Expertise
and insight
for the future

HyeSoo Park

Cooperative game design and development

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Cooperative game design and development

19 May 2020

Author Title Number of Pages Date	HyeSoo Park Cooperative game with Clojure and ClojureScript 47 pages + 3 appendices 19 May 2020
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Mobile Solutions
Instructors	Merja Bauter, Researching Lecturer Petri Vesikivi, Principle Lecturer

The goal of this project was set to demonstrate the viability of designing and developing a co-located game that enhances the player's cooperation and social interaction in a playful way.

The methods used in this study include specific design patterns such as high interdependency, a common goal, and shared control among several determinants of social interaction and player's experience. According to the researches, mechanics of the gameplay, the setup, the scoring system determines whether the gameplay is supposed to be competitive or cooperative. The game design plays an important role when crafting the game as an important motivator for players to cooperate and interact with each other.

In this study, Aikakone multiplayer puzzle game was designed and developed as a proof of concept. The expressive representation of data in Clojure and ClojureScript allows writing and manipulating the game state more easily and simply. Additionally, they helped to predict where the game mutation will happen and make a game less error prone. Game functionalities from the Phaser were utilized by interpolating the functions and properties in the Phaser. Sente enabled communication between clients and servers and synchronization of the game states of all clients via WebSocket and Ajax. The User interface components were made with Reagent and 'Re-frame' was used to handle app state to route between different game pages as a form of SPA.

As a result of this study, the game playtesting has proved that interdependency, common goal, and shared control in a game foster users to talk to each other to achieve the goal of the game. The case study showed it is possible to design and implement the game which encourages the players to interact with each other more and also generated the cooperative play experience.

For a future refinement, studies on methods of physiological measures and gameplay metrics are suggested to assess the social aspects of a cooperative game. Since the game was developed to enhance the player's collaboration, evaluating the impacts of game patterns on social factors of a game possibly brings more meaningful findings in the project.

Keywords	Cooperative game, Game design, Serious game, multiplayer game, Clojure

Contents

List of Abbreviations

TCP	Transmission Control Protocol
LISP	List Processing
FP	Functional Programming
OOP	Object-Oriented Programming
JS	JavaScript
JVM	Java Virtual Machine
CPU	Central Processing Unit
SVG	Scalable Vector Graphic
API	Application programming interface.
UI	User Interface
DOM	Document Object Model
SPA	Single Page Application
HTTP	HyperText Transfer Protocol

1 Introduction

Technology connects people and makes their lives more convenient. It also urges us to stay connected with our eyes on coming texts, emails, or news. Paradoxically, we seem to lose human connection and psychosocial well-being as a result of remaining socially connected with the help of technology such as smartphones [1.] The technology seems to have two sides of coins in regard to human connection and social interaction in people's lives. The author of this thesis wishes to reconcile learning collaboration with playful aspects of a game by designing and building a cooperative serious game. Digital games or video games enables us to spend time together in a playful way and can enhance the concept of togetherness that is overlooked in the high-tech age [2, 1121].

Several types of research have proved that games can work as a tool or medium to give a positive impact on social interactions and player experience with particular design patterns and certain gameplay settings. A study from Brigham Young University's School of Family Life found that kids who were co-playing with parents showed positive indicators such as better behavior, feeling more connected to the families, and stronger mental health [3, 16]. The research from Pontifical Catholic University found that a collaborative game for the multitouch tabletop encouraged the youth with autism to ask for help with verbal and gestural expressions with other players to cooperate and meet the game goal [4].

This project began with concept design discussions with supervising lecturers, Merja Bauters, and Olli Alm in December 2016. Game design, architecture design, and technical implementation were created after the concept design for the game. Due to personal matters and technology issues, the implementation was delayed for 1 year. Technical implementation of the project was conducted mainly between February and June in 2018 and at the end of June, the game was collaboratively playable and responsive to the different screen sizes of devices. Finally, the first digital prototype of the cooperative serious game application, Aikakone was completed in the middle of July 2018.

Chapter 2 presents the fundamentals of game design. The fundamentals include different types of gameplay, game elements, iterative game design process, and player

experience with example games. It points out what aspects can be considered when designing a game. It introduces what technologies exist for a real-time multiplayer game and what to consider when developing a game in Clojure/ClojureScript. Chapter 3 presents Aikakone, a case study application that was designed and built to enhance people's collaboration and social interaction in a playful way. It describes the process of the project from game design to implementation to create a cooperative serious game. Chapter 4 shows the process and results of three user testing and the analysis of what kind of player game experience game elements and game mechanics generate.

2 Game design and development

2.1 Fundamental of Game design

2.1.1 Playing games and Cooperative play

Game design is about crafting basic game elements and how they come together generates play. **Playing games** is commonly compared to watching movies, reading books, or playing music. [5.] These activities have common parts between them, but the big difference is that game players need to interact to bring changes to a game. Depending on how actively the players take roles in playing a game, it makes a different impact on the substance and quality of play experience. The game itself is a medium that makes a play when all the elements interact with players.

When all the elements in a game are harmonized, they create different dynamics. As an example, a car as a system has objects and dynamics. The car's object is described as turning wheels, lights, brake pedal, and car's dynamics include direction changes, indication, deceleration. Both the objects and dynamics interact with each other to make a car work and create a driving experience. If a game is seen as a system, a game design creates relationships between game objects and dynamics. It brings meanings and purpose to a game and players decide the purpose of their play experiences. The play which a game as a system generates is listed as funny play, fast play, silly play, serious play, cooperative play, competitive play, expressive play. [5.]

Competitive play is a widespread form of play. However, sometimes people want to collaborate. When each player is in sync and it goes well, the cooperative play enables players to experience one of the best kinds of fun they can have. Exquisite Corpse game is a good example of **cooperative play**. [5.] It begins with a folded sheet of paper. Each player draws in one of the blank panels and the adjacent panels need to be slightly overlapped. When all the panels are filled with drawings, the players can see a final image of cooperation. The goal is to create drawing together and bring a playful experience.

Valve's Portal 2 is one of the most famous collaborative games. The game's two-player cooperative mode encourages players to act and think cooperatively. The two need to solve the spatial puzzles together. Sometimes, they need to time their actions; on the other hand, one player needs to make portals for the other player. It is also a good example of **symmetrical cooperative play** [5.] In a game, player characters are different, but they have the same way of movements and actions such as shooting portals, running, and jumping. One important characteristic is that the characters were not assigned to certain roles. It leaves space for the players to decide how to develop and make strategies during the play.

Matt Leacock's board game Pandemic is a great example of asymmetrical **cooperative play** [5.] Players cooperate in fighting against a set of four severe diseases to protect the world. Without cooperation, the players will lose easily. The players have certain roles such as Dispatcher, Medic, Researcher, Operations expert, and Quarantine specialist. For example, while the Quarantine specialist can prevent disease outbreaks in the city and all cities connected to that city, the Medic can try to cure the disease quickly. The players figure out together what is the best way to maximize the different roles in the characters to accomplish the common goal.

Coco & Co.'s videogame Way shows a good example of symmetrical cooperative play in a unique way. The two puzzle players take turns solving puzzles over the Internet. The interesting part is that the active player can not see the puzzle in the playspace, but the inactive player can see the puzzle. Way quickly starts to have the ability to learn how to communicate through nonverbal hints to solve the problem. While the active player tries to perform a certain task or move to a specific direction, the inactive player tries to deliver a message with body language. It is defined as a **symbiotic cooperative play.** [5.] It

means each player relies on each other tightly, so they can not survive without cooperation.

Several games were designed to encourage players to cooperate with each other to meet a goal. They create collaborative experiences for players through cooperative play.

2.1.2 Elements and definition of Cooperative serious game

Game elements are the foundation of a game and provide a structure to a game. It distinguishes the game from other media or interaction. Game Design Workshop proposes seven elements of a game such as player interaction pattern, objective, rules, procedures, resources, boundaries, and outcome. [6; 7.] The following definition helps to understand each game element in detail.

Player interaction pattern is about how players interact during the play. The interaction pattern is described as a single-player, one-one-one, team versus team, multilateral, unilateral, cooperative play, and multiple individual players each working against the same system. The **objective** is what the players are trying to accomplish from the play and when some players win the game. A few different goals keep the game interesting. **Rules** gives boundaries to the players by instructing them what they may do and not do in the game. Many rules are explicitly described or instructed in the game. The game gives feedback or further context to all players and the players understand these rules implicitly. **Procedures** are about what types of actions or how players take during the play. The interaction of many rules often defines it. Some of the procedures are implicitly defined in the gameplay context. **Resources** are elements that have value in the game, and it has various types. It includes money, token, health, items, land, and game points. **Boundaries** are about where a game ends and reality begins. **Outcome** tells how the game finishes. It includes both final and incremental outcomes. In a game of chess, puzzle, racing game, or card game, they have a final outcome. One player wins and the other player loses. On the other hand, numerous Role-playing types of games often have an incremental outcome. [6.]

Game and Game-like experiences can be divided by design intents. **Game thinking** is defined as ‘the use of games and game-like approaches to solve problems and create

better experience'. **Virtual world** is a computer-based environment that simulates the real world and is designed and shared by individuals. **Gameplay** is about identified challenges, aims, and rules designed to engage the players. **Non-purposeful** means a pure game is mainly designed for entertainment. On the other hand, Serious games have an educational or other purpose behind the game, which is more than joyful, fun, or playful. Both a **serious game** and a **pure game** have four design intentions such as game thinking, virtual world, gameplay, and game elements. A serious game and a pure game are different under a **non-purposeful** category. [8.]

The different ways of gameplays impact greatly on player experience and interaction such as social presence and enjoyment. Additionally, another research about impacts of the game pattern on player experience and social interaction in co-located multiplayer games also proves that small changes in a game design make a significant impact on the interaction and communication between players [9]. The game pattern in the research means a collection of particular mechanics and rules that create entire gameplay. The research model of social player interaction is illustrated in Figure1.

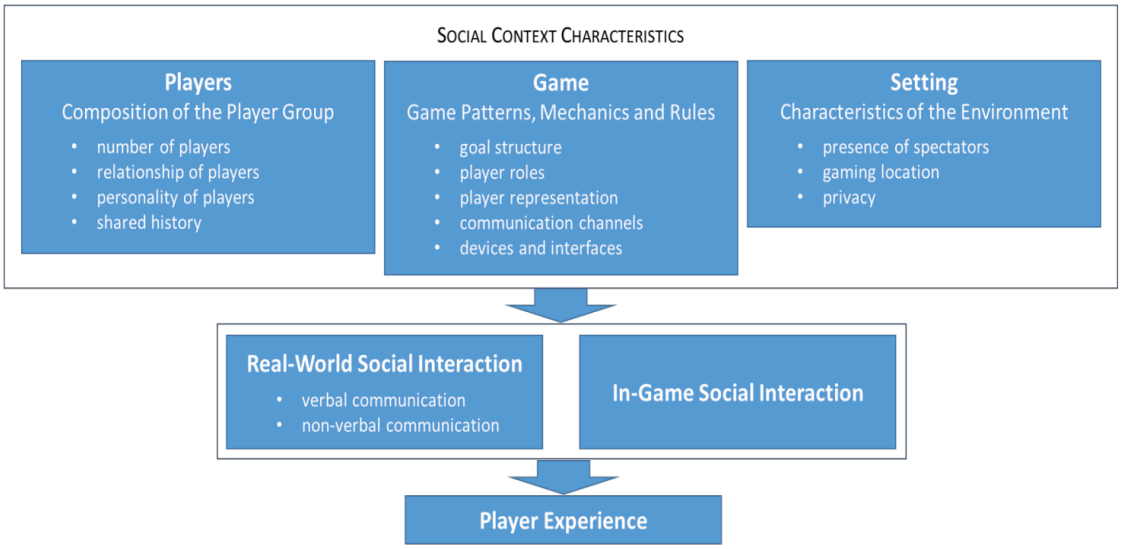


Figure 1. The research model of social player interaction.

As can be seen in Figure 1, player characteristics, game pattern, and game setting influence the player's social interaction and player's experience. Among several determinants of the social experience, the research focused on three aspects of game patterns such as **player interdependence**, **time pressure**, and **shared control**. The testbed

game was made as a co-located multiplayer game. Its result found that higher player interdependence correlates with more communication and less frustration. Regarding communal control in a game, players felt that they have less perceived competence and autonomy. On the other hand, the time pressure did not indicate impacts on players' communication and interaction. [9.]

