

# Dynamic Generation of Game Objects in a Casual Mobile Game Using a Generative Algorithmic Music System

Julia M. Angerer



MASTERARBEIT

eingereicht am  
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juni 2018

© Copyright 2018 Julia M. Angerer

This work is published under the conditions of the Creative Commons License *Attribution-NonCommercial-NoDerivatives 4.0 International* (CC BY-NC-ND 4.0)—see <https://creativecommons.org/licenses/by-nc-nd/4.0/>.

# Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, June 21, 2018

Julia M. Angerer

# Contents

<b>Declaration</b>	<b>iii</b>
<b>Preface</b>	<b>vi</b>
<b>Abstract</b>	<b>vii</b>
<b>Kurzfassung</b>	<b>viii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Objective . . . . .	1
1.3 Document Structure . . . . .	2
<b>2 Related Work</b>	<b>4</b>
2.1 Algorithmic and Generative Composition . . . . .	4
2.1.1 Interactive Digital Instruments . . . . .	4
2.1.2 Composition Techniques . . . . .	5
2.1.3 Related Applications . . . . .	6
2.2 GameFlow and Playability in Casual Mobile Games . . . . .	7
2.2.1 Casual Games Discussion . . . . .	8
2.2.2 GameFlow . . . . .	10
2.2.3 Playability . . . . .	12
<b>3 Audio System: Conception</b>	<b>15</b>
3.1 Characteristics of LaLuup . . . . .	15
3.1.1 Concept . . . . .	16
3.1.2 Requirements . . . . .	19
3.1.3 Aesthetic goals . . . . .	21
3.2 System Details . . . . .	22
3.2.1 Detailed Gameplay Description . . . . .	22
3.2.2 Tools and Design Decisions . . . . .	23
<b>4 Audio System: Implementation</b>	<b>25</b>
4.1 The Pärt Algorithm . . . . .	25
4.1.1 General explanation . . . . .	25
4.1.2 Specific implementation . . . . .	25

4.2	System Architecture . . . . .	27
4.2.1	Basic Audio Modules . . . . .	27
4.2.2	Composing Modules . . . . .	29
4.2.3	Other Structuring Modules . . . . .	30
4.3	Prototype Configuration . . . . .	33
4.3.1	Sound Generation . . . . .	33
4.3.2	Level Design . . . . .	34
<b>5</b>	<b>Dynamic Object Creation</b>	<b>35</b>
5.1	Dynamic Objects in LaLuup . . . . .	35
5.1.1	Idea and Evaluation Concept . . . . .	36
5.1.2	Implementation . . . . .	37
5.2	Chosen Evaluation Setups . . . . .	38
5.2.1	Static Version . . . . .	38
5.2.2	Event-based Collectibles . . . . .	39
5.2.3	Event and Time-based Optional Collectibles . . . . .	40
<b>6</b>	<b>Evaluation</b>	<b>43</b>
6.1	Expert Reviews using Heuristics . . . . .	43
6.1.1	Method . . . . .	44
6.1.2	Results . . . . .	49
6.1.3	Improvements . . . . .	50
6.2	Playability Test using GEQ . . . . .	52
6.2.1	Method . . . . .	52
6.2.2	Results . . . . .	54
<b>7</b>	<b>Concluding Debate</b>	<b>58</b>
7.1	Result Analysis . . . . .	58
7.2	Further Prospects . . . . .	60
7.3	Conclusion . . . . .	61
<b>A</b>	<b>CD-ROM/DVD Contents</b>	<b>62</b>
A.1	Project . . . . .	62
A.2	Thesis . . . . .	62
A.3	Evaluations . . . . .	62
	<b>References</b>	<b>63</b>
	Literature . . . . .	63
	Audio-visual media . . . . .	65
	Online sources . . . . .	66

# Preface

When looking back, I remember always being interested in the way interactive applications and especially games are developed. Starting with my education I spent a lot of time on researching, investigating and (re)creating game mechanics and aesthetics. These collected skills eventually gave me the chance to specialize on interactive applications and game development not only as a student but also as an employee, finally being able to realize an interesting idea from scratch in a small team of visionaries, inventors and designers. Getting here would not have been possible without the support of my family, friends, fellow students, thesis supervisors and co-workers at Kunabi Brother, therefore I want to thank them for letting me follow my dreams.

When Kunabi Brother told me about the game idea they had prototyped on paper, I instantly was amazed by it and wanted to create this game concept as a digital prototype. I had already found a liking in games that focus on exploration and experimentation rather than competing in highscores or just solving level by level before. After all, playing games to me is about the fascination for things that others realize with their imagination by developing worlds that no one has seen and creating experiences no one has sensed before. This again shows that at least when it comes to interactive media, imagination truly has no limits.

I hope that people will enjoy exploring and experimenting with *LaLwup* as much as I enjoyed developing the audio system, integrating it into the game and creating the prototype versions for this thesis.

# Abstract

This thesis elaborates on the usage of dynamic content in casual 2D puzzle games for mobile devices to increase variation in gameplay, mechanics and audiovisual aesthetics and measure the effects of these objects based on a heuristic model established in games literature. It therefore focuses on a particular game prototype called *LaLuup* which utilizes a generative algorithmic audio system (the thesis project) to emphasize exploration and experimentation within the levels.

At the beginning, related projects, applications and generative composition techniques are described, giving insights into the influences they have on the development of *LaLuup*. Furthermore, important game theory such as casual games, playability and the *GameFlow* model is discussed in correspondence with the prototype.

As a next step, the thesis project is introduced by first stating concepts, requirements and desired features and their aesthetic effects. A detailed description of the gameplay and audio generation algorithm is then followed by insights into the development including details about the internal structure and components of the project. The general configuration description of the game prototypes is given to set the scope of the following evaluations. After investigating dynamic content in regards to *LaLuup*, three different prototype versions either containing or not containing dynamic content utilized in various ways are developed.

Finally two evaluations are conducted, the first one performed with experts to find usability and playability issues based on *GameFlow* heuristics. After improving the prototypes, the second evaluation using the *Game Experience Questionnaire* is done, depicting differences in the game versions and matching the results again to the *GameFlow* model. The notably good and stable results show rather small differences between the prototypes, however, the use of dynamic objects may contribute to a more challenging gameplay experience.

# Kurzfassung

Diese Arbeit geht auf die Verwendung von dynamischen Inhalten in *casual 2D puzzle games* für Mobilgeräte wie etwa Smartphones und Tablets ein. Dabei sollen dynamische Objekte die Variation im Spielablauf und in den Mechaniken sowie die audiovisuelle Ästhetik erhöhen. Um die erzielten Effekte dieser Objekte auf Basis eines heuristischen Modells aus der Videospieelliteratur messen zu können, wird ein spezifischer Spielprototyp entwickelt. Dieser Prototyp mit dem Namen *LaLuup* verwendet ein generatives algorithmisches Audiosystem (das Projekt dieser Arbeit) um den Erforschungs- und Experimentiercharakter des Spiels hervorzuheben.

Zu Anfang der Arbeit werden themenverwandte Projekte, Applikationen und generative Kompositionstechniken behandelt um deren Einflüsse auf *LaLuup* darzustellen. Weiters wird über mit dem Projekt zusammenhängende Videospieletheorie im Bezug auf *Playability*, *Casual Games* und das *GameFlow* Modell diskutiert.

Im nächsten Schritt werden die Konzepte, Voraussetzungen und notwendigen Features des Audiosystem-Projekts vorgestellt. Danach folgt eine detaillierte Beschreibung des Spielablaufs und des Algorithmus zur Audioerzeugung sowie Einblicke in die Entwicklung einschließlich einer Beschreibung der internen Struktur und deren Komponenten. Weiters wird eine generelle Konfiguration für die Spielprototypen festgelegt um diese auf die folgende Evaluierung vorzubereiten. Nachdem dynamische Inhalte im Bezug auf *LaLuup* untersucht wurden, können schließlich drei verschiedene Prototypen mit und ohne unterschiedliche dynamische Objekte entwickelt werden.

Daraufhin werden zwei verschiedene Evaluierungen durchgeführt: Die erste Evaluierung besteht aus mehreren Expertengesprächen, bei denen vor allem auf *Usability* und *Playability* Probleme mithilfe der *GameFlow* Heuristiken eingegangen wird. Nach dem Anpassen und Verbessern der Prototypen soll die zweite Evaluierung (eine Spielebefragung auf Basis des *Game Experience Questionnaire*) Unterschiede zwischen den einzelnen Spielversionen im Bezug auf das *GameFlow* Modell feststellen. Die auffallend guten und stabilen Ergebnisse zeigen hierbei nur kleine Unterschiede zwischen den Prototypen, allerdings trägt die Verwendung von dynamischen Objekten zu einer herausfordernden Spielerfahrung bei.

# Chapter 1

## Introduction

### 1.1 Motivation

When characterizing recent well-known casual mobile games, one can conclude that most of them have an abstract or simplified look and their developers focus on generating dynamic content rather than designing level by level by hand. While one may argue that this results in very interchangeable, even replaceable game design, it certainly saves money, time and also matches the current aesthetics in digital interactive media. Another point that should not be missed is that casual games frequently utilize dynamic content generation to facilitate gameplay diversity and replayability. Especially mobile games that are only played for a few minutes at a time profit from dynamic game content such as spawning game items by certain rules, adapting levels based on a difficulty grade and even creating them based on player profiles. However, there are some casual mobile games that lack many of these dynamic factors and are still played frequently. This raises the question if player enjoyment (which in fact induces people to play games) is really connected to dynamic content generation.

In the case of the corresponding thesis project *LaLuup*, an audio-driven 2D puzzle game prototype, gameplay is centered around an algorithmic composition system based on Arvo Pärt's Tintinnabuli style (which is explained in detail in chapter 3 and chapter 4) that offers many solutions to a single level. The game prototype should therefore encourage the player to explore and experiment with the system rather than just solving the level and starting the next. In this case dynamic content could create a diversion from the exploration factor of the game and its auditive aesthetics, but in fact it could also be helpful to increase variability and long term replayability. In performing experiments with different versions of the game prototype, the author wants to explore the effects of dynamic elements to player enjoyment and the aesthetic experience to find a fitting solution for *LaLuup* that can also be used as a reference for similar casual mobile games.

### 1.2 Objective

The idea of the thesis project is based on a paper prototype for an interactive application called *LaLuup* by the company *Kunabi Brother*. As already mentioned, *LaLuup* is an audio-driven 2D puzzle game for mobile devices that is centered around the generation

of sounds and harmonics with user input and corresponding algorithmic composition. In the levels different puzzle tiles are spread across a hexagonal grid and their patterns can be drawn and redrawn by the user. When the users tap grid tiles, impulses recreate the pattern they painted with the beat. The puzzle is solved by generating a loop within the grid and thereby trapping the impulse in it. In addition to the puzzle tiles that spawn impulses, there are also tiles that have to be hit by the impulses and in this way collected to solve the level. These so-called collectibles can either spawn and stay in their positions without changing over time or shift positions dynamically after a certain time has passed or a certain event has happened.

The most criticized factor in the informal user tests conducted while developing the game prototype for *LaLuup* was variation. Many players stated independently of each other that they would need more variation in sound aesthetics as well as gameplay to keep focused on the game longer than a few minutes. In fact, as it is a game prototype, there are still feature extensions and new game objects to be introduced during further development to increase variation. Another aspect that should be mentioned is that the tested levels, as they are the first ten in level progression, could not be too difficult for the test users playing for the first time. Introducing new game objects may therefore not be the best way to raise variation. Additionally, adding those new game objects does increase difficulty in understanding the game within minutes, which is very important because of the way casual mobile games are played.

In terms of raising variation and thereby replayability without expanding the difficulty considerably, it could instead help to utilize the already available elements and the possible effects of dynamic content generation to conduct an experiment that can determine if variation and therefore player enjoyment can be increased by simply modifying the way game objects are spawned or behave over time. In the case of *LaLuup*, these modifications should help to determine if dynamic objects can create the variation needed or if they distract the users from experimenting with the levels' puzzles. Furthermore, the findings could also help to increase variation in similar games.

The term player enjoyment as the most important factor in computer games was first introduced by Sweetser and Wyeth in 2005 in an article that revealed a new model to evaluate the named term, called *GameFlow*. This model will be modified to fit the needs of the game genre and utilized in the experiments performed. The aim of this thesis is to explore the effects of dynamic elements to the *GameFlow* model and the aesthetic experience in the case of *LaLuup* and to possibly help drawing conclusions for similar games.

### 1.3 Document Structure

In the first section of chapter 2, related projects, composition techniques and applications to the generative algorithmic music system developed for *LaLuup* are described to show the influences of game ideas and musical aesthetics on the thesis project. In the second section the focus shifts towards the definition of casual games, the *GameFlow* model and playability in current literature. For each of the subsections the connection and context for *LaLuup* is established too.

Chapter 3 contains information about the development of the thesis project, the audio system of *LaLuup*. At first, concepts, requirements and features including their

desired aesthetic effects are explained. This section is followed by a detailed gameplay description. The Pärt algorithm and its usage in the game prototype are also described in detail in this chapter.

Chapter 4 sheds light on the internal architecture by explaining the components of the audio system in detail. Furthermore, the configurations of the evaluation prototype are explained and justified as well.

In chapter 5 dynamic content creation is investigated in correspondence with *LaLuup*. Different concepts on how to spawn and modify dynamic elements in *LaLuup* are derived and explained in detail. Two of these concepts and a non-dynamic control version are illustrated and then used as evaluation setups for the following experiments.

Chapter 6 deals with the evaluation from two different experiments. The first section elaborates on the expert reviews using *GameFlow* heuristics which were conducted to assist the general aim of this thesis and locate optimization potential within and between the three game prototype versions. In the second section a small-scale playability evaluation conducted with target group users is described to reflect the opinion of the target group on the three prototype versions and depict differences between them. For both sections the methods and evaluation procedures are explained followed by a detailed overview and discussion of the results.

To conclude this thesis, chapter 7 offers a detailed result analysis in the first section followed by further prospects and the drawn conclusions and plans for *LaLuup* based on the evaluation results in the second section. Finally, a conclusion is given to summarize what has been done in the course of this thesis.

## Chapter 2

# Related Work

This chapter is split into two different topic fields with the first section concerning the related work for creating the generative audio system for *LaLuup*. The second section covers the basic and also recent specific literature on *GameFlow*, the model's heuristics and how it is used to design as well as evaluate enjoyment in different game genres. Another important topic in combination with *GameFlow* is playability which is also described in the second section.

### 2.1 Algorithmic and Generative Composition

The project corresponding to this master thesis is centered around the creation of the generative algorithmic audio system for the game prototype called *LaLuup*. With its functionality it considerably contributes to player enjoyment, game aesthetics and playability and is therefore also integrated in the experiments and evaluations following later on. Consequently, it is important to describe the concepts and projects that influenced some of the ideas incorporated into the music system. The following sections provide insight into influences for the user interface, player feedback, algorithmic generative composition and finally reference related applications.

#### 2.1.1 Interactive Digital Instruments

One of the well-known instruments that is based on a digital interface is *reactTable*. It was invented by a group of researchers from the Music Technology Group at the Pompeu Fabra University in Barcelona, including S. Jorda and M. Kaltenbrunner. It follows the concept of tangible user interfaces (TUI) as a tabletop instrument that is operated with the use of plastic plucks that can be placed on top of it as seen in figure 2.1. The idea behind *reactTable* is simple, yet very well thought-out: The plastic plucks symbolize parts of a synthesizer and their proximity determines the connections and relations to each other. A computer vision system detects proximity, type and also rotations of those plucks and the instrument processes and reacts to the changes in the system performed by the user. The *reactTable* was both designed to be used as part of an installation as well as in concerts and can be played by a scalable amount of users. It abstracts the sound generation to a higher level with still giving the user precise feedback on every operation performed on it [10].



**Figure 2.1:** The *reacTable* tangible tabletop instrument in action [10].

This project was developed in the early 2000s and quickly became known not only in academic conferences but also to the public, as it is sold as an instrument to different music performers around the world. It was inspired by the tangible interfaces built by Ishii, Ullmer and researchers at the Tangible Media group [11] and also inspired back as Colter, Ishii and others presented *soundFORMS* at the ACM CHI 2016, a world-wide conference for human-computer interaction. *soundFORMS* is a pin-based shape-shifting display that users can interact with through touch [1]. As much as these tangible music interfaces differ from the thesis project, they also show certain similarities that make them particularly interesting in research: The user interaction techniques and the corresponding generative and compositional sound feedback were an inspiration for the music system as well.

### 2.1.2 Composition Techniques

The flexible nature of games and other interactive applications pose a very special challenge for sound composers generally. A score for a game cannot behave like a score for theaters, movies or any passive media; it has to react to scenarios that are generated by human-computer interaction and adaptively programmed quick-time-events. Therefore, visual programming environments such as Cycling’s *MAX* and Ableton’s *Live* are used to compose generative music that changes by parametrization. Audio middleware such as *FMOD*, *WWise* and *Fabric* are then used to rebuild the functionality to fit the needs of the chosen game engine. So in fact, typical game music composers have to be part musicians but also part technicians to enable them to map classic linear composition to the adaptive generative sound needed in games [28].

To create adaptive and yet memorable sound, several different algorithmic techniques are used, many of them first introduced in other areas of information technology. There is a broad variety of creation algorithms, many working in real-time applications as stochastic approaches using probability distributions such as Markov chains, formal grammar algorithms based on language processing, machine learning with neural networks, genetic algorithms and almost infinitely more. Parameter mapping plays an important role in representing user feedback in any of those approaches and is also done in slightly different ways in audio systems for interactive applications. Numerous papers for those topics exist and some also connect tangible interfaces with generative composition tools.



**Figure 2.2:** The musical tool *January* by *Disasterpeace*, developed by Richard Vreeland as a composing experiment [35].

Nuanain and Sullivan for example recently proposed a prototype for creating algorithmic composition on the former mentioned TUI *reacTable* that incorporates stochastic techniques as well as formal grammar and hybrid algorithms using both statistics and serialism [20].

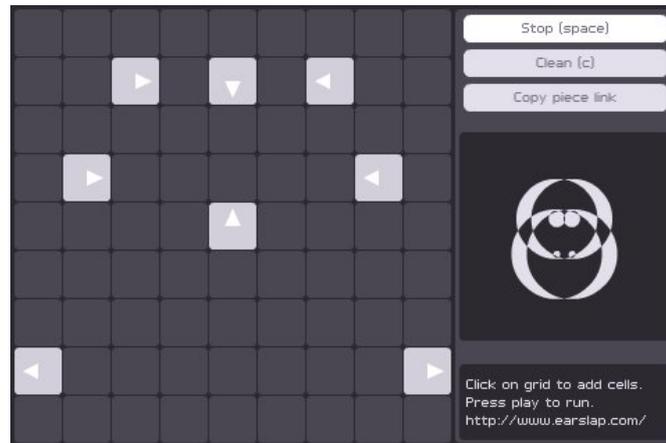
### 2.1.3 Related Applications

Generative composing in interactive applications similar to the game prototype has also been done before, except that the author did not manage to find a game exactly matching the Tintinnabuli styled algorithm and combination of interaction ways used to control *LaLuup*'s music system. During research however, some interesting projects were found that inspired different game components of *LaLuup*.

