requirements are also known as non-functional requirements (NFR). This project used the ISO/IEC 25051 as a reference to determine the quality requirements. ISO/IEC 25051 is an international standard for the evaluation of software quality [38]. Figure 5.1 shows an illustration of the ISO/IEC 25051 standard.

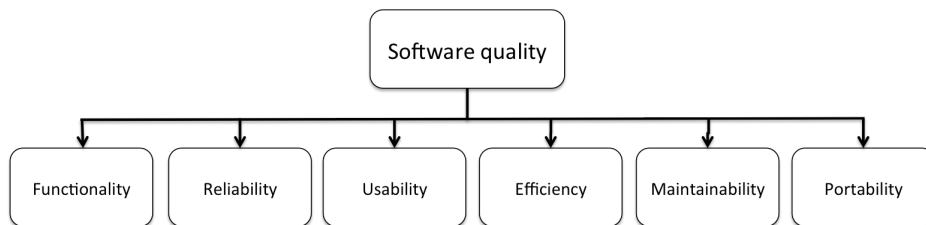


Figure 5.1: *ISO/IEC 25051*

Quality requirements describe system properties which exerts quality. Quality means delivering an expected behavior to the target group. The quality requirements listed in Table 5.2 is prioritized by the following manner; “Requirement” (X). X is a placeholder for the importance of the requirement for the system and is classified as either low, medium, or high.

In order to provide the reader with clear definitions, this section describes the non-functional requirements in a closer detail.

NFR1 Persistent connections

- Connections between a ServerApp and multiple UserApps and GameWallApps are fragile. In order to be experienced as a stable and quality service, the connectivity between network modules must not be broken by web pages being refreshed or messages delayed.

NFR2 Password protection

- In order to protect administrative functionality from users with malicious intent, this functionality should be password protected. FIGA should provide a simple password protection system to deter the majority of unauthorized accesses.

NFR3 Look and feel

- The FIGA applications should be experienced as close to the look and feel of native applications.

NFR4 Responsiveness

- Responsive applications exert quality and insures the user that the application is reacting to the commands it receives as quickly as possible. If applications are perceived as unresponsive, the user experience will be degraded.

NFR5 Platform support

- The ServerApp should be executable on as many platforms as possible. The Windows platform (XP, Vista, 7), OSX (Leopard, Snow Leopard, Lion), and the most popular Linux distros (Ubuntu and Debian) should be supported as a minimum.

Chapter 6

Architecture

The development cycle of the architecture was a continuous process. Finding needed features was done through the implementation of several prototypes in search of general functionality needed across different applications. Features and functionality were expanded upon discovery. The architecture provides an application foundation to build upon, which is easy to use, intuitive, and architecturally sound. The performance of the application needs to take into account if the underlying framework limits the applications potential. To ensure a wide support of different applications, from real-time to event-driven, FIGA was developed with a major focus on flexibility and modifiability.

This chapter is written with a technical point of view. Some readers might find the chapter too detailed and beyond their scope of interest. The chapter concludes with an example of how to create a basic “Hello World” application, to sum up how easy it is to develop with FIGA and its features.

6.1 Architectural Components

A FIGA application consists of three sub applications, namely ServerApp, GameWallApp, and UserApp. Each of these requires different functionality from a framework. This led to the separation of FIGA into three framework modules, respectively FIGAServer, FIGAUser, and FIGAGameWall.

A typical FIGA application is described in the following manner. A ServerApp holds the current state of the application and calculates logic. Different devices

with different views, in most cases GameWallApps and UserApps, are connected to the ServerApp. Their task is to display the current state of the application, or to manipulate it as desired.

A major architectural difference is found between FIGAServer and the two other framework modules. FIGAServer is focused on the server-side role of the application, while FIGAUser and FIGAGameWall are focused on the client-side. Figure 6.1 shows how the different sub applications are dependent on different framework modules.

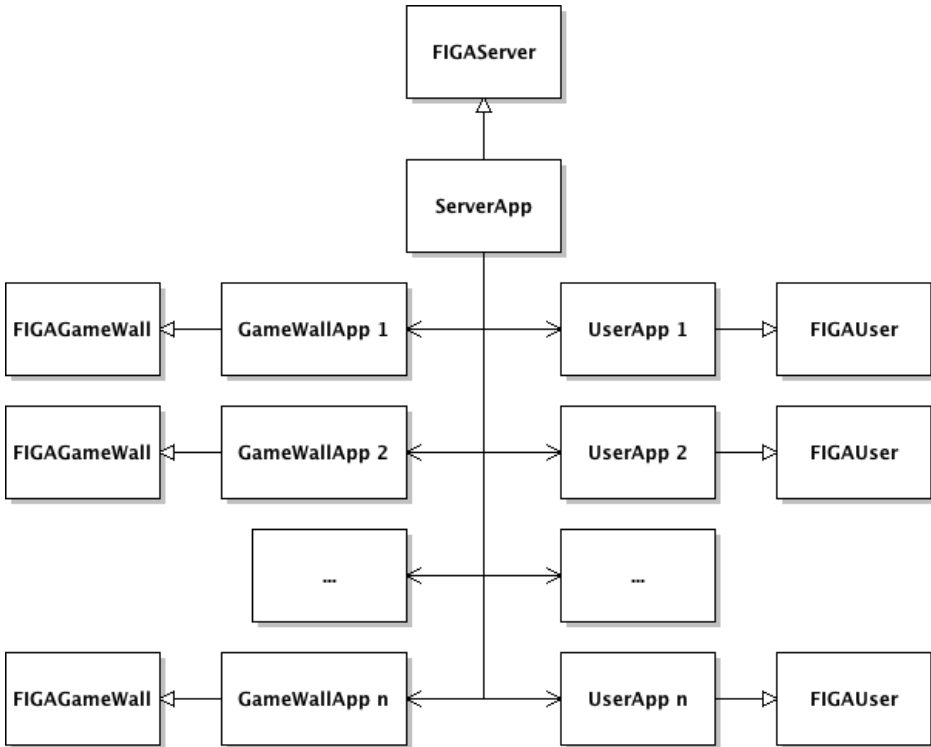


Figure 6.1: Architectural components with their respective framework dependencies

These three framework modules are written in the JavaScript programming language and will be either included as utility functions, as for both FIGAUser and FIGAGameWall, or be used in an inheritance model, as for the FIGAServer. Utility functions are functions created using the functional programming paradigm [47]. They treat the functions in a mathematical sense where identical input gives the same output regardless of state. It is easy to include the framework modules

and integrate them with different existing libraries.

6.1.1 ServerApp

ServerApp focuses on the server-side of the application. Figure 6.2 presents the ServerApp component, and its relations with FIGAServer. An application built upon FIGA creates application specific logic that interacts with FIGAServer in order to perform communication with UserApps and GameWallApps.

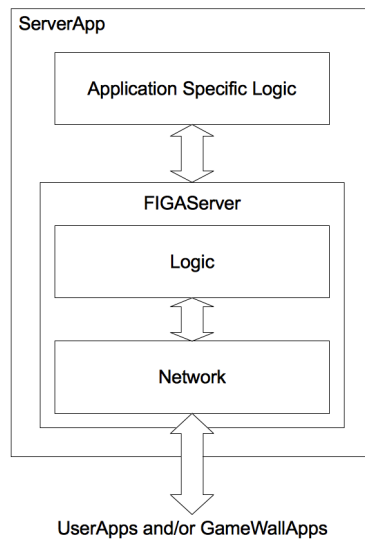


Figure 6.2: *ServerApp component*

FIGAServer

Applications using FIGA implements an underlying server-client network model, where FIGAServer provides the server-side functionality required. This framework module contains the necessary components, such as Node.js, for server-side networking and client session control. FIGAServer was implemented using JavaScript and an overview of the Unified Model Language (UML) class diagram can be seen in Table 6.1. UML 2.0 is a standardized modeling language for creating diagrams in the field of software engineering [35].

FIGAServer
port : integer socketBindings : array storeOldConnections : bool connections : object
init() : void setPort(newPort) : void addBinding(event, func) : void addChannelBinding(event, channel, func) : void bindSocket(socket, channel) : void sendDataToChannel(event, data, channel) : void sendEventToSocket(socket, event) : void sendDataToSocket(socket, event, data) : void addOldConnection(socket) : void reSyncSocket(socket, oldSocket) : void addUserApp(userApp) : void removeUserApp(userApp) : void getUserAppsConnected() : array addGameWallApp(gameWallApp) : void removeGameWallApp(gameWallApp) : void getGameWallAppsConnected() : array dataObjectIsEmpty(object) : boolean

Table 6.1: *FIGAServer class diagram*

Every application that needs the functionality of a FIGAServer must import it and create a class that inherits from FIGAServer class. This gives the developer access to FIGAServer functionality through the use of inheritance.

The following paragraphs describe the different attributes and functions contained in FIGAServer. These form the functionality that can be used by applications that extends the framework.

FIGAServer Attributes

FIGAServer contains the following attributes that an application inherits by using the framework. The attributes can be seen in Table 6.1.

port - The network port the ServerApp listens for incoming connections. Must be available in order for the ServerApp to establish connections.

socketBindings - An array containing all of the different bindings for both UserApp and GameWallApp sockets. These bindings executes given functions when receiving specific events.

storeOldConnections - A boolean value that makes the ServerApp store connection information if a connection is ended, in case it will reconnect shortly after.

connections - An object containing three arrays, respectively *userApps*, *gameWallApps*, and *oldConnections*. *userApps* contains all connected UserApp sockets. *gameWallApps* contains all connected GameWallApp sockets. *oldConnections* contains all recently disconnected sockets along with their previous state information.

FIGAServer Functions

FIGAServer contains a series of functions that is used to maintain connections, send messages, and other utility functionality. The different functions can be seen in Table 6.1.

init - Starts up the ServerApp. Must be called from an application using the FIGAServer.

setPort - Used to set the port attribute. Must be called before **init** in order for the FIGAServer to use the port number.

addBinding - Adds a binding to all incoming connections. If a connection sends a message with the bound event, then the corresponding function will be executed.

addChannelBinding - Adds a binding similar to **addBinding**, but to a specific channel, such as UserApp or GameWallApp channels.

bindSocket - When a new socket connects this function will be called. It traverses the bindings in the *socketBindings* array and binds functions to the socket if the socket is of correct type.

sendDataToChannel - Sends a message to a channel with an event, payload, and a channel. Makes it possible to send messages only to a specific channel, such as UserApp or GameWallApp channels.

sendEventToSocket - Sends an event to a specific socket. Used in situations where an event without a payload is enough.

sendDataToSocket - Sends a message to a specific socket with an event, containing a payload.

addOldConnection - Adds a disconnecting socket to the *oldConnections* array for later reference.

reSyncSocket - If a socket reconnects and a matching old connection is found in the *oldConnections* array, it will copy the old socket state data to insure persistent information.

addUserApp - Adds the connecting socket to the *userApps* array.

removeUserApp - Removes the socket from the *userApps* array, if found.

getUserAppsConnected - Returns the *userApps* array.

addGameWallApp - Adds the connecting socket to the *gamewallApps* array.

removeGameWallApp - Removes the socket from the *gamewallApps* array, if found.

getGameWallAppsConnected - Returns the *gamewallApps* array.

dataObjectIsEmpty - Utility function used to determine if a socket contains valuable state information to store, to insure persistent connections.

Persistent Connections

FIGA supports persistent connections to UserApps and GameWallApps. If a user refreshes the web page or loads another web page using the same device, then developers might wish that session relevant data persists. Node.js automatically closes the connection between a ServerApp and a UserApp or a GameWallApp if a user refreshes the web page, but FIGA has underlying functionality to handle such events. FIGA gives the developer power to decide if this functionality is wanted in the application, since session control is not always needed.

Security

If an application requires web applications containing a control panel or administrative functionality, then a security system should be available for the sake of integrity. If the administrative web page is displayed on a large screen, it takes no more than a watchful eye to see the URL of the web page and malicious users can try to log in with the same administrative rights. A password protection system will protect the system by requiring a password to authorize the user. FIGA solves this problem by storing a text file on the ServerApp hosting device with a

password. This means that the file containing the password is only accessible on the ServerApp device.

6.1.2 UserApp

UserApp focuses on the client-side of the application. The relations between UserApp and FIGAUser can be seen in Figure 6.3, where the functionality of the FIGAUser provides connectivity with the ServerApp.

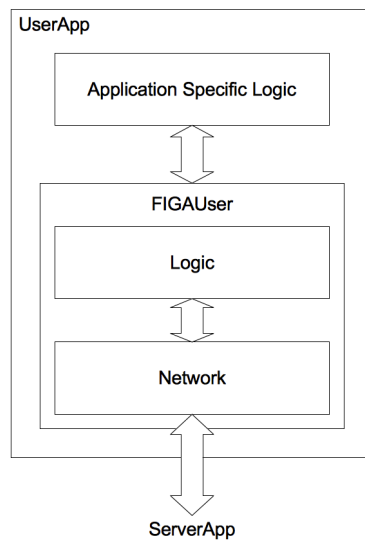


Figure 6.3: *UserApp component*

FIGAUser

While FIGAServer provides server-side functionality, FIGAUser handles the client-side functionality. FIGAUser gives the developer access to a network socket for communication with the ServerApp. While FIGAUser is code-wise independent from FIGAServer, the message events might not. Table 6.2 shows an UML class diagram of the implemented FIGAUser.

FIGAUser Attributes

The attributes are collected from Table 6.2.

FIGAUser
socket : object
addDefaultUserAppBindings() : void addBinding(event, func) : void sendEventToServer(event) : void sendDataToServer(event, data) : void connectToLocalServer() : void connectToServer(address) : void

Table 6.2: *FIGAUser class diagram*

socket - An object used for sending and receiving messages to and from the ServerApp.

FIGAUser Functions

The following functions can be seen in Table 6.2.

addDefaultUserAppBindings - Binds the default events for UserApps.

addBinding - Specifies a binding for the socket with a custom event and a related function.

sendEventToServer - Sends an event to the ServerApp.

sendDataToServer - Sends an event with a payload to the ServerApp.

connectToLocalServer - Makes the socket connect to the ServerApp, assuming the location of the UserApp files is hosted by the same device that runs the ServerApp.

connectToServer - Makes the socket connect to the ServerApp with a specific IP address.

6.1.3 GameWallApp

Similar to the UserApp, the GameWallApp focuses on the client-side of the application. Since the GameWallApp is responsible for the large common display of a FIGA application, its main task is to display the current state of the application. Figure 6.4 presents the GameWallApp component, and how the use of FIGAGameWall makes it possible to send messages to the ServerApp.

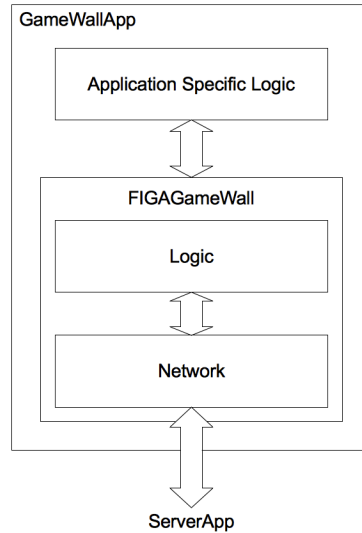


Figure 6.4: *GameWallApp component*

FIGAGameWall

FIGAGameWall provides similar functionality to the developer as FIGAUser. It gives developers access to a network socket that can send and receive messages to and from the ServerApp. Table 6.3 describes the implemented FIGAGameWall using an UML class diagram.

FIGAGameWall
socket : object
addDefaultGameWallBindings() : void addBinding(event, func) : void sendEventToServer(event) : void sendDataToServer(event, data) : void connectToLocalServer() : void connectToServer(address) : void

Table 6.3: *FIGAGameWall class diagram*

FIGAGameWall Attributes

The attributes are collected from Table 6.3.

socket - An object used for sending and receiving messages to and from the ServerApp.

FIGAGameWall Functions

The different functions are collected from Table 6.2.

addDefaultGameWallBindings - Binds the default events for GameWallApps.

addBinding - Specifies a binding for the socket with a custom event and a related function.

sendEventToServer - Sends an event to the server.

sendDataToServer - Sends an event with a payload to the server.

connectToLocalServer - Makes the socket connect to the ServerApp, assuming the location of the UserApp files is hosted by the same device that runs the ServerApp.

connectToServer - Makes the socket connect to the ServerApp with a specific IP address.

6.2 Physical View

To ease the understanding of how the different components are connected Philippe Kruchten designed a view model named *4+1 view model* [1]. The 4+1 view model specifies how software architecture can be described using five different, but related, viewpoints. These viewpoints represent the perspective of the system stakeholders. The system stakeholders are people, groups, or entities, which have an interest in or concerns about the realisation of the architecture.

One of the views in the 4+1 view model, the physical view, describes software onto hardware mapping. The view depicts the system from a system engineers point of view. Figure 6.5 presents a physical view diagram of a typical use case scenario of a FIGA application.

Figure 6.5 contains three nodes, each representing a device. The three devices in this case is a smartphone, a tablet, and a PC. Both the smartphone and

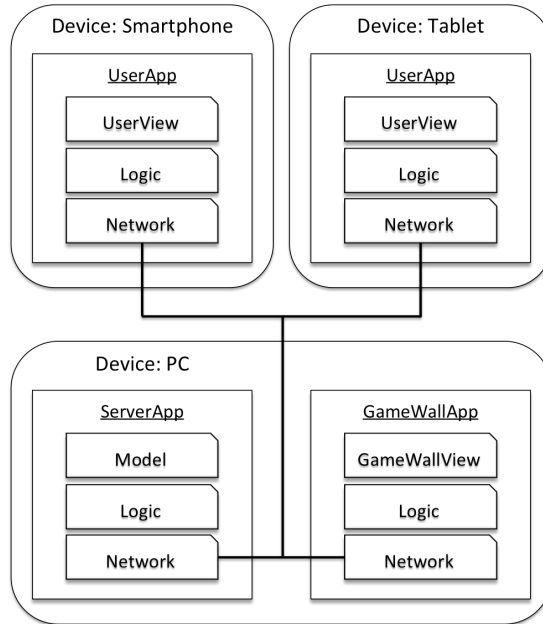


Figure 6.5: A use case scenario of a FIGA application presented with a physical view

tablet runs a UserApp, while the PC runs both ServerApp and GameWallApp at the same time. This scenario is related to an arbitrary lecture, where the teacher brings a laptop to the classroom and uses it with a projector during the lecture. The teacher starts the ServerApp on the laptop, and opens a web browser with a GameWallApp instance to be shown with the projector. The students participating the lecture use their smartphones and tablets to interact with the GameWallApp through a UserApp.

Another typical usage scenario of a FIGA application would be to have a dedicated computer only running a ServerApp instance. This will distribute the workload over several devices and increase individual performance. This scenario requires an extra device running the GameWallApp, but otherwise the behaviour is the same as the previously described scenario. Figure 6.6 illustrates the second use case scenario.

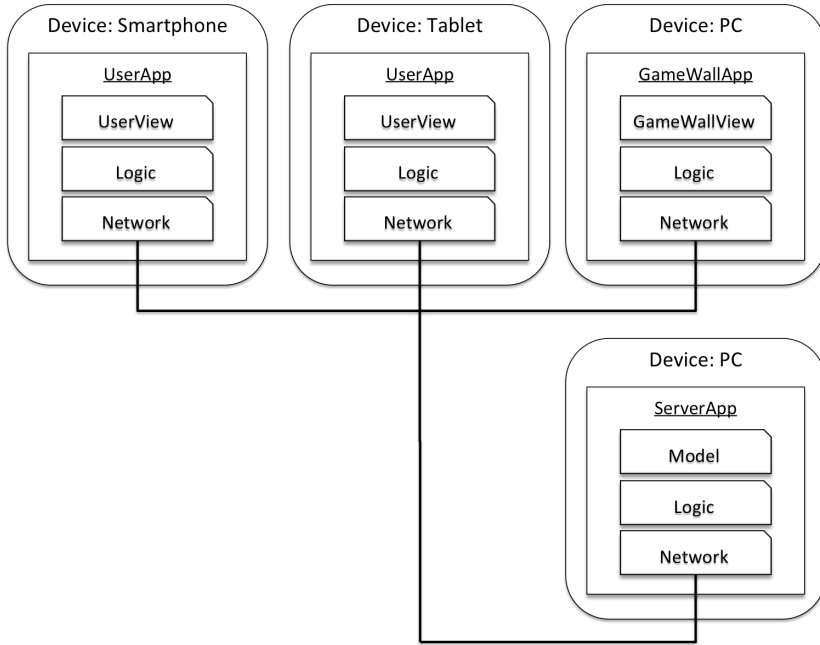


Figure 6.6: A second use case scenario of a FIGA application illustrated with a physical view

6.3 Sequence Diagrams

An UML sequence diagram is an interaction diagram that illustrates how different processes interact with each other and in what order [27].

Figure 6.7 illustrates the connection phase of an application, when a UserApp connects to a ServerApp. The ServerApp receives a connection request from a UserApp. It responds by sending a connection response back to the UserApp and updates its state. The UserApp responds by sending a verification message back to the ServerApp, making the ServerApp add the respective bindings for UserApps. The connection phase is now done and both ServerApp and UserApp is ready for subsequent messaging.

Another common sequential pattern is when a UserApp triggers an event on the ServerApp by sending a message. The message is sent to the ServerApp that updates its own state in accordance to the event. After the internal update the new state is broadcasted to every connected GameWallApp. Figure 6.8 presents

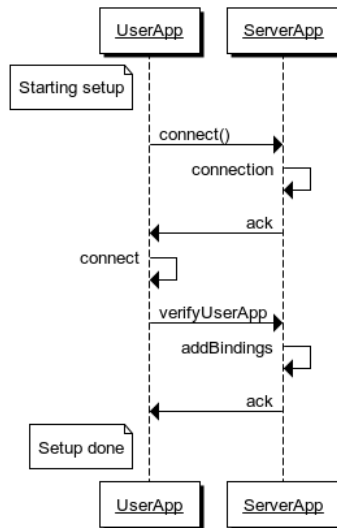


Figure 6.7: A UserApp connects to the ServerApp

the sequence diagram when a ServerApp broadcasts a message from a UserApp to every GameWallApp.

