Andreas Leibetseder

Gamification: Sneaking Fitness Game Mechanics into AAA Games

Masterarbeit

zur Erlangung des akademischen Grades
Diplom-Ingenieur

Studium Masterstudium Informatik

ALPEN-ADRIA UNIVERSITÄT KLAGENFURT
Fakultät für Technische Wissenschaften (TEWI)

Begutachter: Assoc. Prof. Dipl.-Ing. Dr. Mathias Lux
Institut für Informationstechnologie (ITEC)

March 2016

Eidesstattliche Erklärung – Affidavit

Ich versichere an Eides statt, dass ich

- die eingereichte wissenschaftliche Arbeit selbstständig verfasst und andere als die angegebenen Hilfsmittel nicht benutzt habe,
- die während des Arbeitsvorganges von dritter Seite erfahrene Unterstützung, einschließlich signifikanter Betreuungshinweise, vollständig offengelegt habe,
- die Inhalte, die ich aus Werken Dritter oder eigenen Werken wortwörtlich oder sinngemäß übernommen habe, in geeigneter Form gekennzeichnet und den Ursprung der Information durch möglichst exakte Quellenangaben (z.B. in Fußnoten) ersichtlich gemacht habe,
- die Arbeit bisher weder im Inland noch im Ausland einer Prüfungsbehörde vorgelegt habe und
- zur Plagiatskontrolle eine digitale Version der Arbeit eingereicht habe, die mit der gedruckten Version übereinstimmt.

Ich bin mir bewusst, dass eine tatsachenwidrige Erklärung rechtliche Folgen haben wird.

I hereby declare in lieu of an oath that

- the submitted academic paper is entirely my own work and that no auxiliary materials have been used other than those indicated,
- I have fully disclosed all assistance received from third parties during the process of writing the paper, including any significant advice from supervisors,
- any contents taken from the works of third parties or my own works that have been included either literally or in spirit have been appropriately marked and the respective source of the information has been clearly identified with precise bibliographical references (e.g. in footnotes),
- to date, I have not submitted this paper to an examining authority either in Austria or abroad and that
- the digital version of the paper submitted for the purpose of plagiarism assessment is fully consistent with the printed version.

I am aware that a declaration contrary to the facts will have legal consequences.

(Andreas Leibetseder) (Klagenfurt, March 21 2016)

"Every man and woman should play the noblest games and be of another mind from what they are at present."

Plato

ALPEN-ADRIA UNIVERSITÄT KLAGENFURT

Abstract

Fakultät für Technische Wissenschaften (TEWI)
Institut für Informationstechnologie (ITEC)

Master of Science

by Andreas Leibetseder

"*Life is a journey, not a destination*" – a fairly old metaphor reminding humankind to evolve and enjoy its existence rather than getting lost in boring, repetitive routines. Yet, many tiresome tasks we ought to undertake can not be avoided easily. Wouldn't it be great if those activities instead were things to look forward to? Gamification is a recently trending attempt to build upon this notion, rendering mundane endeavors fun and engaging by appealing to our ludic drive. Its techniques can be utilized for enhancing almost any situation: standing at a bus stop, recycling plastic bottles or tedious physical exercise, the lack of which being a major shortcoming in many peoples lives.

Recently, fitness apps and games have tried to improve upon people's unhealthy lifestyle, but often are poorly designed, include expensive additional equipment and get boring quickly, thus, failing to maintain their users' long-term engagement. In order to advance in that area, this thesis takes a novel approach by investigating the user effects of integrating fitness-related activities into a big budget (AAA) as well as a casual game, creating two exergames. For this cause two studies were conducted evaluating experience and feedback of altogether 67 voluntary participants. Results indicate strong tendencies of players preferring the newly introduced AAA approach over the casual fitness game. Surprisingly only 5 persons state that they play exergames, but the majority of them would consider playing games similar to the ones presented at home. Unexpectedly, no significant evidence for a relation between people's workout habits and their conveyed exergaming preferences could be identified.

Acknowledgements

First and foremost, I would like to thank my project advisor Mathias Lux for taking me on, always making time for discussions and supporting me in every way. Also it is particularly important to mention Olvier Vitouch, who came up with the idea for this thesis' studies that Mathias and I could subsequently elaborate on. For their continuous efforts helping me to complete all user studies, Martina Steinbacher and Rudolf Messner have been invaluable colleagues of mine at the ITEC institute. Furthermore I would also like to thank several other colleagues for providing me with very helpful suggestions, advice and pre-studies feedback: Florian Bacher, Christian Kreuzberger, Stefan Petscharnig, Daniel Posch, Benjamin Rainer and Sebastian Theuermann. Finally, a collective 'thank you' goes out to my many student colleagues, who likewise supported me during my thesis but as well throughout my studies at the Alpen-Adria university of Klagenfurt.

Contents

List of Figures

List of Tables

List of Program Code

Abbreviations

AAA	Video Game Classification term - pronounced 'triple **A**'
AAU	**A**lpen-**A**dria **U**niversity
ACSM	**A**merican **C**ollege of **S**ports **M**edicine
AI	**A**rtificial **I**ntelligence
ANSI	**A**merican **N**ational **S**tandards **I**nstitute
AP	**A**rmor **P**oints
API	**A**pplication **P**rogramming **I**nterface
AR	**A**ugmented **R**eality
ASCII	**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
BFG	**B**ig **F**ucking **G**un
BLAP	**B**adges, **L**evels & **L**eaderboards, **A**chievements and **P**oints
BPM	**B**eats **P**er **M**inute
CDC	**C**enters for **D**isease **C**ontrol and Prevention
CTF	**C**apture **T**he **F**lag
CPU	**C**entral **P**rocessing **U**nit
DLL	**D**ynamic-**L**ink **L**ibrary
ESA	**E**ntertainment **S**oftware **A**ssociation
EE	**E**nergy **E**xpenditure
FC	**F**lappy **C**ycling
FFA	**F**ree **F**or **A**ll
FPS	**F**irst **P**erson **S**hooter
FPS	**F**rames **P**er **S**econd
GPS	**G**lobal **P**ositioning **S**ystem
GPU	**G**raphics **P**rocessing **U**nit
GWAP	**G**ames **W**ith a **P**urpose

HCI	**H**uman-**C**omputer **I**nteraction
HB	**H**eart **B**eat
HD	**H**igh **D**efinition
HP	**H**ealth **P**oints
HR	**H**eart **R**ate
HUD	**H**eads-**U**p **D**isplay
IBI	**I**nter **B**eat **I**nterval
IDE	**I**ntegrated **D**evelopment **E**nvironment
IP	**I**nternet **P**rotocol
ISFE	**I**nteractive **S**oftware **F**ederation of **E**urope
ITEC	**I**nstitute for **I**nformation **T**echnology
KRC	**K**endall **R**ank **C**orrelation
LAN	**L**ocal **A**rea **N**etwork
LED	**L**ight-**E**mitting **D**iode
LOC	**L**ines **O**f **C**ode
MDA	**M**echanics, **D**ynamics and **A**esthetics (Framework)
MMO	**M**assively **M**ultiplayer **O**nline
MMORPG	**M**assively **M**ultiplayer **O**nline **R**ole-**P**laying **G**ame
MOBA	**M**ultiplayer **O**nline **B**attle **A**rena
MUD	**M**ulti-**U**ser **D**ungeon
MWW	**M**ann-**W**hitney-**W**ilcoxon
OS	**O**perating **S**ystem
PC	**P**ersonal **C**omputer
PPG	**P**hoto**p**lethysmo**g**ram
Q3A	**Q**uake **3** **A**rena
QS	**Q**uantified **S**elf
QVM	**Q**uake **V**irtual **M**achines
RBC	**R**ank-**B**iserial **C**orrelation
RPG	**R**ole-**P**laying **G**ame
RPM	**R**evolutions **P**er **M**inute
SDT	**S**elf-**D**etermination **T**heory
TCP	**T**ransmission **C**ontrol **P**rotocol
TDM	**T**eam **D**eath **M**atch

TS	**T**ip-**S**leeve
TRS	**T**ip-**R**ing-**S**leeve
UDP	**U**ser **D**atagram **P**rotocol
UI	**U**ser **I**nterface
USB	**U**niversal **S**erial **B**us
VM	**V**irtual **M**achine
VR	**V**irtual **R**eality
VS	**V**isual **S**tudio
WFE	**W**orld **F**amous **E**lectronics
WHO	**W**orld **H**ealth **O**rganization

Formulas

Ohm's Law	I	$=$	$\frac{V}{R}$
Total resistance of resistors in series	R_s	$=$	$R_1 + R_2 + ... + R_n$
Total resistance of resistors in parallel	$\frac{1}{R_p}$	$=$	$\frac{1}{R_1} + \frac{1}{R_2} + ... + \frac{1}{R_n}$
Two resistor voltage divider	V_{out}	$=$	$\frac{V_{in} * R_2}{(R_1 + R_2)}$

Symbols

C	Capacitance	F (**F**arad)
Hz	Hertz	f (**f**requency)
I	Electric current	A (**A**mpere)
R	Resistance	Ω (**O**hm)
V	Voltage	V (**V**olt)

p	pico	10^{-12} ($1pF = 10^{-12}F$)
n	nano	10^{-9} ($1nF = 10^{-9}F$)
μ	micro	10^{-6} ($1\mu A = 10^{-6}A$)
m	milli	10^{-3} ($1mA = 10^{-3}A$)
k	kilo	10^{3} ($1k\Omega = 1000\Omega$)
M	mega	10^{6} ($1MHz = 10^{6}Hz$)
G	giga	10^{9} ($1GHz = 10^{9}Hz$)

Dedicated to my family, who always supported me throughout good and bad times, my friends for having created invaluable fond memories for me and people, who encouraged and inspired me during my journey.

Chapter 1

Introduction

Since the establishment of modern regular working hours[1], time seemingly grew ever more precious to many individuals, as becomes apparent when regarding countless psychoanalytical articles[2] or merely observing increasingly stressed out people in the streets. This phenomenon leads to populations suffering from *time poverty* [2], a term aptly referring to their inability to pursue certain work, interests, leisure activities, etc. due to an already overly dense agenda. It therefore comes as no surprise that people are trying to optimize their daily routines by adopting widespread behavioral patterns, like handling several tasks at the same time, unfortunately more often being rather unsuccessful as studies have shown [3–5]. However, regardless of a person's performance in attempting endeavours like juggling multiple online conversations simultaneously, watching TV while doing homework or answering emails during a meeting - most frequently the underlying motivations are to increase productivity, maximize the time that can be spent on matters deemed more important or to make the task at hand more enjoyable. All of these are valid incentives for improving one's daily life, but especially the latter two emphasize a common point of view: many activities we ought to undertake are not very interesting and hence burdensome to accomplish. Some children would, for example, gladly switch their vegetable plate for a box of chocolate chip cookies, a sales representative might be content if the company's product would magically sell itself or a lazy person may be happy to maintain a healthy body without moving a muscle. No matter how dreaded the task - we will always have to grit our teeth and get on with it, conforming to well known slogans like "business before pleasure". But why should "business" always be boring? Are we destined to complete tedious and repetitive duties over and over throughout our lives? A topic called *gamification* may be able to assist.

[1] The Industrial Revolution defined today's 8 hour work days and approximately spanned over a 100 year period (∼1750-1850) [1].

[2] E.g.: The Economist 2014, "Why is everyone so busy?" (`http://tinyurl.com/m7y4r72`)

Conceptionally old yet only quite recently trending[3], gamification roughly is concerned with making any task attractive and enjoyable, even up to the point of rendering it addictive. To realize this ambitious goal, game design elements are utilized, which appeal to people's ludic drive in order to overcome their natural programming [6]. When executed well, this can be enormously effective - for instance, as part of a competition[4] game designer Kevin Richardson designed a system to reduce speeding, which successfully lowered the amount of fast drivers by an average of 20% in Stockholm, Sweden: a checkpoint records the speeds of all passing cars and all vehicles within the speed limit are entered in a lottery as potential candidates to win the fines of speeders. Thus, simply by introducing a competition to a common mechanism, the radar speed check system, he appealed to the peoples' desire to win money, triggering their awareness of how they may succeed and inciting them to comply with the overall goal: driving more cautiously.

Similar concepts have been used all through human history, but first signs of gamification's modern interpretation emerged early in the 20$^{\text{th}}$ century, where canteens employed the nowadays familiar 10:1 sales policy to increase their revenues [6]: buy ten (cups of coffee), get one for free. Driven by the effectiveness of such strategies many others followed, starting with shopping rewards cards, which cleverly clouded people's perception of value: how do earned reward points relate to currency? Frequent-flyer programs even improved commercial success by appealing to the peoples' desire for status: loyal customers do not have to stand in line, frequently get upgrades or more exclusive Internet connections. Finally modern computer age paved the way for virtual gamified systems, for example Stack Overflow[5] rewarding badges as performance achievements, Amazon's[6] user review and rating scheme or video games[7] that offer virtual goods in exchange for real money. Today, an abundance of gamified applications exist, constantly increasing in numbers, but the concept certainly is no panacea. For one, much thought must be put into gamifying something in order to ensure the desired effects can be achieved. It also cannot truly eliminate time poverty, perhaps at most loosen tight agendas, if there indeed is a way to speed up some contained tasks. At last, most gamified systems require maintenance because, as is often the case with any game, people get bored playing and need motivation, for instance new features, to continue enjoying it. Nevertheless there still is much to be gained from employing gamification techniques, no matter if it is used to engage employees, heighten user experience, improve one's personal efficiency, attract people's

[3] The term "gamification" was coined in 2002 by Nick Pelling (`http://tinyurl.com/o6ye4ow`) and gained popularity around 2010 (`http://tinyurl.com/n9hcgsw`)

[4] "The Fun Theory" (an initiative of Volkswagen), 2010: `http://tinyurl.com/32pgce3`

[5] `http://www.stackoverflow.com`

[6] `http://www.amazon.com`

[7] e.g. `https://www.farmville.com`, `http://candycrushsaga.com`, `http://supercell.com/en/games/clashofclans`, `https://www.zynga.com/games/words-friends`

attention or enhance learning - the range of applications is too vast to cover in this thesis, therefore, only a small selection will be outlined in the Chapter 3.

As a scientific contribution this thesis will be using gamification to address a globally widespread problem: lack of physical exercise. Due to today's variety of constantly available multimedia, entertainment and technology assisting us in every way, many people unfortunately have very little need or incentive to summon up any bodily effort during or between their daily routines. The resulting unhealthy lifestyle can lead to negative psychological [7] and especially unfavorable physical conditions, such as obesity, as becomes apparent when examining the newest *World Health Organisation* (WHO) fact sheet[8] on that matter, which claims that the worldwide amount of highly overweight individuals have more than doubled since 1980. Regular physical exercise is indeed the foremost proclaimed countermeasure [8] to many possible deficits, although it definitely can not halt or prevent all of them, as there generally exist much more factors that influence health negatively, for example malpractices in dieting [9]. To a lot of people, however, exercise is equivalent to hard work and therefore can get very exhausting, specifically when in poor athletic shape. Naturally something this tiresome will be considered not desirable or in a nutshell: too big a cost for too little gain. Here again gamification can step in and greatly improve the incentive for them to start being more active. There already exist a great deal of mobile applications and devices that simply entice people to undertake fitness activities more frequently or aim to improve their user experience while working out (see Chapter 3). Additionally fitness games have been around for many years[9], which mostly combine video gaming with physical exercise often using specifically crafted input devices. This genre, gaming combined with fitness or frequently called *Exergaming*, as the thesis' title might already suggest, will precisely be its main concern and it will comprise a twofold user experience evaluation using different video game projects.

