
MUSICPARTY HERO

Design and development of a local multiplayer music party game



Ignacio López Rivas

Advisor: Dr. Luis Amable García Fernández

Department of Engineering

Universitat Jaume I de Castellón

This dissertation is submitted for the bachelor's degree of

Videogame Design and Development

Abstract

It is a common situation that a group of friends meets at someone's house. Maybe for dinner, to watch a movie or for just having coffee. It is also common that sometimes someone brings the idea of playing a game, this kind of games, called party games, are mostly designed to be playable for a fixed quantity range of people and here is the trouble. Some people often have to pair for playing like one, or even worse, some of them are left out of the game making uncomfortable the people who got to play.

This issue has never really being addressed by the board gaming industry because it is very difficult to design a consistent game experience for any number of players. Video games, on the other hand, benefit from the number of players in many ways. It is a fact that a well designed multiplayer video game becomes more fun as more players join the game as the interactions of the players are more unpredictable than the possible pre-programmed ones by the game developers.

The only obstacle that party video game industry has not addressed at all is to give the potential players the possibility of playing by having to have enough gamepads for every player and of course the cost that this issue implies.

Music-party Hero is an experimental project that tries to address this problem by making use of one piece of high technology everyone almost everytime always carries around, his/her smartphone. By using it as a game controller and a PC/console/Smart TV as the main machine that executes the game on a big screen, it is possible to connect them via WiFi connection and have almost any number of people get together to play the same game at the same time.

Keywords: Multiplayer, WiFi, party game, game design, smartphone

Table of contents

1	Technical Proposal	8
1.1	Motivation	8
1.2	State of the Art.....	9
1.2.1	Lan video games.....	9
1.2.2	Android	9
1.2.3	Local multiplayer	10
1.2.4	Music video games	10
1.3	Game Preview	10
1.4	Objectives.....	11
1.5	Justification	13
1.5.1	VJ1222 – Video Game Conceptual Design.....	13
1.5.2	VJ1227 – Video Game Engines.....	14
1.5.3	VJ1228 – Network & Multiplayer Systems.....	14
1.5.4	VJ1236 – Sound Production and Realization Techniques.....	14
1.5.5	VJ1229 – Mobile Devices Applications.....	14
1.6	Tools.....	15
1.6.1	Game Engine	15
1.6.2	Programming tools.....	15
1.6.3	3D Modeling and texturing.....	15
1.6.4	2D and UI design	15
1.6.5	Debugging, testing and profiling	16
1.6.6	Version control system.....	16
1.6.7	Video	16
1.6.8	Audio	16
1.6.9	Documentation	16
1.7	Project Plan.....	17

1.7.1	Project Schedule.....	17
1.7.2	Deliverables.....	18
2	Game Design Document	20
2.1	Gameplay	20
2.2	Scenario Overview.....	20
2.3	Player Controls.....	21
2.4	Game Mechanics	22
2.5	Art and UI Design.....	24
2.6	Colors.....	25
3	Game Development.....	26
3.1	General Overview.....	26
3.2	Core game.....	27
3.2.1	MIDI.....	27
3.2.1.1	Understanding MIDI	27
3.2.1.2	Reading MIDI	31
3.2.1.3	Parsing MIDI	31
3.2.2	Setting up the scenario	32
3.2.3	Gameplay.....	33
3.3	The Smartphone as a Controller.....	34
3.3.1	Unity networking: HLAPI vs LLAPI	34
3.3.2	Connecting devices.....	35
3.3.3	Commands, RPCs and raw network messages	36
3.4	Game Over screen.....	37
4	Testing & Evaluation.....	38
4.1	Testing.....	38
4.1.1	Verification	38
4.1.2	Validation	38
4.2	Evaluation.....	39
4.2.1	Hardware	39

5 Results	41
5.1 Technical proposal.....	41
5.2 Game design document.....	41
5.3 Technical Report.....	41
5.4 Game application	41
5.5 Project Defense Video.....	42
5.6 Project Defense Presentation	42
6 Project deviations	43
6.1 Game concept.....	43
6.2 Project schedule.....	44
7 Conclusions.....	46
7.1 Future work.....	47
References	48

List of figures

Figure 1 – Scenario.....	21
Figure 2 – Player controls	22
Figure 3 – Button pushed	22
Figure 4 – Scenario with Happiness Bar	23
Figure 5 – System Overview.....	26
Figure 6 – Scenario (Right side view).....	33

List of tables

Table 1 – Documentation Breakdown.....	17
Table 2 – Game Development BreakDown.....	17
Table 3 – Game Design Document BreakDwon.....	18
Table 4 – Evaluation BreakDown.....	18
Table 5 – Deliverables Breakdown.....	19
Table 6 – Header chunk.....	28
Table 7 – Track Chunk.....	28
Table 8 - MIDI Channel Event Structure.....	29
Table 9 – Note On Event Structure	29
Table 10 - Note Off Event Structure	30
Table 11 – Note Conversion Table.....	30
Table 12 – Actual Documentation Breakdown.....	44
Table 13 – Actual Development Breakdown.....	44
Table 14 – Actual Game Design Breakdown.....	44
Table 15 – Actual Evaluation Breakdown.....	44

Chapter 1

Technical Proposal

In this chapter, an overview of the project will be presented. The project consists of a combination of two software applications. The first one it is a PC executable and the latter is an Android app, combining they two in a music hero like game. The distinctive feature of this project is the usage of two different devices that would display different information in real time of the game besides of being multiplayer through a local area network. This way multiple Android devices could play a game in which they would hold private information at their screen and common information at the PC screen.

1.1 Motivation

Within the last years, it has been happening an approach of the PC to the living room (1) (2). The common scenario for PC games is being single player or online multiplayer. It has been left out a segment of the market that consoles filled out long ago, the local multiplayer. We see how gradually the video game digital market show an increase offer of local multiplayer capable games however they are still focused mainly on online multiplayer. This leads us to find the specific local multiplayer offer to be very small. One of the main reasons of this issue is the requirement for the player of owning several gamepads in order to enjoy the experience. This circumstance is not common and therefore the potential buyer ignores the game just for not having the hardware requirement.

This project the aims to tackle this issue by investigating the possibility of using as a controller a piece of hardware that many people already have, and Android device. This is possible as many modern homes have already a WiFi connection. This opens the door to network them to a PC through the TCP/IP protocol. This way, using Unity for the implementation of the game and the network protocol it is possible to develop a game that could be played multiple people without limitations of available gamepads or USB ports available.

The proposed game is a music party game where the players have to press buttons as if they were playing the song that is sounding at the time the game is happening. The game

will present a visual representation of the music that is to come before the button would have to be pressed so the player will know which button and when it has to be pressed. The key aspect of the game is that it has to be fully synchronized with the music played to create the illusion of being playing the song for the player and having him to believe it. Parallelisms with *Guitar Hero* game type are clear as it is the main source of inspiration for this game.

1.2 State of the Art

A theoretical introduction to the concepts dealing with the system should be presented before continuing with this technical proposal. In addition, different lines of existing work in the targeted area should also be discussed.

1.2.1 Lan video games

As people of the specialized press (3) and the industry (4) have stated, LAN gaming is dying due to the growing access to high speed internet connection of the last decade. Online gaming is killing it for natural reasons, in the late 90s LAN gaming was the only way of getting a real sense of multiplayer experience and that has changed. Now is time to explore some other uses to this technology with more modern devices like the smartphones.

1.2.2 Android

Android is the leading smartphone operating system with about an 86.8% market share followed by iOS, the second most popular one with a share of 12.5% (5). Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007. It is based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets but also used among other devices like SmartTVs, watches, cars and even game consoles.

The proper way of developing for this platform is making use of the official languages supported by the platform like Java and more recently Kotlin (6). Still, our main goal is to develop a multiplatform game and we are going to use the multiplatform feature of Unity for this purpose.

1.2.3 Local multiplayer

Some of the earliest games were multiplayer like *Tennis for Two* (7) but not through networked machines. The early stage of multiplayer gaming was sharing the screen among the players while each one of them had their personal input controller.

This game type got stronger at the arcade era with games like *Gauntlet* (8) and among the home consoles from the 80s to the mid 00s where the internet connection led the market towards the online.

1.2.4 Music video games

“A music video game, also commonly known as a music game, is a video game where the gameplay is meaningfully and often almost entirely oriented around the player's interactions with a musical score or individual songs.” (9) This is a well round up description for the game genre of the game this technical proposal is about, in addition, there are three types of sub-genres:

- **Music memory games**

These games test a player's musical memory although the majority of such games primarily emphasize rhythm as the major gameplay-determinative musical element.
- **Freeform music games**

These games are those in which the creation of music takes predominance over gameplay.
- **Hybrid music games**

These games are intended to permit the player's direct interaction with the soundtrack and to encourage the creation of a synesthetic experience.

1.3 Game Preview

In this section a general preview of how the game should be like is presented, a more detailed and in-depth description can be found at the game design document. The game is a compound of two applications and they are described separately:

- PC application:

The main menu will contain at least two buttons (*Play* and *Quit/Exit*). Other optional buttons like *Options*, *About* may or may not be at the release depending on the ongoing development of the game.