American composer Richard Vreeland, who is also known for the soundtrack of the games *Fez* and *Hyper Light Drifter*, created *January* under his artist alias *Disasterpeace*. In this interactive explorative music tool, one controls a child in a 2D environment to catch snowflakes with its mouth (shown in figure 2.2). The caught snowflakes generate notes with different pitches that are chosen by a set of rules. However, the player is free to choose from chords and determine when and how notes are played, which creates a simple and yet beautiful back and forth communication between the player and the application.<sup>1</sup> The idea of *LaLuup*'s communication between the player and background voices is kind of similar, but more concrete than in *January* as the player influences the application in a more simple and straightforward way. This is however needed to create a more casual game with a broad target group that still keeps the player exploring and experimenting rather than just solving level by level. *January* only has one level and aesthetic mood to explore, but *LaLuup* will presumably have more in the final product state and the gameplay has to stay interesting over time. *LaLuup* also does not require the player to have any knowledge about music in general to play and understand it as *January* does.

Another interesting experiment is the application called *Otomata* (as seen in figure 2.3) by Batuhan Bozkurt alias *Earslap* in Istanbul, Turkey. He created a generative

<sup>1</sup>For more information visit <http://disasterpeace.com/blog/january>.



**Figure 2.3:** Otomata’s sequencer interface is very simple and features a grid like *LaLuup* [33].

sequencer that uses cellular automation rules to create sequences that possibly never repeat but theoretically play forever. The application consists of a grid in which the user can paint cells and change their orientation before pressing play to create gradually evolving and complex sound sequences with incredible simplicity.<sup>2</sup> Interesting in reference to *LaLuup* is in this case the sequence creation using a similar grid with only 4 directions instead of 6. *Otomata* was also developed for mobile devices and some of the interaction methods used in it helped in developing the early draft of the interaction system of *LaLuup*.

An upcoming audio tool called *foreverloops* created by Marlene and Ulrich Brandstätter in Linz, Austria, can also be mentioned as an inspiration to *LaLuup*. Their application allows users to simply and dynamically generate soundscapes and visualizations at the same time while not needing to be experienced with classic audio programs. It works with digital gears and motors as main interactable objects (as seen in figure 2.4) that play sounds at user defined beats.<sup>3</sup> Its main functionality can be compared to the *reactTable*. While it is much more of a composing tool than a game, *foreverloops* lets users explore music through simple controls and the feeling of experimenting through these understandable gestures also inspired *LaLuup*’s interaction system.

## 2.2 GameFlow and Playability in Casual Mobile Games

Considering the terms casual game, flow and playability are very abstract, their meaning and context should be well defined before conducting an evaluation containing corresponding heuristics. Some of those abstract concepts are greatly accepted and employed without considering their broad meanings. For this reason and to show recent work in their related fields this section is devoted to defining their meanings in the context of this master thesis as well as create the basis of the following evaluations.

<sup>2</sup>For more information visit <http://earslap.com/page/otomata.html>.

<sup>3</sup>For more information visit <https://www.foreverloops.com/>.

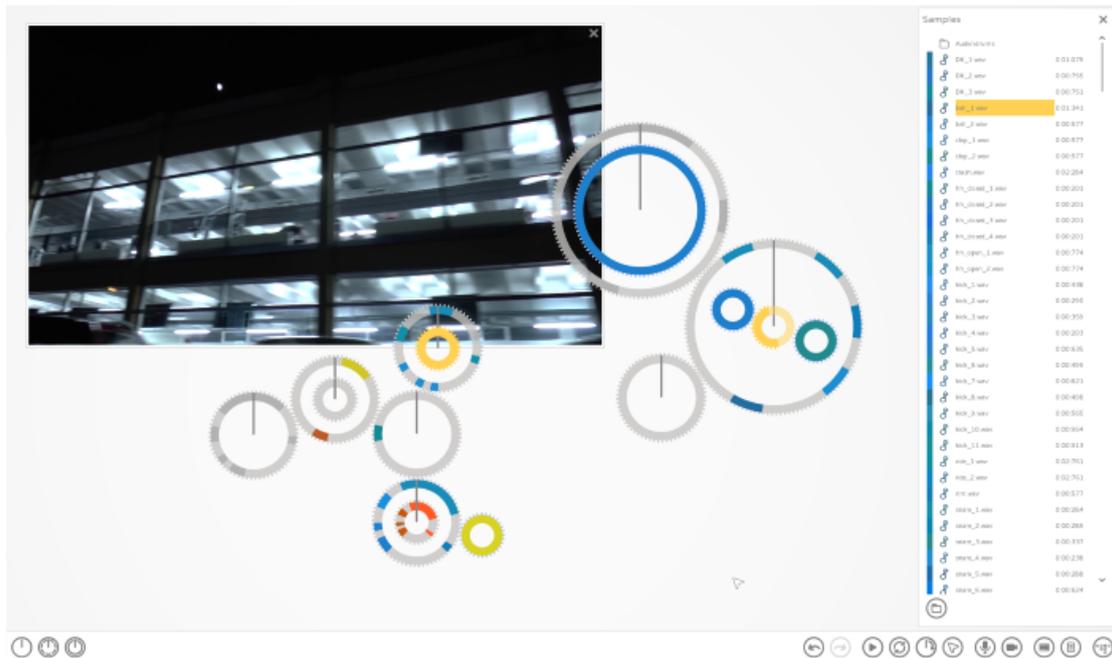


Figure 2.4: A look on the interface of foreverloops [34].

### 2.2.1 Casual Games Discussion

There has been a time between the early years of video games including *Pac-Man*, *Tetris* and *Space Invaders* and the today when not to play video games was the norm and playing them was the exception. At that time digital games were often targeted to the male adolescent users that we still have in our mind as the typical gamer stereotype. But for more than 10 years now there has been a noticeable shift in demographics in gaming as playing video games gets the norm in nearly every age. Games that are appealing to broad audiences are developed again, featuring easy-to-understand controls, simple gameplay and a clear and structured game design. We call them casual games, and the ones playing them casual players [12].

Today casual games are everywhere, they can be played on mobile devices, desktop computers and game consoles. In games cultures there is still no distinct definition for what is *casual*, let alone that there exist a lot of terms combined with this word: Games can be *casual*, as well as game players, playing styles, business models and many more. Kuittinen et al. try to shed some light on the term *casual* in reference to existing games cultures in their conference paper discussing casual games [16]. In a less recent publication, Juul asks the same questions about what and who is casual and interprets the changes in gaming demographics [12]. Following now are the defined terms that are mentioned specifically in this thesis too, describing what they are and mentioning with which other terms they should not be confounded with.

**Casual Game** A casual game is by Kuittinen's definition a term that should be treated ostensibly or virtual. However, the paper still offers certain *casual* properties that can be

Casual Game Design Element	Description
Fiction	What the game is ostensibly about, often a more positive and familiar setting than in traditional hardcore games.
Usability	Easy-to-use in the way that the interface is simple to control and the player understands his/her actions.
Interruptibility	Being able to play and stop in small time slots instead of having to save manually at save points.
Difficulty and punishment	A game that is easy for beginners but yet offers challenges that improve the player's skills and still let him/her recover from failed tasks easily.
Juiciness	Excessive positive feedback to the player's successfully performed actions.

**Table 2.1:** Juul's definition of casual game design is structured into 5 different components [12].

seen in games: Captivating content, simple and easy-to-master controls and gameplay, instant rewards and short game sessions can cause a game to feel *casual* to the user but nonetheless, these properties may vary depending on the game [16]. Juul tries to define casual games in a bit more detail by their design and depicts the five components of casual game design as fiction, usability, difficulty and punishment, interruptibility, and juiciness. They are described extensively in table 2.1.

**Casual Playing** This term refers to the way games are played, for example that play sessions are rather short and the players do not have to rigorously strain their minds to accomplish the game's tasks. It should not be confused with casual gaming which describes aspects of the game cultures that are characterized as casual, such as playing games as a leisure time activity [16]. Casual games do not have to be casually played, as Juul found out in a survey that there are many casual players that invest a lot of time, have a deep knowledge about the casual game they play and even prefer difficult game tasks [12].

**Casual (Game) Player** A person who plays games labelled as casual can be seen as a casual game player, a casual gamer however is someone playing games in a casual way and does thereby not necessarily play casual games [16]. A casual player could therefore also be a person playing *World of Warcraft* occasionally for a short time.

## LaLuup

The game prototype used and partly developed in the course of this thesis features the following characteristics and goals that assist classifying it as a casual game supporting

casual playing elements. It therefore also fixates the target user group for the evaluations as casual game players.

*LaLuup* is specifically tailored to mobile devices and should also support casual playing by level design. Admittedly, the fact that a game is developed for mobile devices does not label it as casual straightaway, but it is a strong indicator that certain device-specific properties will be defining the characteristics of the play sessions that fit into the scheme of a casual game. The battery service life and the typical product usage of mobile devices predict rather short play sessions without even taking a look at the level design. However, the level design is adapted to a casual playing style too as the levels can be solved within a few minutes and the game state is saved every time after completing another level.

Gameplay and controls are aiming to be simple and understandable as *LaLuup* should be accessible to a broad group of users. By the time the prototype is evaluated in this thesis a tutorial about the game controls exists only in the developers' minds and game design documents, but as the gameplay and controls can be explained in a few short sentences, it can be assessed that the concepts will be understandable without verbal help in the final product. By changing the soundscape and visuals after solving the level *LaLuup* aims to provide a reward for experimenting with the game and thereby accomplishing one of many solutions. Additionally, the game does not aim to be extremely difficult to solve as the goal is not to offer a rigorously hard puzzle gameplay but to encourage the players to experiment and explore the game rather than solving the level by pure skill and calculation.

### 2.2.2 GameFlow

There are many heuristic models in user experience and interaction design with which digital games can be evaluated. However, these heuristics are mostly centered around usability rather than player enjoyment and cannot predict if players will actually enjoy the final product in the end. Most of the heuristics guidelines and usability evaluation processes utilized by game developers focus on three main topics: game interaction, game mechanics and game play [5]. Naturally usability is a big part of creating an immersive and enjoyable experience, but there are other factors that can have a strong effect on enjoyment too. Before *GameFlow* there was no model for evaluating player enjoyment in games and there is still no other comparable one since its development for RTS (real-time strategy) games in 2005 [29] and the recent extension to different game genres in 2017 [30].

#### Model

As already mentioned earlier, the *GameFlow* model was developed to measure and evaluate player enjoyment in games. It is based on *Flow*, a concept developed by Csikszentmihalyi in 1990. Csikszentmihalyi's research about the optimal experience showed that enjoyment as a feeling seems to be the same for everyone around the globe. He then describes a total of eight elements that together form a flow experience [3] and should later be the foundation of the *GameFlow* model too. In their work, Sweetser and Wyeth mapped these eight elements to the matching elements games and corresponding literature consist of, as seen in the table 2.2.

<b>Games Literature</b>	<b>Flow</b>
The Game	a task that can be performed
Concentration	ability to concentrate on the task
Challenge and Player Skills	perceived skills and challenges should fit and must be higher than a certain threshold
Control	a sense of control over the actions
Clear goals	the task's goals must be clear
Feedback	immediate feedback must be provided
Immersion	time and self are sensed in a different way, effortless but deep involvement
Social Interaction	-

**Table 2.2:** This is how the psychological elements of *Flow* were mapped to elements in games literature. The first eight (considering Challenge and Player Skills as separate) were directly matched with flow components of Csikszentmihalyi's studies, whereas the last component was introduced by Sweetser and Wyeth because it was frequently mentioned and studied in user experience literature [29].

Since the first element is the game itself it cannot be evaluated directly, but for each of the other eight elements they present in their work various criteria are given to support the evaluation using *GameFlow* heuristics. The heuristics were then verified by the evaluation of two RTS games.

### Modifications and Extensions

The *GameFlow* model was since its development frequently utilized in expert evaluations of games from different genres, also including mobile games like a memory game for seniors [31] and PyramidBloxx, a casual mobile game [22]. In the course of these repeatedly performed tests not only the games but also the model could be evaluated and over time some weaknesses to the heuristics were found. Sweetser et al. recently evaluated mobile games with *GameFlow* heuristics in expert reviews and described that Immersion as a criteria is difficult to review in self-report. Additionally, as many mobile games are simple single-player games, the Social Interaction component is not possible to evaluate too. However, the results showed that the ratings of the chosen mobile adventure games utilizing the *GameFlow* criteria were still very close to the professional ones done [30].

Nonetheless, some additions and changes to the model were made to serve various game genres. For example the PGF (*Pervasive GameFlow*) model aims to emphasize the characteristics of pervasive games such as mobile or place-independent gameplay, interaction between the players and integration of virtual and physical world [9]. Another model especially designed for e-learning games is *EGameFlow* which adapts the heuristics to a more learning-based usage [6].

Playability Component	Description
Functional	Attends to how well key mappings, control peripherals and configurations match the gameplay requirements.
Structural	Contains structures, rules and patterns of how interaction between a player and these facets in the game work.
Audiovisual	Focuses on the relation of functional and structural elements and the appearance and aesthetics of the game.
Social	Analyzes social practices in media usage that the product fits in various contexts of culture and use.

**Table 2.3:** Järvinen et al. define playability as a collection of evaluation criteria structured into these four components [8].

## LaLuup

Many of the heuristics that the *GameFlow* model introduced are worth evaluating for the casual mobile puzzle game. Especially the elements listed in the table 2.2 from Concentration to Feedback are very interesting when it comes to the effects of dynamic object generation. As already mentioned, Immersion is difficult to evaluate objectively in expert reviews as they will be done in this thesis and the Social Interaction criteria is not possible to assess in *LaLuup*. Additionally other fitting heuristics have to be developed and utilized to cover the effects of dynamic content on sound aesthetics and the flow experience in more detail too.

### 2.2.3 Playability

The term playability is often discussed but most times not clearly specified in current games literature, at least not in the extent as the term usability is covered. There are many definitions that contain circular references or do not really describe what playability means as Korhonen states in his thesis about evaluating the playability of mobile games with the expert review method [14].

A very useful definition however provide Järvinen et al. in describing playability as a design and evaluation term, it refers for them to guidelines on how to design a game that meets the requirements in player entertainment. Therefore playability should be treated as a multifaceted criteria collection used to evaluate games as usability is in HCI (human computer interaction). Järvinen et al. structure the definition of playability in four components which are shown in table 2.3 [8].

Sánchez et al. also analyzed playability and user experience in 2012. They explain playability as a set of properties defining player experience by using a specific game system that provides as a main objective enjoyment and entertainment by being credible and satisfying, no matter if the user is playing alone or in company. The definition of playability is therefore characterized by terms that also exist in usability concepts but with differing meanings. Learnability for example should not be challenging in a usability environment, but when applied to the term playability it should. Their identified

Playability Attributes	Characterizing Properties	PX Field
Effectiveness Learnability	Completion, Structuring Game Knowledge, Skill, Difficulty, Frustration, Speed, Discovery	Process/Product Process/Product
Satisfaction	Fun, Disappointment, Attractiveness	Process/Product
Immersion	Conscious Awareness, Absorption, Realism, Dexterity, Socio-Cultural Proximity	Process/Product
Motivation	Encouragement, Curiosity, Self-Improvement, Diversity	User
Emotion	Reaction, Conduct, Sensory Appeal	User
Socialization	Social Perception, Group Awareness, Personal Implication, Sharing, Communication, Interaction	Group

**Table 2.4:** The playability attributes and their characterizing (sub-)properties defined by Sánchez et al. As indicated by the grey bars and the last column, they are structured into different categories based on the area of player experience they are centered around: Process, Product, User and Group [26].

playability categories are Learnability, Satisfaction, Effectiveness, Immersion, Motivation, Emotion and Socialisation, meaning that they split the assessment of playability on both player- and game-centered properties as listed in detail in table 2.4. They further define six facets of playability to create detailed evaluation measurements: Intrinsic (game design and gameplay), mechanical (software system and game engine), interactive (interface and user interaction), artistic (aesthetics and story), intrapersonal (perceptions and feelings) and interpersonal (group perceptions and awareness) [26].

Korhonen offers a narrower and seemingly more concrete description of playability in his thesis published in 2016 before illustrating his own set of heuristics [14]:

A game has good playability when the user interface is intuitive and the gaming platform is unobtrusive, so that the player can concentrate on playing the game. Fun and challenge are created through gameplay when it is understandable, suitably difficult and engaging.

In this way he incorporates the gaming platform, user interface and the gameplay into his definition of playability and leaves social factors affecting the game from the outside and not being directly part of it to the side. His majorly game-centered playability heuristics are split into the modules Game Usability, Gameplay, Mobility, Multi-Player and Context-Aware. However, he also includes player-centered criteria in some of the evaluation modules. He also addresses the relationship between player experience and playability in the way that playability helps in achieving a positive player or user experience of the product or game [14]. Also, player experience closely relates to player enjoyment, just as flow experience and enjoyment are heavily interconnected too.

Another recent definition of playability by Paavilainen is even more game-centric than Korhonen's. According to him playability describes game quality and is defined by the three components functionality, usability and gameplay. Functionality is focused on the technical quality concerning bugs, crashes and smooth operations whereas usability centers on the game interface only. Gameplay is defined as the game's internal operations, like the rules of the game creating the mechanics and dynamics, and does not include human play activity. In this approach playability is not player-centered at all and good playability is also not required for players to have fun [21]. This hugely differs to Korhonen's definition as he indicates good playability affects the player experience positively.

### LaLuup

As Korhonen defines software and hardware stability and social components out of range for being part of playability, his description and heuristics are particularly fitting for this thesis and the evaluation process. Järvinen et al. as well as Sánchez et al. define playability with a social component that seems difficult to evaluate for a game prototype like *LaLuup*. The evaluation should rather be centered around the intrinsic parts of the game and the effects of dynamic game content rather than on social and technical influences from the outside. When talking about functionality, the definition and heuristics Korhonen propose also match better as Paavilainen incorporates more details on the technical quality which do not support the aim of this thesis.

Additionally, the relations between Korhonen's definition of playability, player experience and player enjoyment in general supports smoothly integrating the heuristics with the *GameFlow* model for this thesis' evaluations. Korhonen's heuristics also contain the criteria for aesthetics and style that have been mentioned to be missing before.

## Chapter 3

# Audio System: Conception

*LaLuup* was created as a paper prototype by the company *Kunabi Brother* using a test composition of the Pärt-inspired algorithm in Ableton *Live* that had a simple and yet interesting sound already fitting to the game principles in this early stage. The gameplay worked a little different than in the current version of the digital prototype as the player had to place puzzle tiles that redirected a given impulse to form a loop as seen in figure 3.1. However preliminary user tests showed that the interaction would be more exciting and experimental if users could draw continually reoccurring patterns just as in the company's first mobile game *Blek*<sup>1</sup> illustrated in figure 3.2. A detailed gameplay description explaining the drawing mechanism can be found in section 3.2.1. As *Kunabi Brother* aim to create games that focus on experimenting and exploring rather than on quick and common gameplay that revolves around level-solving only, the second approach was utilized to implement the current version of the digital prototype.