Just as humankind always or at least throughout passed on history liked to play games in general[10], video or computer games are no exception. On the contrary their popularity even rose to the point where they nowadays rather are identified by the single word "games", while others in everyday speech often require the usage of compound terms like "card games", "board games", "social games" and so on. Therefore, as a convenience, throughout this thesis "games", unless further refined, will always refer to video games. These games, in contrast to popular belief, are far from being nonsensical "kid's stuff": according to the Entertainment Software Association (ESA) 74% of all *gamers*[11] in

[8] WHO fact sheet nr. 311, 2015: "Obesity and overweight" (`http://tinyurl.com/62hyt96`)

[9] Already around 1987 Atari games like "Foot Craz" used pressure-sensitive mats to capture user input [10].

[10] See Figure A.1 in appendix A.1 containing a popular infographic on the history of social games.

[11] "Gamer" is a commonly used identification for a person regularly playing games.

the USA, who total an average of nearly 49% inhabitants, are adults[12] and the overall average age of players is 35. This makes sense when considering that games already exist for more than 60 years [11]. Similar popularity can be discovered by examining the Interactive Software Federation of Europe's (ISFE) quarterly GameTrack surveys, where the most recent numbers suggest that an average of nearly 63% of UK's, France's, Germany's and Spain's combined population from ages 6 to 64 play games[13]. Lastly, by now the global game market revenues amount to 74.2 billion dollars a year according to recent statistics[14], which more than justifies the Games Industry to be viewed as a serious business. The reasons for this huge market shares are simple, but effective: games are fun, challenging and often claim their players undivided attention. There is a broad variety of more specific factors that drive a person do play a game, for example social aspects, being creative or escaping reality for a bit, most of which constitute very simple human needs or desires. Furthermore losing at a game a player enjoys usually will not stop him or her from trying again, instead actually provide even more encouragement. This makes games, if well designed, perfect motivators for sportive activities. Home consoles, as already mentioned, picked up this idea very early and fitness games have recently become fairly popular[15], at least since the release of the Nintendo Wii[16], which was first boxed with popular exergame "Wii Sports". Such a purpose-built fitness game, named "Flappy Cycling", serves as the basis for the first part of this thesis' evaluations.

Unfortunately exergames, although often quite popular in the beginning, may exhibit downsides, for example they frequently lack in game design, can be outsmarted easily resulting in players not having to make any effort at all and usually grow boring soon. This is in part because they largely are very device specific and can therefore not be ported well to many systems: Nintendo Wii games are controlled with the "WiiMote" in combination with the "Nunchuck", Sony Playstation Games with the "Move Motion Controller", Microsoft's Kinect system captures movement using an RGB camera with depth sensor and there are many other systems shipping with their own hardware. This greatly restricts the access to those games and maybe is the cause for yet another big problem when it comes to fitness games: they are made with much less budget than so-called *AAA games* (pronounced 'triple A games'), which in film industry terms would be described as *blockbuster movies*: they are carefully designed, generally utilize cutting edge technology and typically are most anticipated among many people. Good AAA games maintain their popularity much longer, sometimes even are played for decades[17],

[12] ESA fact sheet, 2015: `http://tinyurl.com/ketagyk`
[13] ISFE GameTrack Digest of Quarter 2, 2015: `http://tinyurl.com/ok9ua7l`
[14] Superdata Global Games Market Report, May 2015: `http://tinyurl.com/olc3zz9`
[15] Fitness- and Exergaming arose as trend words after 2010: http://tinyurl.com/nra6x3k, http://tinyurl.com/oxcteqg
[16] `http://wii.com/`
[17] E.g. "Counter-Strike"(Valve Corporation, 1999: http://preview.tinyurl.com/ccat4e) and "World of Warcraft" (Blizzard Entertainment, 2004: http://tinyurl.com/39vdalk)

while enjoying a cult following of fans. Thus when it comes to choosing which game to play, gamers naturally more likely go for the option that satisfies them the most: the fun and entertaining AAA game instead of the boring, exhausting fitness game. This thesis therefore takes a novel approach in the second part of its evaluations: a classic fitness device that many households possess, an ergometer is used to enhance the user experience when playing a AAA game.

Structurally the thesis can be categorized into three coherent segments: background information (Chapters 2-3), description of studies (Chapters 4-5) and their evaluations (Chapters 6-7). Starting the background information segment, Chapter 2 provides a proper definition of gamification and outlines related important aspects like basic player motivations and commonly used techniques in different areas. The subsequent Chapter 3 exhibits work related to the topics discussed, i.e. gamification and fitness games. Next, initiating the studies setup segment, Chapter 4 addresses overall goals of the thesis and points out research questions. A general user study projects overview is presented in Chapter 5, also containing relevant implementation details as well as information about setup and realization. Afterwards, Chapter 6 starts the evaluation segment, where gathered results of the different user studies are meaningfully listed, interpreted and qualitatively as well as quantitatively evaluated. A conclusion and proposed future prospects finalize this master thesis in Chapter 7.

Chapter 2

Gamification

Gamification has undergone many redefinitions since it became a buzzword around 2008 [12], prevailing terms like *productivity games* [13], *funware*[1] or *playful design* [14]. More and more web applications used virtual rewards like points systems, badges, progress bars or leader boards for improving the experience of their users [15]. Soon, however, a much broader idea emerged, still centered around people, often fittingly addressed as *players* to emphasize their role in gamified systems, and aiming at increasing motivation, engagement or helping them in a large variety of circumstances. Hence, as outlined in Chapter 1, gamification nowadays is mostly context-agnostic. It commonly is characterized by varying following broad definition or using a similar one [12, 16]:

Gamification is the use of game elements and game-design techniques in non-game contexts to solve problems and engage users.

The idea of using game elements and game-design techniques in software applications is not exactly new: early *Human-Computer Interaction* (HCI) studies already used heuristics from games to create enjoyable user interfaces around 1980 [17, 18]. Of course *Personal Computers* (PCs) back then where only up and coming, but still today it is proven that it pays off to invest in technology ease of use, not least by the rise of companies like Apple[2], who always regarded user-friendliness as a top priority. Many domains adopted this kind of thinking and as an estimate by this year (2015) more than 70% of modern companies have at least one gamified application according to research conducted by Gartner[3]. However, they also stated that by 2014, 80% of all such gamified applications will have failed to meet business objectives due to poor design[4], which they

[1] VentureBeat article by Dean Takahashi, 2008: `http://tinyurl.com/6frdgr`

[2] `https://www.apple.com`

[3] Gartner press release, 2011: "More Than 50 Percent of Organizations That Manage Innovation Processes Will Gamify Those Processes" (`http://tinyurl.com/bml3rgb`)

[4] Gartner press release, 2012: "80 Percent of Current Gamified Applications Will Fail to Meet Business Objectives Primarily Due to Poor Design" (`http://tinyurl.com/qfwzjp9`)

claim to have become reality[5]. This gives the notion that successful gamification is indeed much harder than it seems: simply creating a points system with leader boards won't captivate employees forever – often it soon becomes a chore and even can result in negative effects like rivalries between people. Therefore, it is most important to properly understand what is needed to design a fun, long lasting experience for the targeted participants.

The next sections provide insights to most important aspects of gamification, starting with Section 2.1 pointing out characteristics that drive persons, in context of gamification often labeled *players*, to take on activities, i.e. focusing on their motivations, preferences and system progression. It is followed by a description of game mechanics and design elements in Section 2.2. Section 2.3 concludes this chapter and is concerned with gamification's basic types, principles and techniques.

2.1 Target Group: Players

Gamification's main concern, as already outlined, are the end users, i.e. the players of the systems that it tries to design. This section therefore accounts in detail for their abilities, needs, desires as well as preferences, which is carried out in following three Subsections: 2.1.1, concerned with motivational aspects, 2.1.2, discussing individual preferences and 2.1.3, describing levels of skill progression.

2.1.1 Motivation

Chapter 1 already suggests that any kinds of games are very powerful motivators for humans, but why do they generally have such an appeal? What makes us spend hours and hours of playing our favorite games, sometimes even forgetting to eat or depriving ourselves of sleep? Games have the excellent quality of being able to combine the *desire* we have when it comes to sexuality with the *predictability* of enforcement, except without applying any force or pressure, making the ambition of using a well designed system driven entirely by enjoyment [6]. This drive to undertake an activity differs from person to person and certainly depends on individual taste, environment and circumstances. However, there are typical actions that the majority of people consider pleasant, such as socializing with friends, spending time with their families, eating and sleeping or of course simply playing games. Other activities usually are not at all regarded comparably delightful: most individuals work eight hours a day or even more because they get paid

[5] Ivan Kuo for `http://www.gamification.co`, 2014: "Gartners Latest Gamification Research for 2014 with Brian Burke" (`http://tinyurl.com/onqnzy6`)

to do so. They even work harder if this is rewarded with a bonus at the end of the year. The psychology behind these phenomenons have been extensively studied starting as early as 400 B.C., when Aristotle formulated his theory of *behaviorism* [19]. Its modern approach is often tied to the twentieth century, where famous psychologist Ivan Pavlov and several of his successors, most notably B.F. Skinner, laid the groundwork for today's behavioral studies by examining the effects of reinforcement and punishment on animals and hypothesizing about humans based on their discoveries [20, 21]. The goal was to be able to predict human behavior when reacting to external stimuli.

Eventually results of behaviorism studies affirmed a practice, which proved to be successful then and even today to a certain degree [16]: for a person to condition another person's behavior in his or her favor, rewards or punishments need to be applied carefully. Expected actions are driven by what is called *extrinsic motivation*, which means that the affected person only reacts for reasons external to themselves, i.e. rewards or punishments. In contrast, when people really want to do something out of free will, they are led by *intrinsic motivation*. The terms sometimes can easily be confused and it should be remembered that truly any change in behavior provoked by non personal circumstances, comprising actions that are undesirable, even though they may lead to achieving personal goals like better social status or reputation, are driven by extrinsic motivation. As an example, two people in a relationship could be considered, where one partner really wants attend a certain concert because he or she likes the performing artists, i.e. is driven by intrinsic motivation, and the other one does not really care so much about it, but feels the need to accompany his or her partner, hence, driven by extrinsic motivation, tags along regardless.

Most commonly extrinsic motivation is provoked by behavior reinforcement systems, which can be built using schemata of varying positive or negative incentive ratios, i.e. timed applications of rewards or punishments, as depicted in Figure 2.1. The graphic shows the cumulative number of responses (y-axis) of a test subject[6] against the duration of four particular experiment schedules (x-axis). The higher the responses are the more engaged the subjects are.

Such behavior reinforcement techniques, also frequently named *operant conditioning*, trigger behavioral adaptation as a reaction to learning a fixed or variable reinforcement pattern, which operates response rates or time intervals. Table 2.1 illustrates all four possible combinations and explains the methods by means of real-world examples.

The effectiveness of these methods is evident: when reasonably applied, individuals subjected to them will do all that is necessary to receive their rewards or steer clear of

[6]Initially the study was conducted on rats, monitoring their responses (pressing a button) to food rewards, but similar behavior is analogously observable on human beings in everyday situations.

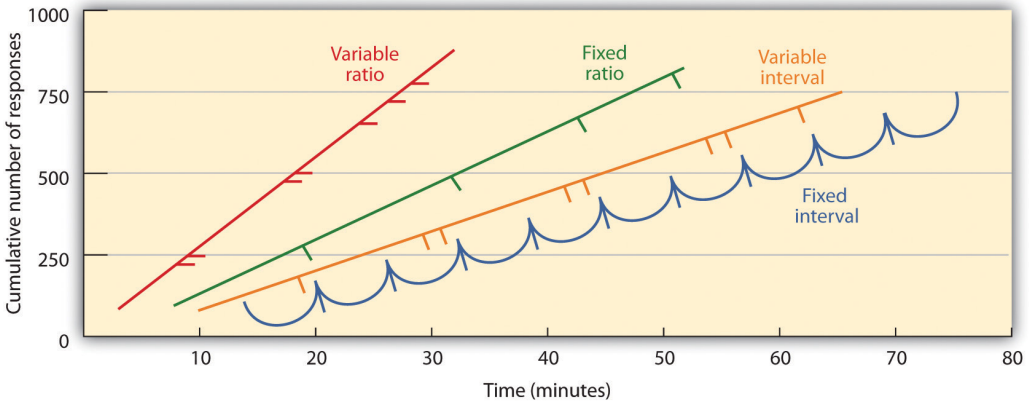


FIGURE 2.1: Response Patterns under Different Partial Reinforcement Schedules [22]. Source: "Beginning Psychology v.1.0", chap. 7 (`http://tinyurl.com/p9yq5j4`), License: `Creative Commons 3.0 by-nc-sa`.

TABLE 2.1: Reinforcement Methods [22].

Method	**Explanation**	**Example**
Fixed-ratio	Behavior is reinforced after a specific number of responses	Factory workers who are paid according to the number of products they produce
Variable-ratio	Behavior is reinforced after an average, but unpredictable, number of responses	Payoffs from slot machines and other games of chance
Fixed-interval	Behavior is reinforced for the first response after a specific amount of time has passed	People who earn a monthly salary
Variable-interval	Behavior is reinforced for the first response after an average, but unpredictable, amount of time has passed	Person who checks voice mail for messages

undesirable behaviors, if reinforcement is delivered in forms of punishment. On the downside, operant conditioning can be very unpleasant for people, considering that they often get very little enjoyment out of extrinsically motivated tasks and hence only grudgingly go about completing them. Specifically in the case of variable-rate conditioning, obvious negative personal effects can be observed, as individuals do generally not know when a reinforcement is going to occur, which provokes addictive or even aggressive behavior. As Table 2.1 already points out, this mechanism is exploited by casinos, tempting players to spend more and more money. Gamification, however, is really all about improving the experience for the affected users and not about boosting efficiency at any cost [16]. It should therefore never make any person suffer and this is why reinforcement techniques should absolutely not be overused.

Since encouraging people through triggering extrinsic motivation in such ways is a behaviorist's area of expertise, shouldn't there also exist a science concerned with intrinsic motivation? Indeed several so-called *cognitivist* theories were developed [16], the most prominent being the *Self-Determination Theory* (SDT) by Edward Deci, Richard Ryan *et al.* [23], which assumes that humans generally behave proactively with a potent ambition for personal improvement, following innate psychological needs. This inherent desire, however, strongly depends on the support of a person's external environment and can therefore possibly be diminished or even extinguished, which is independent on cultural or social classes and results for instance in people alienating themselves from others, predominantly adopting passive habits like watching television all day or completing their jobs apathetically, while biding their time in prospect for the weekend. Not least to counteract such destructive behavior, SDT is concerned with finding the exact human needs that must be satisfied to enable thriving personal growth and well-being. It differentiates between three of such essential needs: *competence* [24, 25], *relatedness* [26, 27] and *autonomy* [28, 29].