The *Play* button will lead to a new screen which ought to be a network lobby that will show off the IP of the server. This way the Android version users would know which machine direction they have to enter. As users join the game, the lobby will fill with their names and random colors. The screen also should have a *Next* button that would open the song selection screen. Only one title will appear at the center of the screen at the time. Once the song is selected, the game starts with the typical scenario of a music hero type game in which there are three lanes for each player and the music notes comes from the horizon to reach the point they should be played and clicked by the player from its game controller. This way each player earns points and the player with the highest score at the end of the song wins the game.

- Android app:

The main menu will simply consist of a couple of buttons that will be *Connect to server* and an *Exit* buttons the former will open up an input box which will ask the user for the server local IP to join the game. Once introduced the screen will tell the user to wait until the host of the game starts it. As the game start, the screen will fill with the three colors of the user buttons to play the game outlined with the assigned color so the player can quickly identify which lanes are theirs.

1.4 Objectives

The course syllabus for VJ1241 - Bachelor' s Degree Final Project states that the generic and specific competence to be acquired is:

- *Ability to perform individually and present and defend in front of a university tribunal an original exercise, consisting of a project in the field of video game design and development of a professional nature that synthesizes and includes the skills acquired in the degree.*

The related learning outcomes are:

- *Individually plan and implement an original project of a professional nature in the field of video games and where the skills acquired in the degree are synthesized and included.*
- *Write a report in English, orally present and defend it in front of a university tribunal an original project of a professional nature in the field of video games and where the skills acquired in the degree are synthesized and included.*

A wide perspective of this statements is that the objective of this project is to learn to develop autonomously a professional video game related project and to properly document it. Besides it also bounds the student with the duty of present and defend it publicly.

Regarding the project itself, we could say that the objective of Music Hero Party is to create an enjoyable experience for multiple users at the same time that goes along with the music. From a more technical point of view, we could say that the objective is to develop a synchronous application that communicates several devices with a master one that executes the main application and acts as a server. These requirements can be split for a better understanding in these points:

- a. Create a game playable by several players in the same room using a single computer.
- b. The game controller for each player will be its own Android device.
- c. The information showed at the computer and at every Android device will be synchronized and appropriate for each player.

From the student point of view we could break the project into some smaller objectives like these:

1. O1 – Technical Proposal: Develop an interesting idea and learn to write a document that properly helps the potential reader to get a correct impression of what the project is about and how is going to be executed.

2. O2 – Work under a deadline: Learn (if not already) to accomplish a non-minor project under the pressure of a deadline and manage to do it while coursing more subjects.
3. O3 – Push boundaries: The trying of not to keep it as a plain and simple game as a final project. Exceed the learning outcome of the degree by going deeper at some area, in this case, the network field of gaming.
4. O4 – Make it scalable: Develop and scalable project so that in case of resulting an attractive outcome, be able to keep working on it in order to reach a public release and hopefully make it profitable.
5. O5 – Technical Report: Write a technical report that summarizes the process of the developing of the game and its difficulties and overcomes, the learning results and how it could evolve.

1.5 Justification

In this section, the relationship between the project and a selection of the courses taught at the bachelor's degree will be stated. Bear in mind that this selection is done following the criteria of the most influential courses realized although almost every subject has had its contribution.

1.5.1 VJ1222 – Video Game Conceptual Design

This was the only game design related course through the bachelor and had to be on this list as the game was designed from scratch. The learning outcome from this subject is:

- *E15 – Define rules that harmonize with the available technological possibilities and which provide fluidity. Set game dynamics as principal elements when making a design.*
- *E15, G10 – Design systems which balance the game mechanics, the objectives to be achieved and possible conflicts inside and outside the game.*
- *E15, G10 – Design balanced scenarios and environments to the game.*
- *G04 – Write a video game technical document in English*

1.5.2 VJ1227 – Video Game Engines

At this subject, we learned how this specific kind of frameworks work internally and how to use Unity as an example of one of the most widely used. The learning outcome from this subject is:

- E12, G09 – Evaluate the technical characteristics of game engines as a technology for video game creating.
- E12, IR06, IR07 – Make extensions and modifications over game engines.
- E12, IR07, IR14 – Make use of game engines for video game creating.

1.5.3 VJ1228 – Network & Multiplayer Systems

Through this course, we learned the fundamentals of every multiplayer system is constructed on and therefore takes a special place at this list of subjects. The learning outcome from this subject is:

- E05, IR05, IR11 – Describe the functioning of game servers.
- E05, IR05, IR11 – Build network applications.
- E05, IR05, IR11 – Make use of actual multiplayer game engines.

1.5.4 VJ1236 – Sound Production and Realization Techniques

As the main art content of this game, the sound and the music have a protagonist part at the gameplay of this game and should have a proper treatment which has been achieved by coursing this optional subject. The learning outcome from this subject is:

- E10, E14 – Develop production of audio in video games projects autonomously.
- E10,E14,IR09 – Get to know and assimilate theoretical fundamental concepts and technical procedures and tools (hardware and software) that allows to manipulate successfully the sound resources for each type of production.

1.5.5 VJ1229 – Mobile Devices Applications

Despite of making use of Unity for the Android app, the outcome learning of this course made us aware of the technical capabilities of these machines and how to use them. The learning outcome from this subject is:

- IR01 – Make use of libraries for creating video games and applications over devices.
- IR01, IR02 – Explain the technologies for design and creation of video games and applications over mobile devices.
- IR01,IR02,IR08,IR09,IR10,G02,G07 – Analyze the technical characteristics of technologies for the video game and application creation on mobile devices.

1.6 Tools

This section lists the tools that we have had used through the development of this game. The tools are grouped according to the function they served to.

1.6.1 Game Engine

- Unity: It is by definition a framework that is primarily used to develop video games and simulations for computers, consoles and mobile devices (10). It is a well-known and reliable software with a high populated community. The downside of this engine is that is poorly optimized for consoles and that has gee-whiz graphics by default that have to be tweaked by customs shaders to get away from that aesthetic. We learned it at VJ1227.

1.6.2 Programming tools

- Visual Studio: Developed by Microsoft and free since late 2014, it has been used as the main code tool. It has official support using it alongside Unity for programming in C#. WE have used it for several subjects through the bachelor's degree.

1.6.3 3D Modeling and texturing

- Autodesk 3Ds Max 2016: 3D modeling software developed by Autodesk that has been used to create some of the primitives used in the game. It has been taught in VJ1216 and VJ1226.

1.6.4 2D and UI design

- Adobe Photoshop: Adobe Photoshop is a raster graphics editor developed and published by Adobe Systems and taught to us in VJ1204 and VJ1223.

1.6.5 Debugging, testing and profiling

- Unity Profiler window: Unity has a built-in profiler that helps us to optimize our game. It can report the percentage of time spent rendering, animating or in our game logic.
- Android Device Monitor: This tool provides a graphical user interface that allow us to read the console and debugging and analysis tools.

1.6.6 Version control system

- Git: An open source version control system designed mainly by Linus Torvalds in order to help the creation of the Linux kernel. Now it is widely used to develop almost any kind of software.
- Bitbucket: As remote repository to keep the project safe.

1.6.7 Video

- Adobe Premiere: Video editing app developed by Adobe Systems used in VJ1230.

1.6.8 Audio

- Cakewalk Sonar: As a Digital Audio Workstation used to compose and edit music.
- Adobe Audition: Audio editing tool used to clean and tweak sounds used in VJ1236.

1.6.9 Documentation

- Microsoft Office Word: It is a word processor developed by Microsoft and released by first time in 1983.
- Microsoft Office PowerPoint: Slide editor by Microsoft used for creating presentations.

1.7 Project Plan

1.7.1 Project Schedule

This section presents the intended planning for Music-Hero Party. The project has been divided into sections for better understanding.

1 Documentation

Elaboration of all documentation related with the project, including a technical proposal, a technical report a video demonstration and a slide presentation.

Table 1 - Documentation Breakdown

Documentation				
ID	Task	Period	Total Days	Total Hours
TP	Technical proposal	30-01-2017 to 9-02-2017	10	14
TR	Technical Report	19-06-2017 to 2-07-2017	14	40
PDV	Project Defense Video	9-06-2017 to 12-06-2017	3	4
PDP	Project Defense Presentation	9-06-2017 to 12-06-2017	3	2
Total				60

2 Game Development

The game development phase could be split into four minor phases that reflects how the game was built step by step. Each of these steps could be broke down again into smaller tasks that were indispensable: analysis, design, implementation and testing. For simplicity those tasks have not been displayed at this table and only the major tasks are showed.