This casual mobile game prototype was built upon a generative algorithmic audio system which was constructed specifically for this application. There was no library or tool already fitting the requirements that the game idea called for: An easy-to-use system implementing a Tintinnabuli style algorithm creating different voices of one given note that incorporates gameplay-specific composing in real-time. Such a system should be reusable (in case of big changes in early stages of the project), stable and of course clearly structured and designed. To create a clear structure the general concept, requirements and prerequisites had to be determined in detail beforehand. This chapter therefore covers the audio system's characteristics, details and requirements within the game prototype.

### 3.1 Characteristics of LaLuup

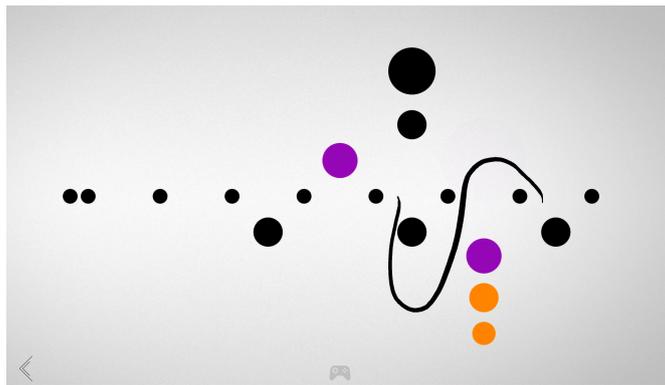
What defines *LaLuup* is the mix of a unique generative algorithmic audio with a very simplified level design that yet allows the creation of complex puzzle patterns and sound structures. Before the concrete implementation many of these conceptual ideas already existed and supported in defining the prototype during development. This section is devoted to the characteristics of *LaLuup* and the corresponding audio system, describing the initial concepts, explaining the detailed requirements and finally shedding light on

---

<sup>1</sup>For further information visit <http://blekgame.com/>.



**Figure 3.1:** A look at the interface of the first paper prototype of *LaLuup* with the gameplay working slightly different than in the current version of the digital prototype.

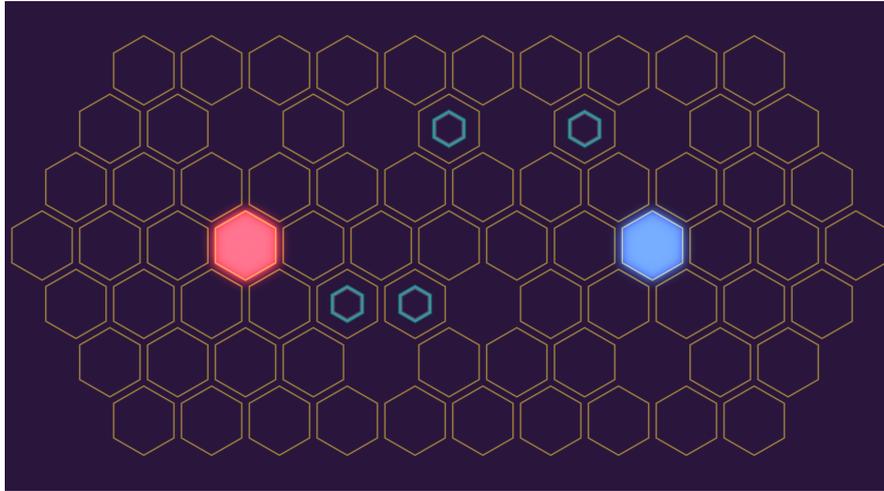


**Figure 3.2:** *Blek* was the first game *Kunabi Brother* developed. Its main focus is on drawing repeating patterns by touch interaction that reach all colored objects in the level. This idea is slightly similar to the repeating patterns in *LaLuup*, yet they are much more limited in directions.

the features and aesthetic goals of the thesis project.

### 3.1.1 Concept

At the beginning of *LaLuup* (its creation as a paper prototype as seen in figure 3.1) many of the concepts also seen in later stages of the project were already there, such as the hexagonal grid and the puzzle tiles also seen in the current version in figure 3.3. The very first concept focused on the tile placing done by the user but was discarded later on as a simpler, more intuitive interaction method surfaced. Now the player paints repeating patterns by dragging them from so-called Spawner tiles (the fully filled hexagons) that have to reach the other Spawner and all Collectible tiles in the level, thus creating a loop



**Figure 3.3:** A simple level structure with Spawners and Collectibles.

that he/she can listen to. That the solution to the puzzle should be a loop and that there should be more than one solution is still identical to the first paper prototype. However, some concepts strengthen and also transform slightly, mainly because of user feedback from different informal tests. The following two topics illustrate the main concepts of *LaLuup* and the corresponding audio system and how they evolved from the first paper prototype to the current digital game version.

### Sound as Gameplay Element

Before *LaLuup*, the company was working on *Frost*, a casual mobile game for iOS devices where the user interacts and controls swarms and flocks consisting of thousands of individual particles.<sup>2</sup> *Frost* is very graphic-centered and aesthetically pleasing as seen in figure 3.4 but lacked a suitable musical setting at first. The audio of *Frost* was later developed gradually along to the level design and consisted of different generative approaches mixed up to match the user particle interaction. In a way it was important to the game to support the immersion, but all the same it was still not the major focus of the game. After creating *Frost* and its audio it was time to try something new and invert the approach, to use sound as a fundamental gameplay element and utilize visual aesthetics as supporting factors.

The concept for the sound was established when one of *Kunabi Brother's* game designers read a book about Arvo Pärt and his Tintinnabuli styled composing, written by Conen [2], describing the musical theory and how Pärt utilized it in his compositions. He first wanted to create a back- and forth communication between the player and the Tintinnabuli styled background voices of the game, giving the illusion of sound-based interaction. However, as soon as the algorithm was built into the sound system created for the player, it was clear that the background should not get too involved with the foreground player interaction as it created confusion and the background feedback was often misunderstood. So the background voices remain at certain events performed by

<sup>2</sup>For further information visit <http://frost-game.com/>.



**Figure 3.4:** *Frost*, the latest iOS game from *Kunabi Brother*, is centered on visual aesthetics by using particle simulations directly processed on the GPU. Sound is generated algorithmically but not the main focus of the game.

the player to manage the needed user feedback, but are not used as much as originally planned.

However, sounds generated by the Pärt algorithm still influence every part of the gameplay as they generate the Melody path sounds, calculate hit sounds for Spawners and Collectibles and also offer the possibility to even change scripted audio in a way that matches the current level. All of these sounds are of course not only generated by the Pärt algorithm, but heavily influenced by the player's actions.

### Puzzle Solving and Exploration

Following the style of *Blek* and *Frost*, *LaLuup*'s gameplay is also centered around puzzles that do not have only one working solution. This is mainly because *Kunabi Brother* focus their games on exploration and experimentation rather than solving puzzle by puzzle just to get to the next level as fast as possible. The gameplay should be revolving around the artistic creations of the player in the game's world, with the game only giving him/her directions and rules to create that must only be followed to a certain amount to reach the next level and start exploring again. The company wants to encourage players to create and influence the game and its elements and to remain and watch the game evolve. Gameplay should be slowed down, thereby creating a game that can be played after coming home from a long work day to relax.

At the moment the digital prototype of *LaLuup* has very restricted possibilities for the player to explore, mainly because not all features are implemented. In the final product there will be more content such as additional gameplay elements and audio components reacting to user input, but as there is only a certain amount of time for creating the prototype and testing if it really works the way it should, this will be considered depending on the feedback of the players and implemented in later stages. The most important factor for the user tests in this stage is rather that the main features already exist and work stably so that the evaluations provide clear and feasible feedback about the game mechanics.

### 3.1.2 Requirements

As every piece of software, *LaLuup*'s audio system has some requirements that were researched before starting with the implementation in order to create a usable, stable and bug-free system without wasting time in the early stages. After a quick investigation phase two areas of requirements surfaced, one concerning the editor and usability design of the system in reference to the game designers and the other one focusing on the technical difficulties that could be met along the way.

#### Internal vs. External Design

The audio system for *LaLuup* is utilized by two game designers and the author as developer and programmer, meaning that it has to be easy-to-use and understandable, providing comprehensible feedback in the game to variable changes in the editor. For that reason it is necessary to create and maintain documentation, to check and adapt variable naming if it is not formulated clearly enough and to be of assistance in any remaining unclear cases.

Designing a system architecture for such an audio system is a trade-off between function and design as the game designers should not need to be aware of all the internal structure to setup and maintain their system configurations. On one hand intrinsic architecture should be functional, meaning it should follow appropriate design patterns and the basic principles of object oriented programming as far as possible, using inheritance and reflection to create a stable, small and reusable code base. On the other hand the exteriors of the system, which are in this case the looks and usage cases in the *Unity*<sup>3</sup> engine editor, have to be well-structured and readable by humans. In many cases the trade-off can be done in the way that a lot of the internal variables are hidden away and the editor input data is just fed and copied to the system. A very useful tool helping to provide a structured external look to the system was *Odin*<sup>4</sup>, a inspector and serializer tool which is also referred to in section 3.2.2. Furthermore, *Odin* and the *Unity* editor implement so-called property attributes that can define variables as hidden in the inspector, read-only or even create foldout group structures for better reading in the editor.

#### Audio Technology

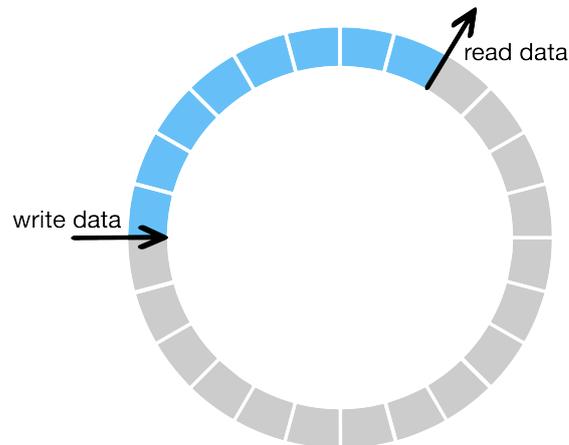
Audio systems that react to user input immediately have to fulfill certain requirements to create the illusion of real-time feedback. Therefore, the most important technological topics for the thesis project were researched and taken into consideration beforehand.

In general, audio processing is never real-time, it depends on a so-called *ring buffer*, a circular buffer that the audio data is fed to. A circular buffer (as illustrated in figure 3.5) can be seen as an array of fixed size where head and tail move around in a cyclic manner, meaning data is stored to the head and processed at the tail. Instead of re- and de-allocating memory every time new audio data is stored to the processing queue, memory allocation can be held consistent and the processing becomes faster, more stable and also more manageable. The downside and consequence of using a cyclic array is quite

---

<sup>3</sup><https://unity3d.com/>

<sup>4</sup><http://sirenix.net/odininspector>



**Figure 3.5:** A ring buffer or circular buffer has both a read and write pointer, meaning that it reads the new data (colored blue) and then releases it to overwrite it again later (colored grey).

obvious as when a lot of audio data is stored into the buffer, the old data is overwritten and lost. Data loss in an audio buffer causes sound output to destabilize, become jittery and click.

Therefore, the size of the *ring buffer* of the DSP (digital signal processing) system is crucial for the sound output as a small buffer size decreases the sound latency because data is processed sooner, but it also decreases the sound stability.<sup>5</sup> On the other side a large buffer increases stability but the sound latency raises too. So choosing the best fit compromise in buffer size is important in the first place but can also be essential in later stages of development if other requirements change.

High quality audio files can be quite large to hold in memory while in-game which is especially important in regards to mobile devices. These devices generally have a smaller amount of working memory available than a typical desktop computer. In the case of small audio samples used by the sequencer to generate foreground and background voices this problem may not arise, but ambience sounds and other audio that may be added to the game could cause stutters in the gameplay due to high memory consumption and so it should not be completely neglected as a requirement. Many audio systems give the user a chance to choose between streaming the audio assets which causes more lag in processing and loading them into working memory. That way the large long files can be treated differently than the small frequent files and this problem can be solved sufficiently.

Another point that is intertwined with the storage requirement is compression. Many audio engines offer a variety of low level compression formats for the sound files. Among the most used are Vorbis, ADPCM and PCM as they all have different benefits and disadvantages. Vorbis is a lossy compression format that utilizes MDCT (modified discrete cosine transformation) much like the well-known format MP3, but is used more frequently in audio libraries as it is open source. It features encoding for multiple chan-

<sup>5</sup>The options in *Unity* for DSP buffer sizes can be found at <https://docs.unity3d.com/Manual/class-AudioManager.html>.

nels (basic MP3 features mono and stereo only), streaming as well as constant and variable bit rate encoding. PCM (pulse code modulation) is a simple analog-to-digital compression that samples the signal in respect to the Nyquist-Shannon sampling theorem, quantizes it linearly to discrete samples and stores those as a binary code. When there is already a discrete signal given it is compressed in a lossless way. ADPCM (adaptive differential pulse code modulation) is only a version of PCM that varies the quantization step size based on a data prediction. Where PCM is larger and lossless it can be decompressed and processed with less effort than ADPCM and with even less than Vorbis.<sup>6</sup>

So the more complex the compression, the more latency the audio system will show when playing this data. However, the less complex the compression, the higher the amount of storage space needed.<sup>7</sup>

### 3.1.3 Aesthetic goals

The concept of *LaLuup* focuses on sound as the main gameplay element and the union of puzzle solving and exploration. The features that add up to this unique game experience are implemented in the digital prototype to the amount needed for a valid evaluation and tracing certain aesthetic goals.

#### Singular Game Mechanics

*LaLuup* is a casual game for mobile devices and should therefore be easy to learn and yet complex enough to create player enjoyment as mentioned in section 2.2.2. Juul's definition of a casual game design also describes difficulty in a similar way as seen in table 2.1. Single game mechanics support this aesthetic goal because they are easy to use and remember, but they still have to be unique and witty to work with complex level structures of later levels too. It requires plenty of time to imagine, develop and create such game mechanics.

The repeating pattern drawing in *LaLuup* can be seen as such a single game mechanic. The author calls it singular or single here because of the fact that it is the only interaction method between the player and the game environment which ensures that once it has been learned by the player, he/she can interact with every level of the game. However, a game mechanic like this also needs to be complex enough to keep the player's interest for a long time, hence the level structure has to assist in creating this complexity much more than if the game mechanics would develop further after every level progression. For this reason singular game mechanics often dominate in games where levels are built of tiled building blocks. The level progression is not implemented as skill improvements of the game character but heavily focused on the rise in structural level complexity.

Even if the positive factors of singular game mechanics for casual games are put aside, there is still an aesthetic advantage to them. The game is easier to explain and understand, even for spectators watching somebody play.

---

<sup>6</sup>Detailed options for audio clips in *Unity* are described in <https://docs.unity3d.com/Manual/class-AudioClip.html>.

<sup>7</sup>A short and straightforward audio file import guide for the *Unity* engine is given at <http://blog.theknightsofunity.com/wrong-import-settings-killing-unity-game-part-2/>.

### Minimalistic Design

Simple design appeals to a broader target group as everyone interprets it in a different emotional way. McCloud, a US comics theorist and cartoonist draws the following conclusion in his theoretical work *Understanding comics* [27]:

When we abstract an image through cartooning, we're not so much eliminating details as we are focusing on specific details. By stripping down an image to its essential "meaning", an artist can amplify that meaning in a way that realistic art can't.

Not only cartoons but abstract representations in general can work on a remarkable meaningful emotional level. Hence, minimalistic design does not always refer to creating a calm and sterile environment, it rather focuses on specific details, leaving things aside that seem unimportant for the current context.

*LaLuup* also aims to keep the design of both audio and visuals simple and scaled-down to appeal to its target group on an emotional level, leaving interpretation of the aesthetics completely to the player. With *Frost* this approach worked arguably well, resulting in a very diversified interpretation of the game reaching from particle symphonies and fireworks to guiding flocking spirits to their home planets. The final version of *LaLuup* should be able to create a similar kind of user feedback.

## 3.2 System Details

Now that the prototype is conceptually defined one can look into the details of the system created. In this section the processes and operations within the game are described and corresponding decisions concerning the design and technical tools used are explained.

### 3.2.1 Detailed Gameplay Description

A detailed gameplay description concerning visual and auditive feedback based on a simple level figure 3.3 is given in this subsection to create a clear foundation for following in-depth information.

#### Interaction and goal

Starting at the very beginning, a Spawner (which is either one of the fully filled hexagons in figure 3.3) is touched and the player holds his/her finger down and moves it in any of the six grid directions. A subtle glow on the grid tiles around the finger indicate the possible directions for drawing, holes are not highlighted in that case. Now the player can draw as far as he/she wants with the only exception of the Collectibles, shown as small hexagonal dots. When the finger is lifted from the touch screen the Spawner starts spawning meaning it creates the impulse that jumps from one grid tile to the next at every beat in the painted pattern playing notes. The system of how the notes are played is explained in detail in the next subsection. When the end of the pattern is reached it is simply repeated farther along the grid which constructs an interesting structure

both visually and sound-wise. This playing pattern is called Melody later on and it stops playing when it hits either another Spawner, a hole or the outer end of the grid. However, when hitting a Spawner, the hit Spawner (its Melody pattern can be manipulated in a similar way) starts spawning too. Now that all interaction principles are declared, the player has only one goal: To get to the next level by closing a loop between the two Spawners using his/her own drawn Melody patterns that include hitting all Collectibles on the way.

### Basic music generation

The music generation for interaction works based on relative direction changes in the Melody patterns on the grid, that means the current direction is always forward and with this information the change in direction can be calculated. Generally, when touching a Spawner, it starts playing using a predefined note for its Melody that gets shifted by semitones when the direction changes. But this so-called *M-voice* does not get played unless there is a direction change, otherwise it is just generated and cached. But in all cases another generated note is played, a Pärt-style *T-voice*, calculated using the cached *M-voice*. The actual amount of semitones stepped in a relative direction can be done in two ways explained in detail in section 4.2.3. A similar scheme is used for generating the drawing sounds as they also play along the patterns, but instead of automatically on beat, they are playing on touch interaction. Small components playing short sounds and the mentioned background voice also use *T-voices* to algorithmically generate a fitting output and are described in detail in sections 4.2.3 and 4.2.2.

### 3.2.2 Tools and Design Decisions

From the first ideas and concepts of *LaLuup* it was planned as a *Unity* engine based prototype for the reason that Kunabi Brother's interest was to create a mobile game out of it if the prototype would succeed. *Unity* prototyping is not only fast and its output deployable on all big players on the device market, it was also chosen because it was already professionally used by all company members involved in the project. Additionally, many of the prototype's components can be reused or just adapted for the final product which is also great for small studios like this where all processes in the development team are iteratively organized. *Unity* scripting can be done in either JavaScript or C#, but for this project the latter one was chosen because both programmers working on this project already had knowledge about it.