People feel *competent* when they successfully handled challenging tasks like winning a spelling bee or learning how to play an instrument, i.e. they effectively interacted with their environment. Also most individuals enjoy social contact, that is to say meeting and talking to friends, relatives or making new acquaintances, which can be summarized as the need for *relatedness*. Lastly *autonomy* is the desire for having control over one's life, meaning that no matter the kind of action, it should conform to personal values and preferences or in other words represent a joyous experience. Any activity relating to one or multiple of these particular needs typically is motivated intrinsically. Most common examples would be hobbies like painting or solving cross word puzzles, pleasurable pastimes such as taking a relaxing walk or entertaining social events like meeting friends at a discotheque. But these experiences are not merely tied to tasks that people relish in their leisure time, also work can be very enjoyable – be it satisfaction from delivering a great product to a customer, helping out a struggling colleague or giving a well accepted presentation in front of all company employees. Since all of these experiences are, as already mentioned, results of self-motivation and on top of that entail satisfaction or happiness, it truly makes sense to focus on how to utilize SDT concepts for gamification.

Just the right amount of intrinsic motivation can be specified by a concept called *flow*, which was developed by often cited psychology professor Mihaly Csikszentmihalyi [6, 16, 30]. Flow could casually be characterized by "being in the zone" and it is achieved, when an activity totally consumes a person, such that time flies by without notice or as a more particular description: a state exactly between boredom and anxiety [6]. Figure 2.2 depicts this state and outlines when it occurs: the higher a one's ability for an arbitrary task the more challenging it must be for not getting overly bored or too anxious. Since

this state is truly most comfortable for everyone, gamification of course tries to recreate it, which is a very demanding endeavour, considering that an engaging gamified system for a broad user base must be very adaptive to every individual user and also gradually refine its posed challenges as progress is made, i.e. people proceed towards mastering that system (refer to Subsection 2.1.3 for more information).

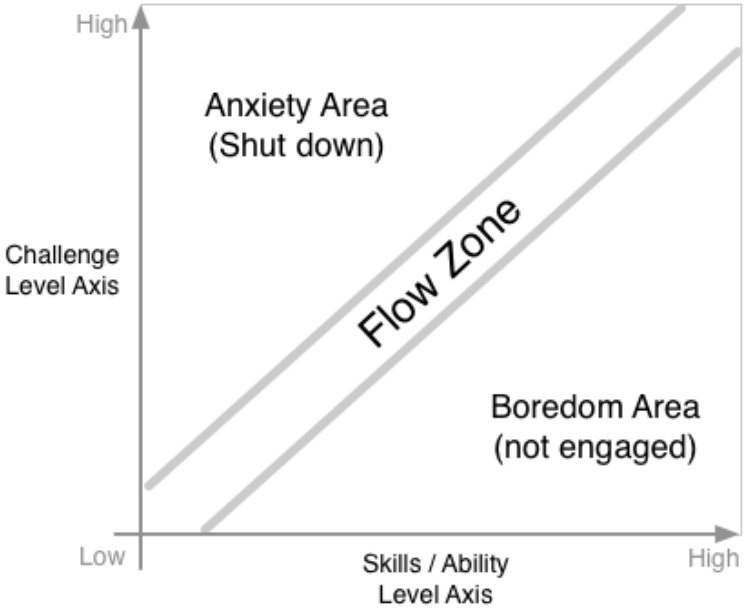


FIGURE 2.2: Concept of Flow: intrinsically motivated persons are inside the flow zone when their skills/abilities meet the challenge at hand. Source: [6].

Considering all motivational factors mentioned above, it is easy to see why games specifically are suited so well for encouragement: they combine all aspects that provide incentive for a player to reach the state of flow. A typical modern *Massively Multiplayer Online Role-Playing Game*[7] (MMORPG) can serve as a perfect example for keeping its players interested and immersed, as is portrayed in Table 2.2 and Table 2.3.

One final question remains now: which of the two motivations is better for gamification? According to some old assumptions, intrinsic drive is superior to extrinsic drive because it means that people are self-motivated, but it really is not that simple because many intrinsically motivated ambitions, like the aforementioned aspiration to reduce weight, need extrinsic motivations, i.e. small rewards for each exercise, to be realized [6]. Practicing gamification therefore does not simply mean using one or the other approach, but oftentimes – of course ultimately depending on the field of application – applying a considerate combination of both.

[7]Players in a typical (MMO)RPG take the role of a character with specific attributes, that can be increased through completing tasks or fighting enemies or other players. Often they enter a fantasy setting, where magic and mythical creatures exist.

[8]Power-ups temporarily boost a characters abilities, e.g. making it stronger or more resilient.

Table 2.2: Extrinsic Motivational Aspects in a Modern MMORPG.

Aspect	Example
Positive Reinforcement	Handing out virtual rewards, like power-ups[8], in a fixed or variable interval, i.e. daily or randomly every few days, to reinforce logging in frequently.
Negative Reinforcement	Dying through a monster or another player can entail penalties like losing equipment, money or incapacitating a character for a short period of time. Frequent dying could even worsen these negative effects, while managing to survive after several deaths for long enough time intervals in turn can gradually reverse this decline.

Table 2.3: Intrinsic Motivational Aspects in a Modern MMORPG.

Aspect	Example
Competence	Seasoned players helping out new, inexperienced by giving them useful tips and explaining the game's concepts.
Relatedness	Socializing with other players by chatting, trading items or going on adventures together.
Autonomy	Complete control over one's character: initially customizing its appearance, choosing which abilities to upgrade and most importantly deciding which virtual activity to undertake.

2.1.2 Preferences

Knowing what motivates people is very important, but at the end of the day everybody has got different preferences, meaning that some person's predominant desires or needs are less important to others – gamers of course are no different for that matter. However generally certain tendencies can be identified that in total are able to characterize specific individuals. Richard A. Bartle, a British professor, researcher as well as author, while developing a *Multi-User Dungeon*[9] (MUD), discovered and documented several kinds of tendencies, each of which are more or less prevailing in every MUD player or, for that matter, really persons playing any kind of game [31], which ultimately can be projected onto many real life situations. Figure 2.3 illustrates these characteristics and breaks them down into four basic player types.

[9]MUDs are usually text-based virtual multiplayer games, often thought of as the spiritual predecessors of MMORPGs.

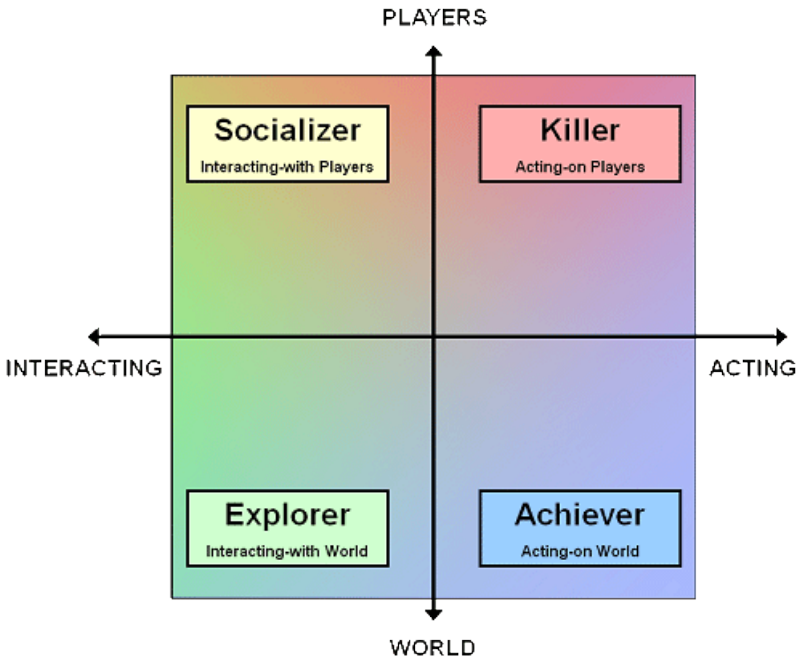


FIGURE 2.3: Bartle's basic player types.
Source: Bart Stewart for Gamasutra, 2011: "Personality And Play Styles: A Unified Model" (`http://tinyurl.com/o3fbffh`)

The graphic's x-axis represents types of actions players can choose from: interacting or acting. Either preferred operation can then be applied to other players or the game world, which is indicated by the y-axis. The diagram's labeled corners portray said basic player preferences [6, 31]:

- ♠ **Explorers** interact with the game world. They initially want to get an idea of the game's magnitude, i.e. discover new areas, find hidden places and, when there is nothing left to explore, they tend to the game mechanics in-depth for gathering comprehensive knowledge as well as finding out tricks, secrets or flaws that might not yet have been discovered by anyone. Collecting information is their main concern, everything else, like progressing in the game, is secondary unless it is absolutely necessary to continue their research. Typical conversations include: "Hmm...", "Haven't tried this, what is it good for?", "There's a time-saving shortcut from A to B ...".

- ♦ **Achievers** act on the game world. They are all about mastering the game and excel others by possessing an impressive amount of treasure, having the highest character levels or overall completed most quests[10]. Typical conversations include:

[10] Quests are tasks the game asks its players to complete in order to receive valuables, power-ups or experience points, which can be used to level up the character, i.e. make it stronger.

"I'm busy...", "5000 more points to go!", "I made a fortune on selling items A and B".

♥ **Socializers** interact with other players. To them the game constitutes merely a means to get to know other people and form wonderful and longstanding relationships. High community status is their primary goal and all possible requirements, like for example knowledge about the game, are only necessary in cases where they help to advance socially. Typical conversations include: "Hi!", "What are you all up to?", "I know how you feel – this can be really frustrating.".

♣ **Killers** act on other players. Bloodthirsty, well trained and always on the lookout for prey, they strike down players for their own entertainment. Although this also can be used to help someone, for example when protecting a weak friend from stronger killers, it much more often is used to cause havoc – the greater the damage inflicted, the more satisfied they are. This generally requires some game mastery, especially a focus on getting strong enough to allow engaging in fights with all kinds of opponents and emerging triumphant in the end. Typical conversations include: "Hahaha!", "Die!", "You wuss!".

Above listing's mixed up bullet symbols are not results of bad character encoding, they're actually clever mnemonics for memorizing the player types by means of an ordinary card deck [31]: explorers (*spades*) are always digging for knowledge and information, achievers (*diamonds*) constantly gather valuables (e.g. treasures), socializers (*hearts*) above all care and feel for others, killers (*clubs*) enjoy beating up others (e.g. using clubs or other weapons).

As one can guess when regarding given descriptions, player types in general are not exclusive. People rather enjoy a mixture of all, ever slightly varying choices dependent on their moods, but often with a substantial tendency towards one specific preference, which can change in time or even differ between games. The diagram of Figure 2.3 allows for a rough player categorization: if someone is at the time more interested in the virtual world than its inhabitants and likes finding out what makes the game tick as well as actively mastering it in terms of leveling, getting powerful or collecting treasure, the person can be placed in the lower half, somewhere between the explorer and the achiever quarter. Once a disposition is settled, less favored play styles either are predominantly treated as necessities to advance in one's primary goals or they constitute a welcome change to prevent the user experience from becoming stale. When designing a multiplayer game or application, it is therefore crucial to think about which player tendencies to incorporate and also to make estimations about their average distribution, as they will constitute the main incentive for people to start playing or continue to do so. Of course

these considerations are primarily subject to the games themselves because their overall target audiences differ frequently: the least common player types in one game can be the most common in another, for example killers in MMORPGs are much more scarce than killers in *First Person Shooters*[11] (FPSs). Nevertheless, specifically games that represent a breeding ground for more than one kind of player need to carefully keep a balance between them because inevitably they will influence each other: for instance in an MMORPG, killers can significantly reduce the entertainment value of achiever's as through interrupting their activities, i.e. killing them while completing quests, they slow down their progress, which perhaps merely drives them to altering their play styles or causes much worse scenarios like making them quit the game. The latter obviously would have negative effects on the games' popularity because in this particular context achievers constitute a big portion of players, as they pursue most MMORPGs' primary laid-out objective – performing well or, in other words, winning.

Disregarding specific game and application contexts, by far the most dominant player type is the socializer [6], which adds up when taking a look at humankind's long history of beloved social games commonly played by many cultures (cp. Appendix A.1, Figure A.1). Many of today's games ultimately appeal to millions of players just because they offer multiplayer experiences, where people can socialize while planting virtual crops in casual games like FarmVille[12] or during strategic space battles in complex MMORPGs like EVE Online[13]. Knowing they talk, interact and fight against real people even drives them to play longer than they would if they played a single-player game. Consequently, it is not unusual for players to pass up going out with their real friends because they promised their online friends that they would participate in some specific virtual adventures. This can happen even if they are not particularly motivated to play, but feel obliged not to disappoint their fellow comrades.

Accordingly, when considering these facts, games or gamified systems created for a widespread audience should optimally prioritize social aspects, followed by providing incentives for achievers, explorers and lastly, comprising the statistical minority, killers. Again, this rough ranking strongly depends on the type of game or application to be designed, hence only serves as a general guideline.

[11] FPSs are games, where players see the virtual world throught the eyes of the games protagonist and typically kill enemies or other players by attacking them with guns, magic or other kinds of weapons in what is often called a "virtual deathmatch".

[12] `https://www.farmville.com`

[13] `http://www.eveonline.com`

2.1.3 Progression

When people learn something new, like playing an instrument, they need to practice in order to become better at it in time – depending on their talents, previously elsewhere acquired expertise applicable and levels of interest in learning the skill, some progress slower, others faster. This certainly holds for games or gamified systems as well because typically new players, also called *newbies* or *novices*, must adapt themselves to the yet unfamiliar environment presented to them. Stuart and Hubert Dreyfus, professors emeriti at Berkeley[14] university, studied people interacting with different systems and discovered significant similarities in their learning processes [32, 33]. Their studies included skill developments of airplane pilots, chess players, automobile drivers as well as adults learning a second language, which led them to devise a five-stage model of people's mental activities while their competences improved, outlined in Table 2.4.

Table 2.4: Five Stages of Skill Acquisition. Source: [33].