Table 2 - Game Development BreakDown

Game Development				
ID	Module	Period	Total Days	Total Hours
MP	MIDI Parsing	19-04-2017 to 28-04-2017	9	50
GM	Game mechanics	29-04-2017 to 12-05-2017	13	40
PCAC	PC-Android Communication	30-05-2017 to 06-06-2017	7	50
AD	Android Development	06-06-2017 to 20-06-2017	14	30
Total				170

3 Game Design

The living design document that works as a guiding vision towards the objective. This is usually the most important document every game ought to have in order to make

the team work in the same direction and leave nothing to chance, as our project will be focused on technical aspects, the time spent on this document should not drain time from the development time. Only in very small teams like two or one person is avoidable, however is still recommended. The GDD will be in chapter two.

Table 3 – Game Design Document BreakDown

Game Design				
ID	Task	Period	Total Days	Total Hours
GDD	GDD production	19-06-2017 to 02-07-2017	13	40
Total				40

4 Game Evaluation

All the testing performed during the development and additional testing done afterwards.

Table 4 – Evaluation BreakDown

Game Evaluation				
ID	Task	Period	Total Days	Total Hours
DT	Development Testing	19-04-2017 to 29-06-2017	70	12
AE	Application Evaluation	30-06-2017 to 02-07-2017	3	8
Total				30

1.7.2 Deliverables

The project has six deliverables as outcomes of the project's development.

- D1 – Technical proposal: a technical document describing the intended work for the project. Due date: *12-02-2017*.
- D2 – Technical Report: a technical report describing all the work done in this final project: *Due date: 21-07-2017*.
- D3 – Game Application: an application that is playable and shows the result of this project. *Due date 25-07-2017*.
- D4 – Project Defense Video: a two minutes video showing a gameplay of the game developed. *Due date 25-07-2017*.
- D5 – Project Defense Presentation: a slide presentation that will support the final presentation in front of the tribunal. *Due date 25-07-2017*.

Table 5 - Deliverables Breakdown

Deliverables				
ID	Deliverable	Related Objective	Related Phases	Due Date
D1	Technical proposal	01	1	12-02-2017
D2	Technical Report	02, 05	1,4	21-07-2017
D3	Game Application	02, 03, 04	2,3,4	25-07-2017
D4	Project Defense Video	02, 05	1	25-07-2017
D5	Project Defense Presentation	02, 05	1	25-07-2017

Chapter 2

Game Design Document

This chapter details a vision for the game and describe the contents Music-Hero Party. The chapter has been divided into sub-sections for readability and to keep the different aspects of the game separate in order to identify them quicker in case of one of these have to be consulted.

2.1 Gameplay

The gameplay follows this structure:

1. **Select a song to play:** At the menu stage, the song has to be selected and will change dramatically the gameplay as it will count as the “level” analog compared to almost any other game.
2. **Hit the notes at the beat of the music:** The gameplay occurs right after the song is being selected. The main game screen is loaded and the music starts to play. As the music plays, notes approach from the horizon and sounds at the time they are at the same location of the activators, the player will have to hit the buttons of his/her controller in order to score points at the step of the music. Synchronicity between gameplay and music is key. Three buttons will be at disposal of the player as the music notes will be divided in three groups attending to their tone and to the button repetition.
3. **Game over condition:** The “game over” condition is failing too many notes, by fail counts striking the button whenever there was no note at the time or to let it pass without striking it. The win game condition is to finish the song with the maximum score possible.

2.2 Scenario Overview

The first thing to know about the scenario is that is a single point of view type game. A static camera will be looking at the three lanes from which the notes will be coming from the horizon. The lanes will cover the majority of the bottom of the screen and will narrow

as they extend towards the horizon as the perspective rules declare. The lanes will be separated only by narrow lighted cylinders like fluorescent tubes.

The three buttons that acts as note activators will be near the bottom of the screen and each of one centered at each lane. There has to be enough space between the buttons and the bottom of the screen to notice if the player has missed a note. Figure 1 illustrates how the scenario should look like.

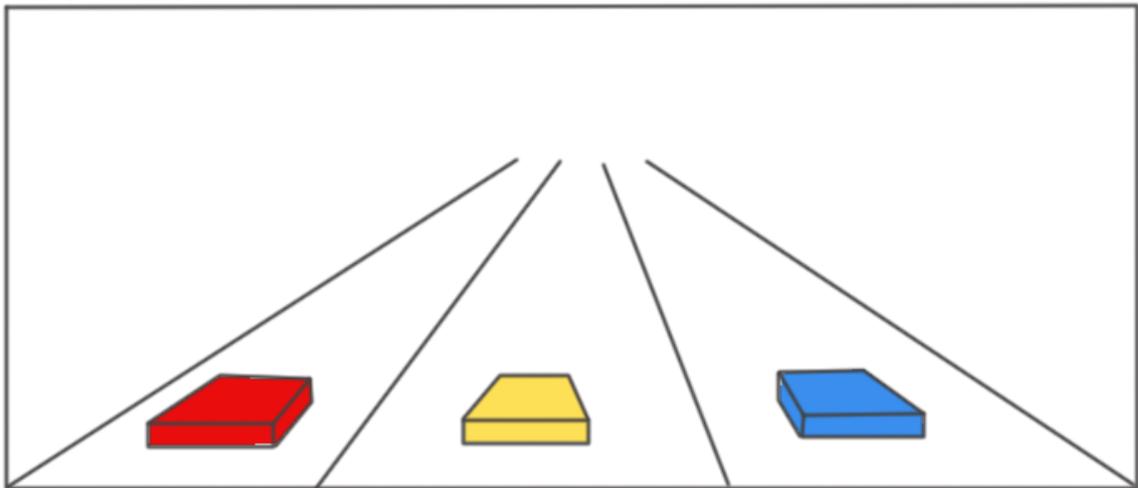


Figure 1 - Scenario

2.3 Player Controls

The player will have to possess an Android device in order to play the game as controls are through the app developed for this project. Once the connection is established and the game started the screen will display three colors and fit the entire screen size as shown in Figure 2. The colors must be identical to the scenario buttons as they represent the way to push them. Every time a color is pushed at the android device screen an

animation of pushing a button like will occur at the main game screen. This is illustrated in Figure 3.

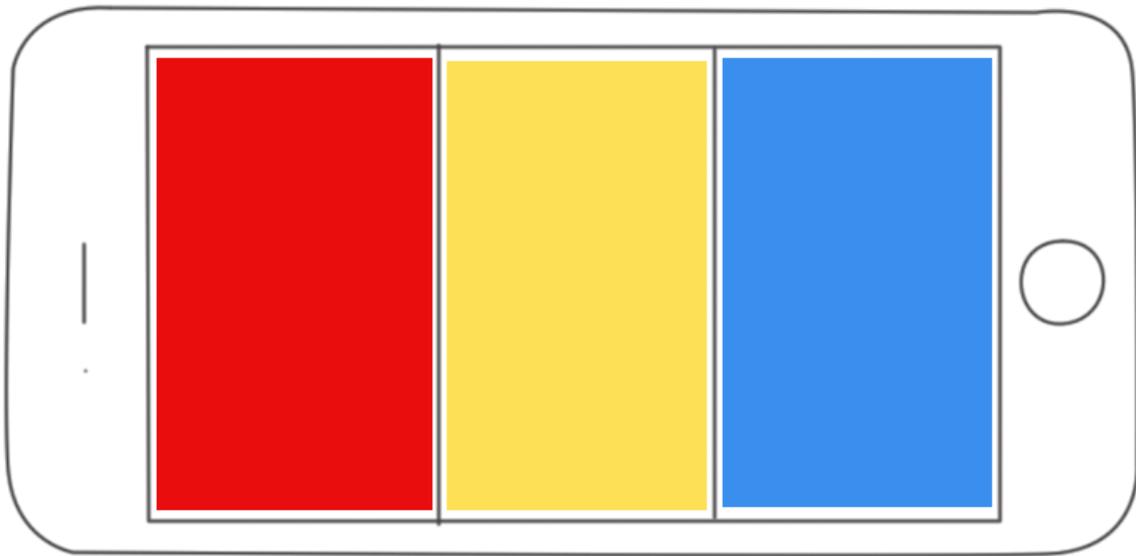


Figure 2 - Player controls

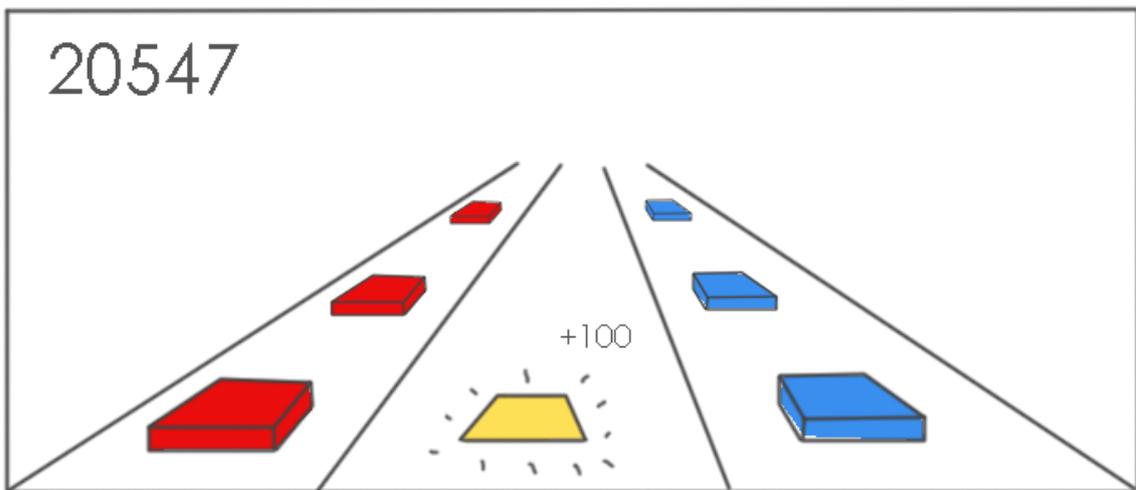


Figure 3 - Button pushed

2.4 Game Mechanics

The gameplay is well-known as the core mechanics will be the same of any *Guitar Hero* game. Each player has a number of lanes (three) with a button like activator at the nearer plane of the camera. The lane will stand from the button to the horizon. As the music relevant notes approaches in time, a particular form of representation (cubes in this game) will appear at the lane it belongs to. As the music plays, the cubes move towards the activators in straight line.