The thesis project also relies on a plugin called *Odin* which is a inspector and serializer tool for *Unity*. In context of the complexity of data containers and structures inside the project it was chosen because it helps to keep the source files and the engine's editor clean. Furthermore, it simplifies the component interfaces inside the editor for the game designer's use. For the audio system there were also some possible options to consider in the beginning. Obviously an audio middleware like *FMOD*<sup>8</sup> and *WWise*<sup>9</sup> could have been used to create the system, but in this case it would have made the implementation much more complex than needed. To keep it clean, editable and understandable an own

---

<sup>8</sup><https://www.fmod.com/>

<sup>9</sup><https://www.audiokinetic.com/products/wwise/>

implementation of an audio sampler and sequencer system was created that fit the needs of *LaLuup* perfectly and could easily be extended too. The very first iteration of this implementation was based on a blog post about designing a music system for games by Huguenard [37] which then soon started to differ in many aspects to fit to the concept of *LaLuup*.

After defining the idea and characteristics of *LaLuup* in detail, researching and setting the project requirements and deciding on the tools the implementation process could be started. The extensive research before the actual development of the game prototype provided a profound base for structuring the architecture as well as creating a suitable level design.

## Chapter 4

# Audio System: Implementation

Based on the conceptual ideas and already defined requirements, the audio system could be built. Using the *Unity* engine and its underlying *FMOD* sound engine an architecture was developed that fit exactly to the needs of the game prototype.

### 4.1 The Pärt Algorithm

The Tintinnabuli-style based on Arvo Pärt's rules is a little challenging to read at first when one is not familiar with musical notation and language, but once grasped the concept is very simple and yet effective. A detailed musical explanation and composition schemes can be found in the book *Arvo Pärt: Die Musik des Tintinnabuli-Stils* by Conen [2]. The following sections explain the general idea and schematic of Tintinnabuli music, how to create a fitting algorithm and the specific implementation for *LaLuup*.

#### 4.1.1 General explanation

The Tintinnabuli method transforms notes of a given scale (called the *M-voices*) into notes on the scale's triad below and above the original pitch [36]. Those are called *T-voices* and the most common six in Tintinnabuli music are:

- T-3
- T-2
- T-1
- T+1
- T+2
- T+3

This means that three of the *T-voices* are below the original note and three are above. So if given a note as *M-voice* including a scale and a triad, T+1 can for example be calculated by getting the note on the triad closest but above the pitch of the original note. T+2 is the second closest and so on, the same applies to the other direction (below the original pitch). A schematic of this technique in A minor can be seen in figure 4.1.

#### 4.1.2 Specific implementation

The implementation of Pärt's Tintinnabuli algorithm in code begins with constructing a Paert object with a given scale and triad. The musical scale and triad are then used

T+3	A5	A5	C6	C6	E6	E6	E6	A6
T+2	E5	E5	A5	A5	C6	C6	C6	E6
T+1	C5	C5	E5	E5	A5	A5	A5	C6
M voice	A4	B4	C5	D5	E5	F5	G5	A5
T-1	E4	A4	A4	C5	C5	E5	E5	E5
T-2	C4	E4	E4	A4	A4	C5	C5	C5
T-3	A3	C4	C4	E4	E4	A4	A4	A4

**Figure 4.1:** In this table the *M-voices* and their corresponding *T-voices* are shown. In this case the musical scale is A minor and the first triad (A C E) is used to generate the six closest *T-voices* to the *M-voice* [32].

to build an array of three notes that contain the actual musical notation for the triad's notes. This is important because the triad object given does only contain relative numbers so it can be reused for any given scale, but the Paert object only uses one given scale and triad.

The next step is already the generation of the six *T-voices* based on one note given as a MIDI number, this is simplified by a helper method that already gets the nearest MIDI note number of a *M-voice* and the triad notes in the array built in the first place, either above or below. The method structure is the following: `NearestMidiFromNote(int midiNoteNumber, Note toNote, bool lower)` where the bool specifies if it should be above or below the original note. These MIDI note numbers are cached in a `List<int>` object and already specify the *T-voices* needed. But as the given original note could not only lie exactly between two triads on the scale, these have to be sorted to be in exact order from lowest to highest. The concept of MIDI note numbers is a great help here because as the notes are already saved to the list as integers, they can simply be sorted afterwards. Finally, when generating the *T-voices* an enumeration is used to access the correct array index (enum `TVoice{T_3, T_2, T_1, T1, T2, T3}`) so it can be accessed just with `tVoiceArray[TVoice.T_3]` or `tVoiceArray[TVoice.T_2]`, for example. The algorithm then looks as follows in plain C# code:

```

1 public class Paert {
2     private Note[] notes;
3
4     public Paert(MusicalScale scale, MusicalTriad triad) {
5         this.notes = new Note[3] {
6             scale.noteBaseOctaves[triad.relativeNoteNumbers[0]].note,
7             scale.noteBaseOctaves[triad.relativeNoteNumbers[1]].note,
8             scale.noteBaseOctaves[triad.relativeNoteNumbers[2]].note };
9     }
10
11     public List<int> GetTVoices(int midiNoteNumber) {
12         List<int> voices = new List<int>();
13         voices.Add(AudioUtils.NearestMidiFromNote(midiNoteNumber, notes[0], true));

```

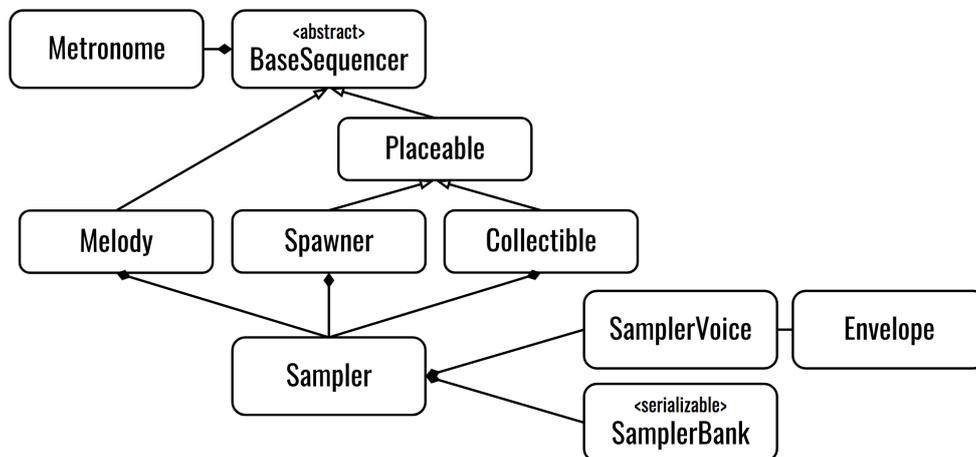
```

14     voices.Add(AudioUtils.NearestMidiFromNote(midiNoteNumber, notes[1], true));
15     voices.Add(AudioUtils.NearestMidiFromNote(midiNoteNumber, notes[2], true));
16     voices.Add(AudioUtils.NearestMidiFromNote(midiNoteNumber, notes[0], false));
17     voices.Add(AudioUtils.NearestMidiFromNote(midiNoteNumber, notes[1], false));
18     voices.Add(AudioUtils.NearestMidiFromNote(midiNoteNumber, notes[2], false));
19     voices.Sort();
20     return voices;
21 }
22
23 public List<int> GetTVoices(NoteOctave nbo) {
24     return GetTVoices(AudioUtils.NoteToMidi(nbo));
25 }
26 }

```

## 4.2 System Architecture

The basic sampler/sequencer system needed to generate the sounds for *LaLuup*'s game-play was based on blog entries of Huguenard about designing a music system for games [37]. By staying close to the functionality of an analogue Synthesizer, his implementation helped to structure the system for the thesis project. Nevertheless, adaptations for sound sampling based on a dynamic base note, additions for algorithmic sound generations and reshaping of the architecture was done very frequently in the first iterations until not that much of the basic structure remained. The following subsections describe the elementary components of the redesigned Pärt audio system and in figure 4.2 the structure is described in a simplified class diagram.



**Figure 4.2:** The basic structure of the algorithmic audio system.

### 4.2.1 Basic Audio Modules

The basic modules, consisting of the Metronome, BaseSequencer and Sampler classes and their associated objects and subclasses, will be described in detail in this subsection.

Together they form the smallest building blocks for the algorithmic generative audio system of *LaLuup*.

### Metronome

The Metronome class has an action `OnTicked` that is called every tick. Ticks are generated by an update loop on top of the *Unity* engine update callback using an accumulator for the time that the tick should occur which is relative to the digital signal processing timer of the engine. The Metronome can be adjusted to use a certain BPM (beats per minute) rate that is parted in an amount of ticks that can be edited too. The action then caches all the extra information like if the current tick is a beat, the tick length and the tick time as parameters in a C# struct called `MusicalStep`. Classes can subscribe to the Metronome to listen for ticks.

The class contains a static component of its own type to provide a main Metronome for every game level. This guarantees that the main Metronome can be accessed and subscribed to from every level component and still ensures the possibility to have a second Metronome in the level if needed. *Unity* also does something similar to simplify access to the provided main Camera component<sup>1</sup>.

### BaseSequencer

The abstract class `BaseSequencer` is a simple building block for sequencers in general. Not only does it have a class field containing a reference to a Metronome object, it subscribes to the Metronome tick action with an abstract method called `HandleTicked`. The following classes extend the `BaseSequencer`'s functionality and play sounds when `HandleTicked` is called:

**Placeable** The `Placeable` class is a superclass to all objects that can be placed on the grid in the editor and can be hit by so-called Melody objects that will be explained in the next subsection.

- **Spawner**

The Spawner, which can be customized with a template object called *Unity Prefab* and overridden per level, is a colored tile in the hexagon grid that contains the data for the Melody it spawns. When it is hit by a Melody on the grid, it starts playing a hit sound using its Sampler. The hit sound is generated based on the note it was hit with.

- **Collectible**

Collectibles are colored objects lying on the grid and have to be hit and in that way collected to close the loop and solve the level. They also play algorithmically generated hit sounds with their dedicated Samplers.

**Melody** The `Melody` class (also customizable as a *Unity Prefab*) is the one actually triggering the gameplay sounds. Every Spawner has a Melody that can be played when the Spawner is touched. Melodies can have two different play modes, either they play

---

<sup>1</sup><https://docs.unity3d.com/ScriptReference/Camera-main.html>

direction-based notes and additionally play Pärt voices only on direction change or the other way around. A Melody's sound is partly based on static data input from the corresponding Spawner like musical scale, triad, direction mapping, play mode, first note and minimum and maximum octave. Because of replacing the first melody note to play with the note the Spawner was hit with it will still sound differently every time it is hit with another note. The Melody class uses two Sampler objects to play its notes, one for the main notes played every beat and one for the direction notes only played on direction change.

### Sampler

The Sampler object (seen on the left in figure 4.3) manages an array of SamplerVoices, their Envelopes and a SamplerBank to play the sounds through the use of simple *Unity* AudioSources. When a note is meant to be played the nearest base note AudioClip in the SamplerBank is found, pitch-shifted to the exact note and then scheduled to be played by the oldest SamplerVoice in the array. Data like sound duration and note is given by the Sequencer through the MusicalStep struct.

**SamplerVoice** When the SamplerVoice now gets to play an AudioClip, it first resets its Envelope with values given by the Sampler and then schedules the AudioClip to play at its corresponding AudioSource. The envelope filtering is done in a *Unity*-specific callback named OnAudioFilterRead which is called by the audio thread. OnAudioFilterRead is called every time a chunk of audio is sent to the filter and allows to insert a custom filter into the audio DSP chain.<sup>2</sup>

**Envelope** The Envelope class is a simple ASR-envelope implementation that helps to keep the SamplerVoice class more readable. It can be called to get output volume values for sounds to create an attack, sustain and release curve.

**SamplerBank** The SamplerBank (seen on the right in figure 4.3) is a data container for AudioClips and their corresponding MIDI note numbers. It allows a sampler to access the AudioClips and pitch-shift them if needed.

## 4.2.2 Composing Modules

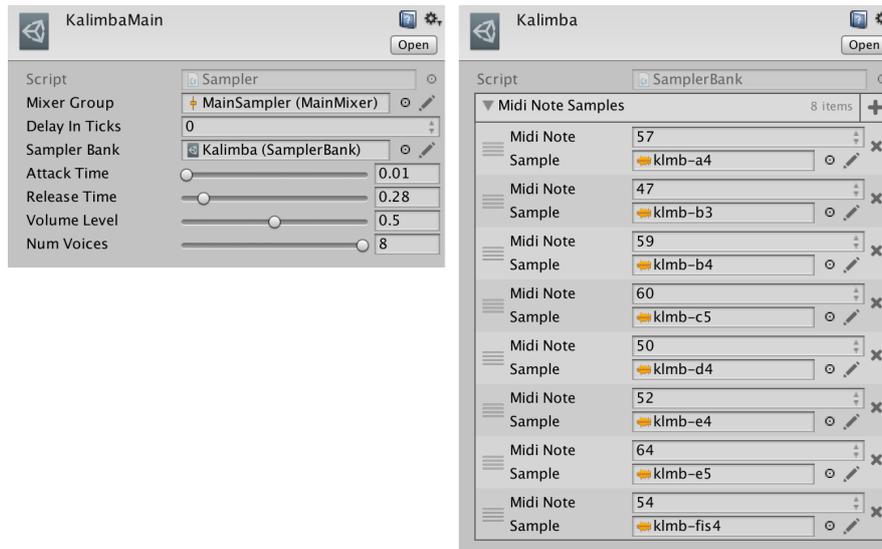
The following classes describe objects not directly part of the basic audio system, they rather specify extensions and utilities of the system's building blocks described before and are therefore not shown in figure 4.2.

### BackgroundVoice

The BackgroundVoice class has no sequencer but a lot of different methods that are subscribed to the LaloopGame Script that manages such things as level solving, level loading etc. It can play different scripted music sequences at every action it is subscribed to, for example when the loop is closed, when a collectible gets hit, or when all spawners

---

<sup>2</sup><https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnAudioFilterRead.html>



**Figure 4.3:** On the left: the Sampler component's options; on the right: the SamplerBank.

have been activated. The background voice component is subscribed to the action `OnPaertPlayed()` and also plays background sounds (chords of *T-voices*) when this action is triggered. A detailed look on the options of this component gives figure 4.4.

**ScriptedMelody** This class is a simple data container for scripted melodies to test their sound within the game. It contains information such as notes, tick lengths and octave shifts. Many `ScriptedMelody` objects are referenced in the `BackgroundVoice` class to get played on certain events.

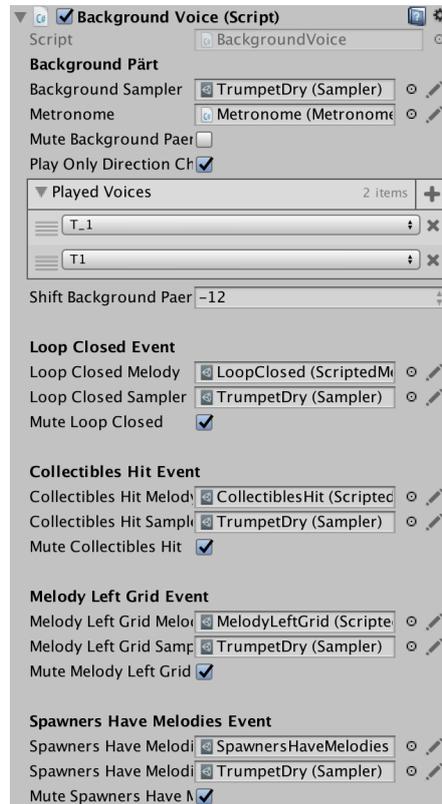
**ScriptedMelodySequencer** While the `ScriptedMelodySequencer` extends from the original `BaseSequencer` class, it has nothing to do with `Placeable`. Its purpose is to play a predefined `ScriptedMelody` when the `BackgroundVoice` wants it to.

## Paert

The `Paert` class implements the algorithm defined by Arvo Pärt's Tintinnabuli styled rule set. Upon creation it needs to know the musical scale and triad it should reference and afterwards it can be called for every MIDI note number that has to produce a *T-voice*. It then returns a list of all 6 *T-voices* that are currently in use in this prototype. As it is an object, it can be reused and does not have to be created anew when another *T-voice* in the same musical scale and triad is needed.

### 4.2.3 Other Structuring Modules

Managing an audio system relying on an algorithm based on classical music theory requires structures and data classes to cache and convert information in a simple and understandable way so that internal changes are also feasible for programmers not working directly on the system.



**Figure 4.4:** The BackgroundVoice component featuring options for playing the background Pärt t-voices and for starting the ScriptedMelody events.

### Important Data Containers

Many data containers are needed to cache, save and load different settings in a music-oriented game prototype. Here are the most important ones that were not mentioned before in detail:

- **DirectionMapping**

The DirectionMapping object (seen on the left in figure 4.5) is used to tell the melody how many semitones it has to go up or down when changing direction. Currently two approaches are implemented, both can be used independently and are chosen for each level by the game designer. The first approach uses a bool array to choose the sign of the semitone step depending on the direction, the amount of semitone steps is given by the amount of steps that are made forward in one direction before changing direction again. The second approach uses simple integers to determine how many semitones have to be stepped in which direction. The direction is either calculated relatively or absolutely to the direction the Melody was going before (a differentiation between relative and absolute directions on the hexagonal grid is shown in figure 4.5 on the right). Therefore a pointy direction system on the hexagonal grid as well as a flat direction system must be specified and a way to convert the directions has to exist.

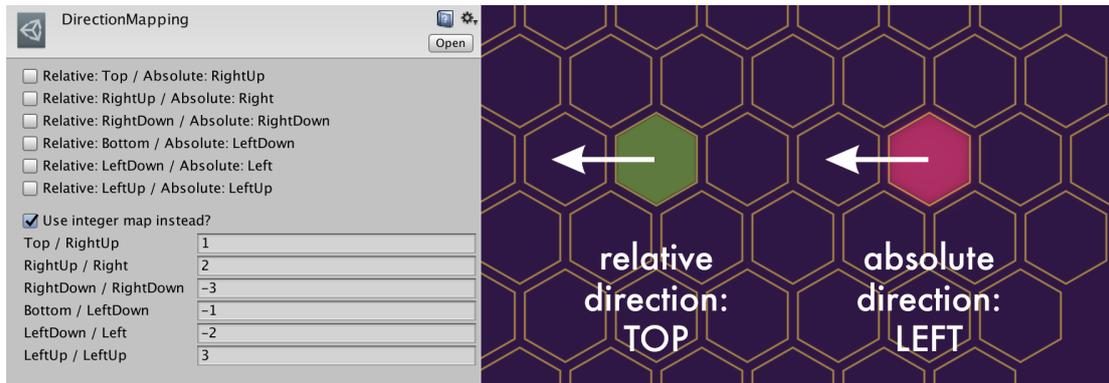


Figure 4.5: On the left: the direction mapping options; on the right: the difference between relative and absolute directions in mapping.