Several design patterns of cooperative games were found by different kinds of research. **Shared goals, players different abilities, shared puzzles or characters, limited shared resources** are examples of these patterns. [10, 3]. A co-located console game study found **mechanics of the gameplay, the setup, the scoring system** determines whether the gameplay is supposed to be competitive or cooperative [11,155].

The model of **Cooperative serious game** is summarized that it has an essential game element as a game and includes game thinking, virtual world, gameplay and other purposes behind entertainment in regard to a serious game. Furthermore, it implies the game is designed to have higher player interdependence and shared control. The common goal and certain rules between players are designed to encourage cooperative play. According to the mentioned research, these designs are seen to significantly influence social interaction and players' experience.

2.1.3 Iterative game design process

The **iterative game design process** is one of the important keys to good design. As Figure 2 illustrates, four phases of the iterative process of design are composed of the Analysis phase, Design phase, Implementation phase, and Testing phase [5].

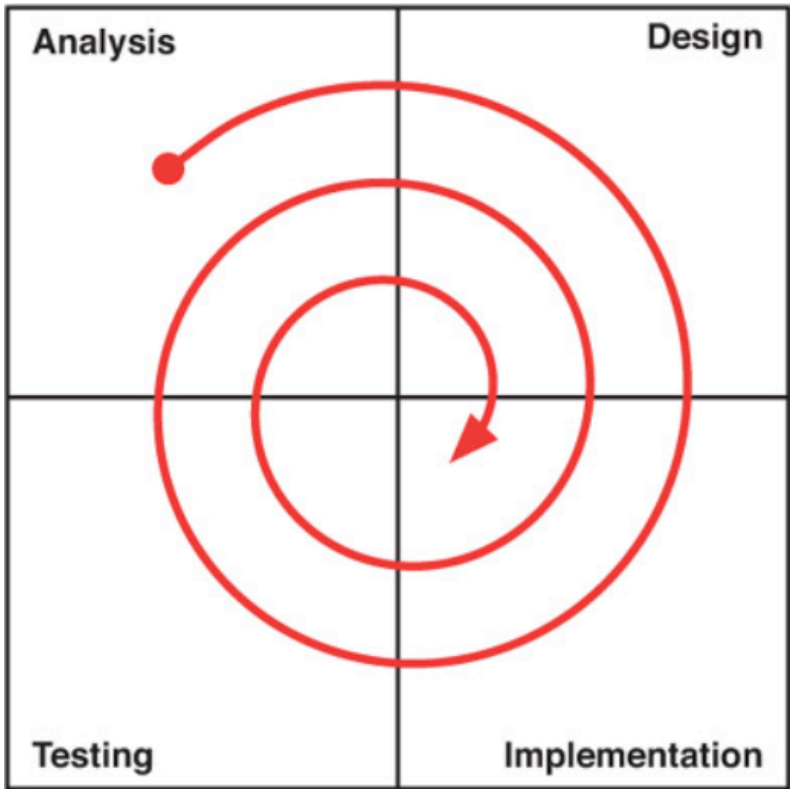


Figure 2. The iterative process of game design.

Analysis phase is about an understanding of the intended audiences, goals, and resources for the game project. The game designers need to understand where they are and what they try to solve with their design. For the analysis, designers can ask four critical questions to themselves.

As a starting point, designers try to ask several questions about target audiences. Researching their stated desires and interests helps them to make a better game. Secondly, it is essential to be realistic and honest about the resources, strengths, and weaknesses of game development. It helps designers to know how to shape their game design and prevents wasting time. Thirdly, 'Imitation precedes creation' quotation can apply to game design. Researching other existing games can often give inspiration to the designers and they can learn how other designers try to solve the design problems and what are their successes and mistakes. Lastly, the question of what is the fastest path to creating a playable game is important to get started to make a game. The core mechanic which players do most during the play can be designed, implemented, and tested first.

In the **Design phase**, with the analysis, designers make a design that solves the problems and available chances around them. It begins with coming up with ideas and ends with a solid plan for the implementation. It is not about your interest and providing a vision to other team members. Jess Schell states in his book 'The Art of Game Design', the most important skill that a game designer can have is 'listening' to the target audience, teams, clients, game, yourself, your gut, your health, and how the designer sounds to other people.

Implementation is about putting design ideas into action. It is important to apply game ideas to playable prototypes as rapidly as possible. The earliest implementations are often just a small portion of the game such as displaying and moving sprites. Testing a small portion of a game often can bring more focus than testing the game from a bigger scale. After finishing the implementation, a playtest is available. The implementations can be carried out as digital implementation and paper implementation. Depending on what to test, an effective method of implementation is differently selected. [7.] For instance, if designers test character movements in a game like Just Dance, the digital implementation is required. However, if designers want to test graphical user interface menu systems or screen changes, paper prototyping can be sufficient enough [6].

Testing is about a process by which a designer opens up a game to end-users and receives their reactions. It helps to find bugs and flaws in a game. Additionally, it helps to analyze functional features and non-functional features such as game levels, balance, and fun. Early testing increases the possibility to get the game on the right track as early as possible. Frequent testing helps the designer understand what is optimum for the play by end-users' feedback. They can give proper changes to the game according to the feedback. Honest feedback from users can bring significant help to improve the game. The designers can encourage end-users to be honest with you so that they can find the flaw of a game [5,6,7].

After running the playtest, designers have ideas on the degree of severity of the underlying issues. The designers and teams propose the solution and apply them to the design phase. The most important thing is conducting a playtest quickly and figuring out whether solutions solve the problems as your intentions. The cycle of this iterative design continues until it reaches the expected results.

2.1.4 Player experience and User interface

The player experience is explained with five layers. The author of Games, Design, and Play mentioned that the idea comes from Jesse James Garrett's book, the elements of User Experience. These five aspects can be a design model for designers to have an idea of what to consider for players. The **five layers of player experience** are about what knowledge players are good to know for playing a game. It consists of a sensory layer, information layer, interaction layer, frame layer, and purpose layer.[5.] For game designers, it is important and essential to understanding how players are aware of the playspace, how much information the game gives, how challenges encourage them to play, and what the game's context impacts a player's experience to design a game for people.

The Sensory layer is about the player's experience with five senses. In this layer, point of view and the shape of the game world are considered. [5.] Designers can ask how the player takes the information in the game world, where to draw players' attention, and how it can be linked to the player's experiences.

In the **Information layer**, the designers consider how players can transform the sensory data into information or knowledge. Many questions are about information space in a game. Which objects are hidden from players, Is the information easy to approach, how much information or clues can be provided to players. In the case of chess, both players can see every element of a game such as the number of pieces, pieces' positions, and capture status of pieces. It is described as '**perfect information**' of a game state. A Werewolf card game is a good example of a game with **imperfect information** space. [5.] Each player knows all the information about their card in their hand, but they have imperfect information about the other player's card. They need to guess and figure it out through play.

The **Interaction layer** is the stage where players combine all the information together and try to understand how the information works together. [5.] If players make a decision and act, they need a working **Mental model** of a game. The mental model is a player's internal image of how a game works. A gap between how a game's space actually works and how players think it works commonly is seen. The more they play the game with

their prior knowledge, the deeper understanding they will develop. In the interaction layer, affordance is an important concept.

Affordance is perceived properties that users think naturally, physiologically, and intuitively about how they can interact with objects. Without being taught, the body knows how to interact with objects and is ingrained to make certain actions. When users see a handle of a cup, they are likely to put their hand into the hand shape hole and grasp the handle [5; 7]. The horizontal bar on the door asks users to put both hands on it and push it. These are the **Perceptible affordances** of an object. If the objects are not used as it is supposed to be, it is called **False affordance.** Users can tell what objects are not meant for. It is defined as **Correct Rejections**. If objects are present but their appearance is not apparent, it is **Hidden affordance**. Affordance helps designers to support players to intuit elements in a playspace to link them to their understanding of the real world.

The prior or later experience makes a basic structure about how players see, experience, and comprehend. It is referred to as the **Frame layer**. Frame gives expectations to players and it helps them to understand the information provided by a game. [5.] Players find meanings in a game through their experiences and references and it frames our understanding of a game.

The **Purpose layer** comes with why questions such as why they decide to play, what they want to learn from. In terms of the player point of view, it is about what they look from gameplay for in a play experience. [5.]

Five layers of player experience are based on a series of theories from sociology, psychology, information science, and other related studies. Questions and considerations on these help designers to understand the design of games.

2.2 Technologies of real-time multiplayer game

2.2.1 WebSocket

Traditionally, clients continuously **poll** the server and check whether new messages come after the last poll to achieve real-time messaging. It was considered inefficient. While it worked, it had a very high network latency as well as high-bandwidth usage, making them unsuitable for real-time multiplayer games. On the other hand, the **WebSocket's** API defines a bidirectional communications channel over a single TCP socket. [12.] TCP is a short term for Transmission Control Protocol. It defines a set of rules on how to establish and maintain network communication between applications. It makes a connection between browser and server more efficient and persistent. It has the ability to push real-time messages much faster than before to active clients.

A WebSocket on the browser is set up as following steps to communicate with a server [13, 293 - 294].:

- instantiating a WebSocket object by providing the server URL
- implementing the open, close, and error event handlers as needed
- implementing the message event handler to handle actions when a message is received from the server
- sending messages to the server
- closing the connection to the server

2.2.2 Chromecast

Google **Chromecast** is designed to stream a video, audio, and game wirelessly from Android, IOS, or Chrome apps to a TV or other bigger screen. The application works as a remote controller to play, pause, seek, rewind, and stop the media. The users do actions on the Chromecast-supported applications, the result will be displayed on the bigger screen. In simple terms, it enables users to bring their small screen experience to the bigger screen such as a TV or bigger monitor.

When players play the Chromecast-supported game, they open the game, tap the Cast button, and choose an available casting device that they want. If players want to play together, they all need to follow the same instructions and connect to the same device.

To build a complete Chromecast application, a **sender application** and **receiver application** are required to be built. The sender application is an Android, iOS, or Chrome application and they control the playback. The Google Cast device or Chromecast is a receiver that displays the playback content on the screen. The players can always control contents in a Cast receiver when contents are cast to it. The sender applications' status and controls should stay synced with playback changes on a receiver. In this way, it can properly handle both multi-sender commands and playback controls from the device's remote control. [14.]

2.2.3 Advantage and disadvantage

WebSocket and **Chromecast** have different advantages and disadvantages when they are used for implementing a multiplayer game. WebSocket over Chromecast for game development are a) players do not have to be on the same Wi-Fi, meaning proximity of players' location is not required, b) players are not required to own Chromecast or a TV but will be able to play the games as long as they have the access to the internet and a web browser, c) lastly, for players to play Chromecast games they have to install an app on their mobile phones but WebSocket games do not require installation or even require players to have mobile phones.

On the other hand, the downsides of using WebSocket over Chromecast is a) the game developers have to host a web server for the backend functionality which can mean more costs and maintenance work needed for the server and more programming required for the backend functionality, also b) Chromecast game players play the same session of the game by being on the same Wi-Fi, so multiple groups of players playing at the same time are taken care of automatically but when developing WebSocket games, supporting multiple groups of players playing different sessions of games requires manual programming from the game developers. Lastly, c) players will need to remember the URL of the game page somehow (by bookmarking or writing it down somewhere) to access the web page for the game, unlike Chromecast games which are apps on your mobile.

2.3 Programming paradigms in game development

2.3.1 Programming paradigms

Programming paradigms are a way to categorize programming languages based on their patterns or features [15]. Problems solved with one paradigm can be solvable with other paradigms. However, certain programming languages are naturally better suited to certain problem sets.

Every paradigm defines a different set of principles for the way programmers write a program. Programming languages can embrace multiple paradigms. While some programming languages enable programmers to write in only one paradigm, others enable programmers to write in more than one paradigm.