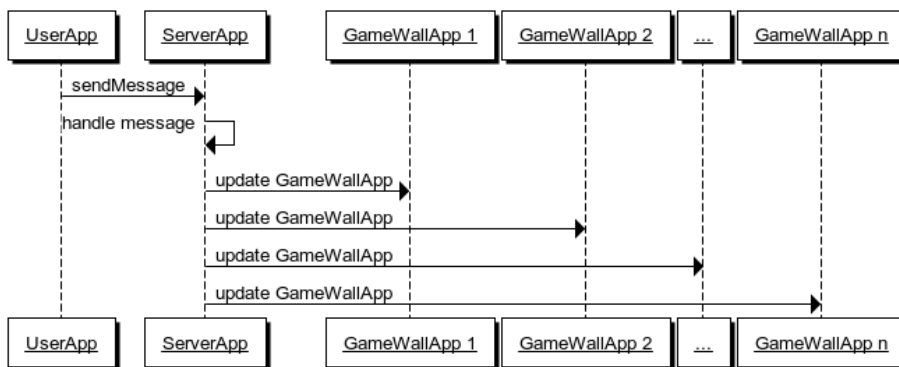


Figure 6.8: A UserApp sends a message to the ServerApp, which broadcasts it to all GameWallApps

6.4 Architectural Limitations

Every architectural design and pattern must be chosen carefully. By standardizing parts of the architecture and using external libraries, it is easy to create architectural drift or performance bottlenecks. The goal was to create a framework that has a net gain in value, if value was measured by helpful functionality coupled with reduced development time.

The network components of FIGA are heavily based on Node.js and Socket.IO. Node.js and Socket.IO use TCP sockets for message transmission, with the JSON format. This gives extra overhead and functionality that might not always be wanted. Especially in cases where low-level control is paramount, and the performance of the system is dependent on fast messaging without the need for the extra functionality offered by the TCP standard. A real-time ServerApp sends messages to each GameWallApps with updates in intervals. Since this information is updated 30 times every second, a late arriving packet is discarded rather than stalling the application waiting for the late packet.

Since FIGA applications are created using HTML, it is difficult to forbid developers of using additional JavaScript libraries. Contrary, it is much better to encourage developers to use existing libraries than to use time developing already existing solutions. However, by using existing JavaScript libraries one also inherits its problems and limitations. Each external library should therefore be closely inspected before integration. The effect and influence of these limitations on the entire system itself is application dependent and often hard to measure. The usage of several JavaScript libraries means more dependencies. This could affect different aspects, like modifiability and scalability, of the developed application.

A FIGA application uses the client-server pattern, making the ServerApp a single point of failure. If the ServerApp crashes or loses its connection to the network, then the entire FIGA application will collapse. This is an issue that needs to be taken into account when designing applications, especially if there is sensitive data that needs to be stored across sessions. In addition to a single point of failure, using only one server might result in an overload of user requests. In the case of a web server, every file needed to display a web page is transmitted to the connecting client. The server might be overloaded by requests if several devices connect simultaneously. One solution of resolving this problem is to keep the files that are required at the smallest size possible, which results in less data transmission. Another solution is to import the files from other server hosting locations, lessening the load on the server.

A web application differs from a static web page in that it actively execute code

and logic. To insure consistency, the server usually handles logic affecting data that should be identical across different connected devices. The computational load can for instance be calculating the application logic before the result is broadcasted to every connected client. The server needs to be able to calculate this without performance loss or else the packets will be delayed to all connected devices, thus hurting the performance of the entire system. To lessen the load on the server, client-side scripting is used to compute logic that can locally differ from other connected clients.

6.5 Creating a “Hello World” Application

To demonstrate how easy it is to create a FIGA application this section will describe how to implement a “Hello World” application. “Hello World” applications are often the first application programmers create when starting out with a new programming language or system. The application is simple, as it only displays the text “Hello World” and then ends. This described example application will behave similarly, only with barebone connectivity between a ServerApp, a UserApp, and a GameWallApp.

6.5.1 Preliminary Work

The first task that has to be done is creating a folder where the respective files can be created. At a minimum, the files that needs to be created and implemented are UserApp.html, UserApp.js, GameWallApp.html, GameWallApp.js, ServerApp.js, FIGAServer.js, FIGAUser.js, and FIGAGameWall.js. The last three JavaScript files, FIGAServer.js, FIGAUser.js, and FIGAGameWall.js, is provided by FIGA and needs no further work from the developers.

After the folder is created in the desired location, Node.js needs to be installed in order to start the ServerApp. There are several different ways of installing Node.js, depending on the device platform. Windows, OSX, and most Linux distros makes it possible to download packages through the terminal or command line. Windows and OSX have ready packages that can be installed using the native operating system installer wizards. Further information on how to install Node.js can be found on their web site [15].

In order for the system to offer the functionality of Socket.IO, Node.js needs additional modules downloaded. It is important that the current directory is the root directory of the ServerApp source files, in order for the installation to be

successful. Further information on how to install Socket.IO can be found on their web site [19].

6.5.2 Developing the ServerApp

The following section describes the implementation details of creating the “Hello World” ServerApp. The FIGAServer.js file is imported in order for the application to extend FIGAServer containing the before mentioned functionality needed. By using the object-oriented programming paradigm the functionality is extended from the GameWallServer class located in the file FIGAServer.js. In JavaScript object-oriented programming this is done through the use of *prototyping*. Prototyping defines inherited attributes and functions to the application specific class. The required lines of code for the ServerApp can be seen in Listing 6.1.

```
1 var Framework = require("../Framework/FIGAServer.js");
2
3 function HelloWorldApp() {
4   this.init = function() {
5     HelloWorldApp.prototype.init(__dirname);
6   }
7 }
8
9 HelloWorldApp.prototype = new Framework.GameWallServer;
10 var server = new HelloWorldApp();
11
12 server.addChannelBinding('verifyUserApp', 'all', function() {
13   console.log('UserApp says: Hello World!');
14   server.sendEventToSocket(this, 'HelloUserApp');
15 });
16
17 server.addChannelBinding('verifyGameWallApp', 'all', function() {
18   console.log('GameWallApp says: Hello World!');
19   server.sendEventToSocket(this, 'HelloGameWallApp');
20 });
21
22 server.init();
```

Listing 6.1: *ServerApp.js*

Line 1 in Listing 6.1 makes Node.js parse and execute FIGAServers source code and store a reference to it as a variable named *Framework*.

Line 3-7 creates a function that defines an *init* function that forwards the call to its framework prototype. This insures that the framework parent class is initiated correctly when the application class specific is initiated.

Line 9 defines the prototype for the *HelloWorldApp* to be the *GameWallServer* class. Line 10 instantiates a new instance of the *HelloWorldApp* class.

Line 12-15 and line 17-20 adds channel bindings, which make the *ServerApp* answer with the correct information depending on the client, if the client is a *UserApp* or a *GameWallApp*.

Finally, line 22 begins the initiation process of the application and the underlying framework. The *ServerApp* is now ready for incoming connections.

6.5.3 Developing the UserApp

This section describes the implementation details of creating the “Hello World” *UserApp*. The *UserApp* consist of three files, respectively *UserApp.html*, *FIGAUser.js*, and *UserApp.js*.

The required lines of code for *UserApp.html* are listed in Listing 6.2.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Hello World UserApp Example</title>
6   </head>
7   <script src="/socket.io/lib/socket.io.js"></script>
8   <script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery
9     .min.js"></script>
10  <script src="FIGAUser.js"></script>
11 </html>
```

Listing 6.2: *UserApp.html*

All of the lines 1-11 in Listing 6.2 are HTML syntax. They are needed for the *UserApp.html* to be shown in a web browser, but the lines of interest for FIGA are line 7, 9, and 10.

Line 7 includes Socket.IO that makes it possible to connect to a Node.js powered server through a web browser by using JavaScript.

Line 9 includes *FIGAUser.js*, which makes the connection to the *ServerApp* and ease the development of *UserApp.js*.

Line 10 includes *UserApp.js* where developers write their implementation code for additional functionality and behaviour.

The required lines of code for *UserApp.js* are listed in Listing 6.3.

```

1 $(document).ready(function(){
2   connectToLocalServer();
3   addBinding('HelloUserApp', function(){
4     $('body').html('Server says to UserApp: Hello World!');
5   });
6 });

```

Listing 6.3: *UserApp.js*

Line 1 in Listing 6.3 is a standard jQuery function that is being called when the document is loaded and ready for input.

Line 2 is function invocation of the method `connectToLocalServer()` in the `FIGAUser.js`, which set up a connection to the `ServerApp`.

Line 3-5 adds a binding on the command “HelloUserApp”. Whenever the `UserApp` gets the command “HelloUserApp” from the `ServerApp` it performs the function content presented in line 4.

6.5.4 Developing the GameWallApp

This section describes the implementation details of creating the “Hello World” `GameWallApp`. Similar to the `UserApp`, the `GameWallApp` consist of three files, respectively `GameWallApp.html`, `FIGAGameWall.js`, and `GameWallApp.js`.

The required lines of code for `GameWallApp.html` are listed in Listing 6.4.

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Hello World GameWall Example</title>
6   </head>
7   <script src="/socket.io/lib/socket.io.js"></script>
8   <script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery
9     .min.js"></script>
10  <script src="FIGAGameWall.js"></script>
11  <script src="GameWallApp.js"></script>
12 </html>

```

Listing 6.4: *GameWallApp.html*

The lines 1-11 in Listing 6.4 is almost identical to the lines of `UserApp.html` in Listing 6.2. The exceptions are line 5, 9, and 10. Line 5 determines the title of the HTML site that now has a `GameWallApp` reference, while line 9 and 10 includes `FIGAGameWall.js` and `GameWallApp.js`.

The required lines of code for GameWallApp.js are listed in Listing 6.5.

```
1 $(document).ready(function(){
2   connectToLocalServer();
3   addBinding('HelloGameWallApp', function(){
4     $('body').html('Server says to GameWallApp: Hello World!');
5   });
6 });
```

Listing 6.5: *GameWallApp.js*

As with the GameWallApp.html which is almost identical to UserApp.html, the GameWallApp.js is very similar to UserApp.js described in Listing 6.3. The only differences are line 3 and 4, where the binding contains a GameWallApp reference and the text that is displayed on the web site is changed.

6.5.5 Starting the Application

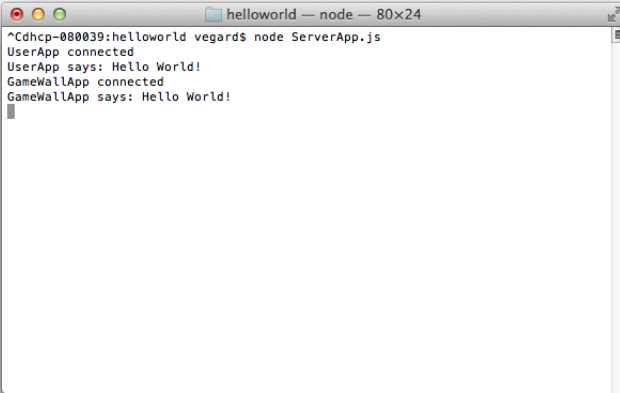
After creating the different sub applications ServerApp, GameWallApp, and UserApp, the FIGA application is ready to launch. To start the ServerApp the developer navigates to the correct folder in the terminal or command line, and then type the given command listed in Listing 6.6.

```
1 node ServerApp.js
```

Listing 6.6: *Start the FIGA application*

This given command starts a ServerApp instance, and the ServerApp is now ready for UserApp and GameWallApp connections. Figure 6.9 illustrates the console output from a ServerApp after one UserApp and one GameWallApp has connected.

Figure 6.10 presents the web browser of a UserApp using the “Hello World” application, and Figure 6.11 presents the GameWallApp.

A terminal window titled "helloworld -- node -- 80x24" showing the output of a Node.js script. The output consists of five lines: a shell prompt, a connection message, a "Hello World!" message, another connection message, and a second "Hello World!" message.

```
^Cdhcp-080039:helloworld vegard$ node ServerApp.js
UserApp connected
UserApp says: Hello World!
GameWallApp connected
GameWallApp says: Hello World!
```

Figure 6.9: *ServerApp using the “Hello World” application*

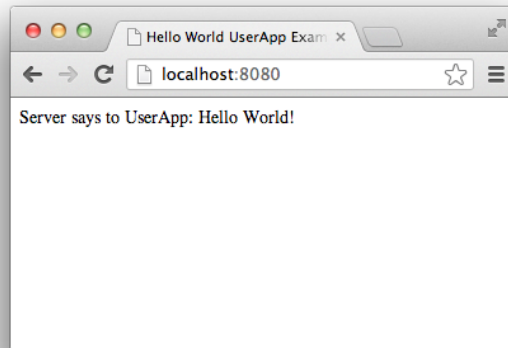


Figure 6.10: *UserApp using the “Hello World” application*

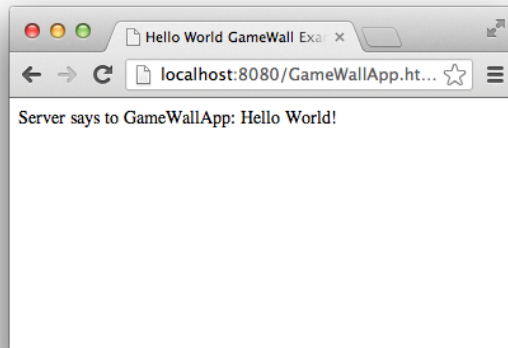


Figure 6.11: *GameWallApp* using the “Hello World” application

Chapter 7

Prototypes

Four prototypes have been developed to test the capabilities of FIGA and to demonstrate its potential. The four prototypes were developed using rapid prototyping and they revealed different aspects of FIGA that needed improvements. Some of the prototypes were developed with a particular technical interest in mind, while other for investigating educational benefits or new ways of interacting with each other. As described in Section 2.2, three of the prototypes were developed to test FIGA applications in a classroom setting. The experiments conducted with these prototypes would find out if there exists any potential learning benefit of using social, multimodal applications in a classroom. The fourth prototype was a video game, developed in an effort to strain the FIGA application and to locate technical bottlenecks.

Some of the applications need functionality not supplied by FIGA, and all but one uses external frameworks in accordance with FIGA. These are described in detail for the relevant prototypes and they will give indications on improvements that can be made to the prototypes for later studies. This chapter documents the process of creating the ServerApp, the GameWallApp, and the UserApp for each prototype. Each prototype has an associated lessons learned paragraph, with helpful knowledge learned by developing the prototype.

7.1 PostIt

Brainstorming with post-its is a well-known, well-used approach. Every participant involved in the brainstorm receives a stack of post-it notes and are told to

write down everything they relate to a specific topic. All the post-its are then collected and placed on a large board for further processing.

PostIt is created to mimic this process, removing unnecessary time waste and making the post-process easier. Every user has their own device, often a smartphone, a tablet, or a laptop, where they can write on virtual post-it notes and submit them. The shared display, the `GameWallView`, is used for showing all the different post-its that have been submitted.

PostIt requires administrative features such as categorization and deletion of post-it notes. This functionality helps the organizer structure the information sent in and filter out unwanted or redundant words that have been mentioned before.

7.1.1 The UserApp

The UserApp is as minimalistic and bareboned as possible. The main focus for the user is to share new ideas and thoughts, and therefore should not be distracted with a complicated user interface. The `UserView` interface will resemble an ordinary post-it, mimicking the physical process as close as possible. The `UserView` consists of a logo, a post-it and a submit button. The user can touch the post-it to start writing new words and press the submit button when ready to deliver the post-it. Figure 7.1 shows the `UserView`.

As the target platform for UserApps are smartphones and tablets, this means supporting different resolutions and screen sizes. The `UserView` is also aesthetically pleasing on larger screens, as shown in Figure 7.1. The idea of using a gesture motion to throw the post-it onto the `GameWallView` was discussed, but this was not implemented due to limited time. Once a post-it is sent from the UserApp, the `UserView` clears all text and is ready for new input.

7.1.2 The ServerApp

The `ServerApp` contains and maintains the state of the application. Every time one of the UserApps commits a new post-it to the `ServerApp`, the `ServerApp` broadcasts the new post-it to every connected `GameWallApp` and updates its state. The `ServerApp` also sends the post-it history when a new `GameWallApp` connects to insure that the new `GameWallApp` displays all previously submitted post-its.

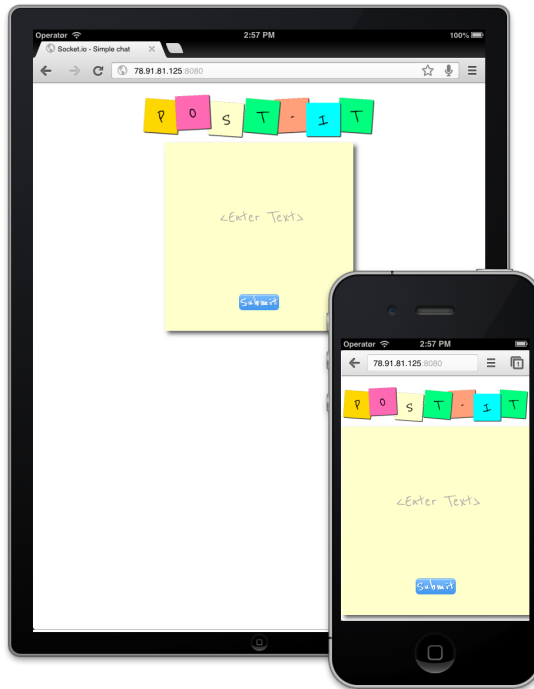


Figure 7.1: *The PostIt UserView on different devices*

7.1.3 The GameWallApp

The GameWallApp functions as an electronic board that displays all submitted post-its. It will also allow administrators to modify the post-its, such as deleting, moving, and categorizing them. Figure 7.2 shows the GameWallView when the application is connected to the ServerApp.

To avoid any misdemeanor from unauthorized users, the application requires a password when connecting a new GameWallApp to the ServerApp. If the correct password is entered, the ServerApp will consider the GameWallApp as validated and start to send it information about submitted post-its. If the password is incorrect, the ServerApp will ignore the connection attempt and the GameWallView will fail to load.

The GameWallApp has the functionality to categorize the different post-its. The organizer has to create these categories, which is done by clicking on the + -button in the lower right of the screen. A popup-box is revealed, as shown in Figure 7.3,



Figure 7.2: *The PostIt Game WallView filled with post-its*

and the user can enter the name of the category and choose a desired color that will be associated with it.

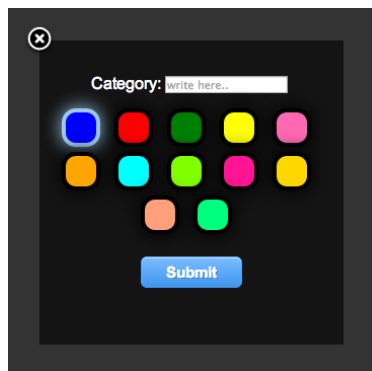


Figure 7.3: *Adding a new PostIt category*

At any given time, a validated GameWallApp can sort all the post-its it has received. This requires that at least one of the post-its have been categorized prior to sorting. Figure 7.4 shows all the post-its categorized, while Figure 7.5 shows the notes sorted based on their category. This makes it easier for the organizer to process the responses from the students.

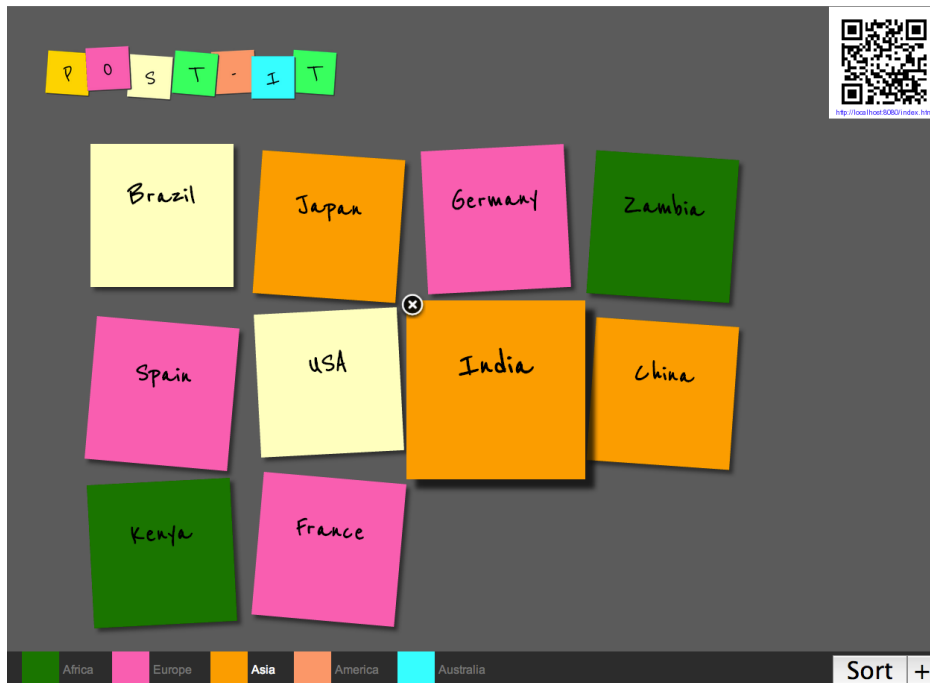


Figure 7.4: *The PostIt GameWallView with a highlighted post-it*

When someone creates an application that shares anonymous user input publicly, it is highly recommended to implement moderation functionality. There is always a chance of misdemeanor when user input is involved, especially if the group contains immature individuals. If the GameWallApp administrator finds post-its which are irrelevant to the topic, then the administrator can manually delete and remove the post-it from the GameWallView. Figure 7.4 illustrates when a post-it is highlighted with a delete button attached.