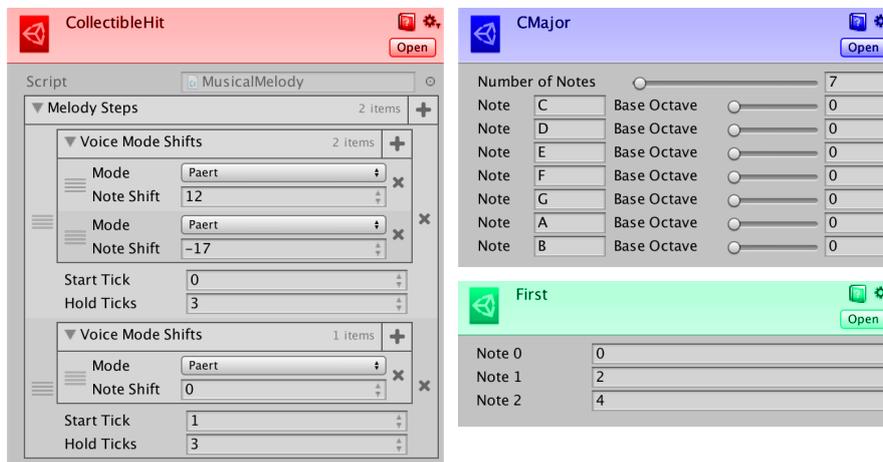


Figure 4.6: A MusicalMelody object (red), a MusicalScale object (blue) and a MusicalTriad object (green).

- **MusicalMelody**

A MusicalMelody (seen on the left in figure 4.6) is in use at every hit of a Placeable object. It defines a sequence of notes and tick lengths to be played if the object is hit based on the so-called hit note. The hit note describes exactly the *M-voice* it was hit with.

- **MusicalScale**

A MusicalScale object (seen in the upper right corner in figure 4.6) defines notes and a base octave for a musical scale that is used in many of the classes mentioned before.

- **MusicalTriad**

The MusicalTriad object (seen in the lower right corner in figure 4.6) defines a triad relative to every musical scale as the semitones to play are given in integer values (for example the first triad of every musical scale is 0 2 4, relatively seen).

### Conversion and Calculation Utilities

In the case of a highly musical and notation oriented project, a lot of conversion methods have to exist. As these methods get used basically in every audio script, they are collectively implemented in a utility class, throughoutly tested and well-documented for reusing. The following conversions and calculations need to be done frequently, stable and in a fast way:

- Get the pitch from one to another MIDI note number,
- convert notes to MIDI note numbers and vice versa,
- get the nearest note to another MIDI note number (lower or higher),
- from a musical scale and a note played of it, choose the next note by just using a relative integer value (with MIDI note numbers or notes and octaves),
- map notes to a given octave spectrum,
- calculate direction mapping differences.

## 4.3 Prototype Configuration

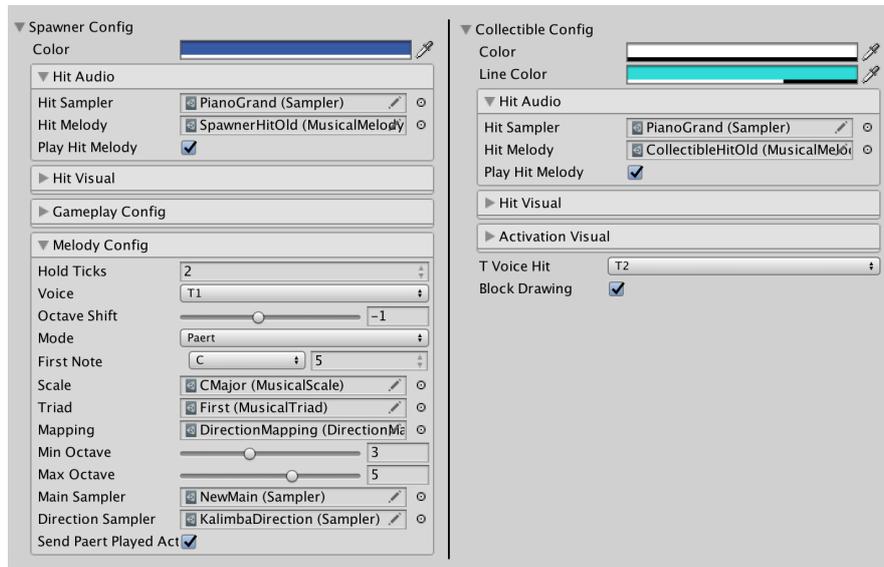
The prototype of *LaLuup* used in the evaluations in the following chapter has certain special configurations in reference to which techniques are used to generate sounds, which interaction feedback is given and how the general level structure looks.

### 4.3.1 Sound Generation

The basic music generation in the evaluation prototype works as already described in section 3.2.1 but there are detailed configurations for the Spawner, Collectible and Melody objects given which can be seen in figure 4.7. For the prototype these descriptions are valid in every test level but they could in fact be overridden per scene.

The general sound generation in the existing test levels is based on the C-Major musical scale and its first triad consisting of the notes C, E and G. Spawners as well as Collectibles reference a unique hit melody that is played when they are hit and which is the same for all levels. Melody objects' samples are held for two ticks per note played before they are released and play the *T-voice* T+1 at every step and additionally their C-Major-based *M-voice* at every direction change. The direction mapping used to generate the *T-* and *M-voices* is an absolute mapping using an integer map to determine the semitone steps that have to be made per direction. When samples get too high or too low, which is below the third and above the fifth octave, they are mapped back to a valid octave in the way that higher values get mapped to the lower end and lower values get mapped to the higher end of the scale. Melody object samples are very clear and close to the African instrument called *Kalimba* while Spawners and Collectibles sound like classic Piano music. Touch interaction creates sounds that resemble the Melody objects.

The BackgroundVoice was retracted slightly after the informal user tests confirmed that some of them were rather confusing, including the background sounds played when the action `OnPaertPlayed()` was triggered. This prototype version therefore only features background sound events when a Melody object leaves the grid and when a loop is



**Figure 4.7:** The Spawner (left), Melody (left) and Collectible (right) configurations of the prototype used in the following evaluations. These configurations apply to all test levels.

closed and the level solved. Background samples are characterized by a soft and heavily synthesized brass sound.

### 4.3.2 Level Design

The evaluation prototype features ten levels with beginner difficulty, instructing the player step by step and letting him/her train and explore the game mechanics. There is no specific tutorial, therefore a short instruction has to be given before a play session can be started. The level design is governed by a certain principle: The levels alternate between instructing and exploring to create a directed and yet organic feel to the gameplay. Game progression is distinguished by level structure and complexity that increases slightly but noticeable from level one to level ten. When a concept is learned, it is often repeated in the next level or combined with another concept from earlier to pose a new and still manageable challenge to the user. This ensures a very linear learning curve that can be mastered with just a few short sentences from the evaluator as a game instruction. The maximum amount of Spawner and Collectible objects in the levels is four, the grid size differs per level. Some level grids also feature grid holes that are indicated by differing tile visuals.

After the Pärt algorithm was rebuilt in software, the system architecture was designed based mostly on research, creating a stable and functional base for developing the game prototype configurations.

## Chapter 5

# Dynamic Object Creation

Most casual mobile games are not designed by hand anymore, but by algorithms defining the levels and objects in them, facilitating gameplay diversity and replayability. A lot of games profit from dynamic content, however there are still games that miss most of these dynamic objects that are played frequently still. Is player enjoyment connected to dynamic object generation in some way? This chapter therefore creates a foundation for the following evaluation of the game using the *GameFlow* model [29] and other heuristics tailored to the prototype to evaluate player enjoyment in games.

*LaLuup* is a first approach in using Arvo Pärt's Tintinnabuli style to create a casual mobile game based on algorithmic music generation. The game prototype was started on paper, based on a lot of research in the field, but also created using trial and error. As a first of its kind the audio system uses Pärt *T-voices* to create the connection between sound and visualization and provide feedback to the player of the game. Like for every new game prototype, it is especially important to ensure that interface, player feedback and the overall game experience follow verified usability and playability guidelines. This can be assured by previously done research as well as through evaluations at different stages of the game prototype. But the evaluations performed with the game prototype *LaLuup* do not only have the goal to discover and erase usability problems, they have another goal: To explore effects of dynamic game elements to the *GameFlow* model and the aesthetic experience in the case of the given game prototype. In the course of this chapter different evaluation setups of the game prototype will be mentioned and explained, two dynamic object spawning setups and a control setup with just the static game. In the dynamic versions of the game prototype so-called collectibles will alternate tile coordinates based on certain parameters. The static version will leave the collectibles at their predefined spawn points throughout the level.

### 5.1 Dynamic Objects in LaLuup

There are many ways to create dynamic content in games, whether by developing objects that move, spawn or change otherwise over time or by events, or by implementing it in concealed ways like dynamically changing difficulty or generated levels. This thesis focuses heavily on the first kind of dynamic objects, requiring them to be visible and understandable to the player.

Games often utilize dynamic objects as part of their core gameplay, in the way that the game does not work without the dynamic elements. This is already used in early twitch-based casual games like *Tetris* and *Space Invaders*. In such games the dynamic objects are used heavily to create more and more time pressure the farther the player's game progress is. In turn-based games such as *Age of Empires* dynamic elements are not needed to progress through the game, the gameplay does not depend on them but rather on the players to complete their turn and then watch the game play out their actions. An easy way to differ between those two game definitions on the base of casual mobile single-player games is by the way the game responds to interruptions: If the game progresses and perhaps is finally lost when it is left unattended, the game features are mostly twitch-based, but if it stays contained in the same state it was left with, it is a turn-based game.

One may argue that twitch-based games with their time pressure and dynamic objects immerse the player faster than turn-based games do, but in fact the effects on immersion alone are difficult to measure as mentioned in section 2.2.2 already. Therefore, another type of measurement and evaluation model has to be used, the *GameFlow* model. By creating dynamic objects with certain twitch-based features within the turn-based game prototype *LaLuup* the differences between these prototype versions can be evaluated and corresponding usability problems can be located.

### 5.1.1 Idea and Evaluation Concept

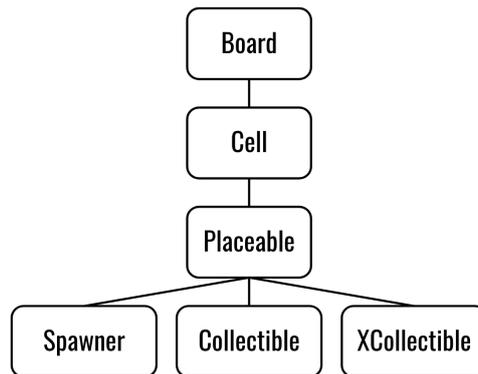
*LaLuup* in its original prototype version uses no dynamic objects in that way, all tiles in the level are fixed and their locations are predefined in a special level editor within the game engine. The reason for this design decision is that the game designers want players to explore rather than to be distracted by moving objects, therefore visible actions within the game should only be triggered by the players themselves. When looking at the importance of collectibles in the levels (as described in detail in section 3.2.1) this decision is understandable, yet it is easy to imagine an alternate version where the collectibles could move. After some experiments with both ideas the question surfaced if it is more enjoyable for the players to have to hunt the collectibles down than just strategically draw *MeLody* paths. Could this help immersion in the game, creating a flow experience in which the players lose track of time easier than if they are just planning, drawing and watching the game evolve at their actions? At the current state of the game prototype it is still worth considering a change in gameplay and mechanics if it increases the players' enjoyment. The internal structure of the prototype is built to be reusable as it is needed to be able to perform adjustments whenever needed.

It is important to mention that the evaluation conducted in the course of this thesis will not prove any effects of dynamic objects to the *GameFlow* model directly, it will rather seek differences between the three prototype versions and discuss them in reference to *GameFlow* and playability criteria. The evaluation results will be valid for the prototype only, and can then be used to fix usability problems and improve certain features in the game. However, some of the findings may be valid to look at when working on other casual mobile games utilizing algorithmic audio systems and similar interaction techniques as *LaLuup*.

### 5.1.2 Implementation

The grid system of *LaLuup* is very modular and its core class is called the `Board`. Within the `Board` there are indices which either contain a `Cell` object or nothing at all. A `Cell` object has different components, including objects to create the look of the grid cells and of course the most important interface for the Collectible system, called `Placeable`. The `Placeable` class is already described in detail in section 4.2.1 in reference to its tasks in the audio system. However, in the case of dynamic collectibles, there is one more derivation of the `Placeable` interface as before. The `XCollectible` class symbolizes `Collectible` objects that do not have to be collected to solve a level, they should rather be a reward for trying new combinations of patterns within one level.

An overview of the grid structure used in the following evaluation setups can be seen in figure 5.1. The grid, and therefore the `Board` object, can have different layouts as well, depending on the number of grid neighbours defined in its properties. It is possible to create quadratic grids with four neighbours as well as octagonal ones with eight neighbours. For this implementation and evaluation procedure the author uses the original configuration of the game with six neighbours and a hexagonal grid to generate evaluation results fitting to this specific level setup.



**Figure 5.1:** The structural connections between the objects involved in the implementation of dynamic collectibles in *LaLuup*. This does not represent a class diagram as unimportant details and objects to this topic are omitted for the purpose of clarity.

A script called `DynamicCollectibles` can then be used to iterate over a list of indices and alter the positions of all collectibles in the game, for both the `Collectible` and the `XCollectible` objects. `XCollectible` objects, which do not have to be collected to solve the level, can be disabled and enabled again and play additional background sounds when hit to symbolize the reward of hitting them with the `Melody` pattern. Their hit states are also reset after they are disabled to show that they are optional gameplay elements. In contrast to these the other `Collectible` objects are only reset if the loop is broken or did not get closed yet. Because of the modular grid system the `Collectible` placement can simply be done by changing the reference index of the object and attaching it to the `Transform` component of the corresponding `Cell` (in the *Unity* engine the `Transform` component of a game object defines the position, scale and

rotation of an object in the game world<sup>1</sup>). The method used to alter the placement is the following: `collectible.AttachToCell(Cell cell)` where the `collectible` specifies the `Collectible` or `XCollectible` object and the `Cell` defines the `Transform` component and visuals it should be attached to.

## 5.2 Chosen Evaluation Setups

Dynamic objects in games can be implemented very diversely and so it is important to define the test cases clearly and also provide reasons why they were chosen in the first place. The following subsections shed light on the way the three different implemented test cases work, what feelings they strive to create for the players and why they were chosen over several other ideas. These test setups are used for both the usability expert reviews and the following play tests with target group users. However, due to the usability reviews before the play test, little changes and improvements can be provided to increase usability and playability for the final evaluation of the prototype done in this thesis. These changes will be described in detail in section 6.1.3.

### 5.2.1 Static Version

In experimental design every playtest needs a control version of the game to find differences in the evaluation results so that these can be inferred to be connected to the changes in gameplay that were made in the other test versions. So the static version in this case is the corresponding control version of the game prototype of *LaLuup*.

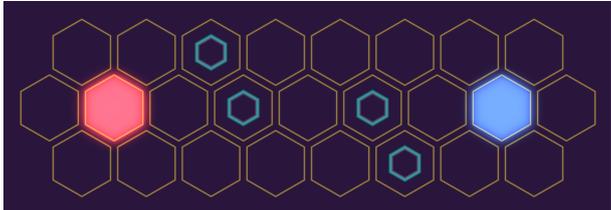
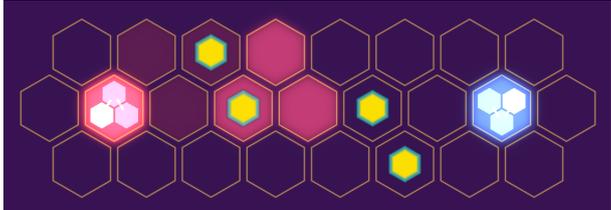
**Setup** This test version features similar gameplay to the one described in section 3.2.1. There are no dynamic `Collectible` objects and gameplay is therefore only revolving around the puzzle and the solution to it. The ten test levels made for the evaluations build up in a tutorial-like structure, first giving the player time to explore and then showing him/her the `Collectible` elements and their purpose step by step. Three of the test levels do not provide `Collectible` objects at all but are included because they are needed in terms of the level and skill progression. These levels can of course be skipped in the other tests if wanted, but are essential for learning the game at first. A short explanation on how a level in the static version works is given at table 5.1.

**Justification** As already mentioned, in the case of the evaluation process conducted in the course of this thesis, the experiments require a control version of the game in which the collectibles do not move to support finding differences in the effects, enjoyment and playability of the game. Additionally, it can be argued that it is easier to depict hidden usability problems within the static version of the game as there are no dynamically moving objects within the levels of the prototype. However, this also changes the way game mechanics are utilized slightly, so this assumption may only be true to some extent.

Another purpose of the static version is of course more game design centered: The first ideas of the prototype as well as the later developed detailed concept revolve around experimenting and exploration, like it is already mentioned in section 3.1.1. At the scope

---

<sup>1</sup><https://docs.unity3d.com/Manual/class-Transform.html>

Game State	Description
	<p>This is the start setup of the static game version.</p>
	<p>When every <b>Spawner</b> has a <b>Melody</b> pattern, the patterns reach the <b>Spawners</b> and all collectibles are hit, the loop is closed and the level is solved.</p>

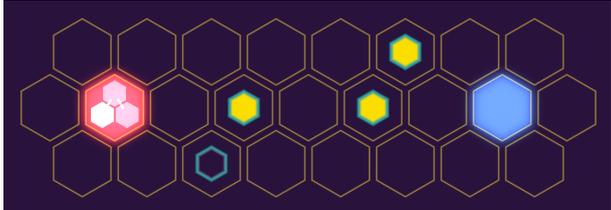
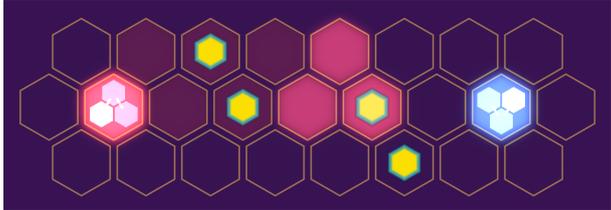
**Table 5.1:** This illustrates the static version of the prototype and how it is solved for test level 5, including a short gameplay description for each step.

of the current development the game designer believes that it could also negatively impact the goal of the game, which is exploring and not only solving the levels.

### 5.2.2 Event-based Collectibles

The idea for this version of the game prototype was created early in development as it seemed to be the most logical and most predictable version of the dynamic **Collectible** implementation at first. However, it is probable that it reduces the player's various possibilities to solve the levels, which has at least happened in some of the test levels due to its implementation.

**Setup** The prototype's levels begin in the same way as they do in the static game version, but the difference is obvious when hitting the first **Spawner** with a **Melody** pattern: The **Collectible** objects will change position at every **Spawner** hit as these **Spawner** hits are specified to be the events on which positions are alternated. Generally, a **Collectible** object only has one to two predefined positions to which they can change as it is difficult to solve levels when having to memorize more than two positions per **Collectible** object. However, these collectibles are not optional to hit to solve the level as they are the standard **Collectible** objects and not **XCollectible** objects. This means that the player has to create different solutions to the static version, sometimes also restricting him/her solution possibilities a bit. In the level illustrated in table 5.2 this is however not really true and the solution to the original version does also still work in the dynamic setup. The solutions of other levels in the test prototype can nonetheless vary in reference to the static control version.