Skill Level	**Components**	**Perspective**	**Decision**	**Commitment**
Novice	context-free	none	analytical	detached
Advanced Beginner	context-free & situational	none	analytical	detached
Competent	context-free & situational	chosen	analytical	detached understanding & deciding, involved in outcome
Proficient	context-free & situational	experienced	analytical	involved understanding, detached deciding
Expert	context-free & situational	experienced	intuitive	involved

Although there are claims that throughout history there have been people that *knew* everything[15], it is doubtful that they also could *do* everything, especially at a level of mastery. That is to say, being proficiently good at something requires more than theoretically knowing how to do it – ultimately only repeatedly engaging in a profession and encountering a sufficient amount of use cases will lead to improvement or as often recalled: "practice makes perfect". This does not mean that theoretical knowledge or rules learned by heart are useless, they are in fact essential and help greatly when confronted with making a decision related to a skill. Situational experiences like this can not easily be defined, hence must be faced by students themselves. They also need feedback regarding their performances to be able to learn, which is obtainable by

[14] Berkeley, University of California (`http://www.berkeley.edu`)

[15] Hmolpedia - Encyclopedia of Human Thermodynamics, Human Chemistry, and Human Physics, 2015: "Last person to know everything" (`http://tinyurl.com/gu5n54z`)

observing persons more skilled than them, written as well as verbal instruction or their own judgements if inventing brand new skills. During this learning process, a person traverses through the aforementioned five skill levels, summarized in Table 2.4, which characterize the growing sophistication he or she experiences at every stage.

1. **Novice**

   Students commence learning with acquiring the ability to recognize so-called *context-free* features, which are distinctly defined skill related facts that resemble the basis for all future advancements. In a board game, for example, these features would be the functions of different pieces needed to play or in a language the meaning of letters that form words. Regardless of the context they appear in, a novice understands them by recalling memorized rules, i.e. a rook can move horizontally as well as vertically or the letter "O" has a circular shape. Since beginners naturally can not yet fully comprehend everything that is relevant to the skill, when assessing their performances, they either personally resort to observing how successful they are in applying all learned rules or confide in a more advanced teacher that provides meaningful feedback for behavior adjustment.

2. **Advanced Beginner**

   Extensive experience through encountering and handling real situations lead to the appliance of more learned context-free rules but also some further advanced, *situational* ones that are hard to define: for instance, people recognize the voices of others that are close to them on the telephone but cannot really account for why or, as another example, somewhat advanced drivers know in which situations passing by other cars can be attempted. Students can handle ever more of these encounters because they recognize similarities to experiences they had in the past and their brains have stored information on how to deal with them – any other form of description, verbal or written, can not really help them in their decisions. Gradually the amount of perceivable context-free as well as situational aspects becomes so vast that people in many situations can not assess which reaction is more important than another, as is observable by expert teachers when regarding their advanced beginner students: in a 1982 study on nurse's journeys from novices to experts the majority of new nursing graduates blindly follow instructions in the given order to handle specific situations involving patients, instead of applying common sense where it is due [34].

3. **Competent**

   To avoid situations getting out of hand, competent students have learned or been taught to perceive all facts of an encountered situation and weigh their options

in order to reach a desirable outcome. Essentially, competent people are goal-oriented, often following an ambition of their own and only using selected rules as a means to an end, rather than complying with all of them: many drivers, set out to reach their destinations as quickly as possible, take the shortest routes, while tending to undertake risky maneuvers and ignoring speed limits. Hence, just like during the daily routines at any company where urgent matters are taken on first, individuals in this stage keep a hierarchical priority list, i.e. an *organizational plan*, and act accordingly, even if sometimes rules must be broken to reach their goals. Other than novices and advanced beginners, who tend to view undesired outcomes caused by their actions applied in good faith as results of rule-misdefinitions, i.e. always act in a *detached* manner, competent people, although also understanding as well as deciding in a detached fashion, deem themselves responsible for their performances or in other words are *involved* in the effects they have caused and therefore must put much care into devising their organizational plans for various occasions. As a consequence of this engagement, they will learn very much from all possible outcomes: favorable results entail the chosen plan to be stored in memory and not desirable developments will be kept as reminders to never make the same mistakes again on future encounters of similar situations.

4. **Proficient**

   A person exhibiting a proficient skill level is truly immersed in the performed task, i.e. has an intuitive, involved understanding, and possesses an extensive collection of past memories on which he or she can base decisions, which, however, still is a conscious, detached act. That is to say, the notion to act might occur intuitively but the actual decisions that can realize said impulse require a conscious analysis of memorized factors. As an example, a proficient gamer playing an FPS may intuitively realize the position an enemy is attacking from, but needs to consciously think about which countermeasures to take: jump, duck, run away or move while responding with fire. Because of proficient people's considerable experience, the strategies used for decisions usually can be greatly narrowed down to just a few, which, analytically weighed against each other, are ultimately reduced to best solution suiting the initial intuition.

5. **Expert**

   Experts fully embody a skill to such extent that an it becomes second nature to them, i.e. everything about it is intuitive. Talking, walking or social interactions are everyday activities that typically do not require any conscious decision making while performed. Of course this implies that nothing out of the ordinary happens, which will often entail some further assessment of the situation. Yet, the cognitive process of experts in such cases comprises critical reflections of intuitions rather

than calculative decision making. In their minds, comparable to the proficient performer, many connections between similar situations are stored but also solution strategies are grouped together helping them to immediately react, without having to think consciously. Accordingly, performance becomes fluent: expert badminton players do not pause to estimate where a returned shuttlecock is going, they instinctively move to that position and overall demonstrate remarkably fast reactions. Chess grandmasters act intuitively on the basis of approximately 50,000 recognized position types [33], which probably compares to a native speakers repertoire of distinct phrases, which is far greater than just the language's vocabulary. Nevertheless, as even experts are still human, their intuitions may not always pan out – circumstances may not have been optimal or unexpected events prevented desirable outcomes. Since they make, however, much more sophisticated decisions than a rule based problem solver, i.e. a computer, they have a better shot at succeeding with their intentions.

Skill levels can certainly increase, but they can also decrease – for example, a former expert pilot at flying fighter jet A, who moved on to fighter jet B, is very likely be merely proficient or even competent when first handling jet A again. People can also be stuck on a level for their whole lives if the skill is sufficiently difficult to master: in chess typically only a small percentage of beginners can advance beyond being just competent players, which would require them to let go of the thing that got most of them to play in the first place – analytical thinking [33]. Still, there are other domains that specifically were designed for every person being able to become expert: driving cars.

As gamification is such a design process, it is crucial to construct systems accordingly, i.e. to accommodate all possible skill levels and ensure that players are constantly interested or in other words: easily achieve a state of flow. This means that it should be possible to enjoy any skill level without the need of progression [6] – advancement therefore has to remain a choice to get more involved rather than a necessity to prevent boredom. Of course it also entails that the gameplay will be different for every progression stage, which emphasizes the importance of carefully designing the novice as well as the advanced beginners levels in a newly designed application, because these represent the first impression of players starting out. The endgame should, as people progress, always be subject to change: people will get bored if there is nothing left to do, i.e. they reached the highest levels, have the best high scores, own the perfect equipment and so on. Adding new challenges for expert players that already achieved everything is a safeguard to preventing an otherwise very likely, undesired consequence: they stop playing.

2.2 Gamification Elements

Just like a craftsman, who needs the right tools to build something, a practitioner of gamification requires certain elements to be able to gamify a system. Since gamification draws its rationale from conventional games, those elements can be regarded as a collection of all the little components that, when meaningfully combined, form such a game, also simply called *game elements*. Of course any game usually comprises obvious elemental parts such as the 32 pieces two chess players line up against each other, but also game rules, like the pawn being able to capture another piece by moving diagonally onto its position, count as basic components because they ultimately represent an integral part of the gameplay. Furthermore, as can be guessed by now, nearly every small detail that in the end plays a part in the final game is considered a game element. However, designing a game, in general, differs from gamifying an application because the two skills usually do not concur in all of their objectives: both should entertain but gamification additionally ought to serve a higher purpose, for instance raising productivity. Therefore, it is not always possible to use arbitrary game design elements for gamification. Nevertheless, if they can be utilized, there is no need to go ahead and build an entire game out of them – gamification, as already mentioned, improves non gaming context with game elements, therefore, it – unless specifically involving games[16] – merely operates using some of their components cleverly to keep its end users focused on ordinary tasks, making them more enticing, instead of plucking people away from reality by immersing them into never-never land, as is typically done in games [16].

This section is concerned with those game elements that have been successfully used in gamification and often are suggested by experts [6, 16, 35]. It is arranged in two subsections that outline a selection of approaches to identify basic components of games usable in non-game contexts: a popular game design framework breaking apart respective aspects in *Mechanics, Dynamics an Aesthetics*, in short MDA (Section 2.2.1), as well as the definition of a *Pyramid of Gamification Elements* (Section 2.2.2).

2.2.1 MDA Framework

Many gamification designers, when defining their set of tools, refer to a framework that classifies games by regarding its Mechanics, Dynamics and Aesthetics (MDA). This breakdown was developed by Robin Hunicke *et al.* [35] and they describe it as a "formal approach to understand games" bringing game design and development closer together. Figure 2.4 shows the scheme's components and their relations to each other.

[16] Games in such contexts are often referred to as *Serious Games* or *Games With a Purpose* (GWAP).

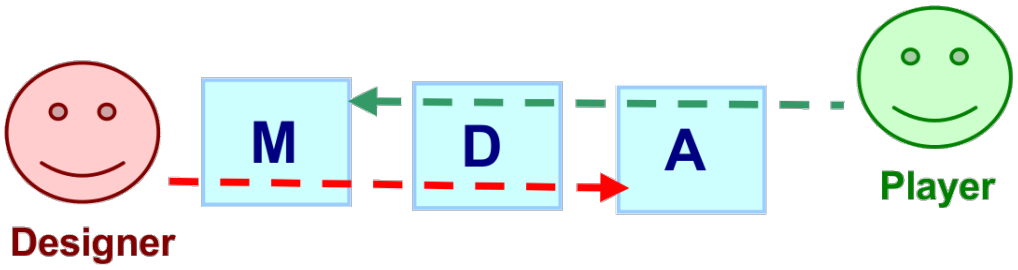


FIGURE 2.4: Mechanics, Dynamics and Aesthetics with their relationships.
Source: [35]

Mechanics comprise a game's basic elements, i.e. its building blocks like the corresponding data and algorithms it incorporates.

Dynamics observe the performance of the mechanics during gameplay, i.e. all effects resulting from in-game processes and interactions with the player.

Aesthetics characterize preferable outcomes of said player interaction, i.e. positive reactions that make the game worthwile.

Every MDA component represents a view on a specific game aspect and all of them logically correlate to each other depending on which involved person's perspective is taken: the game designer's or the player's. As the figure indicates, designer and player approach MDA from two different angles: while the mechanics are the starting point for the designer leading to a dynamic application as well as in turn to delightful aesthetics, the player first gets in contact with the aesthetics, possibly advancing to the dynamics and then mechanics.

The framework suggests to approach the design process by starting from the player perspective. The first step is therefore to define the overall aesthetic goals of a game, which can be one or more of the following basic emotions that people experience when playing [35]:

1. Sensation: a player's senses are stimulated.
2. Fantasy: escaping reality and getting drawn into a virtual world is appealing.
3. Narrative: following an interesting storyline offers entertainment.
4. Challenge: the game, *Artificial Intelligence* (AI) or other people pose a competition.
5. Fellowship: interacting with other players yields social satisfaction.
6. Discovery: exploring maps or the game is tempting (cp. explorers in Section 2.1.2).

7. Expression: acting out one's creativity and self-discovery.

8. Submission: playing for fun or to wind down from exhausting other activities.

A typical AAA FPS could thus include the aesthetics sensation, fantasy and challenge. Since dynamics are the foundation for these emotions, they can be viewed as player developed habits that bring joy or frustration, i.e. well-known phenomenons like camping[17] in FPSs or even game specific behavior, for example diverting weapons from their intended uses – sometimes rocket launchers, if their projectiles' impacts create a sufficiently powerful effect, can be used to boost one's jumping capabilities and, by doing so, it is possible to reach places that would conventionally not be reachable. The weapon itself in such a conventional FPS is part of the game mechanics, which comprises all the tools that allow game dynamics to emerge: power-ups, player models, arena[18] time limits and so on.

In summary, the MDA framework is very much concerned with what ultimately makes a game fun to play and builds everything else around definite aesthetic design goals. It certainly requires designers to employ an iterative process involving a lot of trial and error in order to fine-tune applied game mechanics in such ways as to allow players to experience enjoyable as well as fair dynamics. MDA's structure therefore suits gamification well, since a gamified system that is used everyday usually needs to be adjusted or improved from time to time so it does not become boring.

2.2.2 Pyramid of Gamification Elements

In the course of analyzing over 100 gamified systems, Kevin Werbach and his colleagues developed the Pyramid of Gamification Elements, depicted in Figure 2.5. Just like MDA, it also splits fundamental game aspects into three levels, but for the bigger part defines them differently. *Dynamics*, *Mechanics* and *Components* together form the pyramid's levels with the most abstract concept residing on top, ever more concrete ones towards the bottom [16].

Dynamics are to a certain degree analogous to MDA's aesthetics. They encompasses the overall intentions as well as the effects of a gamified application and therefore also bear restrictions or necessary trade-offs that must be considered along the design process. Werbach *et al.* define five such dynamics: *Constraints*

[17] Campers in FPSs patiently hide in a secluded spot to kill a massive amount of players, hence racking up their points while avoiding dangerous territories.
[18] In a typical FPS, deathmatches are held in so-called multiplayer arenas or maps.

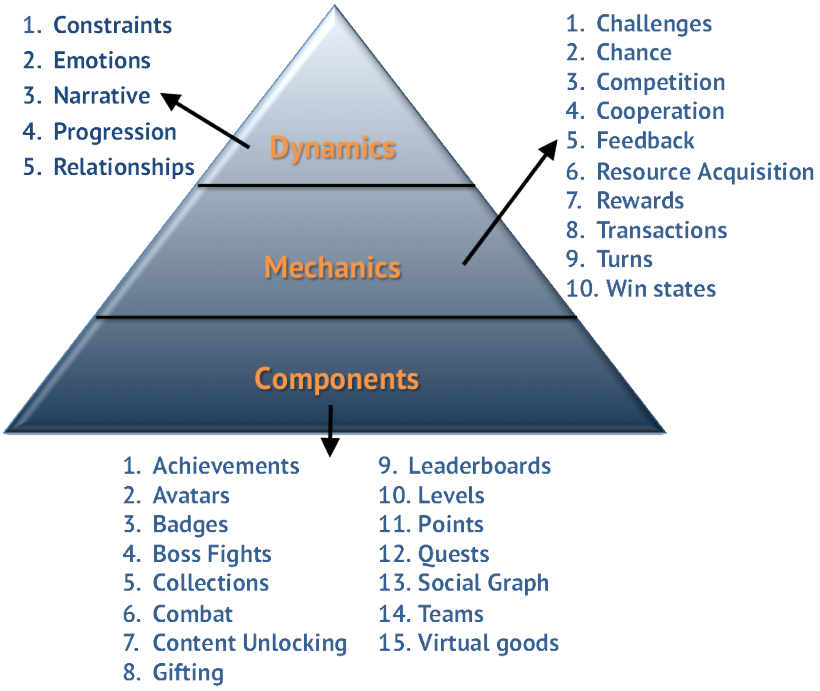


FIGURE 2.5: Pyramid of Gamification Elements.
Source: based on "The Pyramid of Gamification Elements",
Kevin Werbach (`http://tinyurl.com/np9xbto`),
License: `Creative Commons 3.0 cc-by`.