In order to make it easier for users to load the UserApp in their web browser, a QR code image generator that is displayed on the GameWallView is implemented. A QR code is a two-dimensional image containing information that can be scanned to retrieve its information [36]. Smartphones and other devices with a camera

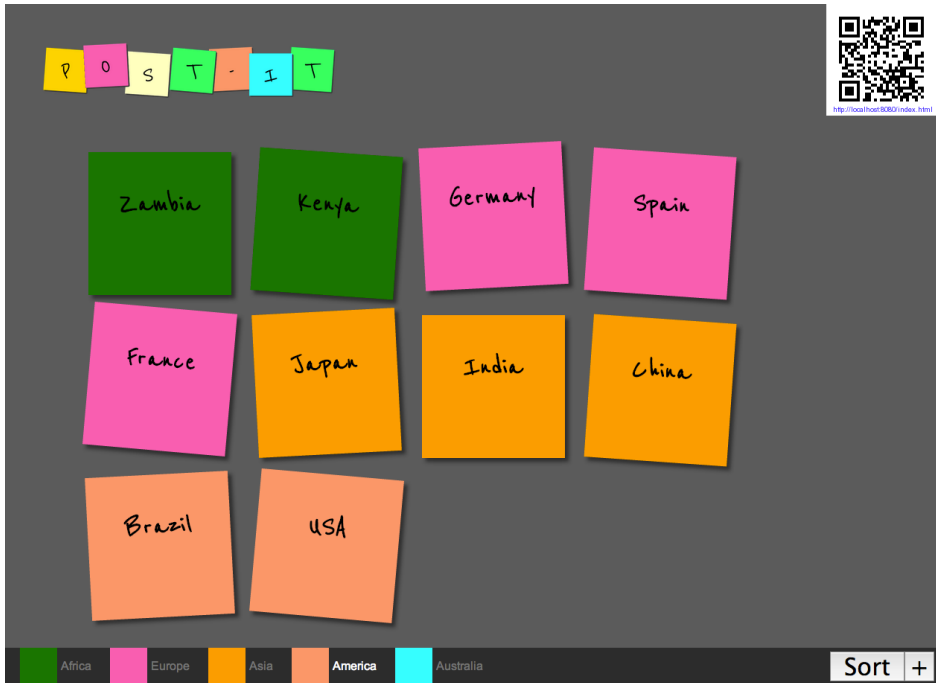


Figure 7.5: All post-its sorted based on their category

and additional software can quickly retrieve this information. The information stored in this case is the URL for the UserApp. The QR code image resizes with an animation if the mouse cursor is hovering over it, making it large enough for everyone in the room to scan it when needed. The QR code can be seen in the upper right corner of Figure 7.2.

7.1.4 Lessons Learned

PostIt give the users looking at the GameWallView the impression of an application not constricted to web application standards in terms of presentation and performance. The focus was to create an appearance that closely resembles native applications in order to blur the lines between web and native applications. This functionality was created using animations, toolbars with buttons, and avoiding user interface elements associated with web technology.

field and a button. Figure 7.7 illustrates the UserView.

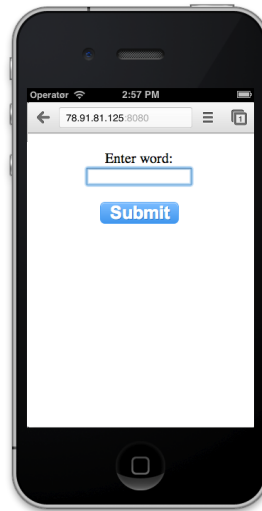


Figure 7.7: *The WordCloud UserView*

When a user submits, the text written in the text field is sent to the ServerApp. The ServerApp then broadcasts the new information to all of the connected GameWallApps. When text is submitted to the ServerApp, the UserView clears the text and awaits further input from the user.

7.2.2 The ServerApp

The ServerApp functions as a session and state holder. It stores all mentioned words submitted by UserApps and how many times they have been mentioned. The resulting list of words and how many times they have been mentioned is sent to all GameWallApps every time an arbitrary UserApp submits a word.

Two issues with information sharing applications are vandalism and improper behaviour from users. The WordCloud application uses no word filter, which means that what is displayed is at mercy of the users. It was decided to not use time implementing a filter that needs constant updates, and is easily avoided by resourceful individuals. It should be noted that there exists libraries and web services that performs word filtering, but this prototype focused on other aspects and functionality.

7.2.3 The GameWallApp

The majority of implementation time was spent on the GameWallView. Displaying the words by their respective mentioned count is performed by the GameWallApp and it requires sophisticated logic calculations. Since every word should be placed together tightly with different font sizes, it quickly becomes complicated to place words without overlapping. At first an own solution got implemented, but the time spent reinventing the wheel was not worth it. This resulted in examining different JavaScript libraries that were suited for displaying word clouds. The popular data-visualizing library D3.js was chosen primarily for its excellent support of visual representations.

Computation time escalates from milliseconds to seconds when the data set becomes large. This means that the time from a user submits a word to the GameWallApp refreshes becomes noticeable when the data set contains a significant selection of words. The computation time depends on the GameWallApp running device, but noticeable delays was experienced with the development environment at around 40 unique words. This delay was accepted since the responsiveness of the application was not of priority. When words are added continuous the view will update as quickly as possible, without giving the impression of being slow and unresponsive unless given special attention. Figure 7.8 shows a screenshot of the GameWallView.

The typical approach when using a word cloud is a two-step process. The first step is collecting user input, and the corresponding visual presentation is created in step two. In this case the WordCloud continuously receives data from its users. This means that the application has to update the GameWallView as users frequently enter new words, meaning step one and step two is being performed multiple times. To accomplish this behaviour, the implementation has to support a frequently updating data set.

7.2.4 Lessons Learned

Time spent reinventing the wheel is time that could be better spent on other tasks. The D3.js library is efficient and reliable, and displays the word cloud in a visually pleasing manner. The library is sufficiently customizable for the applications needs with the ability to define the size of the canvas to render and the size of the different words. One issue is that the placement of the words are random each time the word cloud is updated, meaning that the words will move rapidly around the screen if input is received rapidly. When the data set increases, so does the time required to calculate the new positions of the words

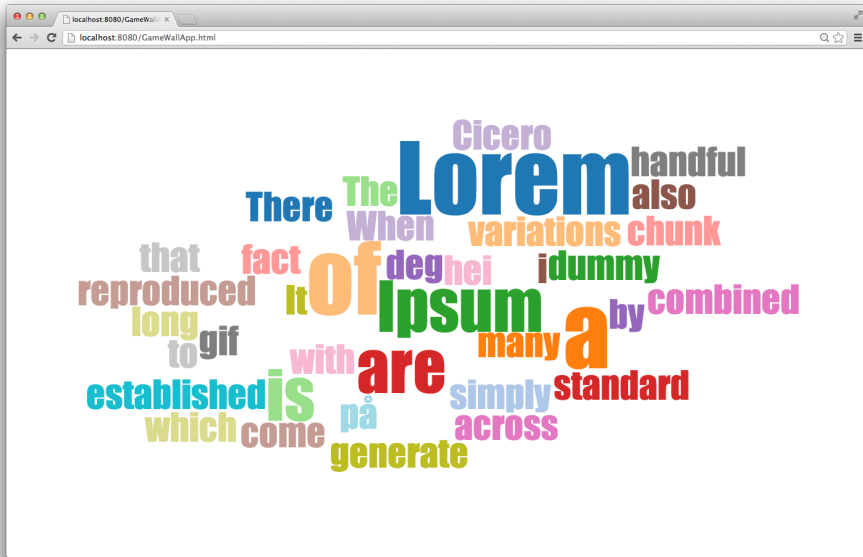


Figure 7.8: *The WordCloud GameWallView*

in the word cloud.

WordCloud is dependent on a large dataset of words in order to display a meaningful visual representation. If all words are mentioned a single time then all of the words will be of the same size on the GameWallView. Word matching is also a problem, where this prototype allowed the words to be capitalized. This means that the following input “test” and “Test” would be considered as two different words. This is trivial to change, and was left in as a design choice.

7.3 Categories

The Categories concept was penned by our supervisor, who wanted to explore the pedagogical advantages and user experience of a more active problem solving application. Categories gives the user two category choices and a serie of terms that must be placed into the correct category, as illustrated in Figure 7.9.

When every connected user has placed all terms in the correct category, a new set of categories and terms is broadcasted to all users. A typical use case for this



Figure 7.9: *The Categories UserView*

application is a teacher that prepares a series of categories and terms that needs to be placed in one of two categories. The theme of the categories and terms is in its entirety up to the teacher and can be anything from history to religion. The touch interface of the UserView gives the user an intuitive approach, dragging the terms into the correct category slot. The GameWallApp is tasked with showing the progress of the entire class.

7.3.1 The UserApp

The main focus of the user is turned towards the UserView, contrary to the other previous applications where the GameWallView has been the center of attention. It shows two rectangular shapes with respective headers. Each of these shapes represents containers of a certain category, and underneath these shapes are several uncategorized terms. The user must place these terms into the correct category by dropping them into the correct category container. When

all terms are placed, a message is sent to the ServerApp with a statement of completion. If the task is correctly categorized then the terms can no longer be moved. The UserApp now waits for a new set of categories and terms from the ServerApp. Figure 7.10 shows a screenshot of the UserView on a tablet where some terms have been placed into the categories.

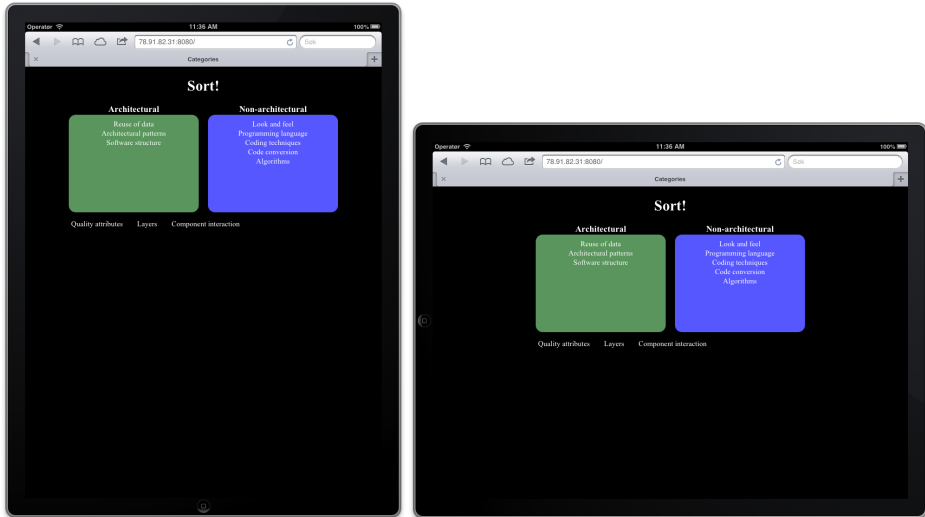


Figure 7.10: *The Categories UserView on a tablet*

7.3.2 The ServerApp

The ServerApp contains all combinations of categories and terms, and works primarily as a content provider for UserApps and as a state holder for GameWallApps. It also contains the solutions to each set of categories and terms, and can determine whether a user has completed correctly the categorization. When a new UserApp connects to the ServerApp it sends a set of categories and terms to the connected UserApp, as well as it broadcasts the updated state of all connected UserApps to all GameWallApps.

When a user is done placing all terms, the UserApp will message the ServerApp with a statement of completion. The ServerApp then checks if the categorization is completed correctly, and answers with a corresponding message. If the categorization is wrong, the UserApp keep on with the current set of categories and terms. However, correct categorization results in a message letting the UserApp

know that the task is completed and that the user has to wait for fellow students to complete the task. The ServerApp also notifies all connected GameWallApps with the new state information. This state information regards the number of completed UserApps. Every time a UserApp sends a message to the ServerApp with a statement of completion it checks if all connected UserApps are finished. If so, it delivers the next set of categories and terms to every connected UserApp and broadcasts a notice with the new state to all GameWallApps.

7.3.3 The GameWallApp

The GameWallView displays the current application state of how many UserApps that are done with the current set of categories and terms, as well as the total number of connected UserApps. This information is displayed as a circular ring that fills up according to the percentage of completed UserApps. In addition to the graphical representation, a label is shown with a textual representation of the completed percentage. Figure 7.11 shows a screenshot of the GameWallView on several different devices.

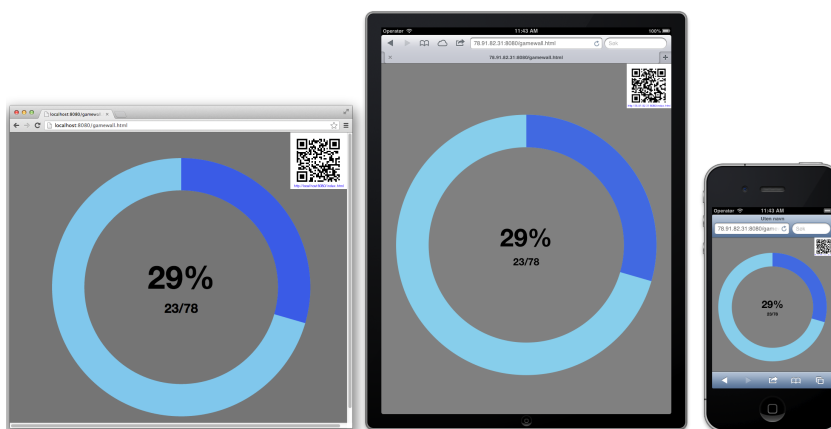


Figure 7.11: *The Categories GameWallView*

7.3.4 Lessons Learned

Categories was implemented to test if it was possible to create a user interface that resembles native application in terms of touch interaction. Touch controls are the de facto standard of modern smartphones, meaning that touch gestures

such as pinch, drag and drop, and swipe are now well-known and expected ways of interacting with touch screen applications. This functionality was implemented using the mobile version of jQuery, which implements touch functionality and mimics the look and feel of native applications [11].

Displaying a meaningful graph on the GameWallView of how many UserApps that had finished the current category set was to show the class progression. Several different graphical libraries that took raw data as input and displayed it using different graph conventions were examined. The D3.js library was selected as it provided polished and powerful ways of visualizing data.

7.4 1814

The possibilities of using FIGA for entertainment purposes with real-time requirements was daunting, but exciting. The video game 1814 was developed to test the network and computational capabilities of the chosen framework components and to observe the social benefits of playing a video game together in the same room.

North & South is a video game developed and published by Infogrames in 1989 for platforms such as Commodore, Atari, and the Nintendo Entertainment System [16]. The game contains several different mini-games that are played differently, and one of these mini-games was found well suited. The battlefield mini-game gives one or two players control over their own army that engages in combat, as illustrated in Figure 7.12.

Each army has three different characters with different combat roles, respectively infantry, cavalry, and artillery. These characters work in the rock-paper-scissors fashion, meaning that different characters are stronger against particular other characters. The infantry is good against the cavalry, the cavalry is good against the artillery, and the artillery is good against the infantry. The gameplay is fast and hectic, and rewards both teamwork and individual performance.

This version of the battlefield mini-game is similar in terms of visual presentation, paying homage to the original while making some adjustments. A localization change is made where the teams consists of Norway against Sweden, instead of the Union versus the Confederacy in the American Civil War during the 1860s. The sole purpose of this change is to use the somewhat humourous competitive relationship between Norway and Sweden to engage players. In the original mini-game the player has to control and navigate an entire army. This makes the game chaotic and hectic, and the player can choose between different game tactics and approaches. This version will differ from this approach, letting a player control

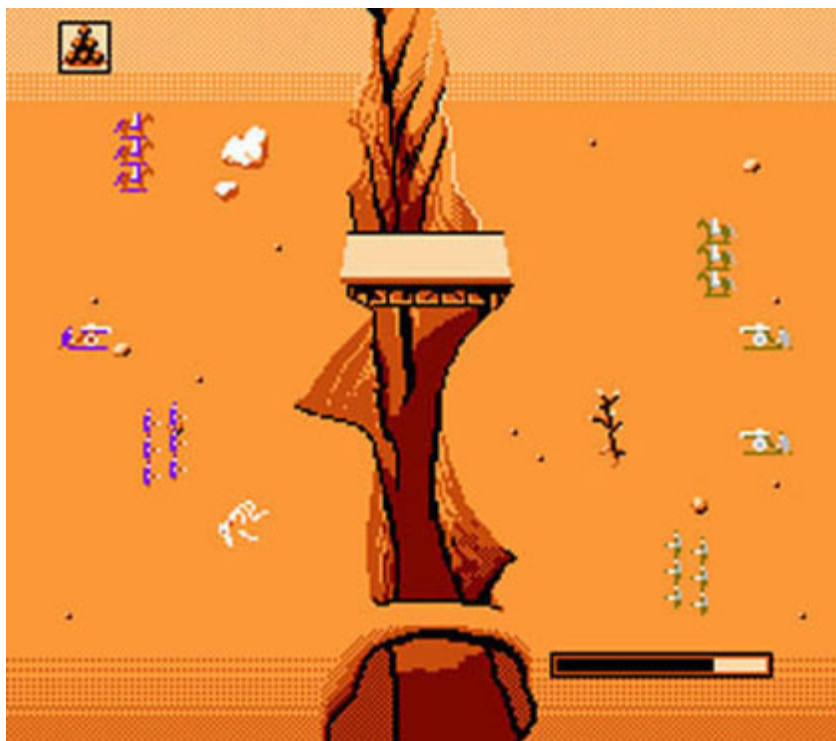


Figure 7.12: *The original game concept of North & South*

and navigate only one character within one of the three combat roles. Thirty players would give thirty different characters on the GameWallView at once. This approach makes the game scalable and unpredictable, and even more chaotic and hectic than the original game.

7.4.1 The UserApp

The UserView is simulating a classic game controller with a directional pad, one attack button, and one information button. In order to accommodate for different devices with different resolutions, the interface is designed using the responsive design technique. The directional pad is attached to the lower left of the screen and the two buttons, attack and information, to the lower right of the screen. This made the design supportive of different screen sizes and resolutions. By having the directional pad and buttons in each lower corner, the player holds the device

and use the thumbs for touch interaction. Figure 7.13 shows how the `UserView` adjust to the screen size and resolution on smartphones and tablets.



Figure 7.13: *The `UserView` is developed with responsive design techniques*

The lack of available screen space on smartphones led to minimalistic design solutions. Displaying too much information will clutter the screen and degrade the user experience. One decision was to let the background image of the `UserView` show the flag of the players' allegiance. This gives the player an immediate understanding of which faction the player belongs to and who to attack. In addition to the background image, the attack button displays an image that indicates the type of character the player controls. The image on the attack button is a rifle, a sword, or a cannon. This makes it intuitive to attain both the team and type of character the player controls, while making the controller design consistent across different type of characters.

The information button is used to find the whereabouts of the players' character on the `GameWallView`. A lesson learned from the *MOOSES* project noted that the players had problems finding their character on the large common screen [24]. While the player presses the information button, the player name will appear over the player character. This makes it possible to temporary identify the character the player controls quickly, without distraction, and only when needed.

7.4.2 The ServerApp

The 1814 video game is a real-time application, opposed to the other prototypes which are event-driven applications. This results in a higher computational load for the ServerApp. The ServerApp calculates the player characters position, move projectiles, and update the score and time, all in real time. These factors make the design and implementation of the ServerApp a complicated affair.

Players are often engaged and have high expectations about the video games they play. Their expectations and demands related to video games increases with each new iteration of games. A new game is expected to outperform the previous games, in terms of simulational complexity or innovative game design. Several technical considerations have been done to provide an enjoyable experience. In order to provide a smooth graphical experience, Claypool et al. suggests 30 frames per seconds [42]. The network latency can not peak above 100 ms, as a noticeable delay between user input and computer feedback will lead to frustration and degrade the user experience among the players [39]. This further points to the critical nature of the design and implementation of the ServerApp.

When a ServerApp starts up, it begins in a configuration mode. The first GameWallApp to connect to a ServerApp currently in configuration mode will display the different settings a game can be configured with. The different settings are for instance the choice between several game modes, or the time limit of a game. The configuration screen is illustrated in Figure 7.14.

The player controlling the first GameWallApp choose the desired settings and submits these settings to the ServerApp. The game is now configured and the ServerApp is ready for incoming connections. If a UserApp connects to the ServerApp before the configuration phase is completed, the ServerApp will respond with an error message informing the UserApp to wait until the ServerApp is configured.

The ServerApp stores for each UserApp connection the players game state. The players game state is for instance team, character type, position, and score. This game state will change during the game, for instance when the player kills, dies, or selects a new character.

7.4.3 The GameWallApp

The GameWallView displays the battlefield with all the player characters on it, as illustrated in Figure 7.15.



Figure 7.14: *The 1814 configuration screen*

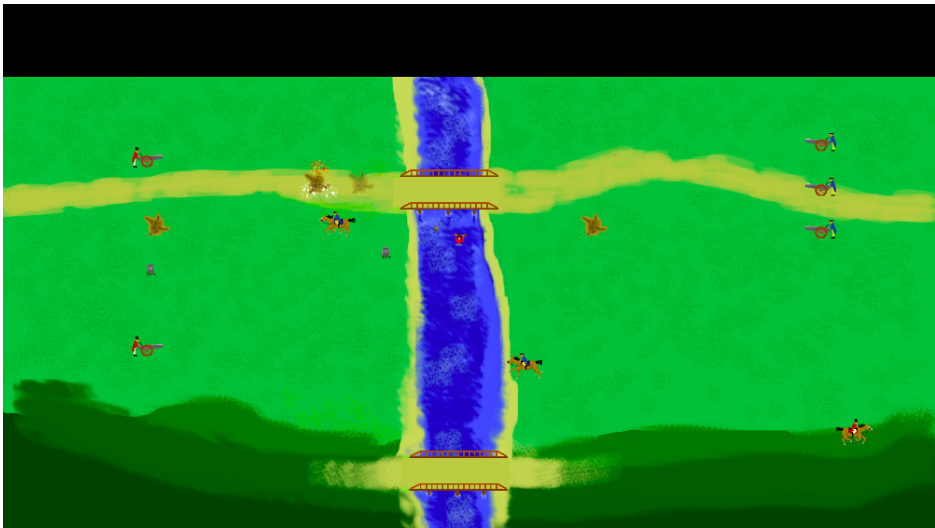


Figure 7.15: *The 1814 GameWallView*

The game is graphically significant more sophisticated than the other prototypes, with real-time rendering and animations. To aid development and to reduce implementation time of the GameWallApp, the application took advantage of importing the powerful JavaScript library Easel.js [6]. Easel.js provides utility functions for working with rich graphics and drawing them on a HTML5 canvas. The ServerApp sends an updated game state to all connected GameWallApps 30 times per second. The received game state contains information such as the players position, projectiles position, and updated score and time. This information makes it possible for the GameWallApp to draw graphics at the correct location on the GameWallView. The main task of every GameWallApp is to represent this information in a fast and consistent manner, making the game feel responsive and smooth for the players.

7.4.4 Lessons Learned

The complexity of creating a fully functional video game should never be underestimated, even in a relatively small scale. The algorithms for collision detection, updating the positions of the players and projectiles, and graphical rendering had satisfying performance.

The network modules communicate using TCP packets, which is the standard of the underlying web socket technology used in Node.js and Socket.IO [15, 19]. This is the recommended technology to use in a system where guaranteed delivery of message is paramount, but in a video game setting this becomes redundant. This is because the only relevant information is the current updated game state, meaning a previous game state packet that is sent from the ServerApp is redundant. TCP packets produce overhead both in terms of creating headers with packet identifiers and requiring acknowledgements for each sent message, doubling the network traffic. The performance of the messaging was satisfactory when the game got tested in small scale, but it is difficult to predict how it would perform in a larger setting.