As the cube collides with the activator the song notes will take place. Besides, if the player pushes the button at the time the cube-note and the activator are colliding, the player earn points as if he were playing the note on a music instrument. If the cube-note does not get pushed, it will pass through the activator and will count as a failed note.

Whether the player push any button that is not colliding a cube-note, it will count as a fault or a try to glitch the game by smashing the button in order to score relentlessly and will be penalized. The penalty for failing or smashing repeatedly will be to low down the audience *Happiness Bar*, an element that take into account how many cube-notes have been pushed on time correctly and how many have not.

The *Happiness Bar* is composed of three colors representing the grade of happiness of the audience. The bar starts at the middle in yellow color and states that the notes scored while the bar keeps at the yellow level will be 100 points. As the player plays the game, the bar can reach up the green level or get down to the red level depending of his/her performance. At the red level, the cube-notes scores 50 points and at green level 150 points. Figure 4 illustrates the *Happines Bar* at the screen scheme.

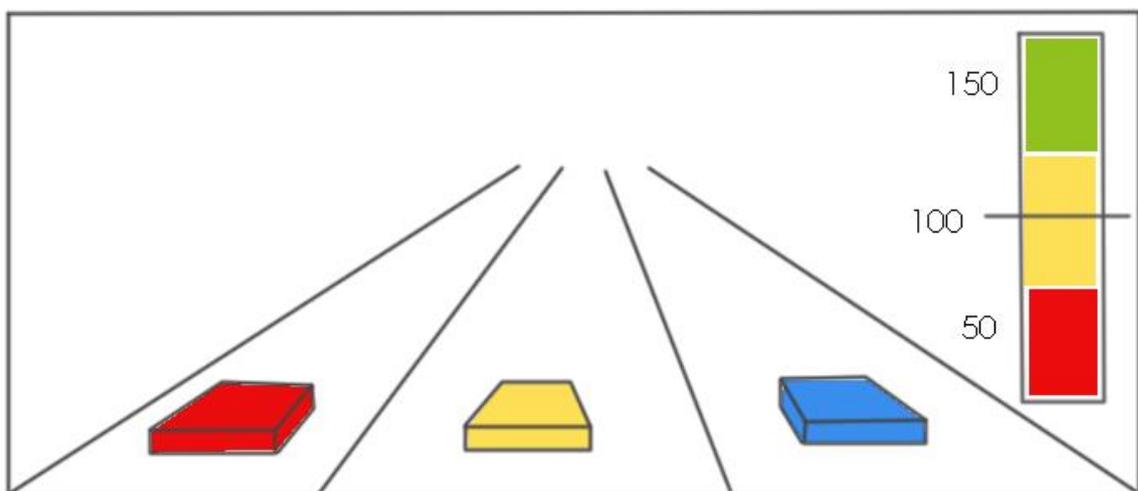


Figure 4 – Scenario with *Happines Bar*

A streak counter is displayed below the *Happiness Bar* and counts how many cube-notes have been played without failing. Every ten cube-notes scored without failing a multiplier indicator grows in one unit. This indicator is showed below the streak counter and returns to one if a single note is failed. A single note scored at the yellow level below the ten notes streak will score 100 points, if it has been scored while 10 and 20 notes were scored without missing, the note will score 200 points.

The player loses the game whether he/she fails too many notes and the *Happiness Bar* get to the bottom of the red level, if this happen, the music will stop and a “game over” sign will prompt taking the player back to the menu. The player “wins” the game by ending the song, at that point, the final score will prompt at the middle of the screen and stored to be beaten again by replaying.

2.5 Art and UI Design

At any time this project was intended to be an arty game in the usual way, it has been taking care of some minimum appearance in order to make the game playable without thinking it is unfinished. This project was carried out with a limited amount of time and tried to focus on technical aspects, so as a result, the art made for this game is the type of art you could find in a prototype, which it is. It respects minimalistic rules of prototyping and allows the future skinning which could lead to a polish marketable product.

The visual style of the game will take a minimalistic approach to the futuristic concepts of the 80s and 90s of the pop culture in which neons , hard edged polygons and synth music were protagonist. Bearing that in mind, the notes and buttons models are crude polygons (in this case rectangles) and the color palette has been chosen to be colorful and to shine in contrast of the dark hollow background. All the music comes from digital synthesis and resembles the music of that age.

The HUD (Heads Up Display) will have to provide the player with information about his/her performance during the gameplay. This information is; score actually achieved, the actual note hitting streak, the best streak done during the actual song, the multiplier value, the actual note value and how far/near is to reach the next level of note value and/or how far/near is the player to lose the game because of his/her poor performance.

For the score, as it expresses one of the most important information the upper left corner of the screen will be reserved for it to be displayed in large number format.

The next thing at the top of the list regarding importance is the happiness of the crowd or the performance meter bar. It will be displayed at the right part of the screen and it will have the form of a vertical three colored rectangle with an indicator of where the player is in form of a thin bar. The rectangle will be filled from top to bottom with green yellow and red. These three colors will express the performance and will state the actual note value, this will be also readable at the side of the bar.

Below the performance meter will be the rest of the valuable information in this order: note score multiplier, actual streak and longest streak. These will be written but using a smaller size of font than the score as they express less important and more circumstantial information.

2.6 Colors

The color identity of the game has been chosen to ease the gameplay and avoid confusion identifying which notes comes from the horizon. With that end colors from far distance areas of the spectrum and quickly recognizable by the player were chosen; yellow, red and blue for the notes. Color blindness is a matter to be aware of, in the optimistic scenario in which this game scales and grow to be released, a color blind option will be included with the release as part of the options.

The scenario has to be understood as a mere excuse for the game and could not take any protagonism, it was thought to be the stage for a music visualizer at the design stage but then again time was onto the project and took it away coming to be a dark grey to avoid stealing any attention.

The *Happiness Bar* or *Performance meter* was made using most recognizable colors of good/regular/bad because it could not again steal any attention from the player as he needs to keep focus on the stage and the note coming. The colors chosen were green/yellow/red correspondingly.

Chapter 3

Game Development

This chapter comprises the project development and its explanation. It covers almost every aspect of the game as other less important like UI or menu functions are ignored as they are found trivial or not highly interesting. The order followed at this chapter reflects the order followed at the development of the project.

3.1 General Overview

This game could be technically described as a piece of software that connect several devices via LAN, take a MIDI as an input chosen by the user, generates the playable content seen at the screen based on the MIDI, process the real-time input by the players devices through the network and show the output only by the main device screen. Figure 5 depicts the description recently made.

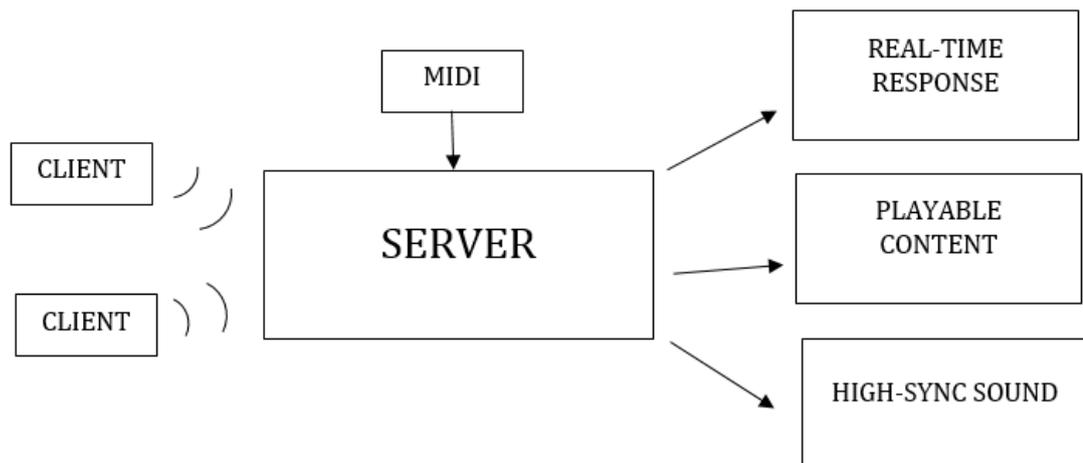


Figure 5 - System Overview

The technical challenges found during its development are described ahead. The most difficult features were not the LAN connection as there are plenty of information of it, it was the music information extraction needed to generate the levels, in other words, the MIDI

parsing. MIDI is a wide-spread data format and still very mysterious because is hard to find valuable, useful information.