Between the 1950s and 1960s, many high-level programming languages were created. They have considerably influenced today's programming languages such as Fortran, Algol, LISP, Cobol, and Simula [417, 421]. Fortran and Algol are classified into imperative languages and LISP, an acronym for List processing, is the second-oldest high-level functional language after Fortran which manipulates special data structures. The most well-known LISP dialects are Common LISP and Scheme. Simula is the first OOP language in which Smalltalk and C + + are inspired from [16, 421]. OOP means **Object-oriented programming**. It is one of the programming paradigms based on creating objects that consisted of data and functions.

In the 1970s, new programming paradigms called OOP and Declarative programming were born. **Declarative programming** can be more precisely split into FP and Logical programming. FP stands for **Functional programming**. The characteristics and language of FP are explained in the next paragraph. Programming language C was designed, and it settled itself quickly, thanks to its ability of direct access to low-level functionalities in the machine and effective compiler. Other languages are Pascal, Smalltalk, Declarative languages, ML, and Prolong. In the 1980s, C++ was created, and OOP concepts were further developed, adding classes and inheritance to C language. In the 1990s, Java was introduced, and it was developed based on C++. JS was also invented in 1995 as an open and cross-platform scripting language for creating and customizing

applications on the internet. [16.] JavaScript abbreviated as JS is the most popular scripting language for creating web pages and it can also be used for non-browser environments such as node.js and Apache CouchDB. It is a multi-paradigm and dynamic scripting language, providing styles of OOP, Imperative, and Declarative [17].

Recently, the **functional paradigm** has taken center stages. Pure functions help developers to reason about errors and easier them to fix. and FP was initially designed to support concurrent and distributed programming. Many programming languages adopt the functional style of building a structure and elements of programs [18.] Later, new languages such as Go, F#, Rust, Dart, Kotlin, Elixr were developed. In 2007, Clojure was created and it supports functional paradigm philosophy.

According to the IEEE research on the top programming languages in 2018, OOP is dominating in the software development industry. Comparatively new languages that contain FP as their paradigm are also on the list [19.] For example, Scala has been used by Twitter, Soundcloud, and Groupon [20]. Kotlin is supported by Google and JetBrains. Clojure has been used by Walmart, Zalando, Apple [21].

2.3.2 Relationship between JavaScript and Clojure/ClojureScript

Clojure is a fruit of an effort in pragmatic and dynamic languages. It endeavors to be a general-purpose language to be used for writing programs in various application domains. Clojure can be similarly used in similar areas as Java. Clojure achieves its purpose by running on an industry-standard open platform, JVM. JVM is a short term of Java virtual machine. Java virtual machine is a virtual machine that provides a runtime environment to run Java programs. Clojure fosters functional programming with an **immutability** and **persistent data structure**. It supports different types of immutable persistent data structures such as vector, map, and list. This structure is beneficial for developers to achieve code readability. The capacity of the Central processing unit abbreviated as CPUs has been better, and concurrent programming more demanding. Clojure provides built-in concurrency support. The immutable persistent data structures make concurrent programs less error prone. [22.]

Clojure is a functional language with a dynamic emphasis. It is a dynamic language which does not restrict the function's return type. It is more flexible about the type when a function returns a type. Clojure supports first-class functions and infers which type function it returns. Also, it has all the abilities that Java has. [22.]

ClojureScript is a compiler for Clojure which emits JS. It has similar characteristics of Clojure, offering the rich immutable data set, FP, destructuring, and state discipline. When an entire program in ClojureScript is compiled to JS, it uses the Google Closure library and compiler to gain its benefits. For instance, Clojure Compiler analyzes code and revises and brings down the code to load faster and make it more efficient. In addition to efficiency, it checks the syntax, types, and variable references, and gives warning on illegal code or future dangerous operations. [23.]

Additionally, Clojure can leverage what the host platform supports such as libraries. With interop, Clojure and ClojureScript directly interpolate with host platforms and utilize host platform libraries easily. ClojureScript. These characteristics prove that Clojure and ClojureScript are practical and fast programming languages.

2.3.3 Consideration in game development

Questions around FP language and game development are often seen in the discussion or question-and-answer based online community. Object-oriented languages have been broadly used for game development. C# is one of the example languages and it is used in a popular cross-platform game engine called Unity. Mutations of game state are an integral part of game development and FP paradigm to minimize the side effects by separating them from the rest of the logic. The mutations of game state and the pursuit of minimizing the side effects seem to be a conflicting concept and not able to be aligned. However, the questions can be asked as whether the program should be purely functional language, or several programming paradigm approaches can be chosen if it is suited better to a certain problem set. Clojure and ClojureSript encourage to minimize the side effects but do not pursue only having a purely functional approach. It allows writing a side effect in an isolated place if it is necessary and the game state management can be a case in a game dev development. [24.]

In addition, Clojure and ClojureScript support interoperating the functions and properties in the host platform libraries or benefit directly from the host platforms. The hosting platforms are Java and JavaScript which have an OOP paradigm. It means a good part of OOP approaches can be utilized by accessing the hosting platform libraries. Clojure and ClojureScript have several game libraries or frameworks. Play-cljs, Chocolatier, Phzr, Quil, and a combination of Re-frame and SVG, an acronym for Scalable Vector Graphics, are game libraries that support browser-based games. [25.] SVG is a vector image format for two-dimensional graphics. It is widely used for defining graphics on the web. JavaScript engine also can be considered, since Clojure and ClojureScript access to the host platform. To evaluate the library, a game designer or developer can ask questions such as which platforms the game library or framework supports, which library it roots from, how well documentations are, what features the library provides. Additionally, familiarity with programming languages, the experience of game development, possible time, and working environments are important to be considered, before choosing the programming languages and making a plan for the game project.

2.4 Small game development with Clojure/ClojureScript

2.4.1 Phaser and Interop

Phaser is a popular software 2D game framework for creating HTML5 games in cross platforms such as desktop and mobile. It is developed by Photon Storm. It uses both a Canvas and WebGL renderer internally and can automatically swap between them based on browser support. This allows for fast rendering across desktop and mobile. It uses the Pixi.js library for rendering.

It organized the Application programming interface, abbreviated as API documentation and more than 700 examples to use as references. It has been widely used and thus developers may leverage a fan base and community power to ask for help or support in the Support forum, Stack Overflow, Slack, or Discord.

Phaser has a number of supportive features to build game ideas into a real game application such as canvas, preloader, sprites, animation, scaling. It has four main classes which are **Game**, **World**, **Camera**, and **Stage**. [26.] Those classes have OOP paradigms

and to instantiate the object from the classes. The properties or methods of instantiated objects are useful to implement the game features. It is a JavaScript library thus if the game is going to be implemented in ClojureScript, ClojureScript **Interop** makes it possible to access properties or invoke methods in Phaser. Interop stands for interoperation and Dot special form is used for both purposes. [27.]

2.4.2 Reagent

Reagent is a ClojureScript wrapper for utilizing **ReactJS** features. Reagent extracts away difficult concepts to React. It allows developers to define the React component efficiently and easily, using ClojureScript functions and data. The UI, an acronym for the User interface is described with Hiccup-like syntax. As a brief explanation of Hiccup, it is a library for rendering HTML in ClojureScript. It uses vectors to represent HTML elements and maps to represent an element's attributes. In other words, it describes DOM which stands for the Document object model. It is a programming interface for an HTML and XML document on the web. To manage states, Reagent's own version of an atom is used. It keeps track of every time it is dereferenced. [12; 28]. Any place in components that uses an atom is re-rendered automatically, based on a value of state changes.

2.4.3 Reframe

Reframe is a Reagent framework for writing a **SPA** in ClojureScript. SPA is a short term for a single page application. It dynamically rewrites the existing web page with new content from a web server without reloading pages. It provides a better-improved user experience and overall performance. It is explained as a functional framework and a reactive framework. As a functional framework, it has data and the function manipulates the data. In terms of Reactive programming, the data coordinates the functions. Reactive programming is a programming paradigm to solve the problems and focuses on three aspects such as data stream (The data objects that start from somewhere and consumed elsewhere), functional programming and asynchronous observer. It enables us to transform the stream of data with functions. It consists of asynchronous and event-driven programs. The multiple streams of data can be combined and filtered out as a result. [29.]

It was influenced by early-Elm concepts and Clojure projects such as Pedestal App, Om, and Hoplon. It developed ideas such as event handler middleware, coeffect accretion, and de-duplicated signal graphs for the first time. [29.]

ClojureScript is a modern dialect of the LISP programming language. Reframe leverages the ClojureScript approach to **code-as-data**, in other words **homoiconic**. Homoiconic means developers are programming in data. Code and data share the same syntax. To leverage data, the re-frame has a data-oriented design. **Events, Effects, and DOM are data in re-frame**. The functions that manipulate data are officially listed and searched by data. [29.]

An **everlasting loop** was implemented as a Re-frame architecture. In the app, the pure functions will be placed on certain parts of this loop. The re-frame follows the data flowing into and out of transforming pure functions around its loop. The conveyance of data in re-framing is similar to a water cycle by different forces such as gravity, evaporation, and convection. Data coordinates the code. [29.]

Each iteration of the loop has a six-domino cascade. A prior domino triggers the next domino until it reaches the last domino. When it is in the last domino, dominos are reset to the beginning and it is ready for the next iteration. The first domino is 'Event dispatch'. The event starts with event triggering such as a button click and new message arrival in a web socket receiver. It tells re-frame is an event-driven framework. 'Event handling' comes as a second domino. In regards to an event, the event handler function computes a description of the required side effects in a declarative manner. The 3rd domino is 'Effect handling'. The returned side effects to the SPA's 'application state' are performed. It changes an application state. The re-frame accepts the nature of effects in a controlled and hidden way. One famous formula was designed for Facebook's React library. The following dominoes 4-5-6 are related to re-running ' f ' to compute and show an updated view to the user. The function computes the DOM nodes based on ' s ' , the given app state. A view, as a result, will be displayed to the user. Whenever the app state changes, a function will re-run and recompute the view for the users.

The function is illustrated below:

$$v = \mathrm{f}(\mathrm{s}) \quad (1)$$

v is a view
f is a function
s is a given app state.

Domino 4 is about taking out data from the ‘app state’. It formats data correctly and provides it to the view function. Domino 5 is View. View functions known as ‘Reagent components’ create UI DOM, utilizing the application state through the querying of the data in Domino 4. The data is reactively delivered while querying it. Domino 6 is ‘DOM’. It is managed by Reagent/React. The hiccup-formatted DOM description becomes real and the actual browser DOM nodes are updated [29.]

2.4.4 Sente

Sente is a small client and server library that makes real-time web communications easy and reliable for Clojure and ClojureScript. Sente supports bidirectional asynchronous and synchronous communications over both WebSocket and Ajax. It has automatic and sensible support for multiple client’s connections and devices simultaneously. It provides a number of useful features, such as Ajax fallback support, keep-alive, buffering, data encoding, and Ring security. [30.]

On the server-side, ‘sente/make-channel-socket!’ function needs to be called to initialize the Sente. This function handles the server adapter and a map of initialization options. ‘:client-id’ can be set in ‘:user-id-fn’ option as a request parameter. Otherwise a :uid key from the Ring session is a default for this parameter. Ring is a Clojure HTTP server library that abstracts the details of HTTP into a concise and unified API. HTTP stands for HyperText Transfer Protocol. It is an application layer protocol for transmitting hypertext documents. The ‘client-id’ key is automatically specified for each client connection and works as a unique identifier. ‘sente/make-channel-socket!’ provides the useful functions that handle Ajax POST requests, negotiates the initial WebSocket connection, provides a receive channel for the socket and pushes notifications to the client. [12.]