Developing applications and games for a large audience requires testing in a real setting as often as possible. This project had limited time, resources, and manpower to perform these kinds of tests. This made it hard to find issues that could arise when the application has a high workload. Testing with virtual connections is insufficient, since the sporadic and behavioural tendencies of a human player is hard to mimic virtually with the available development time.

Chapter 8

User Experiments

One approach to test a framework is to test the applications made with it. The level of success is measured with the user experience response, the development time, and the experience gained using the framework. Ideally, the framework should provide short development cycles with flexible functionality to support a wide variety of features needed in different applications.

In accordance with our supervisor two user experiments were arranged to gather feedback from users regarding the four developed prototypes. The experiments were to complement a software architecture lecture. The participants were mostly software engineering master students with high technical competence, and every one of them owned modern mobile devices. The first experiment introduced the WordCloud, PostIt, and Categories applications for user feedback. The second experiment was a continuation of the first experiment, in addition to the 1814 video game.

This chapter introduces the techniques used for collecting data from the experiments, the experiment procedures, and experiences gained from both the participants and how the applications were to use from the perspective of an organizer.

8.1 Empirical Approach

There exists several different ways of collecting valuable user feedback. Qualitative methods focus on individual feedback through interviews. Quantitative

methods focus on the average impression through standardized forms. Since the experiments were set in a classroom setting with tenfolds of students, a quantitative method, and not qualitative, got utilized in terms of data collection. This resulted in using forms that consisted of five parts, namely general information about the participant, SUS questions, technical considerations, an application comparison, and a comments section. The final forms used in the experiments can be found in Appendix C.

8.1.1 General Information

In order to collect background information about the participants, the survey included some general information questions at the beginning of the form. This information was later used for analyzing the results. The questions about the general information were as follows.

- Age (e.g. 24 years)
- Gender (male/female)
- Study programme (e.g. MTDT)
- Mobile device (e.g. iPhone 4S, Samsung Galaxy SIII, MacBook, Samsung Galaxy Tab)
- Web browser (e.g. Opera, Safari, Firefox)
- Connectivity (e.g. Edge, 3G, Wi-Fi)

8.1.2 System Usability Scale

SUS is a standardized form used to gather empirical data from user experiments [22]. The form contains ten questions that can be answered on a scale from 1 to 5, where 1 is strongly disagree and 5 is strongly agree. Every other question is formulated negatively, making the user reflect upon the questions more carefully. The sum of the SUS form gives a score from 0 to 100 regarding the usability, where the higher score means a better user experience. See Table 8.1 for a complete overview of the questions asked in the SUS form.

SUS forms have been proved to be robust and reliable, and a valuable evaluation tool. It has been used for a variety of research projects and industrial evaluations

<i>SUS questions</i>	
1	I think that I would like to use this system frequently
2	I found the system unnecessarily complex
3	I thought the system was easy to use
4	I think that I would need the support of a technical person to be able to use this system
5	I found the various functions in this system were well integrated
6	I thought there was too much inconsistency in this system
7	I would imagine that most people would learn to use this system very quickly
8	I found the system very cumbersome to use
9	I felt very confident using the system
10	I needed to learn a lot of things before I could get going with this system

Table 8.1: *The SUS questions*

multiple times.¹

8.1.3 Technical Considerations

Additional and specific questions regarding the technical aspects were desired in addition to the SUS questions. The technical inquiries were to map the user experience regarding the technical performance of the applications. Since the applications were made with FIGA, the best-case performance relies on the framework performance. Questions regarding the ease of connecting to the applications through a web browser, as well as latency issues, are some of the questions asked. See Table 8.2 for a complete overview of the questions regarding the technical aspects.

The technical considerations are based on two articles, namely *A Pervasive Game to Know Your City Better* and *Lecture Quiz - A Mobile Game Concept for Lectures* [49, 26].

Similar to the SUS questions, the technical considerations are answered on a scale from 1 to 5, where 1 is strongly disagree and 5 is strongly agree. To simplify the results for the reader, the following tables in this chapter is a simplification where

¹This report acknowledge that SUS was developed as part of the usability engineering programme in integrated office systems development at Digital Equipment Co Ltd., Reading, United Kingdom.

<i>Technical considerations</i>	
1	I found it easy to setup and start using the applications through the web browser
2	I did not manage to interact with the GameWall system with at least one web browser
3	I found it satisfying that I did not have to install any of the applications on my smartphone
4	I found the applications unresponsive and/or with noticeable network latency
5	I felt that the applications had consistent quality

Table 8.2: *The technical considerations*

both strongly disagree and disagree, and agree and strongly agree, are merged together. The complete tables with the whole scale range and respective results are located in Appendix B.

8.1.4 Application Comparison

RQ5, described in Section 2.1.5, led to an investigation of the best-suited application for a classroom setting. Table 8.3 lists the eleven questions regarding the application comparison. The application comparison is answered on a scale from 1 to 3, where 1 represents WordCloud, 2 represents PostIt, and 3 represents Categories.

<i>Application comparison</i>	
1	The application that was the most fun to use
2	The application that was the most engaging
3	The application that I believe has the highest learning outcome
4	The application that I would like to be used in most classes
5	The application that made me think the most
6	The application that demands the most creativity
7	The application that made me initiate social interaction
8	The application that was the most challenging
9	The application that made me most active
10	The application that held my attention
11	The application that made me contribute the most

Table 8.3: *The application comparison*

The application comparison is related to the educational benefits of using the applications in a classroom setting. They are based on the work done in the articles *EGameFlow: A scale to measure learners' enjoyment of e-learning games* by Fu et al. and *Improvement of a Lecture Game Concept* by Wu and Wang [34, 48]. These articles describe the value of using the immersion and fun inherent in video games within a serious application field. Fu et al. introduces several factors that contribute to users feel of enjoyment while using the application [34]. The factors can be seen in Table 8.4.

<i>EGameFlow factors</i>	<i>Factor description</i>
1: Concentration	The application should keep the users attention without stress and information overload
2: Goal clarity	The tasks to be done should be known and be explained at the beginning
3: Feedback	The user should know his/her knowledge level and what is required for the ultimate completion of tasks
4: Challenge	The application should challenge the knowledge and skills of the user
5: Autonomy	The user should be encouraged to take initiative and have control of his/her actions in the application
6: Immersion	The user should be in a state of immersion while using the application
7: Social interaction	The application should become a tool for social interaction
8: Knowledge improvement	The application should improve the users knowledge in a relevant field

Table 8.4: *The eight different factors that EGameFlow uses to measure the users enjoyment of an educational application*

Each of these factors are measured by the use of several questions leads to a very detailed review of an application. Wu and Wang mention eight characteristics of good educational games [48]. These characteristics are listed in Table 8.5.

Table 8.4 and 8.5 gave inspiration for the questions regarding the educational prowess of the different applications. The user has to compare the different applications to each other in the application comparison questions, in order to force the user to prioritize one application over another.

<i>Concept</i>	<i>Description</i>
1: Variable instructional control	How the difficulty is adjustable or adjusts to the skills of the player
2: Presence of instructional support	The possibility to give the player hints when he or she is incapable of solving a task
3: Necessary external support	The need for use of external support
4: Inviting screen design	The feeling of playing a game and not operating a program
5: Practice strategy	The possibility to practice the game without affecting the users score or status
6: Sound instructional principles	How well the user is taught how to use and play the game
7: Concept credibility	Abstracting the theory or skills to maintain integrity of the instruction
8: Inspiring game concept	Making the game inspiring and fun

Table 8.5: *Eight good characteristics of an educational game*

8.1.5 Comments Section

To receive both positive and negative feedback the survey contained three comment sections, one for each of the applications PostIt, WordCloud, and Categories. The participants were stressed to write down everything they could think of regarding the applications in these three comment sections.

8.2 The First Experiment

The first experiment, containing an evaluation of the PostIt, WordCloud, and Categories applications, was conducted 12. March 2013 in a lecture with the subject TDT4240 Software Architecture at NTNU. The lecture theme was “Testing and Implementation” and consisted of problems surrounding software development testing. Attending the lecture was 30 students, mostly software engineering master students. All of the students experienced the applications for the first time. The student population was 93% male with an average age of 24 years.

The purpose of the experiment was to collect empirical data regarding the usability of the applications, as well as some technical considerations and an application comparison to investigate if lectures can benefit from using complementary educational applications.

8.2.1 Experimental Approach

The lecture was taught traditionally with the three applications merged into the lecture. First the students got some information about the subject “Testing and Implementation” within software architecture, and then PostIt was used for brainstorming with the premise “Write keywords about challenges related to implementation and architecture”. Figure 8.1 shows PostIt being utilized in the lecture.



Figure 8.1: *Brainstorming with PostIt*

After the PostIt session, another subject got reviewed and a new brainstorm session was started using WordCloud. The theme to brainstorm with the second time was “Write any keyword about how testing is related to software architecture”. Figure 8.2 presents the WordCloud GameWallView displayed with two projectors and two smaller screens.

To test the students’ knowledge of the course material, Categories got utilized



Figure 8.2: *Brainstorming with WordCloud*

at the end of the lecture. A various selection of expressions had to be placed into the correct category, as illustrated in Figure 8.3. These expressions were selected by the organizer from previously taught lectures within the same course, but they were not taught in the current lecture. This exercise was initially done individually, but the students were encouraged to help each other if they needed assistance.

After the lecture the participants had to fill out a survey. The survey and its questions are described in Section 8.1, and can be found in Appendix C.

8.2.2 Results

After analyzing the results, the range of devices was found significantly larger than the development environment. Samsung Galaxy SII/SIII, iPhone 4/5, Xperia Mini SK17i, HTC One X+, HTC Desire, Xperia Sola, Samsung Galaxy Note 2, and Huawei Honor was used. A total of 78% of the participant used a smartphone during the experiment, and the remaining 22% used a laptop. Almost everyone used Wi-Fi connectivity, except one that used 3G. The web browsers used was Opera, Chrome, Safari, and Firefox.

Some of the participants had problems filling out the survey correctly. Of all the participants, only 20 participants filled out the SUS part completely and 27

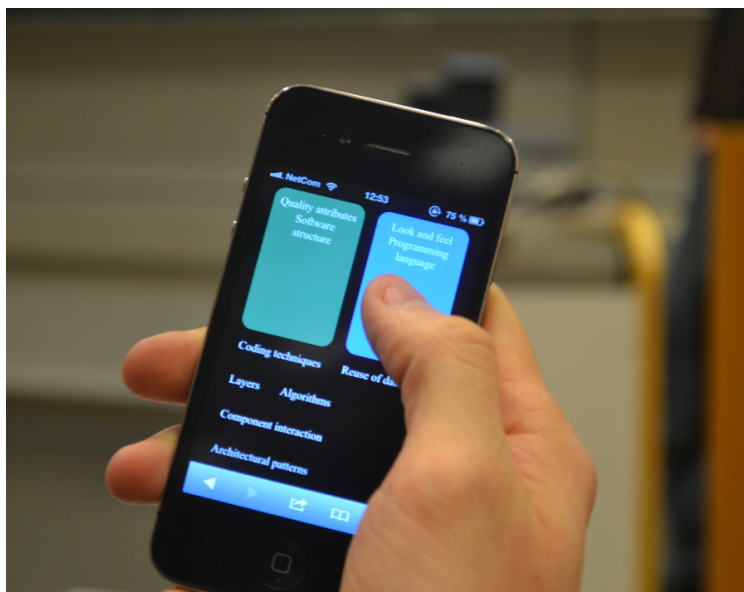


Figure 8.3: *Categories in action*

participants answered the technical considerations. Regarding the application comparison, 26 participants marked the form correctly.

SUS Results

The SUS score calculation is a straightforward process. For each question, the score is ranged from 0 to 4. Since every other question is asked negatively this means some difference in calculation. For every odd numbered question (1, 3, 5, 7, and 9) the score value is the scale position minus 1. The scale position is ranged from 1 (strongly disagree) to 5 (strongly agree). For the even questions (2, 4, 6, 8, and 10) the contribution is 5 minus the scale position. The overall value is obtained by multiplying the sum of scores by 2.5. This total score have a range from 0 to 100.

Table 8.6 shows each questions score distribution, and the SUS score as a whole. The different columns of interest are Avr, Var, and Score, respectively average, variance, and SUS score. The overall SUS score is calculated to 75.13. This is a fairly good result, which indicates that the applications are to some extent easy to use. Table 8.6 shows that question 2, 4, and 10 have a particularly high score.

These questions shows that the applications were simple and not very complex, the participants did not feel the need for support from a technical assistant, and the applications were easy to handle and easy to learn how to use. The users felt no a priori knowledge requirement to use the applications. None of the questions got a significant low score in total, and each of them are above average.

<i>SUS questions</i>		<i>Avr</i>	<i>Var</i>	<i>Score</i>
1	I think that I would like to use this system frequently	3.45	0.58	2.45
2	I found the system unnecessarily complex	1.60	0.46	3.40
3	I thought the system was easy to use	4.15	0.45	3.15
4	I think that I would need the support of a technical person to be able to use this system	1.45	0.58	3.55
5	I found the various functions in this system were well integrated	3.40	0.88	2.40
6	I thought there was too much inconsistency in this system	2.30	0.85	2.70
7	I would imagine that most people would learn to use this system very quickly	4.20	0.27	3.20
8	I found the system very cumbersome to use	2.00	0.53	3.00
9	I felt very confident using the system	3.70	0.75	2.70
10	I needed to learn a lot of things before I could get going with this system	1.50	0.58	3.50
Total score				75.13

Table 8.6: *SUS score distribution*

Technical Considerations Results

The technical considerations results are promising. The majority of participants found it easy to use the different applications in a web browser, they managed to interact with the system, and they did not experience noticeable network latency. A noticeable result was that 93% found it very satisfying that there was no installation procedure. This supports the assumptions and conclusions from the discussion about native versus web applications in Section 4.4. Table 8.7 shows the results from the technical considerations questions.

<i>Technical considerations</i>		<i>Disagree</i>	<i>Neutral</i>	<i>Agree</i>
1	I found it easy to setup and start using the applications through the web browser	11%	22%	67%
2	I did not manage to interact with the GameWall system with at least one web browser	78%	11%	11%
3	I found it satisfying that I did not have to install any of the applications on my smart-phone	0%	7%	93%
4	I found the applications unresponsive and/or with noticeable network latency	78%	11%	11%
5	I felt that the applications had consistent quality	8%	44%	48%

Table 8.7: *Technical considerations results*

Only one participant used 3G connectivity during the experiment. One participant is not enough to take a valid conclusion, but this result gave some indication and ideas for further exploration. This participant who interacted with the applications through 3G experienced noticeable network latency and found the applications unresponsive. Previous research supports this result and assumption, that 3G used as connectivity should give a much higher latency than Wi-Fi [25].

Another noticeable result was that 89% answered either strongly disagree, disagree, or neutral regarding if the participant did not manage to interact with the applications with at least one web browser. Why the remaining 11% did not manage to interact with the applications is unknown. A possible explanation could be that they did manage to interact with the applications, but that the interaction did not work as satisfying as expected. Another explanation could be that the QR code did not work on their device, or that the entry of URL was incorrectly typed.

Application Comparison Results

The participants found the WordCloud application most engaging and creative. PostIt scored best at being most fun to use, made them contribute most, and the application they would like to use most. Categories received the most decisive results regarding highest learning outcome, made them think the most, and was the most challenging. Both WordCloud and PostIt were used as individual exer-

cises during the lecture. An interesting observation is therefore why WordCloud got a significantly higher score regarding social interaction than PostIt. It also received almost as good results as Categories where the participants were stressed that they could help each other if needed. Figure 8.4 presents the results from the application comparison questions.

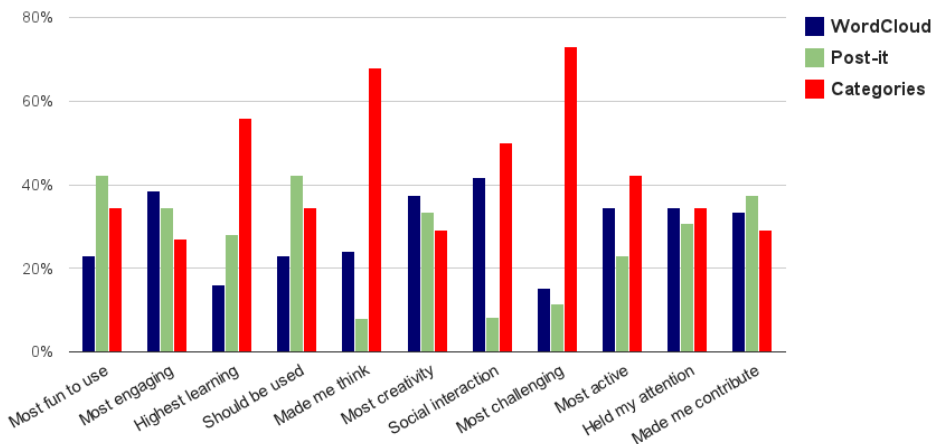


Figure 8.4: Application comparison results

8.2.3 User Comments

The participants gave both negative and positive written feedback. Some of the participants found the applications useless; that it was a distraction and waste of valuable lecturing time. Other participants found it educational and fun to use the applications, and that the applications created variety in an otherwise monotonous lecture. To prove exactly how effective and useful these kinds of applications will affect students regarding motivation and educational learning is left for a more in-depth study.

The experiment gained valuable knowledge of how users perceived the applications. Table 8.7 shows that the process of not having to install any additional software on the user device was warmly greeted. However, the result regarding setting up and starting to use the applications was disappointing. To reduce the number of steps of connecting to the application a QR code was displayed on the GameWallView, as illustrated in Figure 8.5.

One problem with using QR codes was that just a few of the students participating



Figure 8.5: A QR code scaled up on the GameWallView

the experiment had the required QR reader software on their device. This resulted in that most students had to type in the URL anyway, not taking advantage of the QR code available. The user experience is degraded if the applications need complicated configuration and installation procedures in order to interact with them. The effort of joining the collective application should be reduced to a minimum.

Two persons tried to connect to PostIts password protected GameWallApp during the experiment. Why they tried to connect will remain unknown, but the factor that an unauthorized person tried to connect to a password protected page tells you how important it is to always make sure that malicious users are kept away and that applications should be developed with security concerns in mind.

8.2.4 Organizer Experiences

Our supervisor was interviewed after the experiment to get his opinion on organizing and running the lecture using the FIGA applications. Our supervisor had several notable observations and comments regarding using the application in a pedagogical setting.

“The technical aspects were promising since it was easy to integrate the applications into the lecture. While I got assistance starting up the applications due to time constraints, I still feel confident that I could quickly learn using it without help. I have experience using command line tools, but I recommend creating easier ways of starting the ServerApps if the applications are to be targeted towards less technical adept persons. The use of QR-codes might be unnecessary, as I saw

few students using their mobile phone camera with a QR-reader.

Using the applications helped segment the lecture into several parts, each with a distinct theme with practical tasks at the end. As any educational tool, the usage and preparation are important to insure that there are any gains from using it. In this case, the application usage was not prepared well enough because of limited experience using them. The Categories application seemed to have too difficult questions. It was somewhat disappointing to see so many of the students struggle with the questions that were supposed to be easy. Knowing this would make it easier to plan ahead, but it did help me realize that the material was not as common knowledge as I thought. The brainstorming applications, PostIt and WordCloud, made it possible to stream the thoughts and associations of the students in real time. Some students did not take the assignment seriously and posted irrelevant information and jokes instead. It is difficult to avoid such problems when the applications need to be as accessible as possible, and it is highly dependent on a mature audience to maximise productivity.

Some changes need to be done before the next experiment. WordCloud should be used continuously, only showing the resulting word cloud after everyone is done sending their words. Categories should be tested in group assignments, so that the students helps each other with questions a single individual might find difficult. This also helps with the social aspects of the application. The applications are promising, as the students did seem to enjoy using them. I look forward to the next planned experiment.”

8.3 The Second Experiment

The second experiment was planned to verify and expand on the findings from the first experiment. The second experiment was held the 11. April 2013 in a lecture with the same subject and with many students with some experience from the previous experiment. Some minor tweaks were made on the applications, but their concepts stayed the same. In order to collect the user response on the applications the participants received a survey at the end of the lecture. The same survey was kept with some minor adjustments from the first experiment, to ease the comparison between the two experiments.

8.3.1 Experimental Approach

A major difference in the second experiment compared to the first experiment was how the educational applications were integrated into the lecture. The first

experiment was not well enough integrated in the lecture, meaning that it could feel rushed and the questions used in the Categories were too difficult to categorize. The theme of this lecture was the software technology “cloud computing” and the goal of the lecture was to introduce this concept to the students.

The lecture started with a brainstorming exercise using WordCloud. The students had to enter any associations related to the material being taught the following three next slides, as illustrated in Figure 8.6. This was to make the students more active and give the organizer input on their thoughts and associations with the information as it was presented.

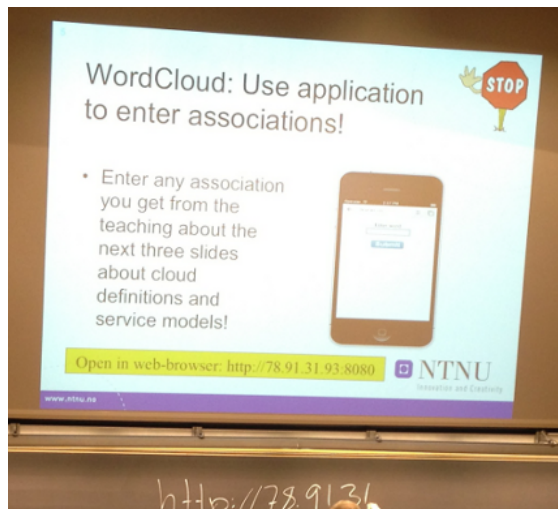


Figure 8.6: *WordCloud during the second experiment*

The exercise was conducted without showing the GameWallView, so the students did not see the result displayed on the wall. After three slides, the application was shut down. This was done in order to stop the incoming messages and to present the GameWallView to the class. Keeping the GameWallView hidden meant that it was possible to moderate the words sent from the students, in order to keep the displayed information on a serious and mature note, without them noticing it. The resulting GameWallView is presented in Figure 8.7.

After a short review of the WordCloud result, the students were to use the PostIt application in a group exercise. They gathered in groups and had to find the five most important issues related to economy and cloud computing. Figure 8.8 shows some of the issues that the students committed.