3.2 Core game

Even though the most attractive aspect of the project might be the usage of various android devices as controllers to take the input, it ought to be demonstrated its usefulness by applying it onto something. The game developed was not initially designed to be played by network as it is an old game concept (first guitar hero was released on 2005), still this game technology is very versatile and many game design and playabilities could be adapted. The subsections that follows explain the game developed to make use of this technology.

3.2.1 MIDI

MIDI stands for Musical Instrument Digital Interface and it is a standard music technology protocol and allows to connect and communicate a wide variety of electronic musical instruments. MIDI is very lightweight as it was created in 1983 when there were very limited hardware capacities, but it is still, very powerful and reliable.

The most important thing to notice to carry out our project is that MIDI format carries data and not wave information. This is key to our project because that means that we are able to read it and extract useful information from it. The objective we are pursuing is to be able to create a song and to be able to play it without having to create the level for playing it. This process is going to be delegate to the computer software that takes the song in MIDI format and generates the level with its data. This automatization of level content creation has not been found in my experience in any musical game I have ever known or that I have been able to find.

3.2.1.1 Understanding MIDI

MIDI format is compound by a header and a body chunk. Each chunk is prefixed with an 8 byte header: 4 byte ID string used to identify the type of chunk followed by a 4 byte size which defines the chunk's length as number of bytes following this chunk's header.

- **Header Chunk:** The header chunk contains information about the entire song including MIDI format type, number of tracks and timing division. There is only one

header chunk per standard MIDI file and it always comes first. It is structured as the table explains:

Table 6 - Header chunk

Offset	Length	Type	Description	Value
0x00	4	char[4]	Chunk ID	"MThd" (0x4D546864)
0x04	4	dword	Chunk size	6 (0x00000006)
0x08	2	word	Format type	0 - 2
0x10	2	word	Number of tracks	1 - 65,535
0x12	2	word	Time division	-

The chunk ID is always "MThd" and the size is always 6 because the header chunk always contains the same 3 word values.

The first word describes the MIDI format type. It can be a value of 0, 1 or 2. We are going to focus on type 0 for this project which is a single layer MIDI track file.

The second word simply defines the number of track chunks that follow this header chunk. A type 0 MIDI file may only contain a value of 1, because they can only contain one track. Type 1 and 2 MIDI files may contain up to 65,536 tracks.

The third and final word in the MIDI header chunk is a bit more complicated than the first two. It contains the time division used to decode the track event delta times into "real" time. This value is representing either ticks per beat or frames per second. If the top bit of the word is 0, the following 15 bits describe the time division in ticks per beat. Otherwise the following 15 bits describe the time division in frames per second. We are not going to go deeper and stick to the usage of time division in ticks per beat.

- **Track Chunk:** Track chunks contain all of the information for an individual track including, track name and music events. Here is an overview of a track chunk's organization.

Table 7 - Track Chunk

Offset	Length	Type	Description	Value
0x00	4	char[4]	Chunk ID	"MTrk" (0x4D54726B)
0x04	4	dword	Chunk size	(see following text)
0x08	track event data (see following text)			

The chunk ID is always "MTrk" and the size varies depending on the number of bytes used for all of the events contained in the track.

The track event data contains a stream of MIDI events that define information about the sequence and how it is played. MIDI carries event messages

that specify notation, pitch and velocity, control signals for parameters such as volume, and clock signals that set and synchronize tempo between multiple devices. Track events are used to describe all of the musical content of a MIDI file, from tempo changes to sequence and track titles to individual music events. Each event includes a delta time, event type and usually some event type specific data. In addition every one of these messages have a special code, however, we are only interested in the note messages and the delta time messages in order to retrieve the information needed for our gameplay.

The *delta time event* is defined by a variable-length value. It determines when an event should be played relative to the track's last event. A delta time of 0 means that it should play simultaneously with the last event. The most important thing to remember about delta times is that they are relative values, not absolute times. The actual time they represent is determined by a couple factors. The time division (defined in the MIDI header chunk) and the tempo (defined with a track event). If no tempo is define, 120 beats per minute is assumed. Below is showed how delta times are structured inside a generic MIDI channel event.

Table 8 - MIDI Channel Event Structure

Delta Time	Event Type Value	MIDI Channel	Parameter 1	Parameter 2
variable-length	4 bits	4 bits	1 byte	1 byte

Note On Event is used to signal when a MIDI key is pressed. This type of event has two parameters. The note number that specifies which of the 128 MIDI keys is being played and the velocity determines how fast/hard the key is pressed. The note number is normally used to specify the instruments musical pitch and the velocity is usually used to specify the instruments playback volume and intensity.

Table 9 - Note On Event Structure

Note On	MIDI Channel	Note Number	Velocity
9 (0x9)	0-15	0-127	0-127

The *Note Off Event* is used to signal when a MIDI key is released. These events have two parameters identical to a Note On event. The note number specifies which of the 128 MIDI keys is being played and the velocity determines how fast/hard the

key was released. The note number is normally used to specify which previously pressed key is being released and the velocity is usually ignored. An important thing to notice is that Note On events can be called with a velocity of zero, then it will be treated as a Note Off event.

Table 10 - Note Off Event Structure

Note Off	MIDI Channel	Note Number	Velocity
8 (0x8)	0-15	0-127	0-127
9 (0x9)	0-15	0-127	0

Before we continue, there is a last important feature that has been mentioned and not explained. The note numbers go from 0 to 127, whether we want to recognize which note we are playing, the MIDI specification stands that middle C is 60 and the rest are relative. This leads to some confusion:

There is a discrepancy that occurs between various models of MIDI devices and software programs, and that concerns the octave numbers for note names. If your MIDI software/device considers octave 0 as being the lowest octave of the MIDI note range, then middle C's note name is C5. The lowest note name is then C0 (note number 0), and the highest possible note name is G10 (note number 127).

Some software/devices instead consider the third octave of the MIDI note range (2 octaves below middle C) as octave 0. In that case, the first 2 octaves are referred to as -2 and -1. So, middle C's note name is C3, the lowest note name is C-2, and the highest note name is G8. (11)

With the software we are using, the note table conversion come up like this:

Table 11 - Note Conversion Table

Octave	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
-2	0	1	2	3	4	5	6	7	8	9	10	11
-1	12	13	14	15	16	17	18	19	20	21	22	23
0	24	25	26	27	28	29	30	31	32	33	34	35
1	36	37	38	39	40	41	42	43	44	45	46	47
2	48	49	50	51	52	53	54	55	56	57	58	59
3	60	61	62	63	64	65	66	67	68	69	70	71
4	72	73	74	75	76	77	78	79	80	81	82	83
5	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107
7	108	109	110	111	112	113	114	115	116	117	118	119
8	120	121	122	123	124	125	126	127				

3.2.1.2 Reading MIDI

The process of getting understandable MIDI comes from a plain C# program that was done by the student developer prior to the game. By using the I/O library, a simple console program was made. The code specifies where the MIDI file is, and then read it in byte form, and write the content into a text file. This was made just to get the output and see if it was possible to get anything reasonably understandable. The outcome of this study was positive and then a newer version of this program was written to be inside the game.

The game itself does something very similar, it has the MIDI files at a folder and read them to get a list of the MIDIs available to the player to choose from. Once the player select the song, the MIDI is read from the folder and parsed from binary to hexadecimal just by a C# I/O command.

The hexadecimal code is not written to a text file like in the study, it is returned as a string variable and kept in memory for the parsing and later usage.

3.2.1.3 Parsing MIDI

Once the MIDI was translated into hexadecimal code, it was just a matter of traverse the string byte by byte and look neatly for all the event messages that we were using for the game. To our purpose, we needed to find the tempo, and all the *Note On* and *Note Off* event messages.

The tempo was inside a system event message, those messages starts with FF byte and the tempo one is coded with 51 after. So, it was just a matter of look for an FF and a 51 after it to retrieve the tempo in beats per minute.

The parsing of the Note On/Off events was done by searching for event messages that started by 90 and 80 respectively. Once they were found, the traverse was made backwards until the delta time message preceding that note event was found. Then the delta time was converted from hexadecimal to decimal and then aggregated with all the preceding delta times in order to get the exact time of the note event that was found. Similar parsing was made with the note event, the hexadecimal content inside the message was translated into decimal and then parsed using the table explained at 3.2.1.1 . These two pieces of information were then introduced into a data collection, which was a List of List of string and therefore a parsed song was extracted. The resultant List contains every note and when they occur.