Game State	Description
	<p>This is the start setup of the event-based dynamic Collectible game version.</p>
	<p>As soon as the red Spawner's Melody pattern hits the blue Spawner, the Collectible objects change position.</p>
	<p>At every hit the positions of the Collectible objects change again. The conditions for solving the level do not change in reference to the static version.</p>

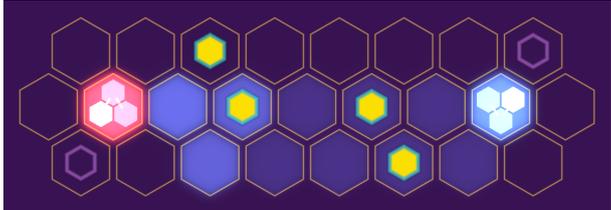
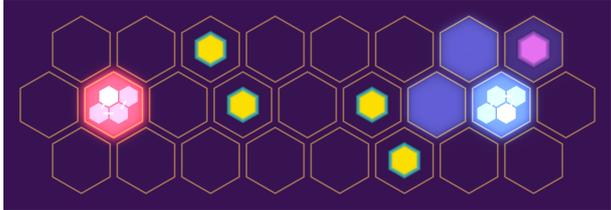
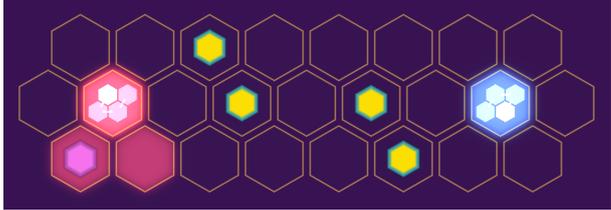
**Table 5.2:** This table illustrates the event-based Collectible version of the game and its states in test level number 5.

**Justification** As already mentioned, this version of the dynamic Collectible objects was conceptually developed early in connection with first asking the question if dynamic objects could have an effect to player enjoyment in casual mobile games like *LaLuvup*. The decision to really implement and use it in the evaluation process in this thesis was made not only because its simplicity in gameplay, but also because there was a strong indication from informal playtests that the players could easily connect the events to the position changes and understand it as a gameplay mechanic.

The predefined positions of the Collectible objects were in fact chosen over a slightly randomized version because they are not only more predictable to the player, but they also let the game designer create a more predictable layout for the puzzle. This enables a more distinct level and challenge progression over time which can be precisely steered by the game designer without destroying the exploration component altogether.

### 5.2.3 Event and Time-based Optional Collectibles

When attempting to create a second dynamic Collectible version of the game prototype several obstacles had to be conquered and the biggest in fact was the problem of position prediction for the player. Alternating positions of collectibles per beat did create more confusion than any version tested before, especially for players that did not have a certain musical comprehension, but also for everyone else it was difficult to

Game State	Description
	<p>This is the start setup of the time-based dynamic Collectible game version.</p>
	<p>As soon as certain trigger Collectible objects are hit on the way, optional purple lined XCollectible objects appear for short time frames.</p>
	<p>As the XCollectibles always appear at the same positions, the player can now adapt his/her Melody patterns to hit the optional objects.</p>
	<p>Even though these objects are optional and conditions for level solving do not change, they produce additional sounds as a reward.</p>

**Table 5.3:** This table shows the time-based optional Collectible version of the game and its states in test level number 5.

concentrate on the changes over time. Therefore, another solution had to be found, connecting event and time-based spawning in an optional way that creates reward rather than frustration.

**First Setup** The first version of the time-based dynamic Collectible objects was centered around the Metronome's beat only. The collectibles switched placed to exactly the positions also used in the event-based version, but rather than waiting for **Spawner** object hits, the position alternation was done per beat, or per a fixed amount of beats. For example, after every fourth beat, the collectibles changed position and everything else stayed the same. This setup created problems like a perceived randomness when and where a Collectible is hit as players were not able to predict how many beats a loop or a Melody pattern had. The whole gameplay suddenly felt random and unpredictable, many players didn't even discover the reason why and when the changes in Collectible position happened.

It was clear that there was no direct, simple solution to fix the prototype test version and use time-based spawning in this manner. However, as time was still a factor the author wanted to integrate in the evaluation process, a concept for a variation of the time-based spawning had to be developed.

**Variation** The follow-up idea finally chosen as a variation of the time-based spawning is so-called time-based optional spawning. In this version of the game prototype there are two types of collectibles, static `Collectible` objects that do not move at all and dynamic, time-based `XCollectible` objects that are only visible for a short time at certain specified events. `XCollectible` objects do not have to be hit to solve the level, instead they reward hitting them with a second Tintinnabuli-styled *T-voice* playing along for a certain amount of beats.

The gameplay now works as follows (also illustrated in an example in table 5.3): When a trigger `Collectible` of all the standard `Collectible` objects in the level is hit and it has a specified `XCollectible` to activate, the `XCollectible` object is activated for a short amount of time (about three seconds in the test version). It can then be hit by a `Melody` pattern and thereby starts playing along the pattern for a certain amount of beats (three in the test version). After the time has run out the `XCollectible` object disappears and can be activated by the trigger `Collectible` again. This variation is therefore not a purely time-based version of the dynamic object spawning, it is rather a hybrid between event and time-based spawning. Additionally, the collection of the extra objects is optional to the gameplay, making them a reward for exploring the level rather than forcing the player to catch them.

**Justification** As already mentioned, time was a factor the author wanted to include in the evaluation process as it is often the main reason for that dynamic objects are spawned in casual games. However, for *LaLuup* a classic implementation that just changes properties over a certain time frame did not work out in a way that can be evaluated both rationally and usefully. Instead, a variation of the time mechanics were found that do not hinder the wanted gameplay and still create the effect wanted for the following evaluation. By making the extra collectibles optional the gameplay test expands to not only include dynamic collectibles, but a dynamic reward for players that explore other level solutions and `Melody` patterns. Thereby the variation gets to be more than a replacement for the simple time-based spawning as it also supports the general concept of experimentation described in section 3.1.1 to a noticeable degree.

## Chapter 6

# Evaluation

The evaluations of the three game versions explained in section 5.2 are structured into two essential parts. First the prototypes are examined for usability and playability problems using the expert heuristic evaluation method with the main *GameFlow* heuristics as mentioned in section 2.2.2 and some additional ones. This first review phase is essential to the following evaluation as the critical usability issues have to be resolved to continue the process. Furthermore, it is important that there are little or at least negligible remaining issues in interfaces, controls, game mechanics, audio, visuals and other game-specific areas so that the outcome of the second evaluation does not only reveal transparent and unadulterated results, but also that the differences in the three prototype versions can be depicted. The second evaluation is a small-scale playability test which consists of a questionnaire filled out by players approximately within the target group of the game prototype. The questionnaire used in this case is based on the core module of the *Game Experience Questionnaire* by IJsselsteijn et al. [24] and it includes many statements that can be connected to the playability and *GameFlow* heuristics in an effective way. It is not only a well structured and profoundly designed document, but it was also tested and validated in a professional way to ensure a robust measurement of the game experience components for which the scoring guidelines are provided.

At the end of this chapter, there should be enough evidence to analyze the depicted differences of the three prototype versions and discuss them in reference to *GameFlow* and playability criteria. Generally the findings can be discussed for *LaLuup* only, however, some results may be relevant to consider when working on other casual mobile games with similar techniques.

### 6.1 Expert Reviews using Heuristics

The first part of the evaluation process focuses entirely on the remaining usability and playability issues within the three game prototypes. The goal of this evaluation is to find and subsequently resolve the remaining critical issues to guarantee a smooth playtest without any players reaching dead ends because of gameplay mechanics or encountering drastic interface problems. In the best case all players will be able to experience all ten test levels without significant interruptions in the final playability tests.

### 6.1.1 Method

A common way to effectively find usability issues in games is the so-called (*expert heuristic evaluation*, *usability expert evaluation* or *expert review* which was introduced by Nielsen and Molich in 1990 [17]. Since then it has become a popular technique to evaluate usability in product interfaces and was adapted for games as well. Its popularity is simple to explain as it can be summarized into three advantages to traditional user testing [14]:

- **Cheap:** The usability expert evaluation was actually an attempt to find a way to create a less expensive usability evaluation for interface designs. As the process relies on the user experience of only a few individuals, it generally reduces the costs of finding issues drastically.
- **Fast:** As already mentioned and going hand in hand with the cost reduction, the heuristic evaluation method is less time-consuming than traditional user tests and accomplishes fast results with less effort. Developers can therefore already resolve issues found on the same day the evaluation took place.
- **Flexible:** Less money and less time already contribute to the flexibility of the method, but not only these two advantages result in it. Flexibility is also provided in the way that the interface evaluated can be in different stages of development and does not have to be entirely complete at the stage of the evaluation. This supports the idea of iterative design and agile development which are two important concepts especially within the software domain.

The first evaluation approach on the expert review method by Nielsen and Molich was conducted in the way that good and bad aspects of the interface were determined in the process. The issues found were reported in detail including ways to fix them to enable the developers to create alternative solutions. By only determining aspects and describing them in detail, this evaluation process can be used for interface prototypes that only exist on paper too. Including to the method the first set of usability heuristics was published in their paper [17] which was later reviewed and published again in 1994 [18]. In general, heuristics in the sense of the expert review method can be defined as a set of rules and guidelines commonly known that support solving problems. A traditional expert review session is structured into four parts [19]:

1. **Initial meeting:** At the kick-off meeting a brief explanation of the game is given, including what exactly should be evaluated, the missing features and target audience. The detailed heuristics should be available on paper so that the evaluators can already have a look at them.
2. **Application review:** The core of the evaluation process is reviewing the application where the experts use their knowledge and experience to independently find usability issues based on the heuristics given before.
3. **Evaluator review session:** The experts then compile a list of all findings together and thereby eliminate duplicates and false ones while redefining unclear issues and their severity ratings.
4. **Final report:** The final result of the heuristic evaluation is a categorized list of usability issues with corresponding heuristics and severity ratings to estimate which problems should be solved first.

The application review within this structure can however be done in many different ways. While Nielsen recommends reviewing the interface twice [19], when looking at game expert reviews, Korhonen suggests structuring the review in three parts, focusing on the interface elements, gameplay and the game interface and game world interaction [14]. However, there are many divergent methods how the review is done in practice as it has to fit to the game prototype as well as to the detailed set of heuristics that should be evaluated [4], [23].

### Heuristics

The heuristics used in the evaluation of the previously described game prototype versions in chapter 5 are, as recommended by Schaffer [7], put together from different sets of heuristics used in common usability literature for game evaluations. The first seven heuristics of the set were taken from the *GameFlow* heuristics by Sweetser and Wyeth [29] and adapted to fit to the game prototype, leaving aside the last component of *GameFlow* as Social Interaction is not useful to evaluate in a single player game as *LaLuup*. Although Immersion is difficult to evaluate with the expert review method (as mentioned in section 2.2.2) it is included because issues directly affecting it could still be spotted.

The remaining three heuristics are roughly based on the playability heuristics for mobile games by Korhonen and Koivisto and include the missing topics the author wanted to evaluate. Interface and Audiovision belong roughly to the game usability heuristics (GU1, GU3 and GU6) and Mobility refers to the mobility heuristics (MO1 and MO3) [15]. Other sets of heuristics as the one by Pinelle [23] and Desurvire [4] can also be mentioned as references as they overlap with some heuristics of the chosen set and were considered as well. The final set of heuristics can be seen in tables 6.1 and 6.2.

### Preparations and Procedure

Usually, the experts invited to participate at a game prototype expert review are at least either experienced in the area of usability engineering or in game design. In the best case they are so-called double experts and bring experience in both areas with them [13]. Developers of the game prototype should not be part of the reviewers because of their involvement in the development. While Nielsen originally suggests 3-5 experts for the evaluation process as new issues found per review decrease rapidly above this amount [19], Korhonen even scales the number of experts down to 2-3 [13]. For the *LaLuup* game prototype, three experts were asked to participate in the evaluations: Two members of the *Playful Interactive Environments*<sup>1</sup> research group with professional backgrounds in game development, design and usability and the likewise experienced co-founder and developer of the previously mentioned audio software *foreverloops* (in section 2.1.3) were invited separately as it was simply not possible to find a mutual date.

A day before the actual reviews took place, the heuristics were mailed to the experts so they could already have a look at them. Before starting the review session, the game was explained shortly, including the interaction methods because there is no actual tutorial contained in the prototype versions. The experts were told that they had to close

---

<sup>1</sup><http://pie.fh-hagenberg.at/>

#	Topic	Criteria	Description
1	Concentration	The player is able to concentrate on the game.	The game should grab the player's attention quickly and also maintain it through the game session. The game tasks should feel worth attending to by providing a suitable amount of work that is within the perceptual, cognitive and memory limit of the player. Players should not be distracted by tasks that don't feel important.
2	Challenge	The game is sufficiently challenging.	The challenges provided by the game should match the players skill level, therefore as the player increases his skill level, the level of challenge also should increase at a suitable speed.
3	Player Skills	The player is able to master and develop skills.	Learning to play the game should be interesting and encouraging. The player should not have to read an instruction manual before being able to start playing. Interfaces and mechanics are easy to learn and use.
4	Control	The player feels a sense of control over his/her actions.	The player should feel in control of his/her actions within the game. Recovering from errors in player actions should be possible and not feel tedious. The game should convey the feeling that the actions and strategies the player uses change the game world.
5	Clear Goals	The game provides the player with clear goals.	Goals should be presented early and clearly to the player.

**Table 6.1:** The first part of the final set of heuristics used to evaluate the prototypes.

the loop between the **Spawner** objects and collect all **Collectible** objects on the way. Furthermore, they were introduced to the touch-and-drag interaction and the concept of the **Melody** patterns. They were also instructed that the **XCollectible** objects are purely optional gameplay elements as this led to minor problems in the first review in which it was unintentionally left out of the game introduction. Since *LaLuup* is a puzzle game that completely relies on exploration and experimentation, it was very important to give the experts the chance to play the game as unhindered as it is possible under the observation of the evaluation moderator before starting the actual review process. Every expert was given another version of the game to play at first before being allowed to try the other two. They also received a handout with all heuristics seen in the tables 6.1 and 6.2 and the metrics which will be explained in detail in the next paragraph. The resulting compiled usability issues can be seen in table 6.3 and the original handouts and tables can be found on the the enclosed CD-ROM/DVD (cf. appendix A).

#	Topic	Criteria	Description
6	Feedback	The game provides the player with appropriate feedback.	The player should receive immediate feedback on his/her actions and progress towards the goals.
7	Immersion	The player experiences deep but effortless involvement in the game.	The player should have an altered sense of time, be less aware of his/her surroundings and self and feel deeply involved in the game.
8	Interface	The game interface is easy to use and understand.	The game interface provides sufficient feedback, is easy to understand and control and does not distract the player from other game tasks.
9	Audiovision	The game's sound aesthetics support the overall game experience.	The game audio should provide support to the visual understanding of the game and the overall game experience. The player should not be distracted by the sound aesthetics but interested in exploring different game configurations to alter them.
10	Mobility	The play sessions are of flexible length and interruptions are handled reasonably.	Levels are of appropriate length to support the short play sessions typical for mobile games. The player should not be punished by the game for interrupting a play session and should be able to quickly get refocused again.

**Table 6.2:** The second part of the final set of heuristics used to evaluate the prototypes.

### Metrics and Result Structure

Normally the final report is written by one of the experts at the last stage of a traditional heuristic evaluation [19] but in the case of this particular review, the time scope the experts had to complete the evaluation was very limited as not even a mutual meeting to compile the list of heuristic violations was possible to organize. The merging of the list was instead done by the author and supported by the intentionally detailed issue descriptions to prevent misunderstandings. There are different ways to use metrics for rating the usability issues in the evaluation process. Ponnada and Kannan for example only evaluate if their heuristics are satisfied for different games and therefore only use a Yes/No rating [25]. In the case of this thesis the actual issues should be rated in detail which renders a Yes/No rating practically unusable. Korhonen uses a severity rating in his publications that is structured into Low/Minor, Medium and Critical/Major [14], [13]. Since it was important to the author to get an even finer granularity to identify the issues that were most problematic, a 5-grade **severity** rating system similar to the one Nielsen originally introduced was integrated [19]:

**0 / No problem:** A heuristic is violated without consequences to the game.

- 1 / **Minor problem:** The negative effect is insignificant or just purely aesthetic.
- 2 / **Medium Problem:** The issue has a small effect on usability or playability.
- 3 / **Major problem:** The issue has a strong effect on usability or playability.
- 4 / **Critical problem:** Playing the game is impossible because of the issue.

Another metric that is not always used in addition to severity is frequency, in the case of *LaLuup*, however, it is interesting because it helps at prioritizing the found issues. The **frequency** rating system used in this evaluation is structured into three different grades of issue occurrence:

- 1 / **Rare:** Only affects a small amount of users or appears in rare situations only.
- 2 / **Frequently:** Affects a large amount of users or appears in many situations.
- 3 / **Constantly:** Affects every single user or appears all the time.

#	Problem Description	Ver.	Heur.	Sev.	Frq.	Score
1	The overall increase in difficulty over time is too fast.	S/E/T	2	3	2	6
2	The level's grid size is too small (played on iPhone SE and 6), there is still space to expand it further.	S/E/T	4/8	2	3	6
3	The purpose of the <b>XCollectible</b> objects is not perfectly clear and they need better audiovisual feedback.	T	6	2	2	4
4	There is some trial and error needed to understand the controls fully.	S/E/T	4	2	2	4
5	A chain of <b>Melody</b> objects is seen as a loop. It is misunderstood that only when a pattern continues playing forever and hits all <b>Spawner</b> and <b>Collectible</b> objects it is an actual loop.	S/E/T	4/5/6	3	1	3
6	The overall difficulty in the dynamic versions is higher than in the static version because of the new concepts that are introduced.	E/T	2/3	1	3	3
7	The time span the <b>XCollectible</b> objects are enabled is not estimable.	T	6	1	2	2
8	The pattern that is displayed in the <b>Spawner</b> objects is not fully clear in meaning at first.	S/E/T	8	1	1	1

**Table 6.3:** The compiled list of issues found by the experts within the heuristic evaluation of the three prototype versions of *LaLuup* (S = static version, E = event-based collectibles, T = event and time-based optional collectibles). The found issues are sorted by their total score which is already suggesting an order to fix them.

The total score  $t$  for the usability issue is then calculated, given the severity as  $s$  and frequency of appearance as  $f$ , by the following equation:  $t = s \cdot f$ . Although this equation is not commonly used and validated in heuristic evaluations in literature, in the scope of this evaluation it was still applied because it delivers fast and easy to understand results of sufficient quality. The total score is utilized to prioritize issues, enabling the author to fix the most urgent and distracting ones before the upcoming playability test. Therefore, the compiled list of issues in table 6.3 is sorted by the total score, ranked from highest to lowest.

### 6.1.2 Results

In the course of the evaluations, the experts found a total of nine usability and playability issues within the three prototype versions. After compiling the total list and merging identical ones there were eight problems left which can be seen in the already mentioned table 6.3. Interestingly, the severity rating always ranged from 1 to 3 and neither 0 (no problem) nor 5 (critical problem) was rated in any of the performed reviews. Severity is in general very uniformly distributed with two major problems and three minor and medium problems each. In the case of frequency, the majority of issues was encountered frequently with two rare and constantly occurring issues each as exceptions. In total, five of the usability issues were discovered in all game prototypes, two only in the event and time-based optional collectibles version and one in both dynamic versions of the game. The heuristics that were found to be violated most often were number 4 (“The player feels a sense of control over his/her actions.”) and 6 (“The game provides the player with appropriate feedback.”) which were mentioned three times. Two times the issues corresponded with heuristic number 2 (“The game is sufficiently challenging.”) and 8 (“The game interface is easy to use and understand.”). For the heuristics number 1, 7, 9 and 10 no usability violations were found in the evaluation process.