(specific limits), *Emotions* (feelings like joy, dissatisfaction, competition, ...), *Narrative* (satisfactory, plausible storyline), *Progression* (gradual advances in difficulty, adequately matching the players abilities) and *Relationships* (social fulfillment).

Mechanics are more concrete elements that control gameplay and facilitate the engagement of players. Their purpose is to provide the means to achieve aforementioned dynamic goals. Ten such mechanics have been identified: *Challenges* (tasks that players have to muster a certain effort for to complete), *Chance* (random surprises to keep the game interesting), *Competition* (for somebody to win, someone else has to lose), *Cooperation* (teamwork to reach a common objective), *Feedback* (information about participants performance), *Resource Acquisition* (collection of desirable items), *Rewards* (advantages granted for achievements or completed tasks), *Transactions* (integrated trading system), *Turns* (interaction between players by taking turns), *Win States* (all criteria leading appointing one or several players as winners, losers or calling a draw).

Components are the most specific game elements and they are closely tied to the mechanics, such that one or more components may ensure the accurate implementation of a certain mechanic. They constitute the largest level of the pyramid with a total of fifteen elements: *Achievements* (specific goals to be reached), *Avatars* (virtual or physical character portrayal), *Badges* (public or private signs

of achievement), *Boss Fights* (demanding and difficult challenges matching the player's current level), *Collections* (sets of obtainable objects or achievements), *Combat* (common fight between characters or monsters, usually carried out quickly), *Content Unlocking* (emerging new possibilities after persons complete particular objectives), *Gifting* (feature to share possessions with other players), *Leaderboards* (illustrative ranking of player performances), *Levels* (specified milestones in character advancement), *Points* (game performance in numbers), *Quests* (challenges that yield rewards when accomplished), *Social Graphs* (visualization of individual social connections), *Teams* (alliance of players for the purpose of completing a shared objective), *Virtual Goods* (virtual valuables, sometimes as well holding real-world monetary worth).

As a collection all of these many elements define an extensive toolkit for creating a gamified system and thereby also beautifully highlight that serious gamification goes way beyond just employing the most popular of its components: *Badges, Levels & Leaderboards, Achievements and Points* (BLAP) [36]. Often believed to be the epitome of gamification, BLAPs nonetheless generally are satisfactory starting points – with good reason because they are straightforward to employ for a great deal of applications, yield good results especially in fresh systems and, since being that common, implicate a certain familiarity for end users. On the downside, they can easily be misunderstood and as a consequence applied in the wrong ways.

By awarding points in a gamified system player progress can be tracked, which is an important feature for users as well as for the designers. They can regulate if somebody wins or loses, provide feedback, indicate participant's status quo and simply produce valuable data for analysis. Most popular concrete point applications use them to resemble accumulated experience, be redeemable for prizes, represent affiliation towards specific topics and support others, i.e. giving the option for users to reward somebody else for doing a good job [6]. Overall they are, however, not very meaningful and can therefore easily lead to confusion or discouragement: which prizes redeemable by points are fair, which are rip-offs? The leading player of a gamified application accumulated an impressive one million points – why bother to participate, if chances of catching up are next to impossible?

Badges and achievements, commonly used synonymously, often help to give points more significance: a simple application would be to indicate that a person successfully accumulated a specific amount of points by putting a fancy emblem on his or her profile in order to serve as a public player performance display. More frequently though they are used for all kinds of achievements, which not necessarily are connected with points. For

example, an achievement in a geocaching[19] system could be to find ten caches that were hidden in spots with altitudes of over 2000 meters above sea level. Of course one could also design goals that can be reached with much less effort like logging into a website for the first time – the possibilities of application are nearly limitless. People look forward to collecting badges because they feel a sense of accomplishment when completing required objectives, want to show-off their capabilities to others competitively or relate to fellow collectors of badges similar to theirs. At the same time they conveniently provide statistical data for designers to interpret: what are a player's favorite functionalities? Is the system heading in the right direction? Hence, as long as these achievements are employed in an appropriate way and regarded meaningful – at least to some extent, they can effectively motivate people. An overuse on the other hand could be the downfall of a gamified application, as it will just be considered silly or disproportionate, equally unattractive like a web page littered with advertisements, only providing very little actual information.

Leaderboards in their traditional form provide a global ranking of a system's players. People browsing these listings are interested how their performances compare to others, therefore their experiences when doing so can naturally be anything from very satisfactory to utterly disappointing (refer to the example above, where new players are put off by the achievements of established ones). Consequently, applications, which mainly use leaderboards for providing incentive, often only appear successful because a few extremely competitive players achieve top rankings, become overly committed and thereby boost the perceived popularity[20]. A good example of that phenomenon was the social network Google Orkut[21]: simply by introducing a sign-up leaderbord the developers created a competition that Brazil as well as India – the most engaged countries – "won", more and more stifling other countries' desires to compete [6]. It is accordingly a better idea not to base leaderboards on just one property and discourage the majority of users, but have several rankings for different properties, which will keep much more players interested in the long run. Also, if possible, designers should try out shorter term leaderboards, which help to avoid people getting bored, as they are provided a quick challenge, i.e. an exciting surprise with a limited duration, for instance, only lasting one day[22].

Levels, comparable to the skill evolution in Section 2.1.3, resemble in-game progress and usually transitions between them indicate variations in difficulty. Similar to badges, they help to give players a sense of accomplishment, but furthermore boost their experiences

[19] Geocachers utilize devices with *Global Positioning System* (GPS) navigational capabilities to find hidden caches or treasures that other people have hidden and marked on a shared map.

[20] Lydia Barbara referring to Jane McGonical, 2015: "Are Leaderboards Bad?", Part 2 (`http://tinyurl.com/qxrywdm`)

[21] Google Orkut was active from 2004-2014, `https://orkut.google.com`

[22] cp. Lydia Barbara, 2015: "Are Leaderboards Bad?", Part 6 (`http://tinyurl.com/jr6ozsr`)

in handling a system by gradually increasing provided challenges. These intensifications could happen in a linear fashion, i.e. with every level gained players are steadily facing more demanding tasks, but really should not, because, although it might be interesting for thoroughbred gamers, it quickly can frustrate casual players, who usually feel that the game early on requires too much of an effort[23]. Instead, exponential difficulty progression is a better solution, accommodating both kinds of players: easy enough for casual gamers to reach higher levels and sufficiently challenging for hardcore players later on when the exponential growth really kicks in. Beating such very hard levels can truly satisfy players and in a way make them feel special or part of an elite group that were able to succeed, while others were not [6]. Levels can also directly refer to skill levels, such as real life degrees awarded by universities, which, at least on paper, should account for a person's abilities in a field.

Considering the advantages, cleverly combined BLAPs may thus not seem such a bad idea and this is really reflected in today's abundance of applications constructed in such manner, but by opening up to a greater variety of elements, for example regarding some of the ones that have been listed in this section, gamified systems can tap into a more versatile, longer lasting potential and by that truly grow.

2.3 Gamification Types, Principles and Techniques

Having identified all the options for basic elements, it is possible to device strategies on how to apply them in a gamification process. There are three types of gamification that define the nature of a finished gamified system and have to be considered by designers before getting to work: *internal*, *external* and *behavior-change* gamification [16], illustrated in Figure 2.6.

An internal gamified activity is meant to be deployed inside a company for its benefit, be it to boost productivity, strengthen team spirit or achieving business goals in time. This gamification type's particular end users – the players – are mainly employees, which exhibit several qualities that must be kept in mind: they mostly know each other, are known by the company and, as common interests, should incorporate the organizational culture. This can be an advantage for gamification designers, but they also have to be careful when integrating new gamified applications, so as not to change current company strategies and practices. In fact, change, as is typically advised in process management [37], should always be introduced in small increments in order to ease the transition for employees. Internal gamification indicates really designing for any community, therefore

[23] Mike Lopez for Gamasutra, 2006: "Gameplay Design Fundamentals: Gameplay Progression" (`http://tinyurl.com/oqdagwc`)

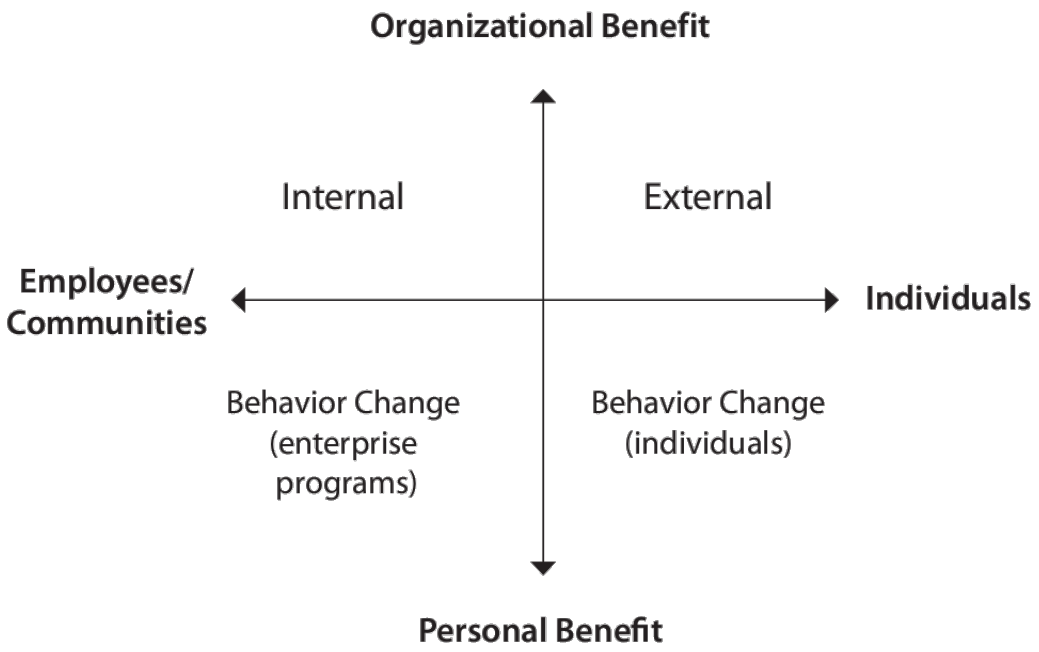


FIGURE 2.6: Gamification types and their relationships.
Source: [16]

it can also be attributed to methods like *crowdsourcing*[24], i.e. the company asks external users to perform desired tasks by themselves, but together regards them as a group[25].

The external gamification type is targeted at an organization's recurring customers, i.e. its user base, and potential ones. Possible pursued goals are usually – but not necessarily – marketing related: increasing sales, accumulating customers or data and research. Gamified applications can effectively arouse interest, strengthen loyalty and motivate people for engagement. Above all, the developed systems must provide a meaningful experience, hence playfully steer users towards the intended objectives while maintaining control – people will see right through poor design, take to their heels and never return.

The third and last type – behavior change gamification – is used to influence people's demeanor in a positive way. Whether this means to encourage them when they just need a little incentive to work out or help them to control their angers, it is often carried out by nonprofit organizations which can benefit from resulting behavioral changes but do not necessarily have to, as generally personal gain is a major driving factor in this kind of gamification. As Figure 2.6 shows it can be applied internally in teams as well as communities or externally addressing specific individuals.

[24]Renowned crowdsourcing example: people contribute to science by carrying out independent tasks in protein folding (`http://tinyurl.com/4ma8k8`).
[25]Kevin Werbach, 2013: "Gamification" Coursera course (`http://tinyurl.com/q4x8hld`)

Finally, for pre-estimating an application's category, i.e. its target audience, designers can try to place it on the chart of Figure 2.6, which brings it all together by putting organizational versus personal benefit in relation to communities versus individuals. Once this more or less easy orientational part is done, much harder tasks remain – more fine-grained definitions and selecting as well as applying the right gamification techniques. Following Subsections 2.3.1 to 2.3.4 will serve as a stepwise overview of a typical agile gamification development process.

2.3.1 Preparations

The often overlooked first step in gamification is of course deciding if it is actually feasible. Sometimes it does not really make sense to gamify – some good inital measures designers can take is to define overall objectives, describe players in detail and their expected behaviors [16].

Like in any software engineering process, all involved parties must understand and agree on the goals of the system: should it improve solidarity of employees, bring customers closer to a brand or help individuals achieve an innate wish of theirs? Even an effective gamified system can be a failure if the developers chose to gamify the wrong aspects of it. As it certainly is possible that objectives change throughout the whole process, they constantly need to be kept in mind and, if necessary, reworked. Preferably a list should be kept containing clearly defined goals, ranked by importance – a well constructed document only includes essential requirements, such that if ultimately only one thing on there would be realized, the application could anyway be considered a success [16].

Before implementing elements and mechanics into a system, players have to be accurately described. It makes sense to start with defining their *target behavior*, i.e. specific actions that will be performable in the final application. Their in-game execution should directly or indirectly support the previously defined goals and could therefore appropriately be included in the objectives list. Meaningful behavior examples would be "work out for 15 minues", "share a blog entry on facebook", "review an article" and so on. Afterwards system metrics, that either directly are used for feedback or only serve for background data analysis, should be appointed to the activities: for instance a user clicking the like button is awarded three points and commenting on a post is worth five. These measurements can also lead to people achieving "win states", which, however, should only be temporary and for that matter visibly declared as such, because both winning as well as losing persons are likely to think that the game subsequently is over, hence move on engaging in something different [16].

Precise player descriptions can furthermore help to get a better estimate of the player's motivations. That is to say, different user bases maintain different motives to participate and accordingly trying to predict that behavior can understandably have great advantages. Here, considering Bartle's various player types, which were explained in Section 2.1.2, will be beneficial and ensure that the system is attractive for a large audience. The easiest way continue with the design process is to create a stereotype user for each kind of player, which should optimally be described in a very detailed fashion – what are their conceptions of fun? What are they good at? What do they like or dislike? [16]. Eventually, it is time to think about their abilities at various skill levels, which, as outlined in Section 2.1.3, change over time, as consequently will their proficiencies in handling the system. Players should never be forced to undertake activities not suited for their capabilities – therefore, typical engagements of every stage should be clearly defined for each player stereotype, either truly resembling the described five level player progression or for the sake of simplicity merely containing its most significant evolutionary steps: novice, competent and expert.

2.3.2 The Perfect Game Design

As already mentioned in Section 2.2.2, starting out with basic reward based gamification elements like BLAPs might be a good idea, but this ultimately depends on the purpose of the system. For a short-term application with the sole purpose of teaching players some new valuable skill, such as proper winter tree pruning or gardening, those techniques perfectly suffice in order to provoke engagement of generally interested people. Then, after the ability is acquired, the system can be abandoned. The reason why this setting is well suited for BLAP gamification is precisely because rewards do not have to be given out over long periods of time – as soon as behavior reinforcement is introduced into a system it must be kept forever since people quickly getting used to it [6]. It of course makes a difference whether the rewards are real or virtual because solid gift obviously cost money, but also constantly adjusting an imaginary environment has its price, which manifests itself in maintenance effort: raising level caps, inventing new achievements, while balancing them with old ones, adjusting values of redeemable points and so on. Designers of a long-term application therefore must keep this in mind and ensure that enough resources can be allocated for keeping the deployed system fresh.