3.2.2 Setting up the scenario

Aside from the obvious procedures of how to build the scenario placing objects in the 3D space of Unity, it is important to explain how the cube notes are placed differently for each song. Usually in this type of games the notes are placed manually by a person in order to make them to seem synchronized with the music. This is a tedious labour but indispensable if the music is only in a wave format, having the music in data format allow us to automatize this work.

To place every note correctly is not a trivial problem, and this is why we wanted the MIDI parsing. We now have access to the time each note is played by summing up every delta time between notes till the last note, now, we are going to place all of them sequentially. Once having the time for each note, we can setup the speed they are going to travel until they hit the position of the activator, and, having the speed and the time lead us to know by basic physic knowledge to the distance every single note is going to need to be placed as the product of time and speed is distance.

Now we have the distance, but, distance to a point is an infinite set of points in 3D space because they conform a sphere. We just want the notes to come from the horizon of our camera and towards the activator sliding through the imaginary floor we have set up at the stage of the game. To achieve this we now resort on mathematics, we can calculate with a couple of points the slope of a line and having as a result a direction vector.

Having a distance and a direction vector leads us to the point in the 3D space every note has to be. This is the way every note is placed just before the start of the song and the game. Figure 6 shows a representation of how the game would look like just before a game starts.

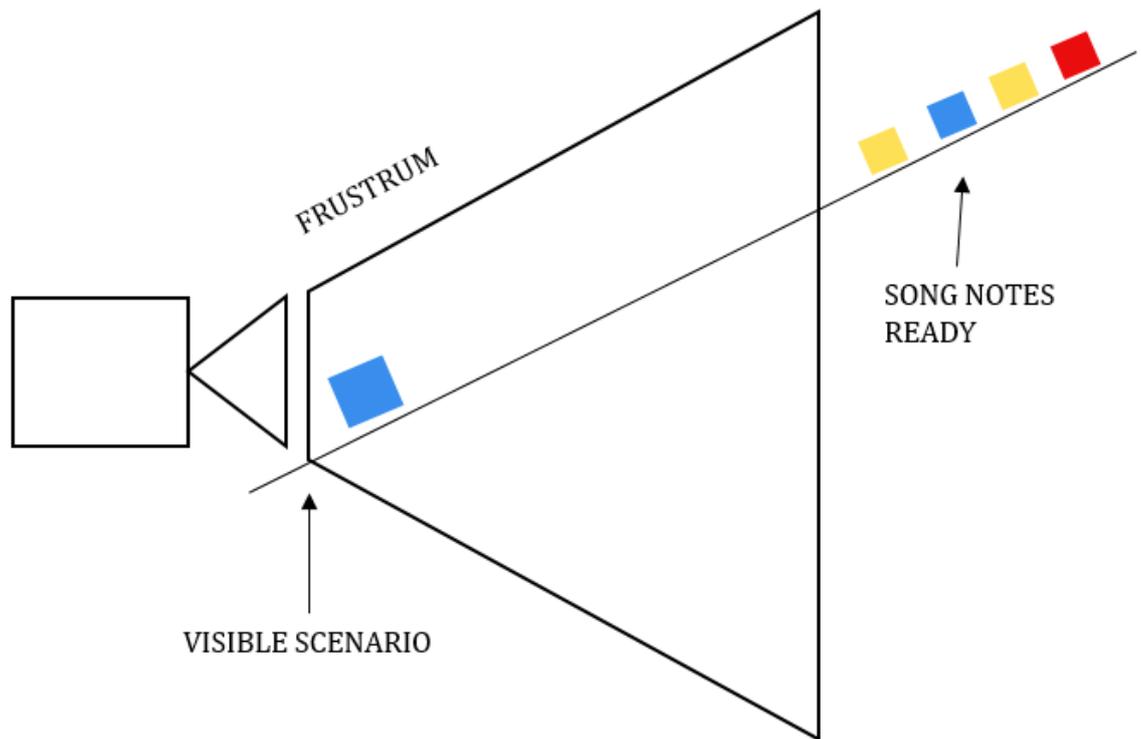


Figure 6 – Scenario (Right side view)

3.2.3 Gameplay

Once everything is ready, the game starts and the notes start moving towards the activator. Every single note carries a collider of its rectangle shape. If the player press the button whenever the activator (which also has a collider) and a note is colliding, the note is removed from the screen and the score grows the amount of points the *Happiness Bar* stipulates multiplied by the *score multiplier*. The score multiplier grows every ten consecutive notes had been scored, so, by the time ten notes are scored, the multiplier grows from one to two and every note scored pass to be multiplied by two. If the player reach to twenty notes scored without failing any note, the multiplier grows to three. The multiplier falls down to one at any time the player lose or fail a single note.

This happens to be very easy in comparison with the work done to set everything up to the gameplay to happen, however, as my game making learning grow, it seems that it is every time like this.

3.3 The Smartphone as a Controller

This section explains how the smartphone side of the application was built and therefore made to work as a controller. The first thing to understand is what it is, its definition in Wikipedia explain it like this:

A smartphone is a mobile personal computer with a mobile operating system with features useful for mobile or handheld use. (12)

Note that it is defined as a personal computer and then we can assume and treat it as a computer that we are connecting to the local network and communicating it with our machine in a typical client-server architecture. Unlike any other common client-server architecture, this communication is not going to be reciprocal, we need the client to send messages to the server, but we do not need the server to update the status for every client. Bearing that in mid, lets address how this was made for this project.

3.3.1 Unity networking: HLAPI vs LLAPI

Before we dig into the networking, we had to knew how multiplayer support was implemented in Unity, and what were the resources that the game engine had in order to manage the multiplayer games.

Surprisingly, it was not long ago since Unity developers started to worry about the multiplayer features its engine was offering. Until the 5.1 version of Unity offered a very crude multiplayer support known as the Low Level API (LLAPI) which was not very pleasant as the programmer had to take care of managing every aspect of the connection through code. With the arrival of 5.1 Unity introduced Unet, known as well as High Level API (HLAPI) which handles the most common scenarios for videogames and ease its making.

Retrieved from its manual (13) comes the best explanation: *The High Level API (HLAPI) is a system for building multiplayer capabilities for Unity games. It is built on top of the lower level transport real-time communication layer, and handles many of the common tasks that are required for multiplayer games. The HLAPI is a new set of networking commands built into Unity, within a new namespace: UnityEngine.Networking. It is focused on ease of use and iterative development and provides services useful for multiplayer games, such as:*

- *Message handlers*
- *General purpose high performance serialization*
- *Distributed object management*

- *State synchronization*
- *Network classes: Server, Client, Connection, etc*

The HLAPI is built from a series of layers that add functionality:

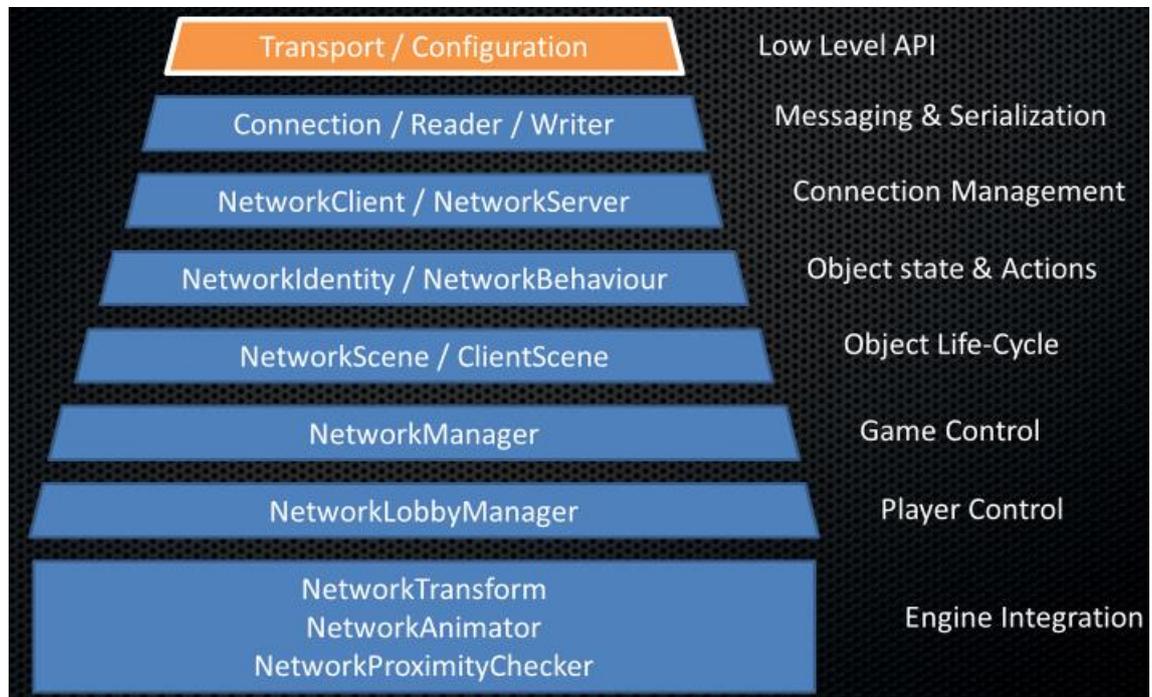


Ilustración 1 - Unity HLAPI Layers

It is important to notice for avoiding confusion, HLAPI and LLAPI are not mutually exclusive and could be used together. There are many people posting on the internet about this and it is hard to reach to the truth as many of them stands that this is not possible.