Of the eight issues found, two were rated with the highest total score of 6, followed by two issues with a score of 4 and four remaining issues scoring under 4. In the following list the problems are ordered from lowest to highest score as in table 6.3 and include a detailed description as well as a possible solution:

1. **Score 6:** The first problem states that the overall increase in difficulty is too fast. Players could therefore become frustrated and stop playing early on before they could grasp the controls and goals of the game in detail. This can be fixed by reducing the overall difficulty, introducing gameplay concepts more slowly and creating additional levels focused on exploring them in detail.
2. **Score 6:** Another issue found was that the grid size is too small as the screen would allow to expand it further without any problems. During the evaluations it was often a problem for the experts to draw the paths they wanted to draw without touching a different tile by mistake. This can be fixed by removing the blank space around the current test levels and zooming the camera further towards to the grid, however it is a challenging task because this could negatively impact the gameplay as the grid size would vary by level.
3. **Score 4:** Problem 3 only affects the event and time-based optional collectible version in the way that the actual purpose of the optional collectibles is not perfectly clear and hitting them needs additional feedback. Players could be distracted from

the main gameplay by trying to find out how the optional collectibles work and get disappointed by their audiovisual feedback when they are hit. By enhancing the background voice that is playing when they are hit and adding stronger visual feedback this issue could be solved.

4. **Score 4:** The next issue is again found in all versions and refers to the trial and error that is needed to fully understand the controls. Unfortunately, the concept of the `MeLody` patterns is difficult to grasp for some players, but it could definitely help to create simple tutorial levels to start with and thereby accomplish more possibilities to experiment and investigate.
5. **Score 3:** An issue that is rarely spotted because it depends on the level configuration (three `Spawner` objects are needed) is problem number 5. It states that a chain of `Melodies` is seen as a loop, meaning it is misunderstood that only when a pattern continues playing forever and hits all `Spawner` and `Collectible` objects it is an actual loop. The solution to this problem is easy to find but a little bit tricky to implement: When a chain is played and the `MeLody` gets trapped between two of the three `Spawner` objects, the collectibles outside of the unfinished loop should be reset to show that the loop is not valid.
6. **Score 3:** This problem states that the overall difficulty in the dynamic versions is higher than in the static version because of the new concepts that are introduced. Because the versions differ in gameplay elements this is not really an issue as only one version will actually be used for further development. If it will be one of the dynamic versions, tutorials will be provided to guarantee a smooth increase in challenge over time.
7. **Score 2:** Another problem is again only found in the optional collectibles version: The time span the `XCollectible` objects are enabled is not estimable and feels rather random to the player. It can be fixed by simply showing the objects only until the next `Spawner` is hit. However, this could possibly change the time when players are trying to hit them slightly.
8. **Score 1:** The last problem found indicates that the pattern that is displayed in the `Spawner` objects is not fully clear in meaning at first. This issue can be solved by a similar approach as problem 4 (by creating more tutorial levels). Additionally, the interface can be reworked in the way that the pattern includes directions in the form of lines.

As supposed, the heuristic evaluation done in this thesis shed light on the most critical usability problems of the three different game prototypes of *LaLuup*. With this information the prototype can be improved before the playability tests are held and further investigation in the field of dynamic objects and *GameFlow* is done. Interestingly, when asking the experts directly after the evaluation, the preferred version of the game was in two of three cases the one that they had played first.

### 6.1.3 Improvements

Due to the short amount of time between the two evaluations only the most critical issues could be solved. By calculating total scores and sorting the problems these critical ones were easy to depict and prioritize. Overall, three out of eight issues were solved,

mostly because of their rating, but also based on the effort to fix them. In the following listing the issues solved including their score are named by their number, ensued by their solution or the improvement that was made to prevent their further occurrence during the upcoming playtests:

- **Issue 1, Score 6:** Reducing the overall increase in difficulty over time was in fact hard to implement in regards to keeping the length of the playtests the same and introducing all concepts to the point they were introduced before. It is not possible to simply add levels because all game versions get longer by doing so. A partly solution to this problem that was chosen for time and comfort reasons regarding to the playtests was therefore to create five more tutorial levels as an own game version and give the user the chance to learn the concepts through them before starting the actual playtest. These five levels do not feature `Collectible` or `XCollectible` objects at all as they are introduced as different concepts in the game prototype versions, they are instead focused on level exploration and experimentation and feature large hexagonal grids to support familiarizing with the pattern drawing mechanics. It does, however, not change the increase in difficulty and just lowers it before playing the actual prototype versions, but this trade-off makes sense to keep the playtest on a rational time scale.
- **Issue 4, Score 4:** The trial and error feeling when learning the controls can be minimized by the same solution as issue number 1. By adding extra levels as an introduction the controls can be grasped easier and slower, changing the trial and error components to experimenting and exploration.
- **Issue 8, Score 1:** The most uncritical issue was solved too because it was a simple and effortless task to do so: By adding direction lines in the patterns displayed in the `Spawner` objects their meaning possibly became clearer. Also, the instruction levels developed for issue 1 and 4 improve the understanding for these interface components slightly.

Issue 2, also having a total score of 6 as issue 1, has not been solved because it would have changed too much within the gameplay and level design structure to be integrated at this stage in a rather short time span. One may think it is simple to rescale the levels but in fact this could create another, even bigger problem: The grid scale would become non-uniform between levels, resulting in a confusing and even more inaccurate gameplay. Therefore this change was left aside as a lightweight solution was not found and has to be considered in the results of the playability tests. As issue 3 only occurs in one version of the game prototype, was only discovered by one expert and the improvement would have needed a considerable amount of time and effort it was left aside too. Issue 5, 6 and 7 were also regarded to not affect the playtest in a substantial amount.

Some cosmetic issues regarding the prototype were also improved: The grid outline color was changed to highlight `Collectible` objects and grid holes even more. Yellow `Spawner` objects were recolored in a darker shade to emphasize the white `Melody` pattern interface because it was nearly invisible on them before. Drawing a pattern and lifting the finger from the screen now starts the `Melody` directly at the last touch instead of the `Spawner` in order to create a better understanding for the interaction method. The most extensive change was that when a level is solved and the loop is closed, both `Melody` patterns now start playing at the same time. The reason for this is that the author

observed that the experts often did not recognize at first that they solved a level in the evaluations. Everything else stayed exactly the same as in section 5.2 for the following playability evaluation.

## 6.2 Playability Test using GEQ

After the improvements determined in the first evaluation of the prototypes were implemented, the second evaluation could be conducted. In this case, the second evaluation process consists of playtests with subsequent surveys to be handed out to the players. The goal of the surveys is primarily to depict differences between the three prototype versions and discuss them in reference to the *GameFlow* model and other playability aspects. Additionally, the playtests generate a broad amount of feedback on the prototypes in general which can be analyzed and interpreted too.

### 6.2.1 Method

The surveys handed out to players had to fulfill certain requirements as reliability, validity, robustness and convenience in the way that it is easy to administer, learn and understand. Developing such measures and verifying them is difficult and time-consuming, taking from a couple months to years. Therefore, appropriate measures had to be found in state-of-the-art publications and literature rather than creating them exactly for the case of this thesis.

Fortunately, a suitable and relatively new survey by IJsselstein et al. called the *Game Experience Questionnaire* [24] was found at the time the exact questionnaire method was researched. The core module of the *Game Experience Questionnaire* and its seven components were chosen because it did not only fulfill the requirements mentioned above, but also matched the research goal.

#### Structure and Metrics

The *Game Experience Questionnaire* (shortly called GEQ) is structured into four modules: The core module focuses mainly on game experience, the social presence module is revolving around the involvement of the player with other entities, the post-game module assesses the feelings of players after they stopped playing and the in-game version of the GEQ core module is used for short, repeated surveys during game sessions.

Since the survey should be performed after playing each prototype, there is no social component in *LaLuup* and the post-game module is beyond the scope of this thesis, only the core module was chosen. The core module features 33 statements that have to be answered on a scale ranging from 0 to 4. The 33 statements can later be summarized in 7 game experience components to use as a basis for the comparison of the prototype versions. The 7 components are called Competence, Sensory and Imaginative Immersion, Flow, Tension/Annoyance, Challenge, Negative affect and Positive affect and their respective statements asked in the playtest questionnaires can be seen in table 6.4. The statements given per component already describe its meaning in detail, for example, Competence means the feeling of competence or expertise a player experiences while playing the game. The survey given to the players included all statements in original

Nr.	GEQ Statement	GEQ Component
2	I felt skilfull	Competence
10	I felt competent	
15	I was good at it	
17	I felt successful	
21	I was fast at reaching the game's targets	
3	I was interested in the game's story	Sensory/Imaginative Immersion
12	It was aesthetically pleasing	
18	I felt imaginative	
19	I felt that I could explore things	
27	I found it impressive	
30	It felt like a rich experience	
5	I was fully occupied with the game	Flow
13	I forgot everything around me	
25	I lost track of time	
28	I was deeply concentrated in the game	
31	I lost connection with the outside world	
22	I felt annoyed	Tension/Annoyance
24	I felt irritable	
29	I felt frustrated	
11	I thought it was hard	Challenge
23	I felt pressured	
26	I felt challenged	
32	I felt time pressure	
33	I had to put a lot of effort into it	
7	It gave me a bad mood	Negative affect
8	I thought about other things	
9	I found it tiresome	
16	I felt bored	
1	I felt content	Positive affect
4	I thought it was fun	
6	I felt happy	
14	I felt good	
20	I enjoyed it	

**Table 6.4:** The GEQ statements ordered by their respective components including their original order number on the survey sheet handed out to players [24].

order number and a scale to indicate the amount of truth they had felt per statement while playing the prototype, ranging from 0 to 4 as in the original GEQ version [24]:

- 0 / Not at all**
- 1 / Slightly**
- 2 / Moderately**
- 3 / Fairly**
- 4 / Extremely**

### Preparations and Procedure

The second evaluation performed consisted of the following steps: At the start the test players were guided through the first level of the five introduction levels and thereby given an overview of the interaction methods of the game. They could then play the remaining four introduction levels to grasp the pattern concept in detail. Afterwards, the test users played each of the three versions of the game prototype and were handed a GEQ core module survey to complete after each version. The sequence in which the prototypes were played was altered randomly for every playtest to make up for the increase in skill and competence the users experienced over time.

After the evaluation process was completed, the players were asked if they had any other remarks corresponding to the game prototypes such as which version they enjoyed most, if there was anything they found disturbing or impressive concerning the overall gameplay, aesthetics and mechanics.

#### 6.2.2 Results

In total, 12 people participated in the GEQ survey, one third of them female and the others male, with an overall range in age of 23 to 31. The test players were chosen with the only prerequisite that they should be accustomed to playing games on a mobile device using touch interaction (smartphone or tablet). The results per GEQ object can be seen in table 6.5(a) for the first four components and in table 6.5(b) below for the remaining three components. For each result set six statistical metrics are given: The sample variance  $s^2$ , measuring how far the data is spread out, is calculated as the average of the squared differences from the mean. The square root of the variance is the standard deviation  $s$  which shows how far the average answer is from the mean. Another metric measuring the spread is the range which is the difference between the highest and the lowest number in the data set. The mode is the most common number in the data set while the median is the one directly in the middle or if the data set is of even length, the average of the two middle ones. Finally, the mean illustrated in the data sets in this result table is the exact average of all values within the component and is therefore mathematically called the arithmetic mean  $\bar{x}$ .

The colors in table 6.5 refer to how far the values are from the desired values for this component: For Competence, Immersion, Flow and Positive affect the best value is 4, while it is 0 for Tension and Negative affect. The Challenge component's desired value is 2, lying directly in the middle of the scale, meaning that the game should be challenging enough for the players to have fun but simple enough to not get frustrated about it. Colors used are a darker shade of green (best score), a lighter green (one ranging step away), yellow (two ranging steps away), orange (three ranging steps away) and red (farthest away from best). In total, the results are very solid, ranging from darker green to yellow only. Generally, the spread in answers per component is slightly higher in many of the dynamic versions compared to the respective static version which is seen in the spread metrics such as variance, deviation and range.

The complete data sheet and survey can be found on the enclosed CD-ROM/DVD (cf. appendix A). The following list discusses the results for each GEQ component shortly and mentions possible reasons for them referencing the observations collected in the playtests:

Version	Competence			Immersion			Flow			Tension		
	S	E	T	S	E	T	S	E	T	S	E	T
Variance	0.37	0.78	0.93	1.40	1.09	1.23	1.17	1.12	1.11	0.30	0.77	0.53
Deviation	0.61	0.88	0.96	1.18	1.04	1.11	1.08	1.06	1.05	0.55	0.88	0.73
Range	2	3	3	4	4	4	3	3	4	2	3	3
Mode	3	2	3	3	3	3	3	4	3	0	0	0
Median	3	2	3	3	3	3	3	3	3	0	0	0
Mean	<b>2.63</b>	2.38	2.53	2.61	<b>2.85</b>	2.69	2.50	<b>2.88</b>	2.75	<b>0.39</b>	0.56	<b>0.39</b>

Version	Challenge			Negative affect			Positive affect		
	S	E	T	S	E	T	S	E	T
Variance	1.41	1.71	1.78	0.34	0.63	0.46	0.55	0.63	0.71
Deviation	1.19	1.31	1.33	0.58	0.80	0.68	0.74	0.80	0.85
Range	4	4	4	2	3	3	3	3	3
Mode	0	3	0	0	0	0	3	4	4
Median	1.5	2	2	0	0	0	3	3	3
Mean	1.52	<b>1.68</b>	1.57	0.46	0.44	<b>0.40</b>	3.22	<b>3.33</b>	3.28

**Table 6.5:** The results of the game experience survey sorted by the GEQ components and split up into two rows to increase readability. Once again the three prototype versions are marked by their abbreviations (S = static version, E = event-based collectibles, T = event and time-based optional collectibles).

	S Mean	E Mean	T Mean	Difference
<b>Flow</b>	2.50	<b>2.88</b>	2.75	0.38
<b>Competence</b>	<b>2.63</b>	2.38	2.53	0.25
<b>Immersion</b>	2.61	<b>2.85</b>	2.69	0.24
<b>Tension</b>	<b>0.39</b>	0.56	<b>0.39</b>	0.17
<b>Challenge</b>	1.52	<b>1.68</b>	1.57	0.17
<b>Positive affect</b>	3.22	<b>3.33</b>	3.28	0.12
<b>Negative affect</b>	0.46	0.44	<b>0.40</b>	0.06

**Table 6.6:** The GEQ components' arithmetic means in order of the amount of difference between all three game prototype version means. The difference is calculated by subtracting the minimum mean from the maximum. The best means per category are marked in green (note that highest mean is not always best depending on the component).

1. **Competence:** The solid overall result of the Competence component is close to 3 with an arithmetic mean ranging from 2.38 to 2.63. However, in regards to the event-based version, most people chose 2 (moderately) over 3 (fairly) as is seen in the statistic averages. This could be caused by the feeling of randomness at which time the **Collectible** objects switch places as some people did not understand when changes happened at first and also had the notion that the **Collectible** objects jump in their way and block their pattern drawings.
2. **Sensory and Imaginative Immersion:** Another good result was achieved regarding the Immersion component with mean values of 2.61 to 2.85 and the event-based version performing slightly better than the others. This could be coherent

with the increased amount of challenge in this version as it was already mentioned in the expert review results in section 6.1.2. Additionally the range of the answers given is very high (4), possibly because of the difficulty of self-assessment at this rather subconscious topic.

3. **Flow:** When looking at the Flow component, the results are quite similar to Immersion with slightly higher differences in the means ranging from 2.50 to 2.88 and an answer range of 3 to 4 (extremely) which can again be related to the problem of self-assessment at this topic. Again the event-based dynamic version performed best, followed by the time-based one and the static prototype version as the last.
4. **Tension/Annoyance:** The results of the Tension component are very good with most people ticking the 0 (not at all) and an arithmetic mean of 0.39 to 0.56. The event-based version has the highest score which possibly coheres with the arguments already mentioned at number 1 (Competence).
5. **Challenge:** The total results of the Challenge component show an arithmetic mean of 1.52 to 1.68 with the event-based version being closest to the desired value of 2. Again the range of answers was really high (4) and the most common ones were 0 for the static and time-based version and 3 for the event-based one. It is again clear that the event-based version was the most challenging and is not the perfect version to start playing with but eventually it seemed to be more interesting during the later gameplay.
6. **Negative affect:** A very good result (and the best of the negative components) was accomplished at the Negative affect component where mean values are between 0.40 and 0.46 and both median and mode are 0. The lowest and therefore best mean was scored by the time-based version which some people found to be very inviting because they felt like they could explore more things and get personal achievements through the optional `Collectible` objects.
7. **Positive affect:** When taking a look at Positive affect one can see that it is in fact the best result of the positive components with an arithmetic mean ranging from 3.22 at the static version to 3.33 at the event-based version and a broad range of answers again. Most people even ticked 4 in both of the dynamic versions which could be caused by the overall increase in perception when games feature dynamic objects because of the constant need of refocusing.

Another interesting data set that can be discussed is the amount of differences of the arithmetic means of the three prototype versions as illustrated in table 6.6. Generally, the differences in means are very low, between 0.38 for the Flow component and 0.06 for the Negative affect component. This could mean that the effects of dynamic objects within the game prototypes are barely influencing the overall result because thereby caused changes in gameplay are nearly undetectable for the players.

When looking at it from a different perspective and taking the differences as important, it could however be a confirmation that Flow, Competence and Immersion are more influenced by the usage of dynamic objects in the prototypes than all other GEQ components. Flow, referring to losing track of time and connection with the outside world as seen in table 6.4, describes the flow experience very similar to the original *Flow* model *GameFlow* is derived from [3]. Sensory and Imaginative Immersion, defined

in table 6.4 as the feeling of exploration and making rich aesthetic experiences, matches the idea of the flow experience too. It can be argued that the Competence component does not really blend in to that picture but when investigating a little further one can see that many of the *GameFlow* components mentioned early on in table 2.2 like Player Skills, Controls and Feedback build upon the concept of feeling skilfull and successful. Therefore, the small differences depicted could really be an indication that dynamic objects can affect the *GameFlow* criteria.