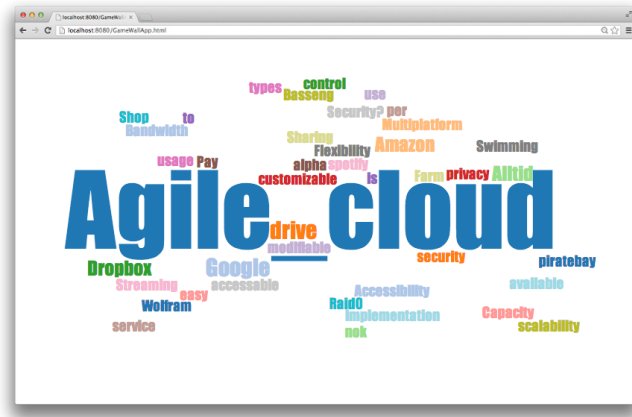


Figure 8.7: The WordCloud GameWallView result in the second experiment



Figure 8.8: The PostIt GameWallView during experiment

A few more slides related to cloud computing followed after the PostIt sequence, until the students were met with a Categories task as illustrated in Figure 8.9. The next exercise was to place different keywords into the correct categories.

Categories was utilized similar to how it was used in the first experiment, only that the keywords to be sorted had been taught in the current lecture. This meant that the students had the keywords and terms fresh in their minds when solving the exercise. The students first tried to complete the exercise individually, but it was stressed that they could help each other if needed.

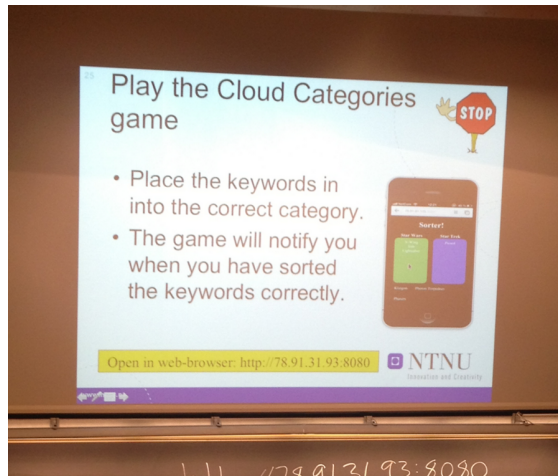


Figure 8.9: *The introductory slide for the Categories in the second experiment*

After the Categories test, the first half of the lecture was finished. In between the first and second half of the lecture was a fifteen-minute break, where the 1814 video game was started up. This meant that if they did not want a break, they could test the game.

As with the first experiment, the participants had to fill in a survey at the end of the lecture. The survey measured the same features as the survey from the first experiment, namely the usability, some technical considerations, and an application comparison. Due to time constraints, this form was delivered to the class only five minutes before the lecture was over, meaning that the majority of the students were quick to reply. This led to very few written comments from the participants.

1814 Game Session

The 1814 prototype was developed primarily to find out if a video game using FIGA was feasible, and as a stress test of the framework components. In order to collect the user experience playing 1814 the participants had to fill out a survey

after a ten minute play session. The game survey contained four sections, namely general information, technical user experience, gameplay and social interaction, and comments. The overall experiment therefore consisted of two individual surveys, the general FIGA survey regarding the educational applications and one survey regarding the 1814 video game.

The general information questions in the 1814 survey were identical to the general survey. These questions were used to get an understanding of the student and some information on which mobile device that was being used, as well as connectivity and web browser. The technical user experience questions were similar to the technical considerations in the general survey. The last question about the consistent quality was removed, but otherwise it was the same questions with some minor corrections. A complete overview of the technical user experience questions can be found in Appendix C.

In addition to the technical user experience questions the survey included some questions concerning the gameplay and social interactions. These questions complemented the technical user experience results, and were used to find out if the game was fun and the benefits of playing together with a large number of people in the same room. Even though 1814 is a video game and have other aspects and features, the classroom setting is well suited for testing a video game. Table 8.8 lists the questions concerning gameplay and social interaction.

<i>Gameplay and social interaction</i>	
1	The game was fun to play
2	The game was not engaging
3	I did not notice that the game was played in a web browser
4	Playing the game in the same room with other people made me less competitive
5	Playing the game in the same room with other people made it more fun
6	I would not like to play games this way (shared common screen w/browser controller) frequently
7	Playing the game in the same room with other people made me cooperate better (vs online play)

Table 8.8: *The gameplay and social interaction questions*

The gameplay and social interaction questions are answered on a scale from 1 to 5, where 1 is strongly disagree and 5 is strongly agree. Notice that the questions are alternated from a positive attitude to a negative one. This is inspired by the SUS form, trying to make the user reflect more carefully and take an opinionated stand.

At the end of the survey a comment box was included, where the participants could write their thoughts and experiences from playing the game. This was to get feedback written with their own words.

8.3.2 Results

A total of 30 technology students participated in the second experiment. Of these, 18 of the students participated in the first experiment as well, while the remaining 12 students were first-time users. The range of devices was similar to the first experiment, but this time two tablets were used as well, one iPad and one iPad 3. Of all the participants, 80% used a smartphone. The only connectivity used was Wi-Fi, and Safari, Chrome, Opera, and Firefox was the utilized web browsers.

As with the first experiment, some of the results were invalid due to incorrectly entries. 29 participants answered both the SUS and technical considerations questions correctly, and 26 participants filled out the application comparison. The following sections describe the results from the second experiment.

SUS Results

The total SUS score was a bit lower than in the first experiment with a score of 73.36. The applications got high scores on the simplicity of starting and using them. The participants felt confident using the applications and they felt that they did not need any technical assistance in order to start using them. They did however feel that there were inconsistencies and that some functions were not well integrated. One explanation for a reduced score might be the timing of when the students got their forms delivered to them. The original schedule was to end the lecture with a WordCloud brainstorm to review the material of the lecture, but it was cancelled due to time constraints. This meant that when the students answered the forms regarding the FIGA applications as a whole, with the experiences from 1814 most fresh in mind. The results can be seen in Table 8.9.

Technical Considerations Results

The technical aspects of the FIGA applications received excellent feedback, as can be seen in Table 8.10. The technical considerations give the impression of high

<i>SUS questions</i>		<i>Avr</i>	<i>Var</i>	<i>Score</i>
1	I think that I would like to use this system frequently	3.31	0.65	2.31
2	I found the system unnecessarily complex	1.97	0.25	3.03
3	I thought the system was easy to use	4.00	0.21	3.00
4	I think that I would need the support of a technical person to be able to use this system	1.38	0.32	3.62
5	I found the various functions in this system were well integrated	3.48	0.33	2.48
6	I thought there was too much inconsistency in this system	2.45	0.61	2.55
7	I would imagine that most people would learn to use this system very quickly	4.10	0.31	3.10
8	I found the system very cumbersome/awkward (in Norwegian: tungvint) to use	1.93	0.42	3.07
9	I felt very confident using the system	3.83	0.36	2.83
10	I needed to learn a lot of things before I could get going with this system	1.66	0.52	3.34
Total score				73.36

Table 8.9: *The SUS score distribution from the second experiment*

user satisfaction. The participants felt that the applications were easy to connect to and use, and the concept of web applications instead of native applications was positively received. The results of the technical considerations can be found in Table 8.10.

Application Comparison Results

The application comparison section of the feedback form made the participants decide which application was best suited in different circumstances. The circumstances was, to mention some, the application that was most fun to use or the application that initiated the most social interactions. The participants had to choose the best application, forcing the participant to prioritize. The feedback regarding the application comparison is summarized in Figure 8.10.

Figure 8.10 shows that Categories is superior in almost all of the questions. It was the most fun to use and the most engaging, most likely because of the activity it required from the user. Categories also was the application associated with

<i>Technical considerations</i>		<i>Disagree</i>	<i>Neutral</i>	<i>Agree</i>
1	I found it easy to setup and start using the applications through the web browser	3%	3%	94%
2	I did not manage to interact with the GameWall system with at least one web browser	87%	10%	3%
3	I found it satisfying that I did not have to install any of the applications on my smart-phone	0%	10%	90%
4	I found the applications unresponsive and/or with noticeable network latency	72%	21%	7%
5	I felt that the applications had consistent quality	3%	31%	66%

Table 8.10: *Technical considerations results from the second experiment*

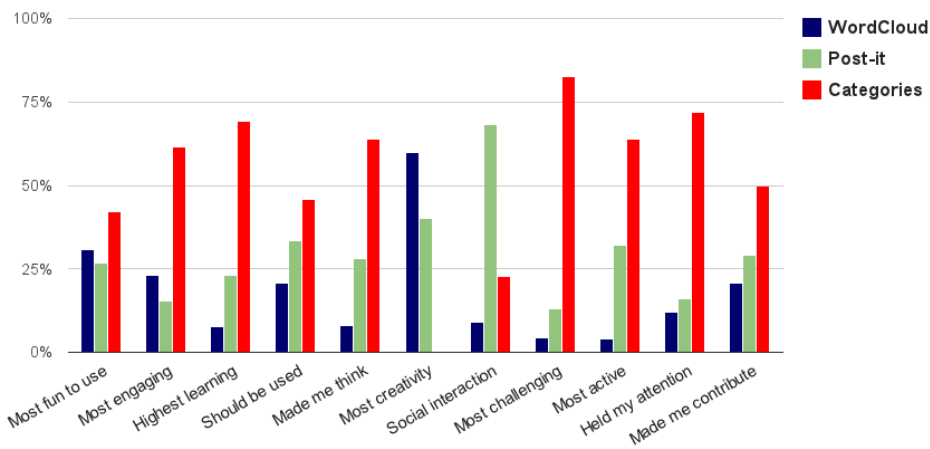


Figure 8.10: *Application comparison results from the second experiment*

the highest learning outcome and the application that made the user think the most, again probably related to the activity it requires from the user to sort the different keywords into the correct category.

WordCloud had a big impact on being the most creative application, followed by PostIt. This was expected, since the tasks associated with both these applications were brainstorming exercises.

The application initiating the most social interaction was the PostIt application.

The exercise that the application complemented was a group assignment. This means that the innate social aspects of the task affected the perception of the application and its use in a social setting.

A perplexing result is the question regarding the application that made the participant contribute the most. The Categories application was again chosen as the best candidate, but it is difficult to give an answer to why. One explanation is that the application made the participant help the person sitting next to him or her. However, WordCloud and PostIt were designed as contribution and brainstorming tools and should therefore have been a better candidate than Categories regarding this question.

8.3.3 User Comments

There was a severe lack of user comments to review after the second experiment. This is connected to time constraints at the end of the lecture, and resulted in that the participants were eager to leave the auditorium and go to their next lecture. This meant that most participants skipped the final comments section. In contrast, the comments section for the 1814 form was answered by almost everyone, probably since they had good time to write their reply. One alternative, which is highly unlikely, is that they did not have any comments regarding the FIGA applications.

8.3.4 Organizer Experiences

Our supervisor was interviewed after the experiment to get his opinion on how the applications worked with his lecture. The following statements describe his experience of organizing the applications.

“The second experiment was very different from the first one. Not that the applications were any different, but the way they were used, the context, was different. The applications were better integrated into the lecture, with the exercises better understood and each exercise produced more replies from the students.

WordCloud got a really good response, with the students paying attention from the very beginning. They seemed more engaged, and sent a lot of words in to the word cloud.

PostIt had a better exercise this time around, with the students getting together in groups. The resulting input from the students was more mature, and the amount of input was much better than the previous experiment. One issue was

the amount of text one could write on a single post-it. This resulted in some unclear post-its, that no one wanted to clarify when I asked out loud if the group could clarify what they meant.

Categories made the students very concentrated, and I feel like fewer students gave up this time than the previous experiment. This might be related to the difficulty of the task. This time the content in the exercise was related to newly lectured material, where the last experiment used material that was general for the entire course.

The last point I would like to make is the importance of clarity in the questions and exercises. Be sure that the questions you want the students to answer is understood by all, before starting the exercise to avoid misunderstandings.”

8.3.5 1814 Results

To push the limits and test the performance of a FIGA application, the 1814 game session was started in the classroom with over 30 students playing simultaneously. This game session was held in the break, in between the first and second part of the lecture. The ServerApp hosting machine was the same as the one used for the educational prototypes. This meant that the participants only had to refresh their web browser to start playing the game. The game got chaotic right away, and people started to shout, laugh, and scream while playing. Figure 8.11 illustrates the GameWallView some minutes into the game session.

Technical User Experience Results

Table 8.11 shows the result from the questions regarding technical user experience.

<i>Technical user experience</i>		<i>Disagree</i>	<i>Neutral</i>	<i>Agree</i>
1	I found it easy to setup and start using the game through the web browser	4%	4%	92%
2	I did not manage to interact with the game with at least one web browser	59%	33%	8%
3	I found it satisfying that I did not have to install the game on my smartphone	0%	13%	87%
4	I found the game unresponsive and/or with noticeable network latency	20%	21%	59%

Table 8.11: *The technical user experience results*

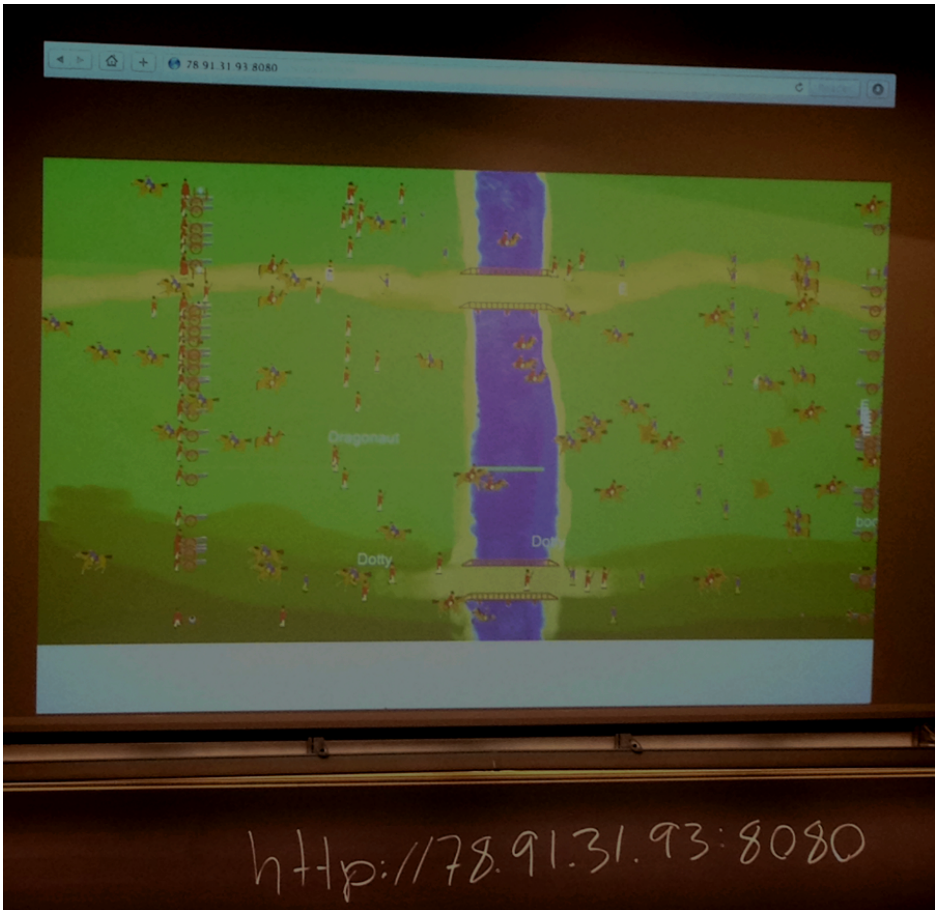


Figure 8.11: *1814 played in a classroom*

The technical user experience results give the impression of an application that is easy to setup and interact with, and that it was satisfying not having to install a native application. The last question regarding unresponsiveness and network latency did receive some inconsistent answers. The high number of players and a graphical bug made it hard to understand if a player was dead or alive. The GameWallView was quickly cluttered with images of game characters that should have been removed. The game was not tested with as many users during development, making it hard to predict the issues experienced at this experiment.

The technical issue was not related to graphical or logical computation load since the GameWallView was able to draw the graphics at the recommended rate of 30 frames per second. The characters, projectiles, and game time progresses smoothly.

The technical user experience questions are almost the same as the questions regarding the technical considerations as in the primary FIGA survey. Nearly every participant answered agree or strongly agree regarding question 1 and 3, respectively if they found it easy to setup and start using the game through a web browser and that they found it satisfying not having to install the game on their device. This corresponds to the technical considerations results, described in Section 8.3.2.

The question regarding responsiveness and network latency is the most concerning one. A total of 59% answered that they found the game unresponsive and/or with network latency. This is a higher percent than expected. One explanation could be that they had problems with finding their character on the GameWallView, and therefore did not notice the movement and actions they performed. Another likely explanation is that the game is not optimized well enough, with room for improvements. Clever algorithms and design patterns can be used to lessen the computational load of the ServerApp. The game was developed with a short development time and with little video game implementation experience. Modern video games have a high expectancy of running smoothly and small flaws will severely degrade the user experience. The combination of the network load and a technical error might be the cause of the poor results in the fourth technical user experience question, since the other educational applications ran smoothly without any issues. The network load issue might be that the ServerApp can not handle the amount of requests and replies necessary for a smooth experience. The technical issue is most likely an identification mapping error, where the GameWallApp creates so called ghost instances that are not animated and removed correctly. This should be further addressed for future improvements.

Gameplay and Social Interaction Results

A total of 24 participants answered the gameplay and social interaction questions. Table 8.12 shows that a slight majority agreed on if the game was fun to play. The result regarding the engagement shows no clear answer in either direction. The result is evenly distributed from disagree to agree, with a minor majority favouring disagree. Table 8.12 shows the gameplay and social interaction results.

<i>Gameplay and social interaction</i>		<i>Disagree</i>	<i>Neutral</i>	<i>Agree</i>
1	The game was fun to play	29%	29%	42%
2	The game was not engaging	42%	25%	33%
3	I did not notice that the game was played in a web browser	55%	33%	12%
4	Playing the game in the same room with other people made me less competitive	75%	21%	4%
5	Playing the game in the same room with other people made it more fun	0%	12%	88%
6	I would not like to play games this way (shared common screen w/browser controller) frequently	42%	21%	37%
7	Playing the game in the same room with other people made me cooperate better (vs online play)	12%	63%	25%

Table 8.12: *The gameplay and social interaction results*

A total of 55% answered strongly disagree or disagree that they did not notice that the game was played in a web browser. Question six, I would not like to play games this way frequently, does not provide a clear result, with an average on neutral and a small tendency towards disagree. These two results combined does not give a definitive answer on the prospects of playing games through a web browser.

Question five, playing the game in the same room with other people made it more fun, is worth pointing out. A total of 88% agreed or strongly agreed to this question, while the remaining 13% was neutral. Table 8.12 shows this compelling result.

Even if the game was not that fun to play and got mixed results regarding its engagement, the results shows that the concept and idea of playing a game in the same room with other people on a shared common screen is fun. The results from question four, playing the game in the same room with other people made me less competitive, supports this statement with 75% disagreement.

Another observation of the gameplay and social interaction results is that the variance on most questions is approximately or more than one, except question five. An explanation of the high variance could be the low number of participants. More participants might reduce this variance and give a more decisive result on the different questions.

8.3.6 Comments About 1814

A frequently repeated comment was the problem of finding the character on the GameWallView. The user got little feedback from their device about the game state. They did not get any feedback if their characters attack hit someone else or if they got hit themselves. Most of the players did not use or understand the use of the information button to help them locate their character on the screen. The game was supposed to play an animation when players got killed, but this animation did not work properly during the game session. Instead the characters remained on the screen, making it impossible to know if the character was dead or alive.

None of the participants had played the original game which 1814 is based upon, making the rules and gameplay difficult to grasp. It is hard to jump right into a new game concept without much explanation beforehand, especially when 30 students play the game simultaneously and can not find their character on the GameWallView. The game concept should therefore have been better explained before starting the session. The participants thought however that the game concept was interesting and that the shared screen concept had potential.

Chapter 9

Evaluation

The collected technical experiences from implementing the prototypes and the user experiences from the experiments forms an evaluation platform. These experiences forms two viewpoints to evaluate the success of creating a framework with value.

This chapter is divided into three parts. *First*, a discussion and experiences of creating applications and a framework in parallel. *Second*, a discussion on organizing user experience experiments and analyzing the results gained from them. *Third* and final, a retrospective on the completion of functional and quality requirements, in order to measure the success of the technical implementation of FIGA.

9.1 Prototype Summary

The four prototypes gave insight on how easy it is to create applications using FIGA, and the inherent quality they gain from using it. This summary describes the overall lessons learned from developing the applications.

9.1.1 Development Methodology

FIGA made it possible to create prototypes in a matter of hours. Native applications of the same complexity have an estimated development time of five to six days based on earlier project experiences. Using the bareboned components

without FIGA is estimated to twice the development time used in implementing the prototypes.

The applications quickly matured with constant updates and bug fixes as they were developed. The possibility of quick iteration cycles makes it possible to center the development around feedback from users without wasting time implementing unwanted functionality.

9.1.2 Application Functionality

FIGA made it easy to create different web applications, with different semantics and usages. The first three prototypes are event-driven applications, where the ServerApp plays the role of an intermediary between users, only updating upon receiving messages. The 1814 application is a real-time video game, where the state of the game has to be updated constantly with updates sent to every connected GameWallApp. The ability to create different applications makes FIGA flexible and modifiable.

Functionality such as security was added to the PostIt application, when discovering that administrative views of the applications could give unwanted access to core functionality to a more technical adept user. This additional functionality was not added to FIGA, due to limited development time and insufficient implementation experience.

9.1.3 The Joys of Web Development

Web based technology is among the most documented technologies to develop with. Simple Internet searches makes it possible to attain a myriad of examples and approaches to different problems. HTML5 and JavaScript makes it possible to quickly prototype a potential web application that can be iterated upon. This means that it is possible to quickly get user feedback on the look and feel of a web application.

Developing web applications is simplified with the incredible amount of scripting libraries and tools already available. This reduces development time and gives developers mature libraries and tools to work with. Examples of JavaScript libraries used in this project are the rendering engine D3.js used by WordCloud, and the animation system provided by Easel.js for the video game 1814. These libraries made it possible to use time on creating the overall applications, instead of using time reinventing readily available technology. These libraries are easy to integrate with FIGA.

An initial good indication regarding performance was that every prototype performed well in the development environment. One concern was that the client-side scripting libraries would not be able to handle complex computations without slowdown. However, even the video game 1814, the most computationally demanding prototype, ran seemingly without any troubles during implementation in the development environment.