3.3.2 Connecting devices

The first attempt of this project was compound of two different Unity projects that were futilely tried to connect. Turned out that Unity has its proper way of sending and receive packages over the network and have coded inside of it in some sort of way to which application it belongs to and wo which version. In other words, there is no easy way to communicate two different projects. Solution; to make a single project making use of the feature called *Platform Dependent Compilation* (14) and separate the scenes that are going to be loaded depending on the machine the game application is running on.

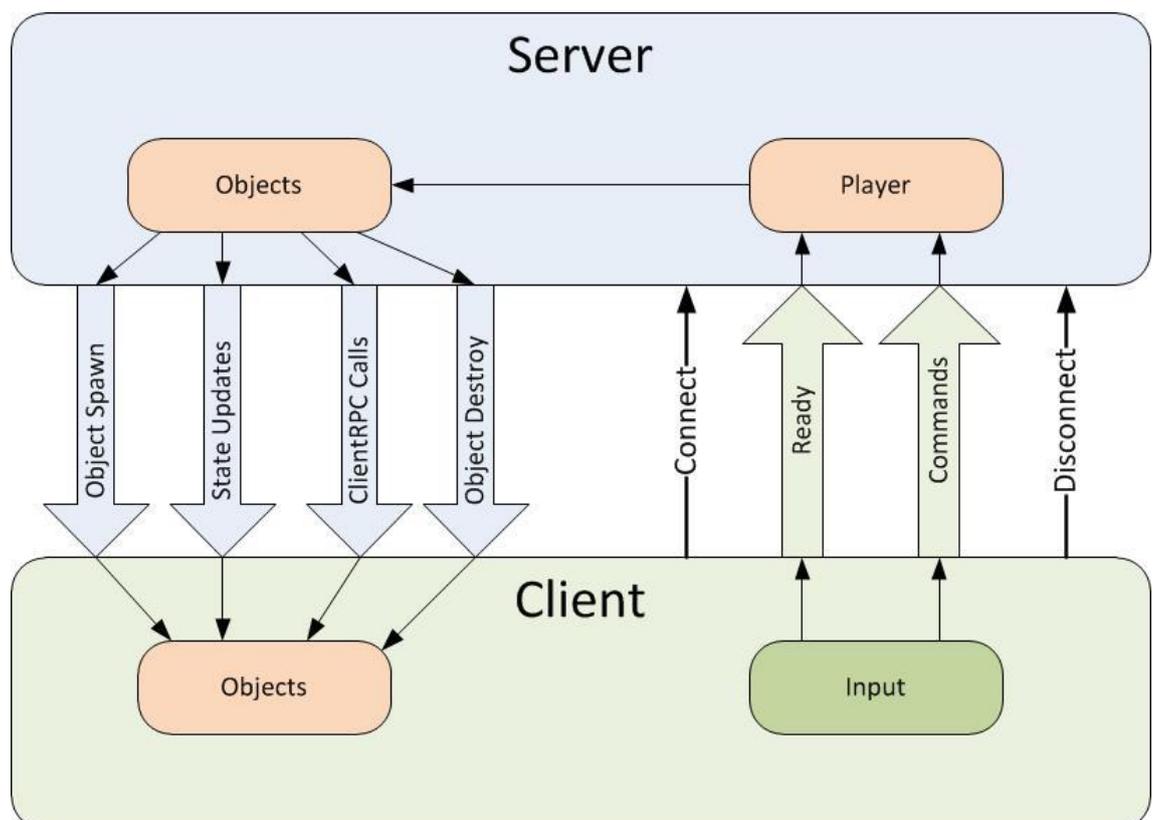
Once this was figured out, “another rock was found in the way”, even though everything in the code seemed to be correct, there was no way of getting the devices connected through the LAN. At some point it came to mind that we could try the same code

with a couple of PC and then we would at least clarify that it has something to do with the smartphone side of the code. The testing revealed that there was not a problem with the smartphone code, the machines were still unable to connect. After several hours, a light turned on and the router ports were checked, every port was obviously closed by the manufacturer, after open the port that we were trying to use to communicate the devices, everything worked as a charm.

3.3.3 Commands, RPCs and raw network messages

The Unity manual reference explains the way of communicating the server and the client very concisely and even illustrate it. (15)

The network system has ways to perform actions across the network. These type of actions are sometimes called Remote Procedure Calls. There are two types of RPCs in the network system, Commands - which are called from the client and run on the server; and ClientRpc calls - which are called on the server and run on clients.



Seems very easy to work out that the feature that fits best to our needs is the usage of *Commands* that are sent from the client to the server. The RPC are used with the authoritative architecture that feels the need to update every aspect of the game to all the clients of the game, since this is not our case, we are not using it.

Another way of getting the job done that Unity offers is to directly send raw messages. Raw messages need to be serialized and deserialized at every time they are sent, this could lead to lower performance and therefore was not used.

3.4 Game Over screen

The game over screen was implemented by using a transparent canvas over the gameplay scene. It is a very popular way of doing it since it was taught at a Unite training day. The content of the screen is a simple message that vary in terms of finishing or losing the song, the score achieved and also holds two buttons; to replay the song in order to get a better score or get back to the menu.

Chapter 4

Testing & Evaluation

This chapter treats the testing made to verify the correct functioning of the application and the overall performance of the project regarding its viability and usability.

4.1 Testing

Even though games are a soft time simulations (16) the time reaction is key in this project and therefore testing took an important role in order to get rid of bugs, and to optimize some aspects. Testing was made through the processes of verification and validation.

4.1.1 Verification

During the phase of development, each new feature is tested during the implementation itself, still, that does fulfill one out of the four testing phases. Testing comes in form of full development builds that are named development phase, alpha, beta and release. Each phase is normally performed by different people, developers themselves, members of the team, beta-testers and users respectively. As this project was carried out with limited amount of time and money, all the testing was done by the developer.

4.1.2 Validation

Validation is a process in which the software is checked to see if its objectives were accomplished. Usually is a process that does not involve the development team because they are biased with their game. Validation has been made at this project by friends, which are not ideal but were the only people reachable at the time the project reached this phase.

4.2 Evaluation

Evaluation is the process that goes beyond the bug detection and rely on the performance of the game. The key factor of this project is the speed of the network messaging system. Evaluation is taking still place at the moment this report is being made, so result may differ from the actual state of the game. Numbers that endorse this evaluation could not be provided as the time left for this aspect of the project shrunk as the code development grew.

It has been noticed that there are some time slots of the day that the performance through the network experience delays. It is suspected that WiFi band noise made by the neighbourhood could have something to do with it.

It is important to remark that four different songs were tested once the MIDI parsing was done to see if the outcome was synchronized with the gameplay. All the results were more than satisfying.

4.2.1 Hardware

Three pieces of hardware are needed in order to play this game, a windows operative system computer, an Android device and a wireless router. The hardware used to develop and test this project is described here:

- **PC: Medion Erazer x7829**
 - Processor: Intel Core i7-4710MQ (2,5 Ghz. / TDP 47W. / 6 Mb. cache L3).
 - Memory: 16 Gb. DDR3L a 1600.
 - HDD: 1 Tb. 5400 rpm.
 - OS: Windows 8.1 (64 bits).
 - Red: Gigabit Ethernet, Wifi-N y Bluetooth (4.0).
 - GPU: nVidia Geforce GTX 870M (3 Gb. GDDR5).
 - Screen: 17,3" Full-HD (1920x1080) LED.
- **Android device: Xiaomi Mi 4**
 - Type: IPS LCD capacitive touchscreen, 16M colors
 - Size: 5.0 inches (~72.3% screen-to-body ratio)
 - Resolution: 1080 x 1920 pixels (~441 ppi pixel density)
 - Multitouch: Yes
 - OS: Android 4.4.3 (KitKat)
 - Chipset: Qualcomm MSM8974AC Snapdragon 801

- CPU: Quad-core 2.5 GHz Krait 400
- GPU: Adreno 330
- WLAN: Wi-Fi 802.11 a/b/g/n/ac, dual-band
- **Router: Comtrend AR-5387un**
 - System-On-Chip: Broadcom BCM6328
 - CPU/Speed: BMIPS4350 / 320 MHz
 - RAM size: 64 MiB
 - Wireless: BCM43225 (14e4:a8d8) 802.11b/g/n
 - Antenna: 2x, non detachable

Chapter 5

Results

A series of deliverables were presented at chapter 1 and could be considered as the main outputs of this project. This chapter outline the outcome of the last five months in which the project was carried out.