## Chapter 7

# Concluding Debate

### 7.1 Result Analysis

The two separate evaluations conducted with the *LaLuup* prototypes provide valuable information about usability and playability aspects and highlight differences between the three game versions. In the following listing, the results and findings of both evaluations are summarized and briefly analyzed:

- **Generally good and stable results:** In both evaluations the test players responded well to all versions of the game prototype, the applications were well received and complimented in terms of the gameplay, mechanics and aesthetics. This general notion is also mirrored in the evaluation data as there were no critical and only two major issues found in the expert reviews (one of them rendered rather cosmetic by the rarity of occurrence). In the playability survey all arithmetic means are above 2 for positive questions and below 1 for negative questions, resulting in a table without any orange and red cells. Especially when looking at the survey, one can see that the results are not only good but also very stable as there were not many outliers in the data set as well. This could also be caused or at least provoked by the rating scheme ranging from 0 to 4, which provides the option of choosing the middle value. However, in the GEQ the middle choice (2, meaning moderately) is not really an equal choice as the scale rates the amount of truth in a statement from not at all to extremely. As there is not really a statement that can be half true, there can only be granularities of the validity of a statement, either rendering it true to some amount or not true at all.
- **Very good audiovisual and aesthetic results:** In reference to the aesthetics of the game prototypes, the evaluation results showed that people enjoyed both the sound feedback and the visuals of the game. In the expert reviews, the game's sound was deemed original, reminding the experts of both old computer games, water drops and forest rain. One of the experts even asked for a system to record solutions to listen to them again and adding the possibility of listening to solutions of others. The visuals were complimented to be so simplified and clear, reacting to all user interactions performed. Both the Immersion and Audiovision heuristics were not violated at all in the conducted usability evaluation. In the second evaluation performed, the GEQ components Sensory and Imaginative Immersion

and Positive affect are in the top three positive results. These components contain statements like *It was aesthetically pleasing, I found it impressive* and *I enjoyed it*. Assuming that player enjoyment is strongly coupled to the audiovisual experience, these results confirm the opinion of the experts.

- **Challenge and difficulty problems:** In the expert reviews, the top scoring issue (6) was in regards to the Challenge heuristic, indicating that *The overall increase in difficulty over time is too fast*. To solve the problem, five tutorial levels were implemented to specifically encourage the users to experiment with the game mechanics before playing one of the three prototype versions. This, however, did not solve the fast increase in difficulty and just lowered it before playing the actual game versions, but was chosen as a compromise to keep the playtest length the same. As expected, the Challenge component of the GEQ showed a broad answer range of 4 with the most common ones being 0 (not at all) for the static and time-based version and 3 (fairly) for event-based prototype, but interestingly the results were not too far from the desired value. The event-based version can be seen as a little more challenging whereas the others seemed to be not challenging enough. Generally, there is some room for improvement concerning challenge as well as difficulty and it was expected as a result because the balancing of puzzle games is always a hard and time-consuming task with no perfect solution.
- **Few interface, controls and feedback issues:** Some issues concerning the interface, controls and feedback of the game prototypes were also found in the expert reviews. Many have been solved by now, but the remaining few problems have to be looked at in detail and improved in the future. The most concerning one is probably the screen resolution issue, stating that *The level's grid size is too small, there is still space to expand it further*. This issue cannot be solved in just expanding the grid as this creates a non-uniform scale changing per level which could negatively impact the gameplay, therefore, other possible solutions have to be created. Other issues such as the clarity of the `XCollectible` objects' purpose, the learning of the controls, the way a loop works and the interface pattern also have to be solved to create an enjoyable and satisfying gaming experience.
- **Small differences between prototype versions:** In the GEQ survey performed, the differences in arithmetic means of the components Flow, Competence and Immersion were larger than all other differences in means between the three prototype versions. As already argued in section 6.2.2, this could still be coincidence because of the small range the differences have and the dynamic objects thereby may not affect the components linked to the *GameFlow* model more than they influence all other components. However, the three largest differences are as expected seen in the Flow, Competence and Immersion components. These components are strongly connected to *GameFlow* (the detailed explanation can be found in section 6.2.2 at the bottom paragraph). Therefore, the possibility of dynamic objects actually affecting the *GameFlow* model and the corresponding flow experience cannot be ruled out.

Many of the stated conclusions are drawn on the basis of inference from a holistic view of all evaluation results and observations collected over the course of this thesis. This shows that there is still room for extensive research and data collection to support

the assumptions made within the result analysis above. However, the author suggests that another evaluation should only be performed in the next stage of development as these prototype versions were sufficiently tested and some improvements should be integrated before repeating the rather time-consuming tests again.

## 7.2 Further Prospects

At the current stage, *LaLuup* naturally has a few solvable issues that are also reflected in the evaluations. It is however important to mention that the game is still a prototype and its audio system is a proof of concept as Tintinnabuli styled music has never been utilized in this form before. Because of the evaluations, however, the next steps in improving the game have already been discussed:

- **Navigating the Player:** As the controls still seem to be trial and error to some test players and the game interaction is not clear for many users, it will most likely be necessary to implement a tutorial at the start of the game. However, this tutorial should be hidden by design, meaning that the users should not be restricted or forced to create exact *Melody* patterns to achieve their player skills. They should rather be encouraged by the level design to perform interactions that lie within the wanted scheme of patterns.
- **Balancing Challenge and Difficulty:** With progressing development and a ready-to-play interaction tutorial, it will finally be possible to create a reasonable challenge curve that fits for most users. Currently it is not planned to have different difficulty levels because all of the levels are designed by hand and with every additional difficulty level the work doubles, but eventually there could be different challenge stages to choose from, enabling the players to start with an easy level or already skip some scenes to experience more challenging levels right away. Ideas like this are currently being discussed in detail because they will be relevant in the near future and should be implemented until the next playtests to considerably improve the player experience.
- **Optimizing Interface and Feedback:** Concerning the open interface and feedback issues, many conceptual possibilities to fix them are already in their test phase. The grid size problem (especially seen on smaller devices) can be reduced by integrating zoom interactions, however this could complicate the process of keeping track of the whole grid. As this is only an assumption made by the author, different zoom mechanisms still have to be tested in the future.
- **Increasing Variation with Dynamic Objects:** As mentioned at the very beginning of this thesis in section 1.2, many players asked for more variation in sound aesthetics and gameplay in the informal user tests performed. After improving the prototypes further in the course of the thesis project the sound aesthetics seemed to become more sophisticated and the later conducted evaluations generally revealed very good results in audio and visual aesthetic aspects. Nevertheless, the gameplay variation did not change that much but for the dynamic prototype versions. In the GEQ survey it can be depicted that the event-based version, admittedly being the most difficult, also has the most top scores in arithmetic means (as seen in table 6.6). It may also be that the dynamic objects actually affect the flow experi-

ence, therefore it could be interesting to integrate dynamic objects within *LaLuup* without overusing them as in the event-based version to keep the difficulty level in a reasonable range.

### 7.3 Conclusion

In the course of this thesis, extensive research in the fields of algorithmic and generative composition as well as flow and playability in casual games has been done. On the basis of this research an interactive audio system for the game prototype called *LaLuup* was designed, developed, integrated in the game and improved over time. Dynamic objects were used to alter the gameplay and mechanics slightly, creating three different prototype versions to evaluate. Two evaluations followed after the prototype construction, revealing usability issues to fix and differences between the prototypes to discuss. The effects of the additional dynamic objects were then studied in regards to the *GameFlow* model, revealing possible correlations between them.

As the developed game prototype is in a rather early stage and *Kunabi Brother* is an indie game company with a small staff, evaluations had to be done within a very small amount of time while still revealing useful results. When looking back, the expert reviews definitely supported identifying usability issues and the subsequently improved prototypes created a good basis for the following survey. The effort and time spent with organizing the reviews turned out to be valuable to find problems that the developers including the author did not know of before. In the case of the GEQ-based playtests the resulting data also seemed promising, however one cannot argue that the tendencies found are statistically relevant as the test group was rather small. This fact complicates the result interpretation and the following implementation of new or improved features within the game prototype. However, it is still very helpful to test such an early prototype with actual casual game players because the observations made in the playtests uncover playability issues and possible solutions for them.

Casual mobile games flourish nowadays but at the same time many of them end up not succeeding because of the largely saturated and highly competitive market. In this amount of content it is needed to stand out and distinguish from the masses, for example by designing levels by hand instead of generating them, or utilizing technologies that were not applied in this exact way before. *LaLuup* tries to achieve these concepts by using a specially designed audio system and levels constructed by the hand of a game designer. Many puzzle-game titles currently on the market are using similar approaches with other special technologies, including AR and VR for mobile devices, to create interesting new player interaction and gameplay methods. As smartphones and tablets are currently one of the most accessible technological devices around today the game sector is still expanding with many possibilities to explore and experiment with.

The future holds a multitude of new combinations of interaction methods, technologies and generative audio and visual aesthetics. *LaLuup* aims to be an example for utilizing a different kind of technology (the audio system) to alter the way casual games are played, shifting the focus on exploration rather than solving level by level. By adding dynamic objects and altering the gameplay to study effects in regards to the *GameFlow* model the author hopes that this example provides relevant information to consider when working on other casual mobile games utilizing similarly special techniques.

## Appendix A

# CD-ROM/DVD Contents

**Format:** DVD-RW, 4.7 GB, Single Layer, ISO9660-Format

### A.1 Project

- **Project/src:** Project files for *LaLuup* (Unity Version 2017.3.0f3)
- **Project/bin1:** *LaLuup* executables of the first evaluation (Xcode 9.4, iOS 11.4)
  - **/static:** Static version
  - **/event:** Event-based collectibles
  - **/time:** Event and time-based optional collectibles
- **Project/bin2:** *LaLuup* executables of the second evaluation (Xcode 9.4, iOS 11.4)
  - **/instruction:** Additional instruction levels
  - **/static:** Static version
  - **/event:** Event-based collectibles
  - **/time:** Event and time-based optional collectibles
- **Project/video:** Videos of the improved executables (second evaluation)

### A.2 Thesis

- **/Thesis.pdf:** Thesis (this document)
- **/images:** Original raster and vector images

### A.3 Evaluations

- **Evaluation:** Original evaluation files
  - **/ExpertEvaluation.pdf:** Handout the experts received filled with results
  - **/PlayabilityTest.pdf:** Survey handout including a short game description, summarized user notes and observations
  - **/PlayabilityTestResults.pdf:** Complete result tables of the GEQ

# References

## Literature

- [1] Aubrey Colter. “SoundFORMS : Manipulating Sound Through Touch”. In: *CHI Extended Abstracts on Human Factors in Computing Systems - CHI EA '16*. San Jose, CA, USA, 2016, pp. 2425–2430 (cit. on p. 5).
- [2] Hermann Conen. *Arvo Pärt. Die Musik des Tintinnabuli-Stils*. 1st ed. Cologne, Germany: Dohr, 2006, p. 202 (cit. on pp. 17, 25).
- [3] Mihaly Csikszentmihalyi. *Flow. The Psychology of Optimal Experience*. 1st ed. New York, USA: Harper Perennial, 1990 (cit. on pp. 10, 56).
- [4] Heather Desurvire, Martin Caplan, and Jozsef A. Toth. “Using Heuristics to Evaluate the Playability of Games”. In: *Extended Abstracts of the Conference on Human Factors and Computing Systems - CHI EA '04*. Vienna, Austria, 2004, p. 1509 (cit. on p. 45).
- [5] Melissa a Federoff. “Heuristics and Usability Guidelines for the Creation and Evaluation of Fun in Video Games”. Bloomington, Indiana, USA: Indiana University, 2002, p. 52 (cit. on p. 10).
- [6] Fong Ling Fu, Rong Chang Su, and Sheng Chin Yu. “EGameFlow: A Scale to Measure Learners’ Enjoyment of E-learning Games”. *Computers and Education* 52.1 (Jan. 2009), pp. 101–112 (cit. on p. 11).
- [7] Katherine Isbister and Noah Schaffer. “Chapter 6 - Heuristic Evaluation of Games”. In: *Game Usability*. Ed. by Katherine Isbister and Noah Schaffer. Burlington, Massachusetts, USA: Morgan Kaufmann Publishers Inc., 2008, pp. 79–89 (cit. on p. 45).
- [8] Aki Järvinen, Satu Heliö, and Frans Mäyrä. *Communication and Community in Digital Entertainment Services*. Tech. rep. Tampere, Finland, p. 78. URL: <http://tampub.uta.fi/tup/951-44-5432-4.pdf> (cit. on p. 12).
- [9] Kalle Jegers. “Pervasive GameFlow: Identifying and Exploring the Mechanisms of Player Enjoyment in Pervasive Games”. Umeå, Sweden: Umeå Universitet, Inst för Informatik, 2009, p. 122 (cit. on p. 11).
- [10] Sergi Jordà. “The reacTable”. In: *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems - CHI EA '10*. Atlanta, GA, USA, 2010, pp. 2989–2994 (cit. on pp. 4, 5).

- [11] Sergi Jordà et al. “The reacTable: a Tangible Tabletop Musical Instrument and Collaborative Workbench”. In: *ACM SIGGRAPH Sketches on - SIGGRAPH’06*. Boston, Massachusetts, USA, 2006, p. 91 (cit. on p. 5).
- [12] Jesper Juul. *A Casual Revolution. Reinventing Video Games and Their Players*. 1st ed. Cambridge, Massachusetts, USA: The MIT Press, 2010 (cit. on pp. 8, 9).
- [13] Hannu Korhonen. “Comparison of Playtesting and Expert Review Methods in Mobile Game Evaluation”. In: *Proceedings of the 3rd International Conference on Fun and Games - FnG’10*. Leuven, Belgium, 2010, pp. 18–27 (cit. on pp. 45, 47).
- [14] Hannu Korhonen. “Evaluating Playability of Mobile Games with the Expert Review Method”. Tampere, Finland: School of Information Sciences, University of Tampere, 2016, p. 280 (cit. on pp. 12, 13, 44, 45, 47).
- [15] Hannu Korhonen and Elina M. I. Koivisto. “Playability Heuristics for Mobile Multi-player Games”. In: *Proceedings of the 2nd International Conference on Digital Interactive Media in Entertainment and Arts - DIMEA’07*. Perth, Australia, 2007, p. 28 (cit. on p. 45).
- [16] Jussi Kuittinen et al. “Casual Games Discussion”. In: *Proceedings of the Conference on Future Play - Future Play’07*. London, Ontario, Canada, 2007, pp. 105–112 (cit. on pp. 8, 9).
- [17] Rolf Molich and Jakob Nielsen. “Heuristic Evaluation of User Interfaces”. In: *Proceedings of the Conference on Human Factors in Computing Systems - CHI’90*. Seattle, Washington, USA, 1990, pp. 249–256 (cit. on p. 44).
- [18] Jakob Nielsen. “Enhancing the Explanatory Power of Usability Heuristics”. In: *Conference Companion on Human Factors in Computing Systems - CHI’94*. Boston, Massachusetts, USA, 1994, p. 210 (cit. on p. 44).
- [19] Jakob Nielsen and Morgan Kaufmann. *Usability Engineering*. 1st ed. San Francisco, California, USA: Morgan Kaufmann Publishers Inc., 1993, p. 362 (cit. on pp. 44, 45, 47).
- [20] Cárthach Ó Nuanáin and Liam O’ Sullivan. “Real-time Algorithmic Composition with a Tabletop Musical Interface”. In: *Proceedings of the 9th Audio Mostly on A Conference on Interaction With Sound - AM’14*. Aalborg, Denmark, 2014, pp. 1–7 (cit. on p. 6).
- [21] Janne Paavilainen. “Playability: A Game-Centric Definition”. In: *Extended Abstracts on Human-Computer Interaction in Play - CHI PLAY’17*. Amsterdam, Netherlands, 2017 (cit. on p. 14).
- [22] Janne Paavilainen et al. *GameSpace. Methods for Design and Evaluation for Casual Mobile Multiplayer*. Tech. rep. Tampere, Finland, 2009, p. 103. URL: <http://tampub.uta.fi/english/tulos.php?tiedot=293> (cit. on p. 11).
- [23] David Pinelle, Nelson Wong, and Tadeusz Stach. “Heuristic Evaluation for Games: Usability Principles for Video Game Design”. In: *Proceedings of SIGCHI Conference on Human Factors in Computing Systems - SIGCHI’08*. Florence, Italy, 2008, pp. 1453–1462 (cit. on p. 45).

- [24] K Poels, Y A W de Kort, and W A IJsselsteijn. *The Game Experience Questionnaire*. Eindhoven, Netherlands: Technische Universiteit Eindhoven, 2013, p. 10 (cit. on pp. 43, 52, 53).
- [25] Aditya Ponnada and Ajaykumar Kannan. “Evaluation of Mobile Games using Playability Heuristics”. In: *Proceedings of the International Conference on Advances in Computing, Communications and Informatics - ICACCI'12*. Chennai (Madras), India, 2012, pp. 244–247 (cit. on p. 47).
- [26] José Luis González Sánchez et al. “Playability: Analysing User Experience in Video Games”. *Behaviour and Information Technology* 31.10 (2012), pp. 1033–1054 (cit. on p. 13).
- [27] Scott McCloud. *Understanding comics*. New York, USA: Harper Collins Publ. USA, 1994, p. 224 (cit. on p. 22).
- [28] Nathan Scott. “Music to Middleware: the Growing Challenges of the Game Music Composer”. In: *Proceedings of the Conference on Interactive Entertainment - IE'14*. Newcastle, Australia, 2014, pp. 1–3 (cit. on p. 5).
- [29] Penelope Sweetser and Peta Wyeth. “GameFlow: A Model for Evaluating Player Enjoyment in Games”. *Computers in Entertainment - Theoretical and Practical Computer Applications in Entertainment* 3.3 (2005), pp. 3–3 (cit. on pp. 10, 11, 35, 45).
- [30] Penelope Sweetser et al. “GameFlow in Different Game Genres and Platforms”. *Computers in Entertainment - Theoretical and Practical Computer Applications in Entertainment* 15.3 (2017), pp. 1–24 (cit. on pp. 10, 11).
- [31] Sharon Lynn Chu Yew Yee, Henry Been-Lirn Duh, and Francis Quek. “Investigating Narrative in Mobile Games for Seniors”. In: *Proceedings of the 28th International Conference on Human Factors in Computing Systems - CHI'10*. Atlanta, Georgia, USA, 2010, pp. 669–672 (cit. on p. 11).

## Audio-visual media

- [32] Guy Birkin. *Mapping Tintinnabuli Transformations - Chart 1*. Mar. 2014. URL: [https://aestheticcomplexity.files.wordpress.com/2014/03/tintinnabuli\\_chart-1.png](https://aestheticcomplexity.files.wordpress.com/2014/03/tintinnabuli_chart-1.png) (cit. on p. 26).
- [33] Batuhan Bozkurt. *Otomata - Generative Musical Sequencer*. July 2011. URL: <https://www.youtube.com/watch?v=k8EfrXihWg> (cit. on p. 7).
- [34] Marlene Brandstätter and Ulrich Brandstätter. *Foreverloops - Hybrid Sandbox Game*. 2017. URL: <https://www.foreverloops.com/images/foreverloops.png> (cit. on p. 8).
- [35] Richard Vreeland. *January - Generative Music Tool*. Jan. 2013. URL: <https://vimeo.com/104276908> (cit. on p. 6).

## Online sources

- [36] Guy Birkin. *Programming Arvo Pärt*. Nov. 2011. URL: <https://aestheticcomplexity.wordpress.com/2011/11/11/programming-arvo-part/> (cit. on p. 25).
- [37] Charlie Huguenard. *Making a Music System*. Aug. 2016. URL: <http://designingso und.org/2016/08/making-a-music-system-part-1/> (cit. on pp. 24, 27).