9.1.4 The Pains of Web Development

Different web browsers and hardware devices make it hard to develop a web application with consistent quality. Web browsers implement different interpretations of HTML5 and JavaScript, which mean that the same source code leads to different results. Various hardware devices changes what can be expected to run without slowdowns, and at what resolutions the web application can be displayed correctly.

Developing prototypes where the look and feel matches a native application is difficult and time consuming. A significant number of work hours went into trying to hide default web browser functionality in order to provide the illusion of a native application. Different JavaScript libraries provide functionality that make it possible to animate the different components in a web page, but they do not perform as well in all browsers, degrading the user experience.

9.1.5 Scaling the Prototypes

The prototypes got tested as they were implemented, but it was difficult to do large scale testing with thirty or more users. This made it difficult to safely conclude that certain parts of the applications were made soundly. This is particularly true in the case of the video game 1814 where the functionality became complex, and various bugs surfaced the moment the ServerApp load increased.

9.2 Experiment Summary

This section summarizes and compares the results found in the experiments. These results can be used in future iterations on the concept, in order to expand on the FIGA concept.

9.2.1 Experimental Approach

The experimental approach for the two different experiments was almost identical. The lectures were taught in a traditional fashion with the applications merged into the lectures. The main difference between the two experiments was the usage of WordCloud and PostIt. The students saw the WordCloud GameWallView and PostIt GameWallView update in real time during the first experiment, while it was hidden until all the words and thoughts was collected in the second experiment. Hiding the GameWallView made it possible to moderate the incoming text, removing distracting and immature input. PostIt was also used as a group exercise in the second experiment, contrary to an individual exercise in the first experiment.

Categories was the last application to be tested in both experiments. In the first experiment, the Categories questions regarded general software architecture questions, while in the second experiment the questions regarded material presented in the same lecture. After the completion of the lectures the students filled out a survey. The surveys were nearly identical in both experiments, to make it easier to compare the results.

9.2.2 Results

Few participants may not give a representative result, as the margin of error can be critical when the sample size is low. However, the results from the experiments give good insight into the user experience of technology students. More experiments should be conducted to map the interest of other target groups.

The first and second experiment consists of respectively 20 and 29 answers regarding the SUS questions. Table 9.1 shows that the first experiment had a slightly higher SUS score on question 1, 2, 3, 6, 7, and 10, while the second experiment had a higher score than the first experiment regarding the remaining questions, namely question 4, 5, 8, and 9. The total SUS score is almost identical from the two experiment results, meaning that the usability is close to equal.

The comparison between the technical consideration results from the two experiments is presented in Table 9.2 with average and variance for each of the questions. A total of 27 participants answered the technical considerations questions in the first experiment, while 29 answered them in the second experiment. Table 9.2 shows that there is a significant positive increase from the first to the second experiment regarding if the participants found it easy to setup and start

<i>SUS questions</i>		<i>Test 1</i>		<i>Test 2</i>	
		<i>Avr</i>	<i>Score</i>	<i>Avr</i>	<i>Score</i>
1	I think that I would like to use this system frequently	3.45	2.45	3.31	2.31
2	I found the system unnecessarily complex	1.6	3.4	1.97	3.03
3	I thought the system was easy to use	4.15	3.15	4.00	3.00
4	I think that I would need the support of a technical person to be able to use this system	1.45	3.55	1.38	3.62
5	I found the various functions in this system were well integrated	3.4	2.4	3.48	2.48
6	I thought there was too much inconsistency in this system	2.3	2.7	2.45	2.55
7	I would imagine that most people would learn to use this system very quickly	4.2	3.2	4.10	3.10
8	I found the system very cumbersome to use	2.0	3.0	1.93	3.07
9	I felt very confident using the system	3.7	2.7	3.83	2.83
10	I needed to learn a lot of things before I could get going with this system	1.5	3.5	1.66	3.34
Total score		75.13		73.36	

Table 9.1: *SUS score comparison*

using the applications through a web browser. It also shows that the two experiments were otherwise similar regarding the remaining four questions. However, one noticeable observation is that the answers had a consistently lower variance in the second experiment compared to the first.

A major difference between the two experiments is the application comparison results. As mentioned in the experimental approaches, the two lectures had a minor different approach regarding the integration of the various applications. If this was the cause behind the major difference found in the application comparison results is unknown. WordCloud received the highest score in the first experiment regarding engagement and creativity, while it only received the highest score concerning the creativity in the second experiment. PostIt received the highest score concerning most fun to use, should be used in classes, and contribution in the first experiment, while in the second experiment it only scored highest on initiating social interaction. Categories received a significant higher score on

<i>Technical considerations</i>		<i>Test 1</i>		<i>Test 2</i>	
		<i>Avr</i>	<i>Var</i>	<i>Avr</i>	<i>Var</i>
1	I found it easy to setup and start using the applications through the web browser	3.74	1.05	4.17	0.65
2	I did not manage to interact with the Game-Wall system with at least one web browser	1.78	1.33	1.79	0.81
3	I found it satisfying that I did not have to install any of the applications on my smartphone	4.41	0.40	4.21	0.38
4	I found the applications unresponsive and/or with noticeable network latency	1.93	0.99	2.03	0.82
5	I felt that the applications had consistent quality	3.48	0.80	3.72	0.49

Table 9.2: *Technical considerations comparison*

most questions in the second experiment, compared to the first.

Figure 9.1 presents the WordCloud results from the two experiments matched against each other. As Figure 9.1 shows, the first experiment had a noticeable better result concerning engagement, making the participant think, social interaction, challenge, activity, attention, and contribution. The second experiment has a higher score than the first experiment regarding creativity and has a slightly better result regarding if it was fun to use. It is hard to explain why WordCloud received higher scores in total in the first experiment compared to the second, but a hypothesis is the importance of integration of the application in the lecture.

PostIt had a significantly higher score in the second experiment compared to the first regarding if it made the participants think and made them initiate social interaction. The first experiment received a higher score regarding its scale of fun and engagement. In addition, the question about holding the attention also got a higher score in the first experiment compared to the second. Otherwise the two experiments got approximately the same result regarding the remaining questions. These results are illustrated in Figure 9.2.

Categories was used by the students in the exact same way in the two experiments, with the questions asked the only difference. The second experiment used questions that the students found easier to answer, as they originated from material presented earlier in the same lecture rather than general software architecture material previously taught. While this difference makes the questions harder to answer, one would assume that harder questions does not degrade the usability of the application. Figure 9.3 shows that this is not the case, where

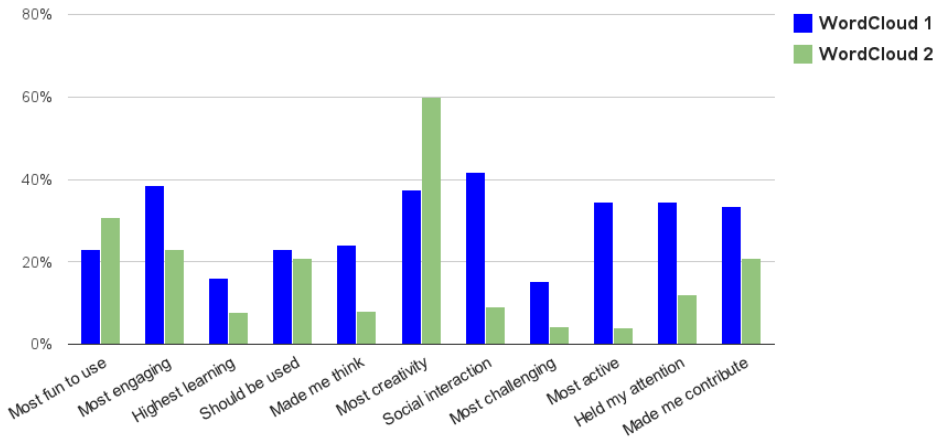


Figure 9.1: WordCloud comparison from the two experiments

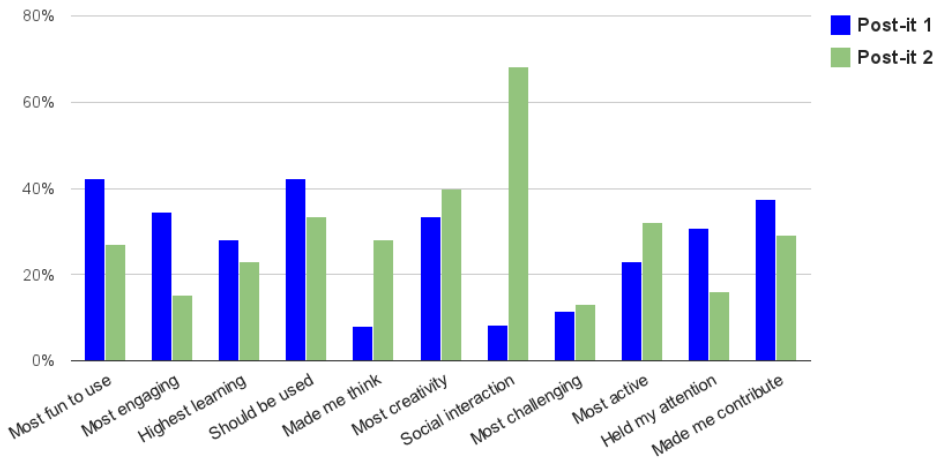


Figure 9.2: PostIt comparison from the two experiments

the questions asked are paramount in how much potential a student sees in the application. The first experiment got a slightly higher score regarding if it made the participants think, and a significantly better result about creativity and social interaction. Otherwise the first experiment received a lower score compared to the second experiment regarding the remaining questions.

One interesting observation from the results was how the participants' opinions

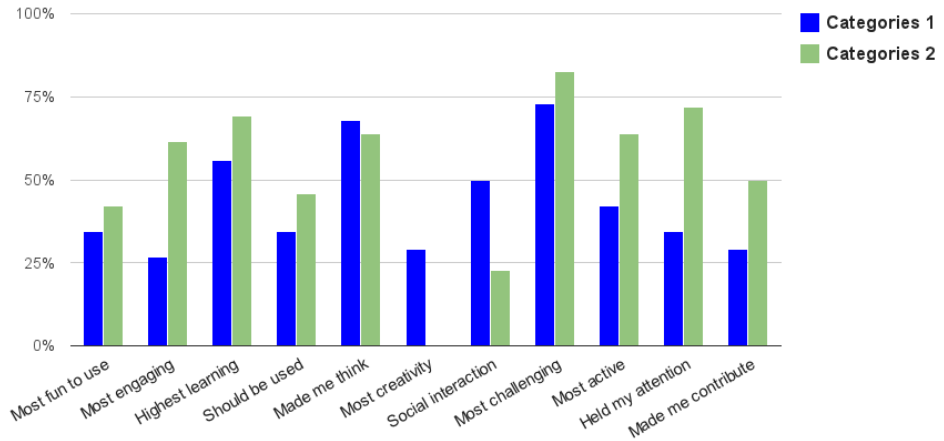


Figure 9.3: *Categories comparison from the two experiments*

changed from the first to the second experiment. Another evident observation was the major difference between first-time users in the two experiments. The survey used in the second experiment included an additional field among the general information, namely “I participated on the previous experiment conducted March 12th 2013 (yes/no)”. This allowed for the results to be divided into two separate groups for further analysis.

Figure 9.4 presents the comparison between the results from the second experiment and the results from the first experiment, where the participants answered “yes” regarding the additional question mentioned above in the second experiment. In other words, a comparison between the results only from the users who participated in both experiments. If a bar in Figure 9.4 has a positive value it means that a higher percent of students preferred it in the second experiment, and vice versa.

There are little to no difference in their answers regarding the questions about learning, usage, challenge, and contribution. However, it is interesting to see how different their opinions are on the remaining questions. The participants found WordCloud significantly more fun in the second experiment compared to the first, while Categories and PostIt were more fun in the first experiment. They also thought that Categories was more engaging in the second experiment, while WordCloud and PostIt was the most engaging applications in the first experiment. The participants answered that WordCloud and Categories were the two applications that made them think in the first experiment, while PostIt demanded most thinking in the second experiment. The participants felt that WordCloud made

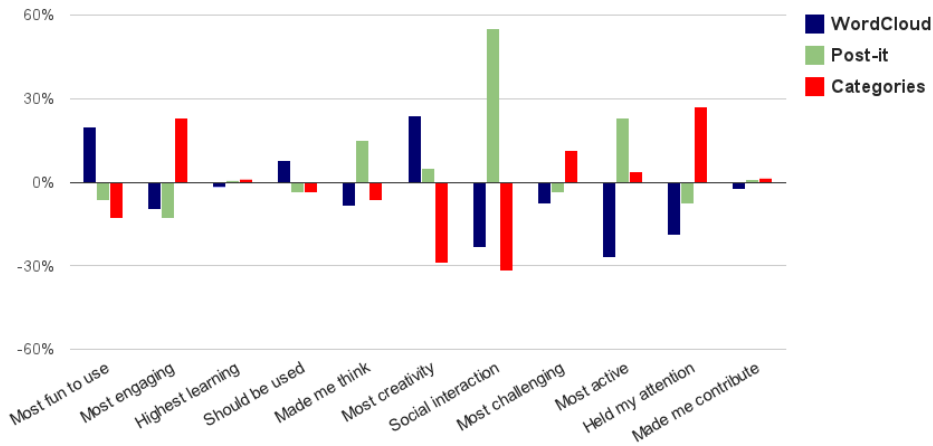


Figure 9.4: *The second experiment compared to the first experiment with only the participants who participated in both experiments*

themselves much more creative in the second experiment compared to the first, and that Categories demanded no creativity in the second experiment compared to the first.

The most convincing result is the question about social interaction. During the second experiment the students used WordCloud individually while the lecture was taught, and when PostIt was used they were put together in groups. This differed from the first experiment where the students were supposed to contribute individually in both applications. Categories was used in the same way in both experiments, where the task was first to be solved alone and then the students could help each other complete the task. As Figure 9.4 shows, PostIt is superior regarding social interaction in the second experiment compared to the first.

The participants answered that PostIt made them most active in the second experiment, compared to WordCloud in the first experiment. They also answered that WordCloud held their attention the most in the first experiment, while Categories held their attention the most in the second experiment.

Figure 9.5 presents a comparison between the results from the second experiment and the results from the first experiment, where only the first-impressions are included and the answers of students in the second experiment that had participated in the first experiment are excluded.

Figure 9.5 gives the impression that the first-time participants from the second experiment favoured Categories in almost every way, except in terms of how the

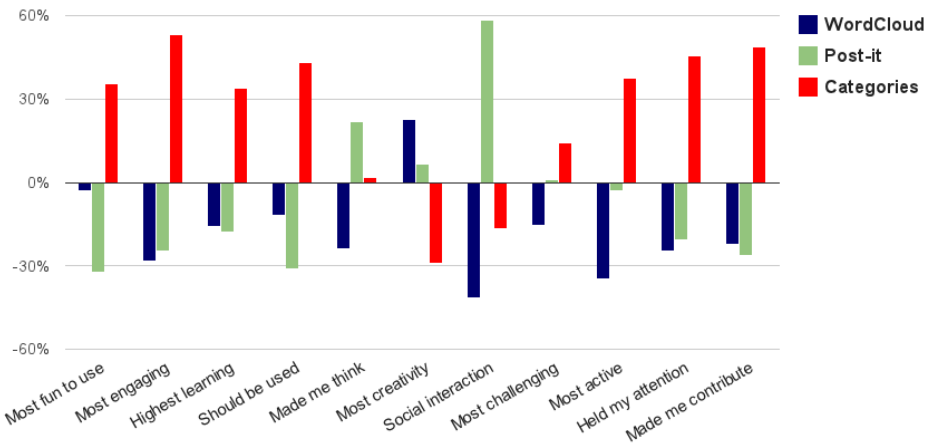


Figure 9.5: *The second experiment compared to the first experiment with only first-time participants*

applications made them think creatively and initiated social interaction.

9.2.3 User Comments

The participants gave more written feedback in the first experiment compared to the second. Some participants in the first experiment found the applications useless, while others found it educational and fun to use. A recurring comment from both experiments was the usage of QR codes for easier connectivity. Only a few participants had installed QR scanner software on their device, making it useless for the remaining students. While the use of QR codes was completely optional, the ones who did not use it found it unnecessary.

There were some unauthorized students that tried to connect to the password protected GameWallApp in both experiments. The first experiment had three students trying to log in to the password protected GameWallApp. The second experiment had one student trying to do the same. Adding a password protection before gaining access to the functionality caught this behaviour. In addition, the IP address of the attempts to connect with wrong passwords was logged and a warning appeared in the application console.

9.2.4 Organizer Experiences

Our supervisor, and the organizer of the two experiments, was pleased with the results gained from the two experiments. Since the results from the surveys had deviations, much attention was directed towards the use of the applications rather the applications themselves. The second experiment had a significant better integration with regards to the course material, giving the application use a better sense of purpose.

If the use of the applications are well thought out, then the use of the educational applications makes it easier to collect input from the students in the classroom since they are more willing to share information anonymously. The effect of being anonymous gave the more immature students a way of broadcasting their distracting comments, but they are a biproduct of an easy to connect application. The organizer can, and did, moderate the results and reduced the volume of immature content.

9.3 Functional and Quality Requirements

The functional requirements were chosen to ensure that FIGA fulfilled technical expectations as a development aid. The listed requirements described in Table 5.1 in Chapter 5 are the most important features that are inherent in FIGA. FIGA will insure a technical standard for all applications since these requirements are fulfilled.

9.3.1 Functional Requirements

The development and implementation of the four prototypes, in addition with the two conducted experiments, shows that all the functional requirements are tested with positive results.

FR1 Supporting shared screen applications - Success

- All of the prototypes were implemented using the concept of a shared screen. They all supported multiple GameWallApps and UserApps.

FR2 Web browser support - Success

- In every prototype, the UserApp is a web application that is executed in a web browser. The results from the conducted experiments shows

that the majority of participants managed to interact with the applications with at least one arbitrary web browser. The different prototypes use different ways of displaying information and user interaction.

FR3 Modularity - Success

- FIGA is based on web technology and can import existing JavaScript libraries such as D3.js, Easel.js, and jQuery. These libraries contain different, industry proven, functionality that reduces the development time and ease the implementation.

FR4 Network simplifications - Success

- Simplifying the networking process was achieved through categorizing the UserApps and GameWallApps together and creating wrapper functions for the messaging. This adds a layer on the low-level functionality, making it easier to handle communication between the ServerApp and all of the UserApps and GameWallApps.

FR5 Application types - Success

- FIGA must support different types of applications, both real-time and event-driven. Both types of applications were successfully implemented, with PostIt, WordCloud, and Categories being event-driven applications and the video game 1814 being a real-time application.

9.3.2 Quality Requirements

The quality requirements were chosen to enhance the quality of the applications, and to ensure a higher user satisfaction. The quality requirements for FIGA are listed in Table 5.2 in Chapter 5.

NFR1 Persistent connections - Success

- In order to provide stable service, persistent connections should be supported. This functionality was implemented and is used in the video game 1814. If the user refreshes the web page, the ServerApp remembers the previous information and does not create a new player instance.

NFR2 Password protection - Partial success

- The PostIt GameWallApp uses a password protection system, but it is application specific. This means that FIGA itself does not sup-

ply a password protection system, but PostIt shows how easy it is to implement rudimentary protection that prevents simple attacks.

NFR3 Look and feel - Success

- The experiment participants felt that the prototypes had consistent quality. The second experiment had 66% of the participants agreeing to this statement. The majority, 90% of the participants, also agreed in the second experiment that it was satisfying not having to install the applications. Quality look and feel of applications requires significant amount of development time, but the conducted experiments shows that web applications can reach satisfactory levels of perceived quality even with limited development time.

NFR4 Responsiveness - Partial success

- A total of 78% of the participants in the first experiment, and 72% in the second experiment, disagreed to that the three educational prototypes were unresponsive and/or had noticeable network latency. These results are just short of the quality requirement goal of 80%. However, if the participants who answered neutral are included, giving the first experiment 89% and the second 93%, then the responsiveness requirement is accomplished.

NFR5 Platform support - Success

- The ServerApps of all the four prototypes can be executed on the required Windows, OSX, and Linux platforms, granted that Node.js and Socket.IO are installed on the ServerApp device.

Part IV

Summary

Chapter 10

Conclusion

The project started with five research questions, stated in Chapter 2. *First*, how much web application development time could be reduced by supplying a suited framework was to be explored. *Second*, a various selection of application prototypes were developed to discover what types of applications that could be made with FIGA. *Third*, locating the performance and limitations of FIGA applications. *Fourth*, what level of usability could be achieved in FIGA applications. *Fifth* and last, finding if there was an increase in perceived learning by using educational applications made with FIGA.

Throughout the project the research methods described in Section 2.2 have been followed to the letter. Four different prototypes have been developed, three educational and event-driven applications and one real-time video game, using FIGA. These prototypes helped form and improve FIGA. The prototypes have also been tested and evaluated in two different lectures, which led to an evaluation of FIGA since all the prototypes are built using it.

This chapter summarizes the projects findings and results by answering the research questions and problem definition. The content of Part II and Part III is mainly the source and basis for the results. The project started with the following problem definition; “*The goal of this project is to design, implement, and evaluate a framework that makes it easy to build multimodal web applications for large displays with multiple users. The framework will use cross platform web technology for creation of simple games or interactive applications. Prototypes will be created in order to evaluate the framework and investigate the potential learning benefits in an educational setting.*” Based on this problem definition the project as a whole is considered as successful.

10.1 RQ1 - Reducing Development Time

RQ1 was concerned with the amount of web application development time that could be reduced by supplying a suited framework. To ease the development of web applications, FIGA delivers a foundation with the most commonly needed functionality for multimodal web applications with a shared common screen. The scale of success is measured by combining the reduced development time with the user satisfaction using them.

In terms of development time, the three event-driven applications were made in approximately one day in terms of working hours. Without the use of FIGA, the development time is estimated to at least two days. This difference becomes even more evident if one compares it with native application development. Own experiences and knowledge would estimate the development time to five or six days in order to develop similar native applications. However, these assumptions are not documented and it is recommended that this should be further researched in the future.

The developed applications received a decent SUS score in the two conducted experiments, and the participants were pleased with the usability of the different applications. Therefore, the conclusion is that FIGAs functionality makes it easier and faster to develop web applications with satisfying usability.

RQ1.1 What functionality should be provided with FIGA?

- FIGA provides the functionality of setting up a web application server and the necessary functionality needed for a `UserApp` or `GameWallApp` to connect to a `ServerApp`. The `ServerApp` inherits functionality required to initialize a listener for incoming connections. The `UserApps` and `GameWallApps` have functionality for connecting to the `ServerApp`. The `ServerApp` has the required functionality for storing, updating, and validating connections. See Chapter 6 for more implementation details.