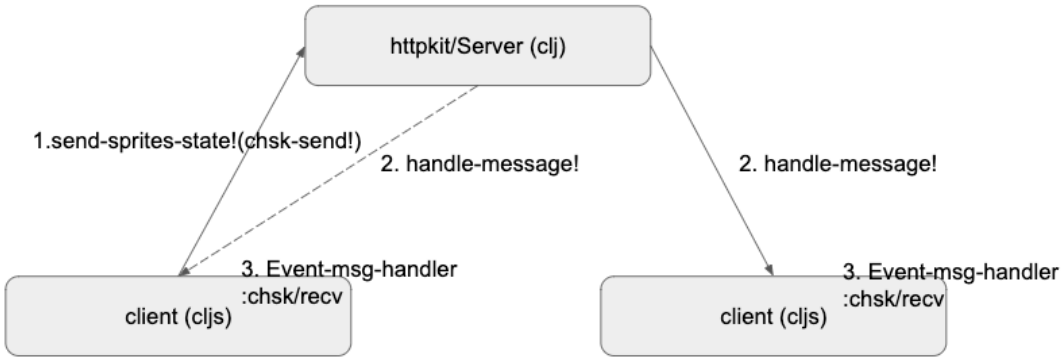


Figure 3. Events handling between clients and a server. (Copyright 2020 HyeSoo Park)

As can be seen in Figure 3, the function receives a map of event description keys as a parameter. The map has the ':id' key to find different message types and ': client-id', the UUID for each client. The chsk-send! function is used to send the message back to the client who sent a request.

The 'core.async go-loop is used to handle the message routing between clients. Sente initializes the message routing by calling the sente/start-chsk-router! Function. It takes handle-message! as the event handler. The 'ring-ajax-get-or-ws-handshake' and 'ring-ajax-post-' functions are needed to use the routes with Sente. On the client-side, the 'sente/make-channel-socket!' function is used to initialize the connection and it aligned with ones on the server. If the ':auto' is set in the ':type' key, the client's discretion to choose either to use WebSocket or Ajax, depending on the availability. The 'ch-chsk' is used to receive the message and 'chsk-send!' function is defined to send messages. The event handling is similar to one on the server-side. The 'event-msg-handler' function works as an event handler and it manages different socket events [12; 30].

3 Proof of concept: Aikakone game with Clojure/ClojureScript

3.1 Process of Aikakone game project

The previous chapters introduced fundamentals of game design, technologies of a real-time multiplayer game, programming paradigms in game development, and supportive libraries with JavaScript and Clojure/ClojureScript.

This chapter describes the overall project process and the aim of the Aikakone game project - from concept design to technical implementation. The process can be divided into **four major stages** as following:

- concept design
- game design
- architecture design
- technical implementation

Along with process structure, an **iterative game design process** was applied to test out the ideas, check implementation possibilities, and know players' points of view. It is divided into four different phases of Analysis, Design, Implementation, and Testing. Even Though the above four stages are listed in order, it was utilized more dynamically. Several stages in four major stages were revisited through an iterative game design process. For instance, after simple playtesting on the initial game, some parts in the game were redesigned and the technical implementation was updated.

Concept design is the first stage of the process where an objective of the project is defined. Ideas of the development steps and different themes were suggested by the supervising lecturers. Along with discussions and brainstorming, collaboration enabled people in the same location was chosen as a theme. Considering the theme and topic, the main focus was decided to build a game application that supports players' collaboration meaningfully in a playful environment. The concept design is explained in chapter 3.2.

The **design stage** includes **analysis phases and design phases of the iterative game design process**. In the analysis phases, the initial requirements, the available resources, strengths, and weaknesses are analyzed. Existing games are reviewed to get inspiration and the fastest path to create a playable game is considered. In design phases, game elements, mechanics of the game, and type of gameplay are decided to make a cooperative serious game. The visual design creates an interface where players directly interact with. Affordance was considered in a visual interface, considering players' experience. The design stage is explained in more detail in Chapter 3.3. The implementation phase of the process is explained as an architectural stage in chapter 3.4 and the user testing phase of the process is explained in section 4.

In the **architectural design** stage, the **technical design for the project** is considered and decided. To meet the goal of the project, possible technologies are reviewed and chosen. The main technologies, Android and Chromecast originally were suggested during the concept design. However, due to the technical, skill, and environment limitation, the core technologies were changed under discussion. Chapter 3.4 describes the architectural design.

The **actual development** was carried out in the development stage. Considering the decisions in the previous stages, the details of technologies are chosen to make features of a game throughout the all implementation process. The user interface was also implemented, and it was made to be responsive to serve different screen sizes of devices. The frontend part of the game was hosted in GitHub and the backend part was deployed in Heroku. The development stage and deployment are explained in Chapter 3.4.

3.2 Concept Design

The topic was to build an application to present the combined stories from the cultural content from Finna API on the screens in public places such as in cafes and pubs of railway stations. - The Finna API is an online service that provides the collection of Finnish archives, libraries, and museum information [31]. The ‘story’ means visitors who deliver their contents by their mobiles, after combining the image, text, audio from cultural open source services such as Finna. As facilitating devices, visitors’ mobiles and Chromecast were considered. The author of the thesis participated in the project in which

two professors, Merja Bauters and Olli Alm supervised. Professor Olli Alm was in the role of a supervising lecturer and professor Merja Bauters was in the role of the customer. In the beginning, the project was started with ideas of the development steps and different themes suggested by the professors in December of 2016. The development steps were proposed as below.

As technical steps, the first data is fetched from Finna API and it can be displayed. Secondly, an Android application sends the contents from Finna API or what platform a user chooses as the receiver application. Thirdly, Chromecast demo-application is going to project the content to the wall.

Additionally, four different themes were suggested for the project. First, it was about 'designing the UX for the representation projected on the wall'. Secondly, 'adjusting the content according to the selection based on recommendation and search' was introduced. Thirdly, collaboration enabled people in the same location was suggested. Lastly, it was about location attached 'infrastructure' where there are various presentations, and which are displayed on a map or in some other manner they can be controlled. Among the four themes, the third one was chosen for the project.

After choosing the theme, the discussion with supervisors was continued to determine what kind of collaborative aspects could be implemented. While brainstorming, many great and creative ideas came up. One was 'a collaborative jukebox' which means that visitors can add their songs into the cafeteria jukebox and the jukebox will contain a playlist in the form of the visitor's collaboration. Another idea of a collaborative game app also was brought up: When the streaming of images, texts, audios, or videos is stopped, visitors will have images on their mobile phones and start to wipe the image together and they can guess what the image is. In addition to those ideas, the puzzle game was suggested as a collaborative game by the author of this thesis in the meeting. The reason for the suggestion was that the game is a composite and synthetic art that includes images, audio, and text and it supports cooperative play.

After the discussions, the idea has been narrowed down to building a collaborative puzzle game. An important question was left with how the puzzle game can be designed to

support people's collaboration meaningfully in a playful environment and how to make meanings with Finna API.

3.3 Game Design

Game design crafts fundamental game elements. In this section, it is explained into two parts which are the Analysis phase and the Design phase. In the Analysis phase, target audiences, resources, and goals for the game project were analyzed. A puzzle game requires cognitive skills such as puzzle-solving and pattern detection, the age limit was decided as more than 7 years old and no specific target gender. A casual game was seen to be suitable for the project since the casual game covers all skill levels and is easier to learn than mid-core or core games and the game aims to cover the large range of age groups. The target place was initially public spaces such as cafes or pubs. However, to cover all the target audience age target places were slightly changed to several public places where not only adults but also kids or teenagers are also welcomed such as a library and museum.

The game was inspired by successful collaborative platforms, Kahoot and Flinga. The structure of the communication in both platforms multiple participants send their ideas or answers through a sender application to the receiver application. The sender application commonly is a smaller device such as a mobile or laptop. The receiver device is a bigger screen such as a TV or a big PC monitor which all the participants are sharing and seeing. SuperBetter is another inspiring game. It empowers players to overcome obstacles covering psychological difficulties, mental pains or sickness and achieve their goal with social support and interaction. It used a gameful method to bring strengths and mindset of gameplay to real life. [32.] Lighthouse secret to the Moon gave inspiration to the puzzle piece control. The puzzle pieces are flipped either in a row or column by clicking on the button controller. The puzzle pieces and button controllers are shown in Figure 4.

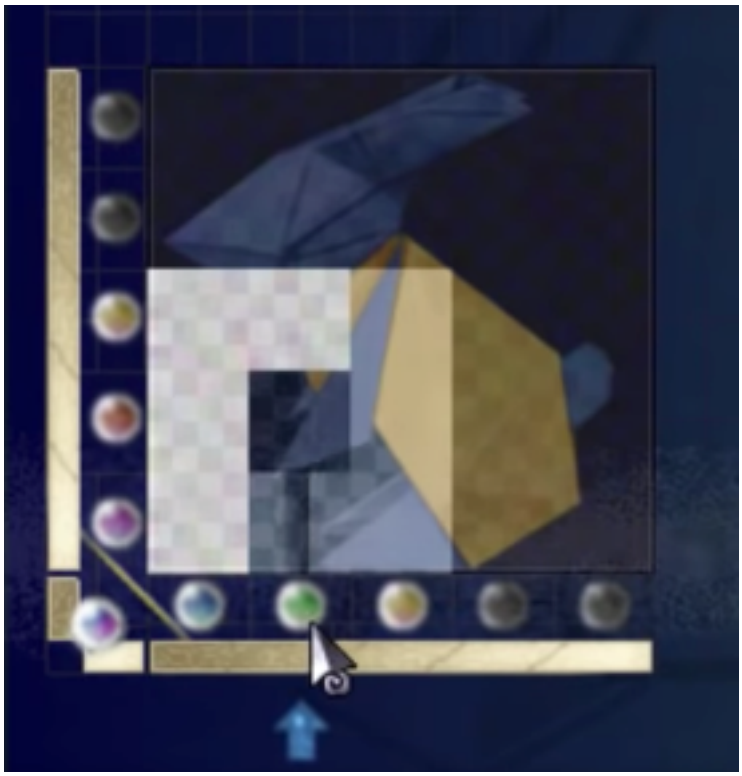


Figure 4. Puzzle pieces are flipped in a row or column by button controllers in Lighthouse secret to the moon game.

The project goal was decided with a discussion with supervising lecturers to build an application as a prototype to promote player's cooperation in a playful way by using a game. In design phases, the designing game elements to enhance the player's cooperation was mainly focused. Game elements are objectives, rules, procedures, outcomes, resources, and player interaction patterns. Books and research papers found that depending on how they are harmonized and combined, players generate a different type of play and dynamics. Interaction between objects and dynamics makes a game work and create players' experience. It was proven that the mechanic, rules, player roles, goals of a game affect social play, cooperative play experience, and social interaction. Player interdependence and shared control were found as determinants of them. [5; 6; 7; 9 ;10.]

Giving thought to goals and those findings, the game elements were designed. The objective of a game is learning collaboration through a game play. The puzzle is shared among all the clients in their browsers; thus, player interdependence was quite high. The puzzle pieces are flipped by button controllers and those are also all shared between the players. Unless other players are not active at all during the entire play, all the players