Commonly the bare minimum of elements for a gamified system is introducing one or several kinds of points that initially can convey which aspects are more important than others, create an economy or build trust, altogether helping developers to define and players to understand the purpose of the application. All the other BLAP elements, like progress bars, multiple achievements and leaderboards, can build on that, but as

a key consideration it should be noted that an excessive use of BLAP components can harm the system by impairing its fun factor, i.e. people's intrinsic desire to play [16]. This phenomenon is often described as the *overjustification effect* [38]: children, when accustomed to being rewarded for doing something, such as reading books or playing an instrument, will – maybe in spite of having fun at first – gradually see the activity as a means to get whatever is is given to them for "compensation". The analogy that can be drawn here is obvious: some people go to work only because they get said "compensation" – money. In both cases the "assigned tasks" are considered work – what else would they receive compensation for? Work may get less enjoyable by the day and soon the only good thing about it is the "light at the end of the tunnel" – the reward. if the reinforcement stops, they will very likely immediately stop engaging in the activity for good, that is unless at some point they manage to recover the fun they might have had prior to being conditioned. Additionally, as already mentioned in Section 2.1.1, there are other techniques employing extrinsic motivation that can produce negative effects – for instance variable ratio operant conditioning can easily trigger addiction and therefore, although perhaps being advantageous for evil companies, should never ever be the intention of gamification.

Nevertheless, as Kevin Werbach states [16], extrinsic motivation is definitely not something to avoid – for instance it is the only way to gamify tasks that never will be fun for anyone: collecting garbage, unpleasant medical examinations or filing taxes. In situations, where intrinsic motivation can be used, however, it can be favorable to only use rewards as support or fallback options, when everything else fails.

Choosing proper game mechanics is a tedious endeavour, especially since an abundance of them exist and were successfully applied. A sometimes very effective brainstorming technique to facilitate these decisions is to simply rely on chance, potentially letting the dice decide: for example, game designer Jon Radoff's list of 42 things that people enjoy ([39] – see Appendix A, Table A.2) could be used as a resource for randomly selecting game elements, as the various resulting combinations might greatly contribute to making a fresh and innovative new system. He also cross-tabs all mechanics to American psychologist Steven Reiss' widely accepted 16 basic human desires (see Appendix A, Table A.1), which can be helpful to determine if specific elements are suitable for selected target audiences.

When finished selecting all the initially employed game mechanics, designers can start describing a meaningful system. Scott Nicholson developed a theoretical framework with key words forming the acronym *RECIPE* [40], which is based on SDT and can be utilized for guidance when attempting this task:

Play: give the player freedom to explore and fail (within boundaries).

Exposition: create stories that are integrated with real world (also let users create their own).

Choice: power should be in the hands of the participants – allow customization, but don't offer too much choices.

Information: use game to allow people to learn more about real world context.

Engagement: encourage users to learn from others.

Reflection: assistance to find other interests and past experiences can deepen engagement.

2.3.3 Lifetime Loops and Cycles

Successful gamification systems usually engage users for long periods of time. This can only be accomplished by constantly providing the players with an incentive to play. Giving feedback can create such encouragement and is therefore a crucial factor to avoid dwindling player motivation throughout an applications lifecycle. Hence, It can be invaluable to devise a perpetual feedback cycle in order to keep users interested as well as engaged. Figure 2.7 depicts an engagement loop, outlining the relationships between feedback, motivation and action.

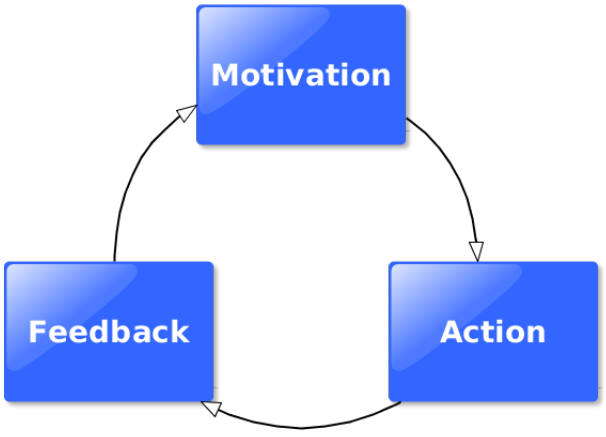


FIGURE 2.7: Engagement cycle. Source: based on [16].

Players perform actions, like completely filling out their social network profiles, if their motivations are strong enough, i.e. their efforts produce something of value to them – e.g. points, achievements or access to previously locked areas. These results in turn serve as motivation for subsequent actions and so on. Engagement loops are a fine-grained description of "what users do, why they do it and what the system does in response" [16].

Another important cycle is depicted in Figure 2.8, which shows a typical recommendation of player progression following a stair-like pattern.

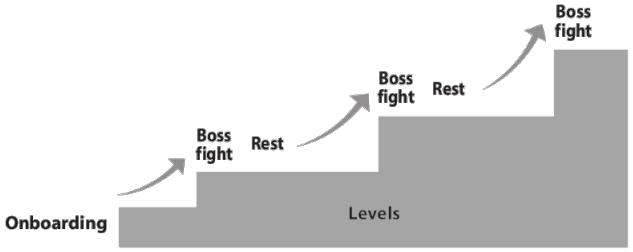


FIGURE 2.8: Progression Stairs. Source: [16].

As already pointed out in 2.2.2, game progression should occur in a non-linear fashion, which is reflected in the depicted stairs. The first step, *onboarding* or drawing the player attention for the first time, is critical because people usually decide very quickly whether they are going to try out a new system or not, i.e. they – depending on their personalities – are more or less used to making snap-decisions like in lots of other daily situations as is referred to in many studies and circumstances as Malcom Gladwell describes [41]. People should never lose in this initial phase and therefore need to be provided with enough guidance throughout their first interactions with the system. After this tutorial-like stage players move on to their first real challenge or *boss fight*, which initiates the beginning of a new, more difficult phase in the game. This cycle should repeat over and over always adjusting to the users' acquired abilities during their journeys. Progression stairs essentially represent an outline of a player's overall engagement with a system [16].

A player's performance in both cycles largely depends on how feedback is given – if properly applied people can even be steered towards certain outcomes: for example, car manufacturer FIAT's study on eco-driving interfaces, which display in-car information about how economically a person is driving, revealed that such kind of feedback positively influences peoples' driving behaviors, reducing their overall fuel consumption by up to 16% [42]. Following types of feedback generally have beneficial effects on players' motivation and behavior [16]:

Unexpected feedback: people like to be surprised, e.g. by not expected reward or achievements. These events raise morale, autonomy and intrinsic motivation. Engaging in a system becomes much more exiting using this kind of feedback, but at the same game designers should be careful, since it resembles variable-interval operant conditioning, which can be dangerous, as already mentioned.

Informational feedback: constant feedback on the users' performance can greatly encourage them to keep going. For instance, an incomplete progress bar can provide

the necessary motivation to finish a social network profile and thereby reaching 100% bar completion. This type of feedback is well suited for getting peoples' attention whenever an important event occurs.

Metric feedback: introducing metrics into the feedback can be a powerful way to highlight aspects that are most important in a system. If, for example, metrics about monthly sales are displayed, while customer satisfaction figures are not shown, users will rather steer towards increasing sales than to improve relations with customers. Therefore, designers should agree on a selection of metrics that best fit the purpose of the gamified application.

In all the considerations taken up until this point the fun factor should never be overlooked. Fun is the ultimate selling point of a gamified application and also the one thing that can separate it from thousands of others, hence should fill a central role in the design process. Game designer Nicole Lazzaro describes four types of fun that can be provided by a system, which target a variety of people [43]: *easy* as well as *hard fun*, *altered states* and *the people factor*. Easy fun emerges from casual entertainment, which does not require lots of thinking or commitment. Hard fun is experienced when attempting or solving a particularly challenging task, like a tricky sudoku puzzle. Games often allow the experience of altered states or a different realities, meaning that through playing people experience alternate emotions that produce soothing and satisfactory feelings. Lastly, the people factor as its name suggests indicates the joy of social interaction of any kind. Which types of fun should be employed heavily depends on the developed application and target users – ultimately, as it is hard to predict what people enjoy, iterative building, testing and refining will be necessary [16].

2.3.4 Development and Deployment

Having already made the majority of crucial decisions beforehand, the first implementation attempt should not be too difficult. As most gamification projects differs from one another and therefore almost certainly can not be perfect right away, they need to go through a refinement process, which – like in regular game design – revolves around building prototypes, testing them and making adjustments according to the test analysis. Besides the technologies to realize the system, certain people's assistance should never be missed [16]: persons understanding the system's business goals, capable designers and analytic experts, who can interpret the data produced. Furthermore special attention should be given to minimizing possibilities of exploitation because – as is the case in many games – if there is a possibility to bend the rules to one's advantage, people will generally take it, thereby greatly influencing or ruining the balance of the system [6].

Finally even at which point in time an initial deployment of the system should occur must be considered carefully, since an at first unfinished or uneven application leaves a mark on its target audience for future installments.

2.4 Summary

Gamification is about making unenjoyable tasks enjoyable. Considering human motivational factors, skill mastering theories and using game design elements, its intention is to create fun, long-lived experiences that improve regularly faced boring or repetitive situations. It is similar to game design but differs in its field of application: real-world activities. Gamification can raise productivity, performance or engagement but should not enforce them. Its iterative design process involves a rigorous requirements analysis, lifetime cycle layout, prototyping, testing and constant refining. A well-designed gamified system is the key to truly engaging a specific kind of audience in certain tasks, preferably appealing to people's self-motivation rather than using external rewards to provoke genuine interest.

Chapter 3

Related Work

Exciting new trends, especially when they are as promising as gamification is made out to be, naturally arouse a lot of interest, inspire people's work and oftentimes even become movements with many engaged supporters. Gamification's evolution is said to already be past the initial hype accompanying any bigger newfound prospect and currently is in the phase of gaining ever more followers, while being regarded with increasing sophistication[1]. When considering its field of application – "non-game contexts" – it becomes apparent that it basically can be utilized in any kind of situation, which is reflected in the great variety and wide range of related work on the subject. Therefore, this section can merely give a small overview on the topic by outlining selected real world scenarios as well as a part of its adoption in academia.

In reference to Section 2.3, the work cited here will roughly be organized into internal, external and behavior change gamification (Sections 3.2, 3.3, 3.4). Additionally, the preceding special Section 3.1 will be concerned with proposed general approaches that are not limited to specific contexts when applied. Finally, despite gamification for fitness improvement can be attributed to the behavior changing kind, concluding Section 3.5 is purely dedicated to it, as it is most closely related to the studies presented in this thesis.

3.1 General Strategies

Many strategies and best practices have been developed in order to improve upon the current conception of gamification and even more will follow, as it is a fairly fresh topic, still quickly sprouting new findings. A few of these strategies have already been outlined in Chapter 2, most notably the approaches of Gabe Zichermann, Kevin Werbach and

[1] Andrzej Marczewski on `gamified.uk`, August 2015: "The Hype is Over – Gamification is Here to Stay" (`http://tinyurl.com/heqrejw`)

Scott Nicholson [6, 16, 40]. Still, lots of interesting ideas can be found by perusing through the massive corpus of publications.

Gamification consultant and author of the recent book "Even Ninja Monkeys Like To Play: Gamification, Game Thinking and Motivational Design" [44] Andrzej Marczewski proposes a simple framework with the acronym *GAME*, defining four steps towards gamifying a system[2]: Gather, Act, Measure and Enrich. The first phase is dedicated to gathering information and recommends that designers should contemplate *what* as well as *why* to gamify, for *whom* and *how* to measure success. This accumulated information serves as input for the second phase, where action is taken by designing a fun solution in every involved party's best interest. When the resulting product is deployed, the measuring phase is used to monitor the users' behaviors in regard to the system's goals, which yields ideas for enrichments that are applied in the fourth phase. The last two steps should constantly be repeated in order to iteratively improve the solution and always keep up with the users' needs. Measurements currently are mostly part of a manual process but could very well be automated in the future, as the analyses of Benjamin Heilbrunn *et al.* point out [45]. Certainly all of these steps are very elementary, lacking any concrete processes, but they are particularly easy to remember and thus at any time offer useful points of reference during development.

Philipp Herzig *et al.* developed a gamification modeling language (*GaML*) for the purpose of improving the concept creation process when gamifying a system [46, 47]. They criticize natural language communication between non tech-savvy designers and developers being too imprecise, often unnecessarily impairing implementations – a more formal approach is needed: by using a gamification taxonomy proposed by Deterding *et al.* [48] GaML poses a platform-independent, verifiable way of precisely describing a gamified system, bridging the gap between domain and IT experts. Of course this early approach of formalizing gamification is limited to its incorporated elements – points, levels, roles, events etc., but could become a valuable tool for designers, as it is still in ongoing development.

Since many gamification strategies still are generalized and do not distinguish between users on an individual level, it is important to consider how these approaches can be optimized using adaptive measures. David Codish *et al.* develop a personality-based model that should eventually adjusts to players' preferences automatically as they use a system [49–51]. They argue that personality plays a large role in the individually perceived playfulness of a system, which they attempt to prove with a series of studies. Preliminary results revealed, for example, that introverted persons favor badges more than extroverted people, while agreeable individuals tend to value progress feedback.

[2] Andrzej Marczewski on `gamified.uk`, 2014: "GAME: A design process framework" (`http://tinyurl.com/h2zv7fr`).

Their framework should interweave such findings into the MDA model, which has been referred to in previous Section 2.2.1. Personalization is also a highly popular research subject in educational gamification because learning is a very individual process, in which preferences as well as speed greatly differs between students [52–54].

Generalized recipes for gamification provide desirable guidelines when approaching new projects. However, Juho Hamari and colleague's often cited study evaluating current publications has revealed that successful gamification is highly context- and user-dependent [55]. Moreover, the term is used very inconsistently across literature, generally too little empirical evidence is produced to back claimed theories [56] and it lacks proper tools to ensure software quality [57]. To address these deficiencies much more work needs to be done, especially developing a broader multidisciplinary view on the subject in combination with exploring a greater variety of game elements across different contexts [56].

3.2 Internal Gamification

As already mentioned, the majority of modern organizations incorporate at least one gamified application. Some simply use generic platforms[3] to gamify their workflow or employ specialized solutions to tackle particular kinds of problems like improving team building[4], increasing employee engagement[5] and motivation[6]. Others develop their own custom solutions: for example, Microsoft's software testing group turned proof-reading Windows dialog-box translations[7] into a game[8], successfully engaging 4 500 employees from all over the world to voluntarily participate and fix errors, while individual company branches competed against each other on global leaderboards. Research of the scientific community is mostly focused on either studying psychological or behavioral outcomes of applied gamification [55], especially in the popular crowdsourcing sector.