5.1 Technical proposal

It is the first chapter of this dissertation and has suffered severe changes since its first version made at February. Those changes came from the need of taking the project to a professional level and as a result of the first research made to carry out the project. It was bigger than we first thought and had to get rid of some aspects and grow at others.

5.2 Game design document

Even though this is a technical project and the game was not intended to be protagonist, it felt necessary to made a game that demonstrate the possibilities of this capability not enough exploited. And where is a game it has to be a game design document. It can be found at chapter 2 of this dissertation.

5.3 Technical Report

The complete documentation carried along this project and a description of the work done is found at this document that you are currently reading.

5.4 Game application

The main objective of this project was to test whether was possible to connect and use a smartphone as a controller and therefore was needed to develop an application to test it. The PC application can be found at <https://drive.google.com/open?id=0ByZWirviWINOY2lEQmVDRk9lOGc> , and work as a

server. The Android client Apk is found at <https://drive.google.com/open?id=0ByZWirviWINORzc4QXRaa2RUZ0k> and work as a client. These two are the result of five month of work but hopefully could lead to a new line of products.

5.5 Project Defense Video

A 2 min video demonstration has been made and can be found at <https://drive.google.com/open?id=0ByZWirviWINOa2oxWG4zbG5ZdWs>.

5.6 Project Defense Presentation

The defense of this project will be carried out with the support of the presentation found at <https://drive.google.com/open?id=0ByZWirviWINOdjBYRFFILUpEb0U>.

Chapter 6

Project deviations

As any software project, video games are not exempt of deviations. Planning is very valuable, however, accuracy at time prediction is only achieved by several years of project management and still not a single developer will bet on its own prediction. Challenges appear as the project grows and overcoming or surrounding them is not always easy.

At this chapter we are going to address the deviations of Music Party Hero. This project was not even going to be called that way but as deviations occurs, name and genre of the game underlying within the project suffered changes.

6.1 Game concept

At the early stage of the project, several ideas were being given thought, finally one was picked up based on the pretended simplicity. A race between a random number of characters was going to happen at the main screen while the players would hold at their handheld devices two buttons to control their character. The game concept started to grow very fast and challenges like control a gun scope mapping the surface of an area of the handheld device became overwhelming.

At the last stage of the student mandatory period of internship at a company the idea of creating a music video game was proposed to us. After some research, the concept seemed to fit the time constraints, doable and scalable. Getting to a first prototype was delayed by the synchronicity need between sound and game. This lead to investing time learning about parsing MIDI file format and then to increased time spent in the game that was not initially intended.

As a result of spending a lot of the development time at the core game, some other aspects of the project have been diminished. A very few time has been given to details like a more appealing menu or more attractive UI design, the background of the game scenario was intended to be a real time sound visualizer, and above all the details, testing and evaluation with several devices were left out.

6.2 Project schedule

Section 1.7 presented a plan and a schedule for the project estimating the duration of each task. At this point we are now able to present the project schedule deviations.

Table 12 – Actual Documentation Breakdown

Documentation				
ID	Task	Period	Total Days	Total Hours
TP	Technical proposal	30-01-2017 to 9-02-2017	10	10
TR	Technical Report	19-06-2017 to 20-07-2017	31	50
PDV	Project Defense Video	-	-	-
PDP	Project Defense Presentation	9-06-2017 to 12-06-2017	3	3
Total				63

Table 13 – Actual Development Breakdown

Game Development				
ID	Module	Period	Total Days	Total Hours
MP	MIDI Parsing	19-04-2017 to 28-04-2017	9	72
GM	Game mechanics	29-04-2017 to 12-05-2017	13	60
PCAC	PC-Android Communication	30-05-2017 to 06-07-2017	36	70
AD	Android Development	10-06-2017 to 10-07-2017	7	20
Total				202

Table 14 – Actual Game Design Breakdown

Game Design				
ID	Task	Period	Total Days	Total Hours
GDD	GDD production	19-06-2017 to 02-07-2017	13	20
Total				20

Table 15 – Actual Evaluation Breakdown

Game Evaluation				
ID	Task	Period	Total Days	Total Hours
DT	Development Testing	19-04-2017 to 21-07-2017	70	12

AE	Application Evaluation	18-06-2017 to 21-07-2017	3	6
Total				18

Chapter 7

Conclusions

This document has presented a party multiplayer video game which is playable through handheld devices. The project included design, development, testing and proper technical documentation.

There is a gap in the market in my opinion that this project is trying to fill, or at least, start walking in the correct direction towards it. It is a very ambitious goal but every way has its first steps. This project was intended to be a very simple game with network capabilities, however I learnt that there is not such a thing as a *very simple game* when it comes at the time of making it from scratch. This is the reason of why MIDI happened to become very important in this project. It may seem that has nothing to do with the initial intention of this project but at some point, it became important just because I ended up learning a lot of new things from it and felt very proud when finally got it to work. I am also very proud of the network part but it did not feel the same way just because is a lot of information about networking on the internet and almost anyone could reproduce that work, which I do not think the same about the MIDI parsing.

The five months the project has been occurring has been a tough journey. During this time the project has been on and off because of the intermission of personal and professional happenings that lead to changes and delays in its realization. It is hard to admit, but it is the only way of learning, that I am my worst enemy in the matter of self-discipline and time organization. Still, I think that very valuable outcome has been learnt with this project, I have had time to practice and get better skills at the usage of the Unity environment, I have had the time to get better at coding, I have had the opportunity of creating a multiplayer game which did not have at the degree, I could experiment and learn a lot about sound wave treatment and what MIDI is and how a file format is made from the inside and some other things that are less important to enumerate but still valuable like music composition techniques, the usage of I/O libraries, how difficult work prediction is and many more. To sum up, I still think that the positive outcome has been very positive and feel very proud of my work.

7.1 Future work

I feel thrilled about the chance of keep working on games on a big screen controlled by handheld devices. I have already several ideas of the next games that I want to make with this technology and I am even thinking of start a studio around this idea.

I feel also that sound has a lot still to say in the video game industry and probably the usage of MIDI is going to be a big step. To synchronize what is happening on the screen and what you are hearing is very difficult with any other sound analysis technique and still those games that tried it became popular even it was poorly done.

References

1. **Gizmodo.** [Online] [Cited: 07 20, 2017.] <http://www.gizmodo.co.uk/2016/07/the-best-ways-to-play-pc-games-in-your-living-room/>.
2. **Valve corporation.** http://store.steampowered.com/app/353380/Steam_Link/. [Online] 7 20, 2017.
3. **Kuchera, Ben.** Polygon. *Gaming has left the LAN party behind.* [Online] Enero 29, 2015. [Cited: 6 9, 2017.] <https://www.polygon.com/2015/1/29/7944755/lan-party-gaming-call-of-duty>.
4. **Zeiger-Guerra, Marc.** Gamasutra. *What is the Future of LAN Gaming?* [Online] [Cited: 6 10, 2017.] http://www.gamasutra.com/blogs/MarcZeigerGuerra/20110115/88774/What_is_the_Future_of_LAN_Gaming.php.
5. **IDC.** *smartphone-market-share.* [Online] [Cited: 06 25, 2017.] <http://www.idc.com/promo/smartphone-market-share/os>.
6. **Cleron, Mike.** android-developers. *android-announces-support-for-kotlin.* [Online] [Cited: 5 17, 2017.] <https://android-developers.googleblog.com/2017/05/android-announces-support-for-kotlin.html>.
7. **Wikipedia Foundation.** Wikipedia. *Tennis_for_Two.* [Online] [Cited: 5 31, 2017.] https://en.wikipedia.org/wiki/Tennis_for_Two.
8. **Mobygames.** Mobygames. *Gauntlet.* [Online] 6 12, 2017. <http://www.mobygames.com/game/gauntlet>.
9. **Wikipedia Foundation.** Wikipedia. *Music_video_game.* [Online] [Cited: 6 14, 2017.] https://en.wikipedia.org/wiki/Music_video_game.
10. —. Wikipedia. *Unity_(game_engine).* [Online] [Cited: 6 28, 2017.] [https://en.wikipedia.org/wiki/Unity_\(game_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine)).
11. **electronics.dit.ie.** [Online] [Cited: 6 18, 2017.] http://www.electronics.dit.ie/staff/tscarff/Music_technology/midi/midi_note_numbers_for_octaves.htm.

12. **Wikipedia Foundation.** [Online] [Cited: 6 18, 2017.]
<https://en.wikipedia.org/wiki/Smartphone>.
13. **Unity Technologies.** [Online] [Cited: 7 7, 2017.]
<https://docs.unity3d.com/Manual/UNetUsingHLAPI.html>.
14. —. [Online] [Cited: 6 29, 2017.]
<https://docs.unity3d.com/Manual/PlatformDependentCompilation.html>.
15. —. *UNetActions.* [Online] [Cited: 07 5, 2017.]
<https://docs.unity3d.com/Manual/UNetActions.html>.
16. **Gregory, Jason.** *Game Engine Architecture.* 2014.