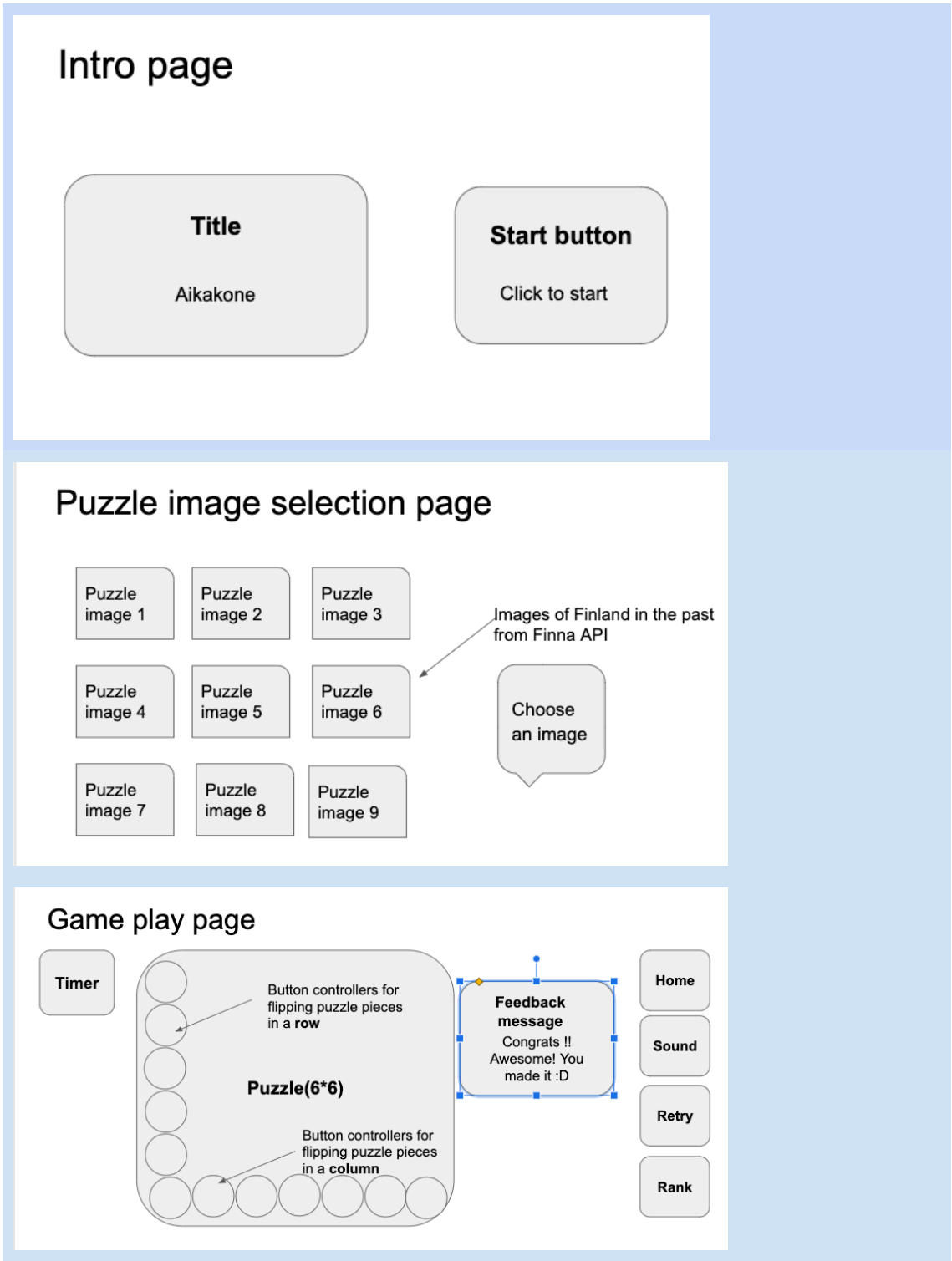
need to communicate verbally or nonverbally to match the understanding of the situation, be on the same page, and solve the puzzle which is a common goal. All players have the same role which plays a puzzle and they have a common goal. Those mentioned aspects show the gameplay is close to symmetrical cooperative play or symbiotic cooperative play.

Game and game-like applications can be split by design intents such as game thinking, game elements, virtual world, gameplay, and purposeful/non-purpose. The Aikakone game was hosted in the browser thus, it utilizes the virtual world. It contains challenges, aims, and rules. These game elements create gameplay. The purpose of the game is behind for entertainment. All the factors show it is closer to a serious game than a pure game.

The puzzle game is categorized as a serious game. **The common goal, high interdependence, shared controls between players** are designed to encourage **cooperative play - either symmetrical cooperative play or symbiotic cooperative play.**

The name of the game, theme, and main colors were decided to convey the meaning of a game. The name of the game, '**Aikakone**' comes from the idea that players contain a glimpse of Finland, while they pick a puzzle image from Finna API and play a chosen image as a puzzle with other players. These image sources tell players about the Finland of the past. The background images and color themes were also designed to tell about representative images of Finland. For example, having a hint from the Finnish flag, blue color was picked as the main color and the red color was used as an accent color in terms of color theme.

Four separate pages were designed as **Intro page, Puzzle image selection page, Gameplay page, and Ranking page** were implemented. With an introduction of Aikakone, players move to the Puzzle selection page. It shows eight images from Finna API which show Finland of the past. Players can choose which image they want to play for their puzzle. In the gameplay page, they can solve the puzzle with the image they chose. They can also see the solving time, reset the game, turn on/off music and go to the ranking page to check the rank. The four main pages are shown in Figure 5.


Intro page
Title
Aikakone
Start button
Click to start
Puzzle image selection page
Puzzle image 1
Puzzle image 2
Puzzle image 3
Puzzle image 4
Puzzle image 5
Puzzle image 6
Puzzle image 7
Puzzle image 8
Puzzle image 9
Images of Finland in the past from Finna API
Choose an image
Game play page
Timer
Button controllers for flipping puzzle pieces in a row
Puzzle(6*6)
Button controllers for flipping puzzle pieces in a column
Feedback message
Congrats !! Awesome! You made it :D
Home
Sound
Retry
Rank

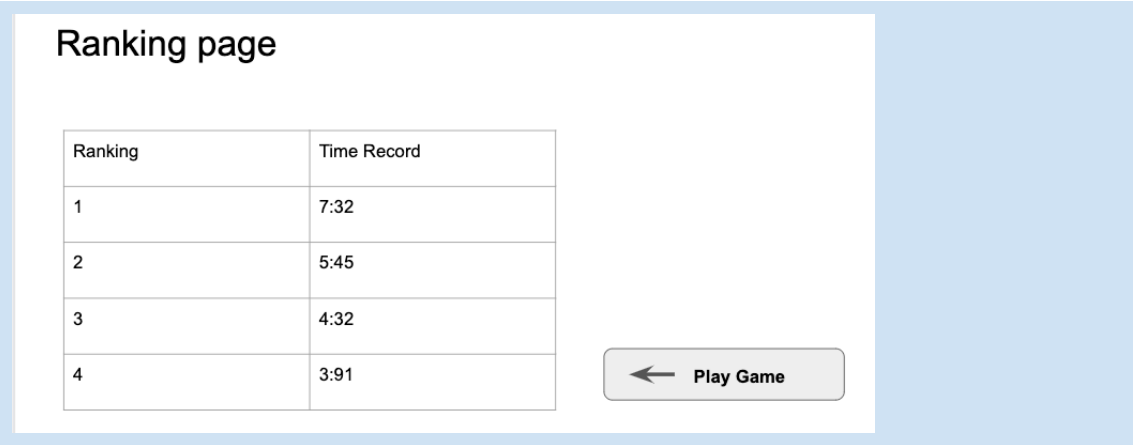


Figure 5. The four-page layout and user interface design. (Copyright 2020 HyeSoo Park)

Affordance was considered in the interface layer. It is a perceived property in which users think naturally about how to interact with objects. It helps players to connect the elements to the understanding of the real world and reduce a gap between how a game's space actually works and how players think it works - which is a player's mental model. For the perceptible affordance, the start button in the start screen and puzzle control buttons have a tween effect to indicate the players where to click to start or play a game. Clicking buttons or an image were designed to lead to each screen as the game flows - starting a game, choosing a game image, playing a puzzle, checking a rank without additional instruction pages.

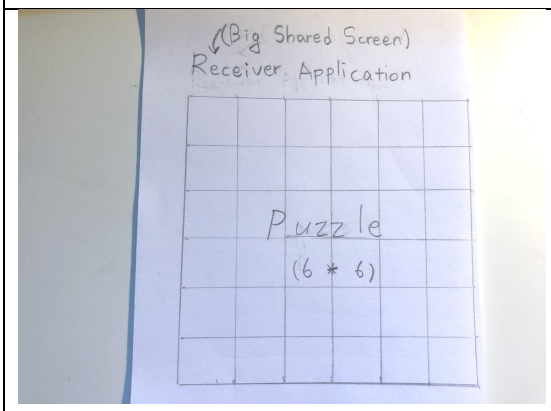
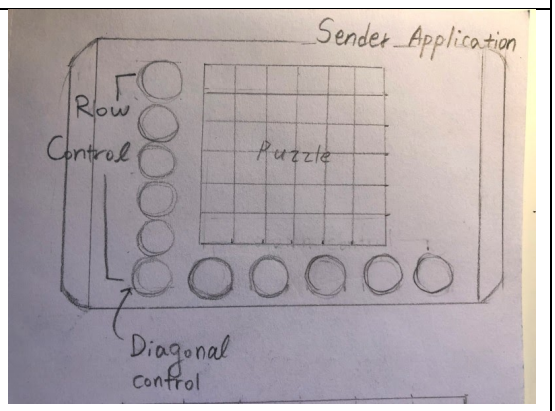
According to the initial specification, analysis, and game design of the project, the **core functionalities** of the game were decided for the implementation as below.

- a multiplayer game which supports cooperative play
- the connection between users and a server
- a puzzle game that has randomly flipped puzzle pieces.
- controller buttons for flipping rows and columns by click
- puzzle Images which fetched from Finna API
- puzzle-solving time to show to a user
- ranking display
- four different pages such as intro, choose an image, puzzle play and ranking in single page application
- responsive web design

3.4 Technical implementation

It has been an exciting time to make a playable puzzle game for the first time. The project was started in December of 2016. A **paper prototype** was drawn in January. A design of Receiver Application and Sender Application in the paper prototype is presented in Table1.

Table 1. A paper prototyping for Receiver Application and Sender Application. (Copyright 2020 HyeSoo Park)

Receiver Application	**Sender Application**
	

With the blueprint of the application, it was moved to development stages. It went through many trials and errors. Several obstacles appeared such as limited available time due to the family situation, lack of technical knowledge, and technical issues. However, the author of the thesis kept learning new programming languages and usages of related libraries. The technical implementation of the project was carried out mainly between January and June in 2018. At the end of June, it gradually came to fruition - the game became collaboratively playable and responsive to the different screen sizes of devices. Additionally, the frontend was deployed in Github.io and the backend was deployed in Heroku, so users can play the puzzle game with their own web browser. Finally, the first digital prototype of the game application was completed in the middle of July 2018. In the future, it is going to be updated by the user's feedback, people's advice, and additional ideas.

The technical implementation can be divided into a functional part and a design part. At first, an attempt to implement Aikakone with Chromecast was tried. Unfortunately, the difficulty in development - the difficulty in getting debug messages, lack of examples, and documentation led us to change the decision on technologies. The initial implementation with Chromecast can be seen in figure 6 [33].

Figure 6. Initial implementation with Chromecast. (Copyright 2020 HyeSoo Park)

After the first attempt to make a game with Chromecast was not succeeded, the technical obstacles needed to be circumvented with new technologies. With Chromecast technology, the client's sender application was an Android application and the shared application was a web-based receiver application. Instead, both clients and servers were decided to be web applications. **WebSocket** was chosen to serve their communication. **JS** was used in the initial implementation with Chromecast. For the new implementation, **Clojure** was selected as a backend language and **ClojureScript** was picked as a front-end language. The reason for language choices was the functional language paradigm generally leads to code more reasonable and less error-prone and other advantages that Clojure has such as simple syntax and consistency throughout the language.

It started with testing the WebSocket part with simple puzzle pieces sprites, using **Sente** which is a client and server web communication library for Clojure and ClojureScript. After testing the communication channel between a client and the server, the main puzzle game was created, using a **Phaser** which is an HTML5 game framework. The reason JavaScript game library was used instead of the ClojureScript game library is that Phaser provides useful game related functionalities such as canvas, preloader, sprites, animation, scaling, and great documentation. Especially some ClojureScript libraries were lacking features like sprite sheets or less performant.

This helped to build a puzzle game more efficiently for the web browser of a laptop and mobile. For example, Phaser helped to load and cut images into the right sizes for the game with its sprite sheet feature. After creating a puzzle board with puzzle pieces, randomizing the puzzle pieces and flipping puzzle pieces by row, col and diagonal flip control buttons were implemented. The following code shows the example of JavaScript **interop** from ClojureScript. The "." prefix in front of the method name is the syntax for calling a method of the following object. In this case, "phaser-loader" is the object, and "sprite sheet" is the method that is being called on the object. Listing 1 shows how puzzle pieces control buttons were created with a phaser-loader and sprite sheet. [34.]

```
(. spritesheet
    phaser-loader
    "puzzle"
    image-src
    (util/get-piece-width-height @util/puzzle-image-width)
    (util/get-piece-width-height @util/puzzle-image-height)
    (* util/row-col-num util/row-col-num))
 (. spritesheet
    phaser-loader
    "flip-buttons"
    "images/control-buttons.png"
    (util/get-button-width util/button-sprite-col-num)
    (util/get-button-height util/button-sprite-row-num)
    (* util/button-sprite-row-num util/button-sprite-col-num))
```

Listing 1. Creation of puzzle pieces and control buttons using Phaser library with JavaScript interop. (Copyright 2020 HyeSoo Park)

The game engine does not provide the functionality for the real time communication with server and other clients, thus Sente library was used for creating a real time duplex web application. It is a small client and server library which makes it easy to build reliable, high-performance real-time web applications with Clojure and ClojureScript.