By splitting up problems into small chunks and wrapping them into fun micro-tasks, there is a good chance that crowdsourcing helps to better understand corresponding subjects or deliver remarkable results, which has been utilized in a large variety of areas: video, image or document retrieval [59–65], multimedia synchronization [66], relevance assessment [67] and pro-environmental activism [68], just to name a few.

[3] E.g. Badgeville (`https://badgeville.com`), Bunchball (`https://www.bunchball.com`), Playlyfe (`https://playlyfe.com`) or [58].
[4] E.g. Loquiz (`http://loquiz.com`).
[5] E.g. Bluewolf (`http://www.bluewolf.com`) or Proof (`http://www.mindbloom.com/proof`).
[6] E.g. ChoreWars (`http://www.chorewars.com`).
[7] Microsoft Windows ships in over 100 languages, making localization a very hard task.
[8] "The Language Quality Game" on Microsoft's TechNet: `http://tinyurl.com/jkkyhv9`.

Another significant amount of research in this section's context is concerned with intra-organizational gamification, predominantly focusing on internal social network services featuring extrinsic rewards [69, 70]. Studies examining the effects of such systems on employees discovered an increase in participation as points, badges, leaderboards etc. were introduced [71–73]. Similar positive impacts also have been reported for enterprise collaboration systems [74, 75] as well as in employee training [76–78], including the broad field of simulation gaming [79].

Lastly, as employee health is closely related to their performance[9], organizations are starting to introduce gamification as part of their health initiatives [80], which is an important measure to increase the fun factor of such programs[10]. Several existing platforms like Keas, Fitbit or Validic[11] offer gamified services in order to provide information as well as challenges for dieting, exercise and general wellbeing.

3.3 External Gamification

The abundance of shining examples in the consumer market has arguably been the reason that initiated the gamification hype a few years ago [81]. Foursquare[12] is considered among the most famous gamification services that increase user engagement: people are given digital rewards for visiting known local businesses and completing challenges. Furthermore, users can rate as well as comment on these places and interact with other Foursquare members. Since online communities like this or social networks in general can not exist without the participation of their subscribers, it is only natural that they maximize their chances using gamification, for example by using badges, profile completion bars, public display of post counts and so on. Therefore, these services can be considered the topmost applications of external gamification.

From an academic point of view, popular online social communities, due to their large user bases, are perfect for conducting research. Comparable to intra-organizational networks, several studies report positive effects such as increased system and user interaction [82–84]. Frith *et al.*, when investigating Foursquare, even discovered that some people resort to cheating by artificially changing their IP addresses in order to get rewards more quickly [85], again highlighting the effectiveness of introducing game elements. However, a big problem in gamified applications used by a massive amount of users is keeping their

[9] Lisa Evans for Entrepreneur, 2013: "What 3 Companies Are Doing to Keep Employees Healthy" (`http://tinyurl.com/henyllk`).

[10] cf. Karsten Strauss commenting on the "2013 Global Workplace Health and Wellness Report" (`http://tinyurl.com/jv33ucj`) for Forbes, 2013: "Employees Don't Have Time For Wellness Initiatives – REPORT" (`http://tinyurl.com/jh7wqzu`).

[11] Keas (`http://keas.com`), Fitbit (`https://www.fitbit.com`), Validic (`https://validic.com`).

[12] `https://foursquare.com`

engagement in the long term: after all achievements have been collected, no game is left to be played and applications can become stale. For instance, Foursquare, launched in 2009, is meanwhile struggling to maintain its users, despite the company's attempts to redefine the service in 2014[13]. Such developments have also been discovered in several studies, which suggest that specific gamified applications may initially be perceived as novelties and thus only experiencing temporary success [71, 72, 86, 87]. Other than general strategies[14], very little is published to address this issue, which may be due to gamification still being a very young subject – some characterize most current naive applications with the term "Gamification 1.0" and speak of a slow ongoing transition to a more sustainable version 2.0 [88].

Within this section's context, another largely popular field of application is gamification for educational purposes, which from a general point of view has already been employed for centuries in one or the other form: for instance, the student grading system is a kind of points awarding scheme that goes as far back as 1792[15] and different academic years can be regarded as levels. However, many of such traditional elements are considered bad gamification, as they fail to engage the majority of students properly[16]. Of course, throughout history there were many attempts to improve upon teaching methods, like for example learning literature that included game elements[17] or flash cards to aid memorization. Still, nowadays technology enables much more versatility: the use of gamified online learning platforms as well as countless educational software products or games[18]. Scientific case studies evaluating the performance of students subjected to gamified curriculums showed remarkable results such as improved interaction with the teaching staff, higher percentages of finished assignments, which also happened in a more timely manner than usual and much higher engagement due to the introduction of competitive elements [91–94]. Although the outcomes were overall positive, there also were issues encountered, confirming that applying gamification indeed should be well thought through [91]: game elements caused unwanted behavior [94], competition was not well received [95] and study effects on students were marginal or even unclear [96, 97].

[13] Michael Carney for Pando, 2015: "Sad Foursquare :(" (`http://tinyurl.com/j3zbsoe`)

[14] cf. Victor Manrique on `gamification.co`, 2014: "5 Key Design Factors for Long-Term Engagement with Gamification" (`http://tinyurl.com/kev82bz`)

[15] cf. excerpt from Thom Hartmann's "Complete Guide to ADHD" [89] on Joe Bower's blog, 2012: "A short history of grading" (`http://tinyurl.com/ab9cmg3`).

[16] cf. Sarah Smith-Robbins on `educause.edu`, 2011: "'This Game Sucks': How to Improve the Gamification of Education" (`http://tinyurl.com/z7vfg8m`).

[17] E.g. "K's first case" by L.G. Alexander and John Holder, published in 1975 [90].

[18] E.g. Growth Engineering LMS (`http://tinyurl.com/gtxtjz6`), Prodigy (`https://prodigygame.com`), Memrise (`http://www.memrise.com`) or Mathbreakers (`http://tinyurl.com/zt9gm3k`).

3.4 Behavior Change Gamification

In a sense, all gamification is utilized to change behavior, even when a gamified system merely ought to increase player engagement because such an effect involves people acting differently than they normally would, hence, changing their behavior. However, although overlapping with some of the aspects mentioned in the other sections[19], the focus here is set on changing the players' behaviors or their attitudes towards something *in the long run*, perhaps even beyond actively using a gamified application. This is as well strongly related to educational gamification because altering habits oftentimes requires learning something new or at least receiving specific insights into various corresponding topics – for instance, encouraging people to eat healthier involves teaching them about food. Nevertheless, as already outlined, behavior also can be influenced by simply making things fun to do – "The Fun Theory" competition, previously mentioned in Chapter 1, brought up many great ideas to support this belief: for example, a piano staircase playing musical notes on every step convinces people to choose a healthier way of ascending from a subway station than using an escalator or a bottle bank turned into an arcade machine shows that it can be fun to recycle[20]. Accordingly, new prospects arise for bettering ubiquitous problems such as environmental pollution, rendering research in this area particularly important.

In order to motivate people for reducing CO_2 emissions, Liu *et al.* developed EcoIsland, a gamified monitoring system deployed at peoples' homes giving them visual feedback about how sustainable all household members behave in their daily lives [59]. A study with six voluntary participating families was conducted and the majority of persons stated that they had been positively influenced, although a quantitative data analysis did not show any significance, which maybe has been due to the small scale of the experiment. Keeping an equivalent goal in mind, i.e. reducing emissions, while additionally saving fuel, some car manufacturers have developed eco driving assistant systems, which have been used effectively, at least for short time periods [42]. Ohad Inbar *et al.* suggest a future gamification-oriented research plan, in order to maximize the persuasiveness of such systems, implementing rewards as well as social aspects [98].

Other work concerned with long-term behavior changing gamification aspects includes using mobile technology for motivating people to reduce their energy consumption needs [99], raising personnel efficiency for emergency situations like fire alarms [100], increasing

[19] See also Figure 2.6.

[20] Piano Staircase (`http://tinyurl.com/yck5dsj`), Bottle Bank Arcade (`http://tinyurl.com/ygulvm3`).

children's financial conceptions[21] and many more topics, including the substantial health-related area. A prominent subdivision of this field – gamification for fitness – is of particular relevance for this thesis and will therefore be discussed in the next section.

3.5 Gamification and Fitness

It is common knowledge that physical activity is essential for maintaining a healthy body and studies confirm that it is a crucial factor in preventing chronic disease as well as premature death [8]. However, modern society struggles in finding the motivation to work out regularly: according to statistics collected by the US *Centers for Disease Control and Prevention* (CDC) only 21% of all American adults meet the recommended levels of physical exercise[22] and similarly the WHO estimates merely about 31% of grown-ups meeting said requirements in Europe [101]. Apart from motivational media that can be viewed or obtained everywhere, technology also brought fairly new ways of attempting to change this undesirable situation – fitness-related applications and games, which are the focus of following two subsections.

3.5.1 Fitness Applications

Whether they are used by fitness enthusiasts or hobbyists, devices able to track physical activities are becoming common accessories. They can record how many steps are made during a day, which jogging routes a person takes, what the highest heart rate is while working out and much more. Most of the software using this captured data can be found online or in the form of mobile apps – by now even apps exist that tie together others, like for example Apple Health or Google Fit[23], which offer central dashboards for a great variety of gadgets and applications. Gamification is what connects almost all of these tools: points can be acquired, challenges completed, progress shared with others, all while chasing one's own personal fitness goals. Table 3.1 shows a selection of currently popular devices, platforms and mobile apps.

Table 3.1: Popular fitness devices, platforms and mobile apps.

	purpose	name	URL
devices	activity tracking	Jawbone UP	`jawbone.com/up`
	activity tracking	Fitbit	`fitbit.com`
	activity tracking	Nike+ FuelBand	`tinyurl.com/op5f9yq`

[21] PlayMoolah platform – a financial empowerment initiative `http://tinyurl.com/hfcvhvq`.
[22] cf. CDC web page, 2014: "Facts about Physical Activity" (http://tinyurl.com/zlxvnxo).
[23] Apple Health (`http://www.apple.com/ios/health`), Google Fit (`https://fit.google.com`).

	social wellness	Keas	keas.com
platforms	social wellness	Fitocrazy	fitocracy.com
	activity tracking	MapMyFitness	mapmyfitness.com
	activity tracking	Runkeeper	runkeeper.com
	activity tracking	Garmin Connect	tinyurl.com/j5z89k6
	activity tracking	Endomondo	tinyurl.com/ja33wnr
apps	daily exercise	7 Minute Workout	tinyurl.com/jo3at2u
	running	Nike+ Running	tinyurl.com/jpwk3uk
	running	Runtastic	tinyurl.com/jyqr7g5

All mentioned devices typically are worn around the wrist or clipped onto pieces of clothing and, besides collecting information, many of them can remind their users to start moving after long periods of inactivity on a daily basis. Transferring data to mobile phone apps or desktop PCs usually is accomplished via Bluetooth or USB port. Platforms normally do provide mobile apps, although in some cases they lack refinement, as becomes apparent when regarding their poor reviews and ratings. Furthermore, nearly all apps on the market are of sub-standard quality when regarded from a fitness coaching point of view: a recent study compared the content of 30 popular exercise prescriptive mobile apps against the recommended *American College of Sports Medicine* (ACSM) guidelines [102] and found that 29 apps scored below 40% when matched against a set of key criteria points in the categories of aerobic, strength/resistance and flexibility training [103]. Moreover, in general lower priced or free versions of health-related software products tend to be of lesser quality than more expensive ones, as Joshua West *et al.* confirm after evaluating 3336 relevant apps [104]. All of these results indicate that there is much room for improvement content-wise, which, however, is neglectable from a gamification perspective.

Cameron Lister *et al.* reviewed the use of gamification, game elements and health behavior theory constructs in 132 Apple iPhone fit or wellness apps, each of which incorporating at least one of the following elements: leaderboards, levels, digital rewards, real world prizes, challenges, social or peer pressure [105]. Results showed that gamification is correlated to both game elements and health-related behavior, albeit for the latter only in a motivational sense, merely comprising 5 out of 13 identified core behavior constructs. Thus, they revealed that apparently gamification in health-targeted apps currently disregards the majority of behavioral aspects, which either fall into the category of psychological and performance-related capabilites or social and physical behavioral triggers. Additionally, on average less than 50% of either gamification, game or behavioral components were integrated, which highlights the lack of an industry standard for this specific software

category. Therefore, besides ideally improving upon their content, future applications can also profoundly increase their user impact by benefiting from both behavioral theory and gamification.

Other related research indicates an overall positive effect of gamification on physical activity – in particular, studies revealed that several social aspects can additionally increase the motivation to exercise [106–109]. Nevertheless, gamified health-related systems of course exhibit general shortcomings such as the previously mentioned novelty effect causing people's engagement to fade over time and the lack of custom-fit in-app personalization.

3.5.2 Fitness Games

Fitness or exergames can be regarded as special cases of gamification[24] because they motivate users for working out by using full-fledged games rather than restricting themselves to selected game elements. Generally, a clear distinction between such games with a purpose (GWAPs) and gamified applications is made, but it is quite possible that the line drawn between the two becomes blurred: for example, the popular GPS-based running app "Zombies, Run!"[25] immerses players in a post-apocalyptic zombie scenario, requiring them to complete narrated missions, while avoiding hordes of zombies. It is basically an exergame but also a gamified fitness tracker because, just like other running apps, it provides its users with all kinds of statistics about their present and past performances – stimuli for continued engagement. Given the game's enormous success of reaching over 1 million sales since its launch in 2012[26] and also the fact that other companies have already started creating their own fitness adventure experiences[27], it is very likely that this trend of merging games with gamified apps will be explored further in the near future.

Long-term benefits from fitness games can only be achieved if people have access to them whenever they like, which of course makes smartphones very attractive mediums for their distribution but also Computers and home gaming consoles: nowadays most of them support motion capturing devices and additionally a variety of attachable fitness gadgets is available. In fact, ever since digital home entertainment ended the golden age of the arcade industry[28], there have been many attempts to make gaming a more

[24] cf. [16]

[25] `https://zombiesrungame.com`

[26] Stuart Dredge for The Guardian, 2015: "Zombies, Run! goes freemium after 1m sales to attract hordes of new players" (`http://tinyurl.com/lfaqbbg`).

[27] E.g. "BattleSuit Runner Fitness" (`http://tinyurl.com/jcantuq`) or "The Walk" (`http://tinyurl.com/zdhenbl`).

[28] cf. Laura June for The Verge, 2013: "For Amusement Only: the life and death of the American arcade" (`http://tinyurl.com/aawzxev`).

immersive experience and encouraging physical activity: pressure-sensitive floor pads allowing players to control games with their feet were among the earliest devices to enable exercise-oriented gaming [10]. Today, many options for indoor as well as outdoor exergaming are available (see Table 3.2 listing selected titles), even platform-independent custom solutions, which can be very expensive, depending on the included fitness device type[29].