10.2 RQ2 - Supporting Different Application Types

RQ2 regarded what kind of application types that could be made with FIGA. Section 4.3 concerns two application types, namely real-time and event-driven applications. The three educational prototypes were all event-driven, and the 1814 video game prototype was a real-time application. Thus, FIGA supports development of both event-driven and real-time applications.

RQ2.1 How easy is it to create an event-driven application with FIGA, and what are the benefits of using FIGA?

- The three educational application prototypes, PostIt, WordCloud, and Categories, were developed over a much shorter iteration cycle than the video game 1814. FIGA helped considerably in making sure that the behaviour of the applications was consistent. The functionality supported by FIGA made it simple to connect the different application components while application specific events and functionality could be defined individually.

RQ2.2 How easy is it to create a real-time application with FIGA, and what are the benefits of using FIGA?

- The main development time of the 1814 video game was close to a week in terms of working hours. This was the most complicated application of the four developed prototypes, pushing the limits of both the graphical and network module. FIGA helped considerably in making sure that every device could communicate with the application and play.

10.3 RQ3 - Locating Performance Bottlenecks

RQ3 was concerned with the performance of applications made with FIGA. The two conducted experiments got very satisfying results, and none of the educational prototypes used in the lectures did experience any major performance issues. This shows the great performance of using FIGA for event-driven applications. The feedback regarding those was mostly on the application usage and its user interface, none regarding responsiveness and network latency. The performance is more dependent on additional algorithms and functionality created by developers than the underlying FIGA.

RQ3.1 How many users can applications made with FIGA handle?

- In the two experiments conducted 12. March and 11. April 2013, described in Section 8.2 and Section 8.3, respectively 27 and 30 technology students participated. The educational and event-driven applications PostIt, WordCloud, and Categories, did not experience any delay or problems in these two experiments, so these kinds of applications made with FIGA have no problems with up to 30 users. The participants did experience some delay or problems during the 1814 video game session, but everyone managed to interact and play. The

ideal would be to gather hundreds of users and test the applications simultaneously, but this scenario is beyond the range of this project. The prestudy revealed that a similar project, Josh Software, had positive performance results using the same network components as FIGA and could scale up to hundreds of concurrent users [18].

RQ3.2 How will network latency affect the user experience?

- According to the experiment results in Chapter 8 experienced most of the students little to none network latency when using the educational and event-driven applications PostIt, WordCloud, and Categories. However, when using the video game 1814 the students changed their opinion regarding the latency. The 1814 game did not work properly during the experiment, and it is hard to conclude whether or not it was the game related problems that led to the feeling of network latency, or if it was the network latency that led to the feeling of unresponsiveness. Either way, this caused a degraded user experience.

10.4 RQ4 - Improving Usability

RQ4 regarded the achieved level of usability present in applications made with FIGA. The evaluation summary in Section 9.2 shows that the applications received an average SUS score of 73.3, a decent score. The participants did not answer conclusively if they would like to use the applications frequently, or if the various functions were well integrated in the applications. The application usage differed from conventional teaching methods and due to limited development time the various functions were not as well integrated in the different applications as wanted. These reasons may be the answers behind the low score regarding the questions, but it is difficult to prove that they are. The level of usability is still considered as satisfying and more than good enough for further development and improvements.

RQ4.1 Can FIGA applications reach the look and feel of native applications?

- The results in Section 8.3.5 presents that most students noticed that the applications were dynamic web pages. Although, every participant found it satisfying that they did not have to install any additional applications on their device. Almost every participant managed to interact with the applications with at least one web browser. It is difficult to achieve the look and feel of native applications with web applications to the full extent as of today, but different tools and

libraries makes it possible to reach satisfactory levels of positive user experience.

RQ4.2 How easy is it possible to make the setup and start using applications developed with FIGA?

- Section 9.2 presents that the participants of both experiments found it very satisfying that they did not have to install any of the applications on their device and found it easy to setup and start using the applications through a web browser. A single line in the terminal is the only requirement for starting a ServerApp. The organizer with no experience with FIGA applications prior to the experiments was confident that he could manage starting the applications without technical assistance.

10.5 RQ5 - Educational Applications and Learning Benefits

RQ5 was concerned with finding if there was an increase in perceived learning by using educational applications made with FIGA. The results from the experiments conducted gave good insight into what types of FIGA applications are most suited for an educational setting. In order to create good pedagogical applications, the presentation and integration of them should be carefully considered. Lecture Quiz, an existing system described in Chapter 3 with similar traits to FIGA and this project, is easy to integrate with traditional lectural methods [26]. The students that tried Lecture Quiz claimed it to increase learning and concentration during the lecture. Lecture Quiz can easily be compared and related to the educational applications developed with FIGA.

Application types in Section 4.3 described that recent experiences show that video games and other educational software can be effective and compelling context for learning [37, 43]. Using these video games and educational software within a classroom can be beneficial for academic achievement and motivation [44]. Based on these experiences, the experiment evaluation, and the Lecture Quiz study, an increase in perceived learning by using educational applications made with FIGA is possible. However, it will require further research and work to find the levels of increased learning and knowledge improvement.

RQ5.1 Which FIGA educational application concept returns highest perceived learning benefits?

- Categories received the highest scores in both experiments. The focus on problem solving and matching terms seems to be a good approach to make the students become active in an otherwise static lecture. Most lectures are monologues, where the students become a passive recipient of the lecturer.

RQ5.2 Which FIGA educational application concept cause increased social interaction?

- The students' perceived usefulness of the educational applications created with FIGA is heavily dependent on the preliminary work done by the lecturer, and how the applications are used in the lecture. The simple change from an individual to a group exercise has obvious effects on how it is perceived as a social interaction initiator. This change is seen from the first to the second experiment as described in Section 9.2, where PostIt was changed from an individual to a group exercise. In addition to making the students cooperate, having a group exercise made the responses more mature and less prone to malicious content.

RQ5.3 Which FIGA educational application concept aids a teacher in getting feedback from the students?

- From the detailed interview with our supervisor after the two experiments it was stated that the experience of organizing a lecture using the educational applications made him structure the lectures differently. Knowing that some of the material was to be reviewed in an exercise made him more attentive to how the material was presented. The organizer also felt that the feedback from the students was improved using the electronic brainstorming methods, compared to traditional pen-and-paper solutions.

Chapter 11

Further Work

There are several aspects of FIGA that needs further development and assessments. The functionality proposed in this chapter has either been out of scope of this project or been a serendipitous discovery while developing FIGA. In either case, this further work should improve upon this project or evaluate the usefulness of it.

The 1814 video game development resulted in several technical difficulties. The results found in Section 8.3.5 points to either limitations in the network module or an implementation bug that becomes evident under high network and computational load. A closer look on the behavior of the applications under high network and computational load should be conducted in order to better conclude if FIGA is suited for high performance real-time applications.

The three educational prototypes performed above expectations, without giving a clear indication on the maximum number of active users being able to interact with them simultaneously. To ensure the capacity of an even larger attending audience, this should be further explored. In addition to a technical evaluation, the educational prototypes open up a larger scope of pedagogical use of information technology in a classroom setting. This should be explored further to evaluate the potential of using educational applications developed with FIGA.

An evaluation of the usefulness of the complementary applications could be performed by having regular use of the applications. By conducting a quiz at the end of lectures using Categories would make the students reflect on what they have learned that lecture.

Since the development of FIGA only took place in-house, it lacks feedback from

external developers. The application prototypes were also developed in-house, where they followed the best practices of FIGA. The experience and knowledge of how FIGA should be used is not likely to be as obvious for external developers. In order to mature, FIGA should be used in third-party development to validate its qualities as intuitive and extendable.

The quality requirements of providing security and session control were both implemented, though limited. This functionality should be improved in order to provide a better service to the applications using FIGA. Security will be paramount in later development, especially in widespread applications with many participants. The experiments experienced several cases of immature behaviour with exhibitionistic tendencies, where the GameWallApp presents the opportunity to show off in front of the crowd. Adding a banning system could reduce this issue, where a user is tagged as inappropriate and exclude all subsequent input from that user. This could be implemented using the same approach as with the persistent connection system, using the IP address of the user as an identifier. This quality functionality should be improved upon in order to make FIGA more appealing.

There exists a great potential in the spontaneous nature of the applications made with FIGA and it shows the promise of localized web applications. This project shows the possibilities of connecting isolated technology platforms, where the trend in today's technology marked is to only cater oneself. Collaboration was one of the main motivations behind FIGA, and this project proves that not only is it possible that vastly different devices communicate effortlessly, but that people interacting together is both fun and improves learning. FIGA allows for rapid prototyping of a wide variety of application concepts, and the future will hopefully show exciting applications building on the foundations of FIGA.

Part V

Appendix

Appendix A

Test Environment

13" MacBook Pro

One of the test and development devices was a 13" Macbook Pro from early 2011 with a screen resolution of 1280 x 800 pixels. 2.3 GHz Intel Core i5 processor, 4 GB 1333 MHz DDR3 memory and Intel HD Graphics 3000 384 MB. Through the project it had Mac OS X version 10.8.2 installed, as well as several web browsers, such as Safari, Chrome, Opera, and Firefox.

15" MacBook Pro

Another test device was a 15" MacBook Pro from spring 2008. Screen resolution 1440 x 900. 2.5 GHz Intel Core 2 Duo processor and 2 GB 667 MHz DDR2 SDRAM. The GPU is a NVIDIA GeForce 8600M GT chip. Mac OSX 10.6.8. The following browsers are installed Safari, Opera, Chrome, and Firefox.

iPhone 4S

The iPhone 4S has a 3.5 inches retina display with multi-touch, and a resolution of 960 x 640 pixels at 326 ppi. 512 MB RAM, 32 GB memory and 2x Cortex A9 800 MHz processor. During the project the iPhone 4S had iOS 6 installed, along with the web browsers Safari and Chrome.

iPad 2

The iPad 2 has a 9.7 inches display, 4:3 aspect ratio at 132 ppi. It consists of 512 MB DDR2 RAM, 64 GB storage capacity, and 1 GHz dual-core ARM Cortex-A9 processor. Similar to the iPhone 4S, the iPad 2 also had iOS 6, Safari and Chrome installed.

Samsung Galaxy SII

Samsung Galaxy SII is an Android mobile phone with the resolution of 800 x 480 and a 1.2 GHz dual-core processor. The device used the native web browser Google Chrome Mobile and Firefox Mobile.

Appendix B

Experiment Evaluation

B.1 Technical Considerations Results from the First Experiment

<i>Technical considerations</i>		<i>Strongly disagree</i>	<i>Disagree</i>	<i>Neutral</i>	<i>Agree</i>	<i>Strongly agree</i>
1	I found it easy to setup and start using the applications through the web browser	4%	7%	22%	45%	22%
2	I did not manage to interact with the GameWall system with at least one web browser	59%	19%	11%	7%	4%
3	I found it satisfying that I did not have to install any of the applications on my smartphone	0%	0%	7%	44%	49%
4	I found the applications unresponsive and/or with noticeable network latency	41%	37%	11%	11%	0%
5	I felt that the applications had consistent quality	4%	4%	44%	37%	11%

Table B.1: *Technical considerations results*

B.2 Technical Considerations Results from the Second Experiment

<i>Technical considerations</i>		<i>Strongly disagree</i>	<i>Disagree</i>	<i>Neutral</i>	<i>Agree</i>	<i>Strongly agree</i>
1	I found it easy to setup and start using the applications through the web browser	3%	0%	3%	63%	31%
2	I did not manage to interact with the GameWall system with at least one web browser	41%	46%	10%	0%	3%
3	I found it satisfying that I did not have to install any of the applications on my smartphone	0%	0%	10%	59%	31%
4	I found the applications unresponsive and/or with noticeable network latency	31%	41%	21%	7%	0%
5	I felt that the applications had consistent quality	0%	3%	31%	56%	10%

Table B.2: *Technical considerations results from the second experiment*

B.3 Technical Considerations Results from 1814

<i>Technical user experience</i>		<i>Strongly dis-agree</i>	<i>Disagree</i>	<i>Neutral</i>	<i>Agree</i>	<i>Strongly agree</i>
1	I found it easy to setup and start using the game through the web browser	0%	4%	4%	50%	42%
2	I did not manage to interact with the game with at least one web browser	42%	17%	33%	4%	4%
3	I found it satisfying that I did not have to install the game on my smartphone	0%	0%	13%	58%	29%
4	I found the game unresponsive and/or with noticeable network latency	12%	8%	21%	46%	13%

Table B.3: *Technical user experience results from 1814*

B.4 Gameplay Results from 1814

<i>Gameplay and social interaction</i>		<i>Strongly dis-agree</i>	<i>Disagree</i>	<i>Neutral</i>	<i>Agree</i>	<i>Strongly agree</i>
1	The game was fun to play	8%	21%	29%	42%	0%
2	The game was not engaging	4%	38%	25%	33%	0%
3	I did not notice that the game was played in a web browser	13%	42%	33%	12%	0%
4	Playing the game in the same room with other people made me less competitive	25%	50%	21%	4%	0%
5	Playing the game in the same room with other people made it more fun	0%	0%	12%	75%	13%
6	I would not like to play games this way (shared common screen w/browser controller) frequently	4%	38%	21%	29%	8%
7	Playing the game in the same room with other people made me cooperate better (vs online play)	12%	0%	63%	25%	0%

Table B.4: *Gameplay and social interaction results from 1814*

Appendix C

Evaluation Survey

This chapter contains the different evaluation surveys used during the project, as well as the respective results.

First, the evaluation survey from the first experiment, followed by its results. Second, the evaluation survey from the second experiment, followed by its results. Third and last, the evaluation survey from the 1814 game session, followed by its results.

GameWall evaluation

General Information	
Age (eg. 24 years)	
Sex (male/female)	
Study programme (eg. MTDT)	
Mobile device (eg. iPhone 4S, Samsung Galaxy SIII, MacBook, Samsung Galaxy Tab)	
Web browser (eg. Opera, Safari, Firefox)	
Connectivity (eg. Edge, 3G, Wifi)	

The questions below on page 1 (SUS Questions and Technical Considerations) considers the overall concept of using a shared GameWall within a classroom. Use the overall impressions from all three applications combined and mark a single box of your choice.

	SUS Questions - GameWall System	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
1	I think that I would like to use this system frequently					
2	I found the system unnecessarily complex					
3	I thought the system was easy to use					
4	I think that I would need the support of a technical person to be able to use this system					
5	I found the various functions in this system were well integrated					
6	I thought there was too much inconsistency in this system					
7	I would imagine that most people would learn to use this system very quickly					
8	I found the system very cumbersome to use					
9	I felt very confident using the system					
10	I needed to learn a lot of things before I could get going with this system					

	Technical Considerations	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
1	I found it easy to setup and start using the applications through the web browser					
2	I did not manage to interact with the GameWall system with at least one web browser					
3	I found it satisfying that I did not have to install any of the applications on my smartphone					
4	I found the applications unresponsive and/or with noticeable network latency					
5	I felt that the applications had consistent quality					

The questions on page 2 (Application Comparison and Comments) considers each application individually. The answers will be used as feedback to improve each of the applications and find the most suited classroom application. Rate the applications from 1 to 3, where 1 is the worst and 3 is the best.

	Application Comparison	WordCloud	Post-it	Categories
1	The application that was the most fun to use			
2	The application that was the most engaging			
3	The application that I believe has the highest learning outcome			
4	The application that I would like to be used in most classes			
5	The application that made me think the most			
6	The application that demands the most creativity			
7	The application that made me initiate social interaction			
8	The application that was the most challenging			
9	The application that made me most active			
10	The application that held my attention			
11	The application that made me contribute the most			

Comments about WordCloud

Comments about Post-it

Comments about Categories

General information								Strongly disagree (1), disagree (2), neutral (3), agree (4), strongly agree (5)										Application comparison																
Nr	Age	Gender	Study programme	Device	Unit (device)	Browser	Connectivity	SUS										Technical					Application comparison											
								1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	1	2	3	4	5	6	7	8	9	10	11	
1	27	Male	MTDT	Notebook	Laptop	Chrome	Wifi	4	2	4	1	3	3	5	X	5	2	5	2	5	2	3	2	2	2	2	3	1	2	3	2	2	2	
2	23	Male	MTDT	HTC Desire S Android	Smart Phone	Opera for Mobile	Wifi	5	1	5	1	4	1	5	X	4	1	4	4	5	1	5	3	3	3	1	1	3	3	1	1	3	1	
3	22	Male	MTDT	Samsung Galaxy SII, PC Win8	Smart Phone, Laptop	mobile Dolphin, win8 Chrome	Wifi	4	1	4	2	3	3	4	2	4	1	5	3	4	1	4	2	1	3	3	3	2	3	3	3	1	3	
4	21	Male	MTDT	Xperia Mini SK17i	Laptop	Opera	Wifi	3	1	5	1	3	2	5	1	4	1	3	1	5	1	3	3	3	3	1	3	X	3	3	3	3	X	
5	22	Male	MTDT	iPhone	Smart Phone	Chrome	Wifi	1	1	4	2	1	4	4	2	4	2	4	1	4	2	1	3	3	2	3	3	2	3	3	3	1	2	
6	43	Male	MIT	iPhone	Smart Phone	Chrome	Wifi	4	4	2	3	4	4	X	2	3	4	1	4	4	2	4	2	1	2	2	1	2	1	3	3	3	2	
7	25	Male	MTDT	Samsung Galaxy SIII	Smart Phone	Chrome	Wifi	3	1	4	1	3	3	4	3	2	3	4	1	5	1	4	3	1	3	3	3	1	3	3	1	1	1	
8	23	Male	MTDT	HTC One X+	Smart Phone	Chrome	Wifi	4	2	4	1	3	2	4	2	3	2	4	1	5	2	3	3	1	3	2	3	1	1	3	3	3	2	
9	22	Male	MTDT	iPhone 4	Smart Phone	Safari	Wifi	3	2	4	1	3	4	4	3	3	1	4	1	4	2	3	X	X	X	X	X	X	X	X	X	X	X	
10	21	Male	MTDT	HTC Desire	Smart Phone		Wifi	4	3	4	2	3	3	4	2	3	1	4	1	5	1	2	1	1	1	1	1	3	1	1	1	3	1	
11	24	Male	Software Engineering	-	Laptop	Chrome	Wifi	3	1	5	4	3	2	4	3	3	3	3	1	5	3	4	2	2	1	2	1	3	3	1	1	2	3	
12	21	Male	MTDT	iPhone 4	Smart Phone	Safari	Wifi	3	2	4	2	3	3	4	3	3	2	3	2	4	2	3	3	2	3	2	3	3	3	2	2	2		
13	21	Female	MTDT	iPhone 4S	Smart Phone	Safari	Wifi	4	2	5	1	4	3	5	X	2	1	2	1	3	1	3	1	1	X	1	X	1	X	3	1	1	X	
14	24	Male	MTDT	iPhone 5	Smart Phone	Chrome	3G	4	1	5	1	5	1	5	1	5	1	3	1	5	4	4	1	1	1	1	2	1	1	3	1	1	1	
15	25	Male	MTDT	Xperia Sola	Smart Phone	Opera	Wifi	4	2	4	1	3	3	4	2	3	3	4	1	5	3	3	3	3	1	2	1	3	1	2	1	2	2	
16	23	Male	MTDT	iPhone 4	Smart Phone	Safari	Wifi	4	2	4	2	4	2	4	2	3	1	3	3	4	2	3	1	1	3	1	3	1	1	3	1	1	1	
17	25	Male	MTDT	iPhone 5	Smart Phone	Safari	Wifi	4	2	4	1	3	3	4	2	4	1	5	1	5	1	4	2	1	3	3	3	3	3	3	3	1	1	1
18	22	Male	MTDT	Galaxy S2	Smart Phone	Chrome	Wifi	4	2	4	1	3	2	3	2	4	1	4	1	5	4	4	2	2	3	3	3	X	3	3	3	3	3	
19	24	Male	MTEL	-	-	-	-	4	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
20	22	Male	MTDT	iPhone 4S	Smart Phone	Google Chrome	Wifi	3	2	4	1	4	2	4	2	4	1	3	2	4	1	4	1	3	3	3	3	1	1	3	3	1	1	
21	24	Male	MTDT	iPhone 4	Smart Phone	Safari	Wifi	3	3	3	3	3	3	3	3	3	3	1	5	3	5	3	X	X	X	X	X	X	X	X	X	X	X	
22	21	Male	MTDT	iPhone 4	Smart Phone	Safari	Wifi	X	X	X	X	X	X	X	X	X	X	2	5	4	4	3	2	2	3	2	3	2	1	3	3	2	2	
23	24	Male	MTDT	Samsung Galaxy Note 2	Smart Phone	Chrome	Wifi	4	1	5	1	5	1	5	1	5	1	5	1	5	1	5	2	2	2	2	2	2	2	2	2	2	2	3
24	23	Male	MTDT	None	None	None	None	1	2	3	1	3	4	3	3	3	3	3	3	3	3	3	X	X	X	X	X	X	X	X	X	X	X	
25	21	Male	MTDT		Laptop	Chrome	Wifi	4	1	5	1	5	1	5	1	5	1	5	1	4	1	5	1	2	2	2	2	3	2	1	3	3	3	3
26	24	Male	Datateknikk	Macbook	Laptop	Safari	Wifi	3	3	2	2	3	2	4	3	3	2	4	2	4	2	3	3	3	3	3	3	3	3	3	3	3	3	3
27	25	Male	Datateknikk	iPhone	Smart Phone	Safari	Wifi	3	1	4	1	4	1	4	2	4	1	5	1	4	1	4	2	1	2	3	1	1	3	1	2	1	1	
28	29	Male	Data	iPhone	Smart Phone	Safari	Wifi	4	2	4	1	3	2	X	4	5	2	4	1	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3
29	23	Male	MTDT	Huawei Honor	Smart Phone	Mozilla	Wifi	4	1	4	1	4	2	5	1	5	1	4	3	4	2	4	2	2	3	2	3	1	X	3	2	2	2	
30	23	Female	BIT	Samsung Galaxy S3	Smart Phone	Chrome	Wifi	3	3	3	1	2	3	4	X	4	1	4	2	3	2	3	2	2	2	2	3	2	X	2	2	2	2	

GameWall evaluation

General Information	
Age (eg. 24 years)	
Gender (male/female)	
Study programme (eg. MTDT)	
Mobile device (eg. iPhone 4S, Samsung Galaxy SIII, MacBook, Samsung Galaxy Tab)	
Web browser (eg. Opera, Safari, Firefox)	
Connectivity (eg. Edge, 3G, Wifi)	
I participated on the previous test conducted March 12th 2013 (yes/no)	

The questions below on page 1 (SUS Questions and Technical Considerations) considers the overall concept of using a shared GameWall within a classroom. Use the overall impressions from all three applications combined and mark a single box of your choice.