The backend server acted as the intermediary of all the concurrent clients - which is players in the game. To sync multiple clients' events were forwarded to all clients except the direct sender. The following code is how it was implemented in Clojure backend code as seen in List 2. Figure 7 illustrates a **communication between clients and a server**. [35.]

```
(defn- send-data-to-all-except-message-sender
     [client-id message-type data]
     (doseq [uid (:any @connected-uids)]
     (when (not= client-id uid)
     (chsk-send! uid [message-type data]))))
```

Listing 2. Backend code that syncs all clients by forwarding events. (Copyright 2020 HyeSoo Park)

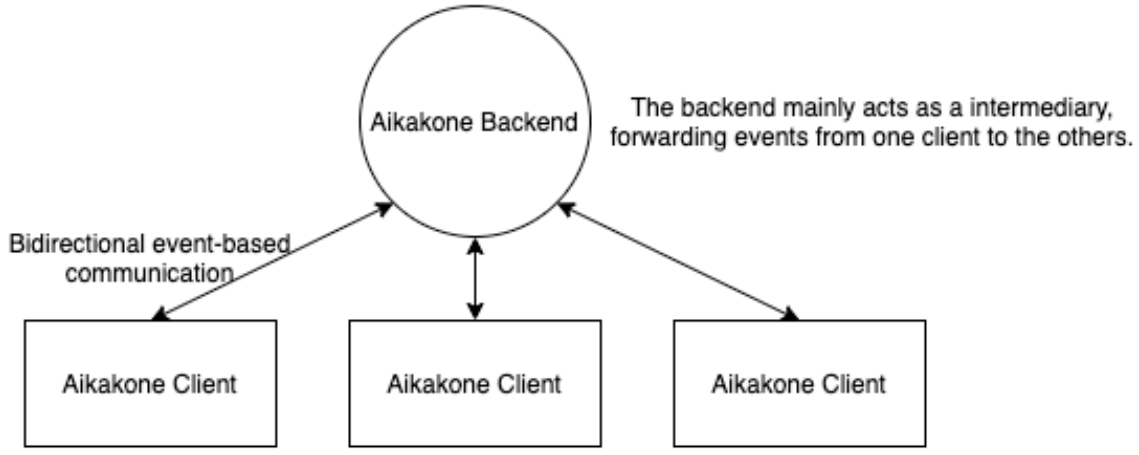


Figure 7. The connection between Client and Server (Copyright 2020 HyeSoo Park)

To support player's collaboration, all properties of game states are able to be transferred through WebSockets from Sente, so that players can share the information of the game state of a puzzle with the server and the other players. While they are sharing a game state information, players can synchronize their puzzles with each other and create a collaborative puzzle game idea. The detailed implementation is described in Figure 8.

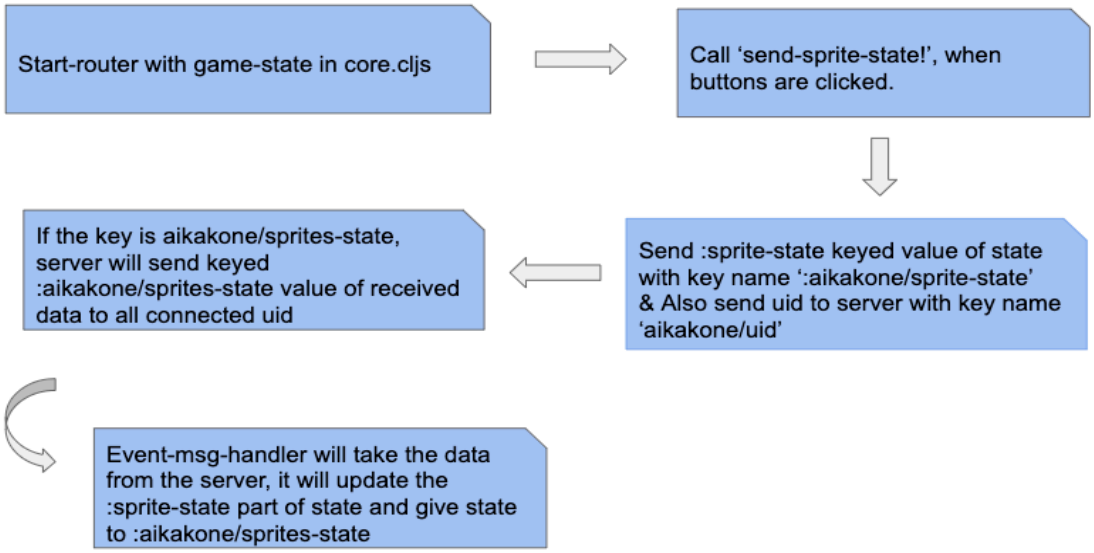


Figure 8. Communication between clients and a server to synchronize a game state between clients. (Copyright 2020 HyeSoo Park)

The client sends values of game state data with a key and client-id to a server. The server will send received data to all connected clients. Clients will use an event handler function to update the game state.

'**Re-frame'** was used to handle app state to route between different screens such as intro screen, image choose screen, game screen, and ranking. 'Re-frame' was developed for writing **SPA**s in ClojureScript and using **Reagent**, which is a thin wrapper around React Js. It is a functional framework and the functions transform that data. It has six steps to handle the event display the data into DOM such as event dispatch, effect handling, Query, View, and DOM. With this setup, Clojure data structure such as vector and map were used to represent the corresponding HTML, which is called Hiccup. This has the advantage of being able to use the full power of general programming language mixed with a markup language. Additionally, the syntax of the programming language is the same as the markup language itself which is quite handy. The framework implements the UI by using the cycle of 6 domino effects.

The **six dominoes** are illustrated in listing 3, 4, 5, and 6:

1. Event dispatch

```
(defn show-game! [search-word]
      (rf/dispatch [:set-game-img search-word])
      (rf/dispatch [:screen-change :game]))
```

Listing 3. Listing 3. Event dispatching code that changes screen to the game. (Copyright 2020 HyeSoo Park)

2. Event handling

```
(rf/reg-event-db
      :screen-change
      (fn [db [_ screen]]
      (assoc db :screen screen)))
```

Listing 4. Listing 4. Event handling code that computes a description of the required side effects to show different screens. (Copyright 2020 HyeSoo Park)

3. Effect handling

Side effects are performed on this step. The returned side effects to the SPA's application state are performed. It changes an application state.

4. Query

Data from the "app state" is extracted on this step.

```
(rf/reg-sub
      :screen
      (fn [db _]
      (:screen db)))

       ... in the middle of a component
      @ (rf/subscribe [:screen])
          ...
```

Listing 5. Listing 5. Query code that subscribes to screen change events. (Copyright 2020 HyeSoo Park)

5. View

View functions (Reagent components) that compute the UI DOM that should be displayed to the user.

```
(defn- puzzle-selection-view []
      (into
      [:div
      [:img {:style {:display ”block"}
      :src    "images/puzzle-selection-bg.png"
      :width ”100%"
      :height "100%"}]
      ...
```

Listing 6. Listing 6. Extract from reagent component that is written in Hiccup syntax. (Copyright 2020 HyeSoo Park)

6. DOM

The browser DOM nodes are mutated on this step. This step is handled by Reagent/React.

The timer and music were implemented as additional features. The solving time is displayed to players and the completion time is marked as their achievement in a ranking page. Players also can create simple music collaboration. Each control button has a different frequency of major scale in the 4th octave. Depending on a pitch frequency, it will make a different tone and take a different instrument to play with. Players can create harmony together, while they play a puzzle game by clicking control buttons.

Four separate pages such as **Intro page, Puzzle image selection page, Gameplay page, and Ranking page** were implemented. The user interface determines the looks and feels of the application. The representative images of Finland were drawn by the author of the thesis as background images. These background images and the color theme were applied to the user interface. Four different pages are shown in Table 2.

Images



Choose image:D
Helsinki
Bite
8.424

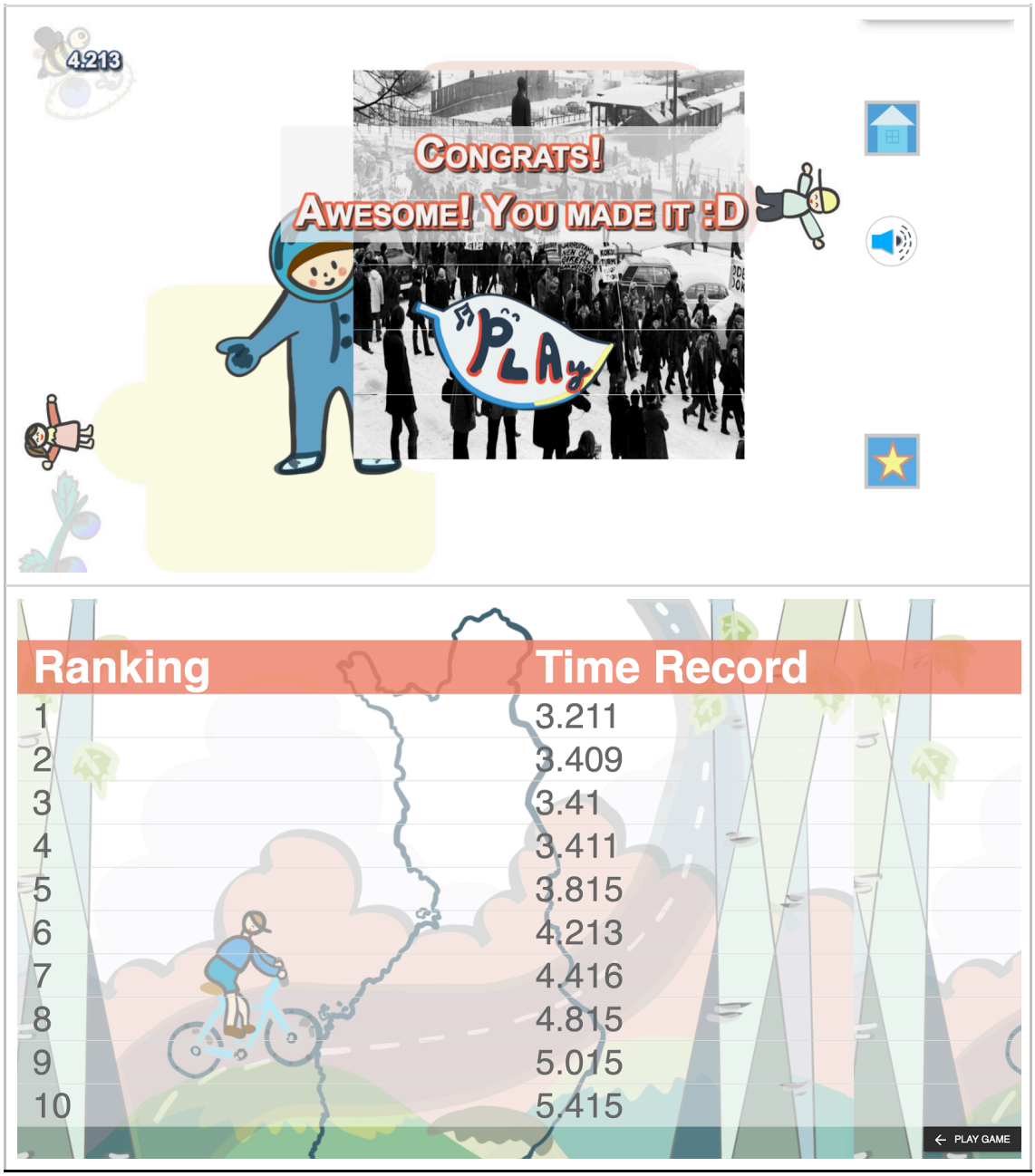


Table 2. The user interface of the Aikakone game in a web browser. (Screenshots of demo application) (Copyright 2020 HyeSoo Park)

Additionally, it is planned to be used in various types of devices such as mobile, laptops, pad, and a big screen. **The user interface was implemented to be responsive to different screen sizes** as seen in figure 9 and figure 10. Especially, in the case of mobile, it supports playing the game in both landscape mode and portrait mode on their mobile.

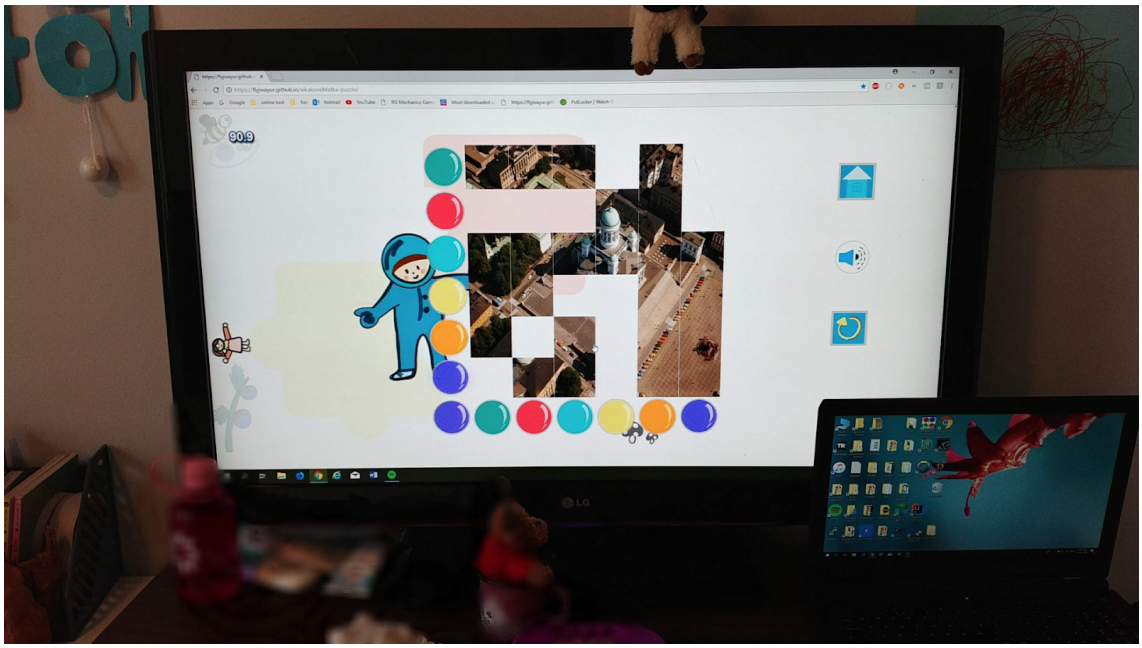


Figure 9. The Aikakone game is displayed on a big shared screen. (Copyright 2020 HyeSoo Park)

Figure 10. The Aikakone game is displayed on the mobile screen. (Copyright 2020 HyeSoo Park)

A front-end part of the first prototype was deployed in a GitHub page [36]. It rendered all images and puzzles smoothly. For the backend parts which include a server-side, it was deployed in the Heroku free platforms. If any users do not use it, the server in a Heroku

free version sleeps and becomes slow. But given that even it is enough to host the 'Aikakone' game, the performance seems to be reasonably usable.

4 Results

4.1 User testing

User testing were conducted three times. Players tried the Aikakone game on different devices such as mobile, laptop, and PC at once. The web browser was a choice of the platform for the game, it enables a game to be played locally and in different places. A co-located setting and multiplayer mode were pursued as much as possible for meaningful testing. The first and second user testing was conducted in a co-located setting and several players played in multiplayer mode, but in the third user testing, players played a game in a single-player mode.

The first user testing was performed with 2 friends at a cafe near Leppävaara on 16th June in 2018. In the playtesting, two players played a puzzle game together via their own mobile. It proved that it is playable as a multiplayer game. As feedback, they suggested giving indicators of whether other players join the same game or not using an avatar or character. The second user testing was carried out with 3 friends at home on the 7th of August in 2018. Gameplays were observed as seen in figure 11.

Figure 11. Co-playing the Aikakone game on a laptop and iPad. (Copyright 2020 HyeSoo Park)

They actively interacted with each other to ask each other many questions such as which image they choose and how to flip the puzzle pieces. They felt the way of puzzle control - which is flipping puzzle pieces in a row or column - is very unique. They felt it was very easy to play and they would like to have more difficult challenges or levels. The most interesting behavior was that they would like to compete with each other rather than cooperating in the beginning. Later, they figured out that if they compete, it became more difficult to solve the puzzle, then they played the game cooperatively. The third user testing was executed in Wärkfestval where makers can demonstrate or share the prototype, the process of making, success, or trials with other participants. It was on the 10th of November in 2018 in the Iso Omena library. The author of the thesis participated in the festival as a maker and presented the game with visitors. [37.] Through the festival, five visitors visited the booth. Each person visited at a different time and they tried the Aikakone game individually. The player tasks were prepared as follows:

- Move from an intro page to the next page.
- Choose one image for playing a puzzle.
- Begin the puzzle.
- Find the controller to play a puzzle.
- Solve the puzzle.

- Check how fast you solve the puzzle.
- Find your ranking.
- Turn on and off the background sound
- Find the other image for the next play.
- Play the same puzzle with other people.

4.2 Analysis

The results (Appendix 1 and Appendix 2) of user testing brought numerous insights and from the feedback, it becomes clearer to know how actual play can be, how players feel about it, and what to improve. When the game was introduced, they were curious about the game. Except for one player in the 3rd user testing, all other players thought the game was fun and playable. Most of the people felt it was too easy or easy to play. Many players want more challenges or variety in a game. Active players explored the game by their own interaction. On the other hand, several players preferred to ask for more instruction right away or looked at how other players were playing. Several players easily found out how to play a game, but other players were not.

The results (Appendix 2) of the surveys and observations in the 2nd and 3rd user testing show that the most important parts to improve were 'adding more instruction of gameplay and how to set up multiplayer sessions. Additionally, it showed that most of the players did not notice that it is a multiplayer game, thus they wondered why their puzzle pieces were being flipped automatically. Several users guessed computer AI is flipping some parts of the puzzle. They also wondered whether they were at the same time or needed to compete with each other. It clearly showed a lack of instruction about multiplayer games. One of the players suggested a great idea which is adding visibility of other players via avatar or characters. Regarding menus, about half the players in 3rd user testing felt easy to navigate through the menu button, but others felt it was difficult.

The bright sign was that several players felt that the game helped them to talk or discuss to each other to achieve the goal of the game. They discussed the plan to match all the puzzle pieces together. After discussion, some of the players decided to take a turn to play, so that they will not interrupt each play. Several players in the 3rd user testing said that they feel connected when they play a game with friends or other people. They all

agree on team building games or multiplayer games can be quite helpful for them to learn how to collaborate with others. Most of the answer was 4 points out of 5 for how much it could be helpful for learning collaboration. 1 was not at all and 5 was very helpful in scale. What to improve for the game was suggested as following:

- Add more instruction in general.
- Add visibility of other players and indicate what is each player's role.
- Providing more pictures to choose and make them appear randomly than static images.
- Add more challenges and variety to the game (not too easy and simple).
- Add stories about the pictures.
- Give a better distinction between interactive UI elements and background images.

The user testing has proved that interdependency, common goal and shared control in a game encourage players to talk to each other to achieve the goal of the game. Once players tried to compete with each other but later they realized that they needed to collaborate to match all the puzzles. **It shows that social interactions between players and game mechanics generated social play which can be a competitive play or cooperative play**. Many players have experienced team building games or multiplayer games. All of them have positive opinions about the statement - those games can be quite helpful for them to learn how to collaborate with others. Even though a number of spaces needs to be improved to make a game more meaningful, it tells the game meets the goal of the project which was enhancing the player's collaboration.

5 Conclusion

The overall process of designing and building the Aikakone game was explained in this thesis. The purpose of this project was to demonstrate the viability of designing a game that enhances the player's cooperation in a playful way. The first prototype of a game was delayed by one year due to personal matters, lack of technical knowledge, and technical issues, but it was continuously developed by steady learning and a playable game was delivered at the end. Even though several features needed to be improved in the game, it met most of the initial goals.

Clojure and ClojureScript enabled representing game data expressively. It was easy to represent the state of the game board because it is easy to write literal data and manipulate data in the language. It encourages to minimize the side effects by isolating them from other code. It means Clojure and ClojureScript allow mutations on a state when it is necessary, and mutation of the game state is an integral part of the game. In the game, the game state was updated, mutating a game state with an atom. Since it supports interoperating the functions and properties in the host platform, the game leveraged useful game functionalities from the Phaser game framework. Keeping in mind that functional language is rarely used for game development, it was a very interesting experimental project to undertake. Reagent and Re-frame enabled to make user interfaces and different pages. Using Sente, it was possible to make stable communication channels between multiple clients and a server. Game states were able to be synced between all clients. The game benefited from those libraries to create a playable game.

Depending on how game elements are combined, players create different ways of playing and dynamics. These game mechanics and elements make an effect on social play, player experience, and social interaction. Based on these studies, the interdependence, common goal and shared control were designed to the game. The case study revealed that this game design helped players to interact with each other more and support cooperating with each other towards the common goal. For better player experiences, the perceptible affordance was applied to design the user interface. It helped players to know how to start a game and how to navigate the menus.

Several ideas are suggested for a future refinement such as checking the number of flips it takes to solve a puzzle for a rank and making input fields for usernames to display it with their achievement. The author of this thesis assumed that checking the number of flips might give time for players to think about the game, reflect, and try out different things with a game. Based on the users' feedback, displaying other users with an avatar or character besides the puzzle can be helpful for players to recognize other players. Additionally, the game difficulty is not challenging enough for players after a few trials. More dynamic images for the puzzle and various challenges will make a game more excited.

Since the scope of the project was the game design and implementation, the subjects of the user testing were limited to the player's background and the game itself. For further studies, studies on methods of physiological measures and gameplay metrics are recommended to assess the social aspects of a game. Since the game was developed to enhance the player's collaboration, evaluating the impacts of game patterns on social factors of a game possibly brings more meaningful findings in the project.

Finally, the case study proved that a certain pattern of game design encouraged the players to interact with each other more and also generated the cooperative play experience. The author of this thesis hopes to see more new experiments and ideas of game design and development.

References

1 Glatter R., Digital Addiction: A Recipe for Isolation, Depression and Anxiety [Online] Forbes; 13 April 2018. URL: https://www.forbes.com/sites/robertglatter/2018/04/13/digital-addiction-a-recipe-for-isolation-depression-and-anxiety/#4df0daaf5f6b. Accessed 2016 - 2020

2 Kappen D. et al. Exploring Social Interaction in Co-located Multiplayer Games. CHI EA `13: CHI `13 Extended Abstracts on Human Factors in Computing Systems, p 1119 - 1124; 13 April 2013.

3 Thilmany J. Girls and Video Games: Playing for mental health: Mechanical Engineering. Vol. 133 Issue 4, p16-16; April 2011.

4 Silva G. et al. Collaborative Strategies in Multitouch Tabletop to encourage Social Interaction in People with Autism. Interacción '14: Proceedings of the XV International Conference on Human-Computer Interaction. September 2014.

5 Collen M, Sharp J. Games, Design and Play: a detailed approach to iterative game design. Addison-Wesley Profession; June 2016.

6 Gibson J. Introduction to Game Design, Prototyping, and Development: From Concept to Playable game with Unity and C#. Addison-Wesley Professional; July 2014.

7 Schell J. The Art of Game Design, 3rd Edition. AK Peters/CRC Press; July 2019.

8 Marczewski A. Even Ninja Monkeys Like to Play: Gamification, Game Thinking & Motivational Design. Blurb.6 October 2015.

9 Emmerich K, Masuch M. The Impact of Game Patterns on Player Experience and Social Interaction in Co-Located Multiplayer games. CHI PLAY `17: Proceedings of the Annual Symposium on Computer-Human-Interaction in Play, p411- 422; October 2017.