	SUS Questions - GameWall System	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
1	I think that I would like to use this system frequently					
2	I found the system unnecessarily complex					
3	I thought the system was easy to use					
4	I think that I would need the support of a technical person to be able to use this system					
5	I found the various functions in this system were well integrated					
6	I thought there was too much inconsistency in this system					
7	I would imagine that most people would learn to use this system very quickly					
8	I found the system very cumbersome/awkward (in norwegian: tungvint) to use					
9	I felt very confident using the system					
10	I needed to learn a lot of things before I could get going with this system					

	Technical Considerations	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
1	I found it easy to setup and start using the applications through the web browser					
2	I did not manage to interact with the GameWall system with at least one web browser					
3	I found it satisfying that I did not have to install any of the applications on my smartphone					
4	I found the applications unresponsive and/or with noticeable network latency					
5	I felt that the applications had consistent quality					

The questions on page 2 (Application Comparison and Comments) considers each application individually. The answers will be used as feedback to improve each of the applications and find the most suited classroom application. Mark a single box of your choice.

	Application Comparison	WordCloud	Post-it	Categories
1	The application that was the most fun to use			
2	The application that was the most engaging			
3	The application that I believe has the highest learning outcome			
4	The application that I would like to be used in most classes			
5	The application that made me think the most			
6	The application that demands the most creativity			
7	The application that made me initiate social interaction			
8	The application that was the most challenging			
9	The application that made me most active			
10	The application that held my attention			
11	The application that made me contribute the most			

Comments about WordCloud

--

Comments about Post-it

--

Comments about Categories

--

General information									Strongly disagree (1), disagree (2), neutral (3), agree (4), strongly agree (5)										Application comparison															
									SUS					Technical																				
Nr	Age	Gender	Study programme	Device	Unit (device)	Browser	Connectivity	Participated last	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	1	2	3	4	5	6	7	8	9	10	11
1	23	Male	MTING	iPhone 4	Smart phone	Safari	Wifi	No	4	2	4	2	4	2	4	2	4	2	5	1	5	1	5	1	1	3	2	2	2	2	3	3	3	3
2	21	Male	MTDT	Sony Ericsson / Lenovo Thinkpad	Cell phone / laptop	Opera	Wifi	Yes	4	1	5	1	4	2	5	1	4	1	5	1	5	1	X	1	1	3	X	1	1	X	1	1	X	X
3	22	Female	MTDT	LG Phone	Smart phone	Chrome	Wifi	No	3	2	4	2	4	2	4	2	4	1	4	2	4	3	4	3	3	3	3	3	1	2	X	3	3	3
4	21	Female	MTDT	iPhone 4	Smart phone	Safari	Wifi	No	3	3	3	3	3	3	3	3	3	3	4	3	4	3	3	3	3	3	3	3	1	2	3	3	3	3
5	21	Male	MTDT	One X HTC	Smart Phone	Chrome	Wifi	Yes	3	2	4	1	4	2	4	2	4	1	4	2	4	2	4	1	2	3	2	3	2	2	3	2	3	3
6	24	Male	MTDT	iPad 1. gen	Tablet	Safari	Wifi	Yes	3	2	4	1	3	2	4	2	4	1	1	5	3	3	3	1	1	2	2	2	2	2	X	2	2	2
7	24	Male	MIT	LG	Smart phone	Chrome	Wifi	No	2	2	4	1	3	1	5	1	5	1	5	1	3	1	4	3	3	3	3	3	1	X	X	3	3	3
8	43	Male	MIT	iPhone 4S	Smart phone	Chrome	Wifi	No	2	2	4	1	4	2	4	2	4	2	4	2	4	2	4	3	3	3	3	3	1	2	3	2	3	1
9	22	Female	MTDT	Galaxy Nexus	Smart phone	Chrome	Wifi	No	4	2	5	1	4	2	5	1	5	1	5	1	5	1	5	X	X	X	X	X	X	X	X	X	X	X
10	21	Male	MTDT	iPhone 4	Smart phone	Safari	Wifi	Yes	4	2	4	1	3	3	4	2	3	2	4	1	4	2	3	2	2	2	2	3	1	2	3	2	2	2
11	23	Male	MTDT	Huawei Honor	Smart phone	Firefox	Wifi	Yes	4	2	4	1	4	1	4	1	4	1	5	3	4	2	4	1	3	3	1	3	1	2	3	3	1	1
12	22	Male	MTDT	iPhone 4S	Smart phone	Safari	Wifi	No	3	2	4	1	3	3	4	2	4	1	4	1	4	2	4	3	3	3	X	3	1	3	3	3	3	X
13	22	Male	MTDT	iPhone 5	Smartphone	Chrome	Wifi	Yes	4	2	5	1	4	2	4	1	4	1	5	2	4	2	4	2	3	3	3	3	1	3	3	3	3	3
14	21	Female	MTDT	iPad3 and iPhone 4S	Smartphone and Tablet	Safari	Wifi	Yes	3	2	4	1	3	3	5	3	3	3	5	1	4	1	3	1	1	1	1	X	X	X	3	X	1	1
15	23	Male	MTDT	Samsung Galaxy SIII	Smartphone	Firefox	Wifi	No	3	3	4	1	4	3	4	2	4	3	4	2	4	4	4	1	3	3	1	2	1	3	3	3	1	3
16	26	Male	MTING	-	-	Chrome	Wifi	-	4	2	4	1	4	2	4	2	3	1	4	1	3	1	4	2	1	2	2	3	1	2	3	3	3	2
17	22	Male	MTDT	None	None	None	None	Yes	2	3	3	2	3	4	3	3	3	3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
18	24	Male	MTEL Exchange student	Samsung Nexus S	Smartphone	Standard	Wifi	Yes	3	2	4	2	3	3	4	2	4	2	4	2	5	3	3	3	3	2	3	1	2	3	2	2	3	2
19	24	Male	Exchange student	-	-	Chrome	Wifi	Yes	4	1	4	2	3	2	3	2	3	2	3	2	5	3	4	2	3	3	1	3	1	1	3	3	3	1
20	23	Male	-	HTC Desire S	Smartphone	Opera for android	Wifi	Yes	4	2	5	1	5	1	5	X	5	5	5	1	5	4	5	2	2	2	2	3	1	1	3	3	2	1
21	23	Female	DIT	Samsung Galaxy S3	Smartphone	Chrome	Wifi	Yes	4	2	4	2	4	3	4	2	4	2	4	3	4	2	3	3	3	3	3	3	1	2	3	3	3	3
22	25	Male	MTDT	Xperia Sola	Smartphone	Chrome	Wifi	Yes	4	2	4	1	3	2	4	1	4	2	4	1	4	1	4	X	X	X	X	X	X	X	X	X	X	X
23	24	Male	MTDT	MacBook	Laptop	Safari	Wifi	Yes	3	2	4	2	4	2	4	2	3	2	4	2	4	2	4	X	X	X	X	X	X	X	X	X	X	X
24	23	Male	MTDT	HTC One X+	Smartphone	Chrome	Wifi	Yes	5	2	4	2	3	4	5	2	4	2	4	1	5	2	4	2	3	3	2	3	1	12	3	3	3	2
25	25	Male	MTDT	iPhone 5	Smartphone	Safari	Wifi	Yes	3	2	4	1	3	3	4	2	4	1	4	1	4	2	3	2	3	3	3	3	2	3	3	3	3	2
26	24	Male	MIT	Android	Smart phone	Chrome	Wifi	No	4	2	3	1	4	2	4	3	3	1	4	2	5	1	4	1	1	1	1	2	2	2	3	2	3	3
27	22	Male	MTDT Exchange student	Samsung Galaxy S2	Smartphone	Chrome	Wifi	Yes	2	2	4	1	3	2	4	2	4	1	4	2	4	2	4	3	3	3	3	3	2	2	3	3	3	3
28	24	Male	Exchange student	Asus X55C	Laptop	Chrome	Wifi	No	4	1	4	1	4	2	5	1	5	1	5	2	4	1	3	3	3	3	3	2	2	2	3	2	3	2
29	22	Male	MTDT	iPhone	Smartphone	Chrome	Wifi	Yes	2	2	4	2	2	4	4	3	4	2	4	2	4	3	2	3	2	2	3	2	2	2	2	2	2	3
30	23	Female	MTDT	iPhone	Smart phone	Safari	Wifi	No	3	1	4	1	4	3	4	2	4	2	4	2	5	2	4	3	3	3	2	2	2	2	2	3	3	3

1814 evaluation

General Information	
Age (eg. 24 years)	
Gender (male/female)	
Study programme (eg. MTDT)	
Mobile device (eg. iPhone 4S, Samsung Galaxy SIII, MacBook, Samsung Galaxy Tab)	
Web browser (eg. Opera, Safari, Firefox)	
Connectivity (eg. Edge, 3G, Wifi)	

The questions below considers the game *1814*. Use the overall impressions from the game sequence and mark a single box of your choice.

	Technical User Experience	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
1	I found it easy to setup and start using the game through the web browser					
2	I did not manage to interact with the game with at least one web browser					
3	I found it satisfying that I did not have to install the game on my smartphone					
4	I found the game unresponsive and/or with noticable network latency					

	Gameplay & Social Interaction	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
1	The game was fun to play					
2	The game was not engaging					
3	I did not notice that the game was played in a web browser					
4	Playing the game in the same room with other people made me less competative					
5	Playing the game in the same room with other people made it more fun					
6	I would not like to play games this way (shared common screen w/browser controller) frequently					
7	Playing the game in the same room with other people made me cooperate better (vs online play)					

Comments about 1814

--

General information								Strongly disagree (1), disagree (2), neutral (3), agree (4), strongly agree (5)							Comments				
Nr	Age	Gender	Study programme	Device	Unit (device)	Browser	Connectivity	Technical User Ex.				Gameplay & Social Interaction							
								1	2	3	4	1	2	3	4	5	6	7	
1	24	Male	MTDT	iPad 1. gen	Tablet	Safari	Wifi	4	1	4	4	4	2	3	4	4	4	3	
2	24	Male	MTEL	Samsung Nexus S	Smartphone	Chrome	Wifi	4	1	4	3	4	4	2	3	3	2	3	A bit hard to understand the rules of the game at the beginning
3	43	Male	MIT	iPhone 4S	Smartphone	Chrome	Wifi	4	5	4	4	4	2	2	2	4	4	4	På iPhone 4S, chrome, ble ikke "control-knappen" riktig skalert på skjermen. "Ned-knappen" var ikke synlig
4	24	Male	MIT	LG something	Smartphone	Chrome	Wifi	3	2	4	4	1	4	3	3	5	3	3	Shot my cannon at a horse for 2 min with no response from the game (was the horse supposed to just live, or die? Questionnaire is full of "not"-questions, which should be avoided: eg "The game was not engaging" should be "The game was engaging"
5	22	Male	MTDT	Samsung Galaxy SII	Smartphone	Chrome	Wifi	5	3	4	5	2	4	3	2	4	2	3	I did not understand why, nor did I like that i swapped character (died?). I had difficulty finding my character, I missed my nametag
6	22	Male	MTDT	iPhone Galaxy Nexus	Smartphone	Chrome	Wifi	4	4	4	3	2	4	2	3	4	4	3	Bad hitbox detection. Sprites was not removed upon death. Should have nicknames over characters to make it easier to find yourself.
7	22	Female	MTDT	Exchange student	Smartphone	Chrome	Wifi	4	3	5	4	4	2	3	1	4	2	4	Hard to figure out who you were / which charater on the screen you control
8	24	Male	MTDT	Asus X55c	Laptop	Chrome	Wifi	5	1	3	1	4	1	1	1	5	1	4	Add more instructions about playing (How to play)
9	22	Female	MTDT	iPhone 4	Smartphone	Safari	Wifi	5	3	4	4	4	3	3	2	4	3	3	Hard to figure out which player I am in the game
10	23	Male	MTDT	Huawei Honor	Smartphone	Mozilla	Wifi	4	3	4	2	4	2	2	2	4	4	4	need indications of hit targets. needs to be easier to know who you are
11	21	Female	MTDT	iPhone 4	Smartphone	Safari	Wifi	4	3	4	4	2	3	2	3	3	3	3	i did not understand who i was and how to do stuff. and my player changed and my team changed and i dont know how you did not say we used only the common screen - the image on the PC looked like a loading screen, so most of the time i thought the game was not ready on my computer yet, and just waited. when my neighbour noticed how it worked, he told me, and i started trying to find myself on the screen, but due to my poor eyesight and the unexplained show-name (aka "?") button, i did not. might be a fun game, but i did not really play it.
12	21	Male	MTDT	Lenovo ThinkPad E130	laptop	Chrome	Wifi	X	X	5	X	X	X	X	X	X	X	X	it did not scale well on my device/browser, hard to see where my character was, seemed like it did not detect when i hit - or taught i hit.
13	23	Female	BIT	Samsung galaxy s3	Smartphone	Chrome	Wifi	4	3	3	5	1	2	1	2	4	4	1	I could not move left/right. Usability: make moving interactive. Pause game at start (lobby). Missing nicknames, difficult to locate player
16	27	Male	MTDT	HTC Desire	Smartphone	Android	Wifi	5	3	4	4	3	3	4	2	4	3	4	
17	23	Male	MTDT	-	-	Firefox	Wifi	2	1	4	4	3	3	3	2	4	5	1	
18	21	Female	MTDT	"The new iPad", iPad3	Tablet	Safari	Wifi	5	1	5	1	3	3	3	1	4	2	1	Vanskelig å se hva som skjedde da jeg angrep. Fikk ikke inntrykket av at jeg traff noen. Også var det litt rart at jeg plutselig byttet side. Først norsk, så svensk.
19	22	Male	MTDT	iPhone 5	Smartphone	Chrome	Wifi	5	1	4	4	3	4	4	2	4	4	3	
20	22	Male	MTDT	iPhone 4s	Smartphone	Chrome	Wifi	5	1	5	4	4	3	1	1	5	2	3	Kult konsept
21	25	Male	MTDT	Xperia Sola	Smartphone	Chrome	Wifi	5	1	4	3	3	2	4	1	4	4	3	
22	24	Male	MTDT	Macbook	Laptop	Safari	Wifi	4	2	4	4	2	4	2	2	4	2	3	
23	23	Male	MTDT	TC One X+	Smartphone	Chrome	Wifi	5	3	5	5	3	2	2	1	4	3	4	I did not know where/who i was
24	26	Male	MTING	-	-	Chrome	Wifi	4	2	3	2	2	4	3	2	3	5	3	hard to find who you controll. no feedback on if you are hit or are hitting someone. feels completly random. because you cant see who is doing well.
25	23	Male	MTING	Macbook	Laptop	Safari	Wifi	5	1	5	1	4	2	2	3	4	2	3	
26	23	Female	MTDT	iPhone	Smartphone	Safari	Wifi	4	1	5	3	3	4	2	2	4	2	3	

Bibliography

- [1] 4+1 Architectural View Model. <http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf>. Accessed: 04/03/2013. 54
- [2] Cascading Style Sheets. <http://www.w3.org/Style/CSS/>. Accessed: 12/11/2012. 30
- [3] D3.js. <http://d3js.org/>. Accessed: 03/03/2013. 31
- [4] Developing multiplayer HTML5 games with Node.js. <http://smus.com/multiplayer-html5-games-with-node/>. Accessed: 18/01/2013. 18
- [5] Document Object Model (DOM). <http://www.w3.org/DOM/>. Accessed: 13/11/2012. 29
- [6] EaselJS. <http://www.createjs.com/#!/EaselJS>. Accessed: 27/01/2013. 85
- [7] Google scholar. <http://scholar.google.no/>. Accessed: 21/01/2013. 13
- [8] Hemisphere games. <http://www.hemispheregames.com/>. Accessed: 18/01/2013. 18
- [9] HTML. <http://www.w3.org/MarkUp/>. Accessed: 12/11/2012. 28
- [10] jQuery. <http://jquery.com/>. Accessed: 14/11/2012. 31
- [11] jQuery mobile. <http://jquerymobile.com/>. Accessed: 14/11/2012. 80
- [12] jQuery Usage Trends. <http://trends.builtwith.com/javascript/JQuery>. Accessed: 07/03/2013. 31
- [13] Microsoft xbox. <http://www.xbox.com/nb-NO>. Accessed: 14/02/2013. 36
- [14] Nintendo Wii U. <http://www.nintendo.com/wiiu>. Accessed: 19/01/2013. 19
- [15] Node.js. <http://nodejs.org/>. Accessed: 16/10/2012. 34, 35, 59, 85

- [16] North & South at MobyGames. <http://www.mobygames.com/game/north-south>. Accessed: 17/04/2013. 80
- [17] Paper.js. <http://paperjs.org/>. Accessed: 08/11/2012. 31
- [18] Real-time Games using HTML5, WebSockets, Nodejs and Socket.io. <http://blog.joshsoftware.com/2012/04/12/real-time-games-using-html5-websockets-nodejs-and-socket-io/>. Accessed: 17/01/2013. 18, 134
- [19] Socket.IO. <http://socket.io/>. Accessed: 17/10/2012. 34, 35, 60, 85
- [20] Sony playstation. <http://no.playstation.com/>. Accessed: 14/02/2013. 36
- [21] SoundHound. <http://www.soundhound.com/>. Accessed: 18/03/2013. 37
- [22] SUS - A quick and dirty usability scale. <http://hell.meiert.org/core/pdf/sus.pdf>. Accessed: 11/03/2013. 88
- [23] Usage of JavaScript libraries for websites. http://w3techs.com/technologies/overview/javascript_library/all. Accessed: 07/03/2013. 31
- [24] Morten Versvik Aleksander Baumann Spro. Multiplayer on one screen entertainment system, jul. 2007. 21, 82
- [25] Eivind Sorteberg Alf Inge Wang, Martin Jarret. Experiences from implementing a mobile multiplayer real-time game for wireless networks with high latency. 2009. 97
- [26] Terje Øfsdahl Alf Inge Wang and Ole Kristian Mørch-Storstein. Lecture Quiz - A Mobile Game Concept for Lectures. <http://www.idi.ntnu.no/grupper/su/publ/alfw/aiw-sea2007.pdf>, 2007. 22, 89, 135
- [27] Scott W. Ambler. UML 2 sequence diagrams. <http://www.agilemodeling.com/artifacts/sequenceDiagram.htm>, okt. 2012. 56
- [28] Victor R. Basili. The experimental paradigm in software engineering. In *Proceedings of the International Workshop on Experimental Software Engineering Issues: Critical Assessment and Future Directions*, pages 3–12, London, UK, UK, 1993. Springer-Verlag. 12
- [29] Gaetano Borriello, Carl Hartung, Bruce Hemingway, Karl Koscher, and Brian Mayton. Multi-player soccer and wireless embedded systems. *SIGCSE Bull.*, 40(1):82–86, March 2008. 21
- [30] Coulouris, George; Jean Dollimore, Tim Kindberg, Gordon Blair. *Distributed Systems: Concepts and Design*. Boston: Addison-Wesley, 5th edition, 2011. 32

- [31] Randy Pausch Dan Maynes-Aminzade and Steve Seitz. Techniques for interactive audience participation. aug. 2002. 20
- [32] Aleksander Elvemo and Vegard Gamnes. *Framework for Multi-player Game-Wall Interaction - A prestudy*. NTNU, IDI, TDT4501, 2012. 4, 12
- [33] David Flanagan. *JavaScript: The Definitive Guide*. O'Reilly and Associates, 2006. 31
- [34] Sheng-Chin Yu Fong-Ling Fu, Rong-Chang Su. EGameFlow: A scale to measure learners' enjoyment of e-learning games. <http://www.sciencedirect.com/science/article/pii/S0360131508001024>, 2009. 91
- [35] Martin Fowler. *UML distilled*. Addison-Wesley Professional, 2004. 47
- [36] Borko Furht. *Handbook of Augmented Reality*. Springer, 2011. 71
- [37] James Paul Gee. What video games have to teach us about learning and literacy, 2003. 36, 135
- [38] ISO/IEC. Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Requirements for quality of Commercial Off-The-Shelf (COTS) software product and instructions for testing. *Computer*, aug. 2011. 43
- [39] L. C. Wolf L. Pantel. On the impact of delay on real-time multiplayer games. 2002. 83
- [40] 82 Queva Vista Novato CA 94947 Loren C. Carpenter. Video imaging method and apparatus for audience participation. Patent, 11 1994. US 5365266. 20
- [41] Salvatore Loreto, P Saint-Andre, S Salsano, and G Wilkins. Known issues and best practices for the use of long polling and streaming in bidirectional [http](http://www.internet-engineering-task-force.org/). *Internet Engineering Task Force, Request for Comments*, 6202:2070–1721, 2011. 32
- [42] Kajal Claypool Mark Claypool and Feissal Damaa. The effects of frame rate and resolution on users playing first person shooter games. 2005. 83
- [43] Seymour Papert. Does easy do it? Children, Games, and Learning, 1998. 36, 135
- [44] Cumsille Marianov Correa Flores Grau Lagos V. Lopez X. Lopez Rodriguez Rosas, Nussbaum and Salinas. Beyond Nintendo: design and assessment of educational video games for first and second grade students, 2003. 36, 135

-
- [45] Jürgen Scheible, Timo Ojala, and Paul Coulton. MobiToss: a novel gesture based interface for creating and sharing mobile multimedia art on large public displays. In *Proceedings of the 16th ACM international conference on Multimedia*, MM '08, pages 957–960, New York, NY, USA, 2008. ACM. 17
- [46] Alireza Sahami Shirazi, Christian Winkler, and Albrecht Schmidt. Flash-light interaction: a study on mobile phone interaction techniques with large displays. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '09, pages 93:1–93:2, New York, NY, USA, 2009. ACM. 17
- [47] Peter Van Roy and Seif Haridi. *Concepts, techniques, and models of computer programming*. MIT press, 2004. 46
- [48] Bian Wu and Alf Inge Wang. Improvement of a Lecture Game Concept - Implementing Lecture Quiz 2.0. <http://www.idi.ntnu.no/~alfw/papers/csedu2011-lecturegame.pdf>, 2011. 91
- [49] Bian Wu and Alf Inge Wang. A pervasive game to know your city better. 2011. 89