10 Reuter C. et al. Game Design Patterns for Collaborative Player Interactions. DiGRA Conference; 2014

11 Quandt T, Sonja Kröger. Multiplayer: The Social Aspects of Digital Gaming, p147-156. Routledge, New York; 2014

12 Dmitri S. Web Development with Clojure, 2nd Edition. Pragmatic Bookshelf; July 2016.

13 Shankar A. Multiplayer with WebSockets. In:Pro HTML5 Games. Apress, Berkeley, CA; 2012

14 Google. Google Cast, URL : https://developers.google.com/cast/docs/developers; Accessed 2016 - 2017

15 Loyola Marymount University. Programming paradigms[Online]. Accessed 2018-2020

16 Maurizio Gabbrielli,Simone Martini. Programming Languages: Principles and Paradigms. Springer London Dordrecht Heidelberg New York; 2010.

17 Mozilla. JavaScript [Online], Mozilla; 10 May 2018. URL: https://developer.mozilla.org/bm/docs/Web/JavaScript. Accessed 2016 - 2020

18 Hughes H. Why Functional Programming Matters. Research Topics in Functional Programming. Addison-Wesley p17 - 42; 1990

19 Cass S, Bulusu P. The Top Programming Languages 2018 : Find the programming languages that are most important to you[Online], IEEE spectrum, 31 July 2018. URL: https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2018.Accessed 2016 - 2020

20 StackShare, Scala [Online], 2018. URL: https://stackshare.io/scala. Accessed 2018 - 2020

21 Hickey R. Clojure: Companies [Online], 24 October 2018.
URL: https://clojure.org/community/companies. Accessed 2018 - 2020

22 Hickey R. Clojure: Features and rationale [Online]. 01 April 2020.
URL: https://clojure.org/about/rationale. Accessed 2016 - 2020

23 Hickey R. ClojureScript: Features and rationale [Online]. 01 April 2020. URL: https://clojurescript.org/about/rationale. Accessed 2016 - 2020

24 Lambda Island. Game Development with Clojure/ClojureScript [Online]. 08 December 2020. URL: https://lambdaisland.com/blog/2016-12-08-game-development-with-clojure-clojurescript. Accessed 2016 - 2020

25 Slant. What are the best functional programming languages for game development [Online]. URL: https://www.slant.co/topics/1014/~best-functional-programming-languages-for-game-development. Accessed 2020

26 Davey R. Phaser [Online], Photon Storm Limited; 2020, URL : https://github.com/photonstorm/phaser. Accessed 2016 - 2020

27 Hickey R. ClojureScript: JavaScript API [Online]. 01 April 2020. URL: https://clojurescript.org/reference/javascript-api. Accessed 2016 - 2020

28 Modrzyk N. Reactive with ClojureScript Recipes: Functional Programming for the Web. Apress; September 2017

29 Thompson M. Reframe: A Reagent Framework For Writing SPAs, in ClojureScript [Online]. URL: https://github.com/day8/re-frame. Accessed 2016 - 2020

30 Taoussanis P. Sente: Realtime web communications for Clojure/ClojureScript [Online]. URL: https://github.com/ptaoussanis/sente. Accessed 2016 - 2020

31 Finna. Finna API (in English) [Online]. URL: https://www.kiwi.fi/pages/viewpage.action?pageId=53839221, https://finna.fi/?lng=en-gb. Accessed 2016 - 2020

32 McGonigal J. SuperBetter: How a gameful life can make you stronger, happier, braver and more resilient. Penguin books, New York; 2016

33 Park H. Finna puzzle game: Initial implementation with Chromecast [Online]. January. 2017. URL: https://flyjwayur.github.io/finna-puzzle/, https://github.com/flyjwayur/finna-fetchAPI, https://github.com/flyjwayur/finna-game-chromecast-receiver, https://github.com/flyjwayur/finna-game-android-sender. Accessed January 2017

34 Park H. Aikakone puzzle game frontend [Online]. 01 April - 09 November 2018. https://github.com/flyjwayur/aikakoneMatka-puzzle/. Accessed 2018 - 2020

35 Park H. Aikakone puzzle game backend [Online]. 21 May - 16 June 2018. URL: https://github.com/flyjwayur/aikakonematka-puzzle-backend. Accessed 2018 - 2020

36 Park H. Aikakone Live Demo (The first prototype) [Online]. 2018. URL: https://flyjwayur.github.io/aikakoneMatka-puzzle. Accessed 2018 - 2020

37 Wärk ry. Wakfestval: Family-friendly festival of science, art, and technology [Online]. 2018. URL: https://www.warkfest.org/en/, https://www.warkfest.org/call-for-makers/?fbclid=IwAR2wM5K5T00EWkz7Oiusk5l8ujWV1oEvluhwm2_6tj9E1kI1jDmJK2TB_OU, https://www.warkfest.org/makers/#aika. Accessed October - November 2018

1. Results of the 1st and 2nd User testing

The first user testing was with 2 friends at a cafe near Leppavaara on 16th June in 2018. The second user testing was carried out with 3 friends at home on the 7th of August in 2018. The questions about the player's background and the game itself were asked using Typeform after gameplay. The first and second testing used the same questionnaires, thus the results are in the same tables. The result of the player background questionnaire is shown in Table 3 and the result of the game survey is illustrated in Table 4.

Table 3. A player's background questionnaire of the first and second user testing. (Copyright 2020 HyeSoo Park)

	Player 1	**Player 2**	**Player 3**	**Player 4**	**Player 5**
What is your age?	Between 25 and 29	Between 25 and 29	Between 30 and 34	Between 25 and 29	Between 30 and 34
What is gender?	Male	Male	Female	Female	Male
Do you like playing a game?	Soso	Very much	Soso	Soso	A little bit
How often do you play games?	rarely	Every day	once in a week	One time per week	Once a month
What is your favorite game? What do you like most about that game?	tomb raider. I like it because of the mixture between puzzle-solving and action	Mount and Blade Warband. Multiplayer and sandbox system.	Roleplay adventure pc game. It gives me the same chance to live in another world.	Super Mario. I like music and interface	Tetris, it's simple
Have you played a game in a classroom? if yes, which game do you remember playing in the classroom?	fiends	Yes, Stellaris and Europa Universalis	Some. Hangman (word game).	Yes. Kahoot	Tetris
What did you most like about it? What did you learn from that game?	puzzle-solving. to tolerate the class	It wasn't educational. I learned to be better at spreadsheets.	Playing as a team. Tactics of choosing letters and finding the right word in the end.	I like its concept. I know more information and react faster	Concentration and completing a task

Do you use a mobile? If yes, have you sometimes felt isolated from people, lonely or anxious, using it?	a couple of time	Not really.	Yes. Not really, as I'm most of the time on do-not-disturb mode.	Yes and I sometimes felt like that when using it	Yes, yes
Do you have any experience that technology helps you collaborate with others? What was it?	confluence, version one.	Slack, Git, emails	Yes. Computer-assisted manuals.	Yes, Slack and Trello are really good tools when I need to collaborate with other people at school or workplace	Yes, game consoles like Xbox

Table 4. A game survey of the first and second user testing (Copyright 2020 HyeSoo Park)

	Player 1	**Player 2**	**Player 3**	**Player 4**	**Player 5**
Was this game fun and playable for you?	Yes	Yes	Yes	Yes	Yes
What was the most fun part of this game for you?	Annoying my collaborators	completing the task	fighting with others	When working together with someone to achieve a common goal.	It was simple and straightforward
Was this game easy to get started and play?	Very easy	Easy	Very easy	Soso	Easy
Was this game challenging enough for you?	Too easy	Soso	Easy	Easy	Easy
Did you play with other players? How was your experience?	Exiting when I starting annoying them	yes and we ended up playing against each other	Fun :) But need more content, a maybe different mode	First quite confusing, but after understanding it we managed to solve them.	It's not clear what is the role of the other person. Are we competing? Or in the same team?

Does this game help you to collaborate/cooperate with other players? If yes, why do you think that?	Makes us talk		we agreed to play in turns	Yes. Otherwise, the game will never end, if not collaborating with the other person.	It wasn't clear for me
Did you learn from others? If yes, what did you learn from them?	Fast clicking	how to cooperate	patience	To wait for your turn.	In the game? No
Did you make any meaningful choices in this game? If yes, what was it?	Giving a role number to everyone		we organized together	Finding the fastest way to finish it.	Small choices, like which row to click
What do you like least/most about this app?	Not much randomness	user interface	not enough variety/multiplayer capabilities	Without instruction, it was difficult first to understand why things move without our own actions. Working with other people.	I liked the fact that it was multiplayer. It would be better if it was clear what is the role of the other player
Do you have any thoughts on how to improve this game? (What is missing in this game?)	More randomness for images	competitive mode would be nice and to know more about the pictures, where they are from etc	UI responsiveness for different screens, more picture options to choose, and make them appear randomly. It would be cool if you can't choose the image at all and can't see it before the game starts.	Visibility of other players.	Some explanation or UI elements to clarify what the other players do

2. Results of the 3rd User testing

The third user testing was executed in Wärkfestval where makers can demonstrate or share the prototype, the process of making, success, or trials with other participants on the 10th of November in 2018 in the Iso Omena library. Five players were surveyed about both the player's background and the game itself. The result of the player background questionnaire is shown in Table 5 and the result of the game survey is described in Table 6.

Table 5. A player background questionnaire in Wärkfestval. (Copyright 2020 HyeSoo Park)

	Player 1	**Player 2**	**Player 3**	**Player 4**	**Player 5**
Do you like playing a game?	Very much	Very much	Very much	Very much	So so
How often do you play games?	1 - 2 times in a week	Several times a day	a few times in a month	a few times in a month	Very rare
Which devices do you normally use for playing a game?	Others	PC	PC	Mobile	Mobile
Have you played any 'team building game' or played a 'multiplayer game'?	TRUE	TRUE	TRUE	TRUE	FALSE
If you tried them, what was the name of the game?	Counter-Strike	Terraria	don't remember	Guitar hero	I did not try the multiplayer game.
What did you like most about, when you play together with other friends or other people?	Feeling connected	All the above answers	More fun	All the above answers	Feeling connected
Do you think team building games or multiplayer games help you to learn how to collaborate with others?	TRUE	TRUE	TRUE	TRUE	TRUE
If yes, how much was it helpful? If you scale them from 1(Not at all) to 5(very helpful)	4	4	4	5	3
If yes, could you elaborate a little bit more about how 'team building game' or 'multiplayer game' help you to collaborate with others in real life?	All the above answers	All the above answers	Developing decision-making skills	Developing decision-making skills	Interacting with others

Table 6. A survey about the game in Wärkfestval. (Copyright 2020 HyeSoo Park)

	Player 1	**Player 2**	**Player 3**	**Player 4**	**Player 5**
What was your first impression/feeling on the game application?	Idea was good	Kid-friendly	really nice and cozy	I was not sure exactly how to play the game but it was cute to look at.	Very interactive
Was this game fun and playable for you?	TRUE	TRUE	TRUE	FALSE	TRUE
Was this game challenging enough for you?	Easy	Too easy	Soso	Easy	Soso
How was it easy to figure out how to start and play a game?	Very easy	Easy	Easy	Difficult	Soso
Did you play with other players? How was your experience?	Don't know :)	Fun to make up own rules	it's cool	No.	I played alone but I think it would be more fun with more players
Was it easy to figure out that the Aikakone game is a 'real multiplayer game'? If it is not, could you suggest any idea to improve?	Not really, more instructions.	More instructions on how to set up a multiplayer session	was easy	No. I did not know there was an option to do a real multiplayer game.	For me, I didn't get it at first that it's a multiplayer game! Maybe a hint would be good in the start
How easy is it to move between different pages(It has four pages such as intro page, puzzle image selection page, puzzle play page, ranking page)?	Soso	Soso	Soso	Easy	Easy
How easy to figure out how to use different navigation buttons(a start, play, reset, turn on/off the sound, ranking button)?	Easy	Difficult	Easy	Difficult	Very easy
Does music (background music and pitches(or notes)from control buttons) make you interested in this application more Or was it pleasant for you?	Music was good	No positive or negative impact	it was pleasant	Yes. The music was cute and nice.	Yes it was

Do you have any thoughts on how to improve this game? (What is missing in this game? ex) More instructions)		Better cue and distinction between interactive UI elements and background images	more cool pics)	Generally, I wish it was easier to navigate the game and also hope it was more challenging.	I think a bit more instructions or maybe an interactive way of showing briefly how to play the game would be good

3. MIT License

The MIT License (MIT)