TABLE 3.2: Past and present popular fitness games, in descending order by first release.

name	release	platforms	description
Ingress	2012	smartphones	*Augmented Reality* (AR) location-based multi-player game that requires users to capture landmarks by physically moving there.
Zumba Fitness	2010	consoles	Exercises based on the Zumba fitness program[30], controlled via motion controllers or Kinect.
Kinect Adventures	2010	console (XBox)	Collection of minigames played by using Kinect-captured body motion.
Just Dance	2009	consoles	Players use a motion controller to follow specific dance routines and are judged by their performance.
EA Sports Active	2009	consoles	Provides custom workout and is played using motion controllers and optionally the balance board.
Personal Trainer: Walking	2009	handheld (Nintendo DS)	Walking / Running Tracker with unlockable minigames.
Wii Sports / Fit	2006 / 2007	console (Wii)	Sports games and fitness program controlled via motion controllers and/or additional balance board.
Gamercize	2006	consoles, PC	Mini stepper connecting to a variety of devices with simple concept: it pauses games or disables inputs (joystick, mouse, ...) when workout is stopped.
EyeToy: Kinetic	2005	console (Playstation)	Exergames making use of the EyeToy camera utilizing gesture recognition to track players' motions.

[29] E.g. Exergame Fitness devices `http://tinyurl.com/jd363ub`.
[30] `https://www.zumba.com`

Yourself!Fitness / My Fitness Coach	2004	consoles	Personalized fitness program with 500+ exercises, some intended for use with workout ball, weights etc.
Dance Dance Revolution / Dancing Stage	1998	Arcade, consoles	Dance game requiring players to follow an on-screen dance routine with precise timing.
Virtual Biking (various games)	1982	arcade, consoles, tablets	Various ergometer controlled biking / racing games.

Most popular fitness games are part of series that are still ongoing to this day. It is noteworthy that all of those games specifically are tailored for exercising with various fitness devices or motion controllers and no attempts have been made to integrate exercising into games that were made without physical activity in mind, let alone into AAA games. Such experiments only seem to be undertaken by hobby craftsmen at the moment[31]. However, as *virtual reality* (VR) equipment like the Oculus Rift[32] goggles are on the verge of becoming publicly available, steps are made towards increasing player immersion by devising novel game input methods: for example, Virtuix[33] develop a virtual reality motion platform that lets players wander through the worlds of their favorite 3D games using their own feet. Similarly, VirZoom[34] uses an ergometer to achieve said effect, although currently only with custom made games. Using these devices of course does not necessarily implicate conducting a thorough workout, but definitely increases the amount of physical activity involved when playing games.

Many academic studies examine the effect of exergames on people's *energy expenditure* (EE), a measure that can be increased by physical activity and should not be exceeded by dietary energy intake, in order to prevent gaining weight. Jennifer Sween *et al.*, studying 27 papers on this subject, found a strong correlation between fitness games and EE [110]. Most of the research they investigated discovered that introducing such games resulted in test subjects meeting the ACSM guidelines for health and fitness. The games used for the studies included interactive aerobic, cycling, motion capture, dance simulation as well as isometric resistance titles of all current major consoles – Microsoft XBox, Nintendo Wii and Sony Playstation. Although dance simulation games caused the most significant EE increases of about three times of a person's resting expenditure, not a single study reported any inconclusive or negative results. Furthermore, similar

[31] E.g. ManaEnergyPotion on Instructables, 2009: "Hooking Up A Treadmill to PC Game" (`http://tinyurl.com/zd2s36f`).
[32] `https://www.oculus.com/en-us/rift`
[33] `http://www.virtuix.com`
[34] `http://www.virzoom.com`

results were discovered in a literature review targeting the physical effect of exergames in healthy elderly [111]. Such outcomes on one hand emphasize the potential of fitness games, but on the other hand do not account for their ability to change people's attitude towards exercising.

Scientific studies examining fitness games' effects on user behavior, depending on their research objectives, either use commercial exergames or create custom solutions. Conclusions are of mixed nature – partly positive [112–116], indicating an increase in physical activity, and partly inconclusive, reporting that no significant behavioral change can be identified [117–119]. Furthermore, in addition to questionable incentive for long-term engagement, the games often simply lack fun [112, 120], which can be addressed using a variety of approaches: psychological or physiological considerations [120], increasing the level of perceived accomplishment [121], introducing social and narrative components [122, 123] etc.

Work most closely related to the projects presented in this thesis comprises user studies that investigate the effects of home fitness devices connected to exergames. Darren Warburton *et al.* investigated 14 low-active young males partaking in a specifically devised exercise program over a 6 week period. They discovered a 30% higher attendance to ergometer training sessions when video games were involved as opposed to exercising the traditional way [124]. Ahn and colleagues turned a treadmill into an interactive running entertainment system, featuring a built-in computer, user-wearable sensor bracelets and an ultrasonic sensor for tracking a players position [125, 126]. They further developed "Swan Boat", a competitive team-based multiplayer racing game, in order to evaluate the entertainment system. A two-week study with 12 participants revealed an overall positive reception, beneficial social effects due to the necessity of team synchronization and increased engagement caused by high competition between teams. Yue Gao *et al.* also utilized a treadmill as well as a Kinect camera with a casual exergame to confirm that short exercise breaks throughout the day yield temporary benefits of cognitive nature, such as improved concentration [127]. Göbel and colleagues developed three exergames aimed at improving the long-term motivation of players: an ergometer controlled set of minigames incorporating vital signals from players as well as two other games tracking user movement via accelerometers and video recognition, respectively [128]. All captured sensor information directly influenced the gameplay, which generally had a positive impact on player motivation and experience, as they found in various focus group tests. Finally, Hoda *at al.* evaluated how the user experience in ergometer-based exercise is dependent on the application context [129]. They examined 20 participants in five different cycling settings: no addition, game, TV, game plus TV and game plus movie. Results showed that integrating the workout into a game has been the most encouraging and motivational environment, while biking without any entertainment was mostly considered tedious.

Some of the casual exergames developed for studies are similar to the Flappy Bird based project presented in this thesis: for example, Göbel and colleages' "Pigeon Hunt" uses the pedaling speed of an ergometer to control the height of a pigeon collecting reward letters [128]. Although there were some shooting type games, involving the use of buttons, sensors or hand gestures to hit targets[35], no true FPS has been part of any discovered research. Furthermore, as already outlined, none of the studies investigating exergames experimented with AAA games in any way.

3.6 Summary

Gamification is used and researched in a variety of fields, including crowdsourcing, information retrieval, training/education and sustainability. Countless reports underline its effectiveness for many purposes like improving user engagement, learning or raising awareness. However, they also point out that its application is very context-specific, hence, should carefully be thought through. Often other shortcomings such as lack of personalization and failure to achieve long-term user engagement are identified. Gamification for health and fitness is an emerging trend that is frequently incorporated into smartphone apps or Internet portals. Games for fitness are closely related to those applications as they share the same goal: motivating people for a healthier lifestyle. This is accomplished by integrating exercise into games, a concept that is widely adopted in development for home gaming consoles and scientific research. Studies showed increased workout motivation when games are involved, but also yielded inconclusive results in terms of significant behavior change. Additionally, introducing vital sensors seems to have a positive impact on user enjoyment. Although the games of some studies were similar to the casual game presented in this thesis, no AAA games have yet been utilized for exergaming research.

[35] cf. ergometer and button controlled duck shooter [129], handheld sensor targeting in "SunSportsGo" [128] or hand gesture based target shooting [130].

Chapter 4

Formulation of Objectives

By now fitness games are ubiquitously available and their substantial estimated market share of over 6.6 billion dollars highlights a general public interest in purchasing them [131]. However, in spite of those facts, the buyers' initial excitement often quickly fades and widely promoted beneficial effects like changing peoples' physical exercise habits are limited to a short period of time, after which the games are shelved[1], making room for new, more exciting games. Several studies underline that notion of rapidly declining interest [113, 119, 120], emphasizing that ultimately fun plays a vital role in any game – a game lacking fun becomes just a plain task. As previously mentioned, AAA games compared to fitness games do a much better job at entertaining players, as they often bear high replay values, feature the latest advanced technologies and are generally designed much more thoughtfully. This of course turns them into the most attractive and anticipated games on the market. Furthermore, since nowadays good games are far from being in short supply, gamers also have to be very selective in choosing which games to play, not least if they like playing a variety of different game genres. Often therefore, players have to devise their own gaming schedule: naturally very little time is spent on playing exergames, while a lot more on popular ones that are hyped by magazines or online review sites. Lastly, as hinted in the introduction, fitness games are often controlled by platform specific devices that greatly restrict their portability – frequently equipment even needs to be bought on a game-to-game basis regardless of the platform used. Such additional devices can cost lots of money and on top of that have little use without their corresponding games.

The problem with most of today's fitness games as well as apps becomes even clearer when regarding their common purpose: gamifying sports strongly advocates a focus on physical activity – fun games are not really fixated on anything other than bringing joy

[1] Michael Rosenberg for The Conversation, 2014: "Sorry gamers, Wii Fit is no substitute for real exercise" (`http://tinyurl.com/pzyopx8`)

and satisfaction to their players. Therefore, a much more interesting approach would be to shift said focus onto the games and enhancing *them* with fitness elements, i.e. applying *fitnessification*[2]. This thesis should be regarded as an impulse to support such a notion, which shall be verified by evaluating all related investigations.

During the course of this thesis, two studies were conducted with the purpose of on one hand confirming as well as elaborating on the above mentioned tendencies and on the other hand studying the effects of using a novel approach: fitnessifying a AAA game. Following Section 4.1 explains the general rationale behind the thesis' idea and in 4.2 research questions are formulated.

4.1 General Idea

The original idea for studying the user experience effects of two comparable fitness gamifcation projects was born out of the notion that exercising should be part of any healthy human being's life and games are very powerful motivators. Both projects should feature a common home fitness device – an ergometer – used to control games hooked up to it. This seems by no means an entirely new endeavour – countless studies like this have already been conducted and even similar commercial products are available. The key difference, however, is that one of the used games shall be a modded version of a AAA game, which seemingly has not yet been part of any similar studies (see Chapter 3 for more information about related work). The resulting findings of this novel approach should afterwards be compared to study outcomes from the other project, which will resemble the most prevalent method of gamifying exercise: creating a game specifically for the utilized fitness device.

The studies have been funded and contribute to the CROSMOS Project[3], which was launched with the key purpose of understanding user interaction and behavior in mixed-reality environments in order to improve design as well as performance of corresponding applications. The modus operandi of both user experience analyses was kept as similar as possible: in a supervised field study, the participants were given sufficient instructions to be able to try out the finished projects for as long as they liked[4] and subsequently were asked to fill out a questionnaire about their experiences (see Chapters 5 and 6 for more details).

[2]The term 'fitnessification' in this context was coined by Mathias Lux in 2016.

[3]CROSMOS: Cooperation, Resource-Optimization and Self-Organization in Mobile, Mixed-Reality Environments (`http://crosmos.aau.at`)

[4]A nonbindung suggestion of five minutes was given every time.

4.2 Research Questions and Hypotheses

As stated above, the main purpose of this thesis is to show that fitness-enhanced AAA games are more enjoyable than games specifically designed for supporting physical exercise. Hence, the main evaluation problem **P** can be summarized as follows:

P Fitnessification of games encourages more positive feedback than gamification of sports.

Based on this goal, several research questions can be developed, summarizing the purpose of this thesis as well as related future aspects:

Q1 How do (sportive) people feel about exergaming?

Q2 Can exergaming be more fun?

Q3 Given the AAA game integration, would it be possible to motivate people to work out more?

Q4 Which conditions must be satisfied for gamers to consider exergaming at home?

As a final specification, six hypotheses are defined that should either be proved or refuted, representing the main evaluation goals of the studies and addressed in concluding Chapters 6 and 7:

H1 Gamers would rather play AAA exergames than casual exergames at home.

H2 Active[5] people would rather play AAA exergames than casual exergames at home.

H3 Regular gamers that are less active would rather consider working out more playing AAA exergames than casual exergames.

H4 Active people would rather play exergames at home than rarely active people.

H5 Less active people would rather consider working out more by playing exergames than regularly active persons.

H6 Regularly active occasional gamers would rather play AAA exergames than casual exergames at home.

Several additional hypotheses, emerging from post-study analyses will furthermore be presented and discussed in Chapter 6.

[5]Active refers to practicing sports in all hypotheses.

Chapter 5

User Studies: Overview and Implementation

User studies and their analyses represent the core of this thesis and in order to collect a proper amount of data for evaluating aforementioned research objectives, two user experiments were conducted, which will be described in this chapter. Each of the studies involved participants controlling an exergame via a fitness device and accounting for their experiences in doing so. To identify and resolve shortcomings in the projects before undertaking the main inquiries, the experiments were preceded by preliminary tests.

Structurally, this chapter is divided into three parts. An overview (Section 5.1) outlines the projects' general architecture and the utilized hardware. Section 5.2 gives a detailed account for methodology, implementation and setup of the first study, which revolved around a casual game named "Flappy Cycling". Similarly, the subsequent Section 5.3 will portray these aspects for the second experiment: a modified AAA game with the title "Quake 3 Arena Fitness Edition".

5.1 Overview

A general requirement of both projects is to convey health information to a personal computer. Figure 5.1 shows an early study-independent architecture sketch, with the purpose of clarifying the general employed HCI.

Participants should control games by using conventional input methods, like joysticks, but as well via performance data captured from a sports device, for example information about the current running speed on a treadmill. The sports device used in all studies

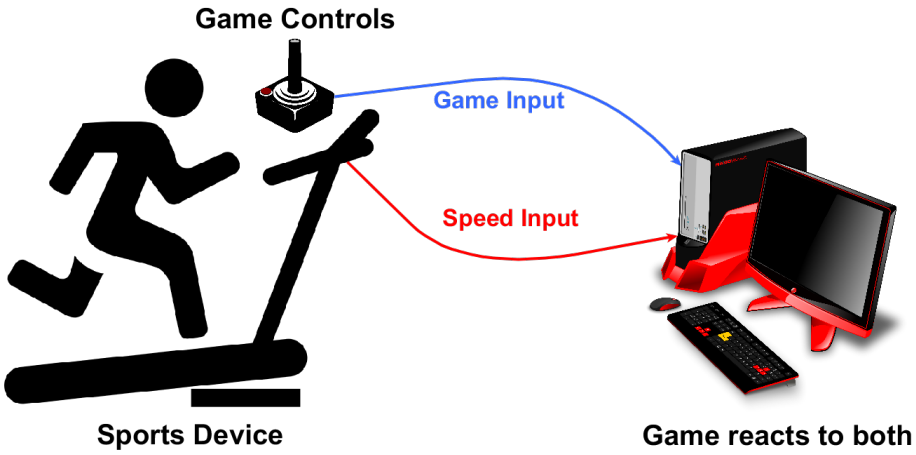


FIGURE 5.1: General Study Architecture.

has been, as already mentioned, an ergometer[1], which provides two kinds of data – the cycling rate and pulse sensor information via two attached handles (see Figure 5.2a). To be able to properly capture signals from the device, an *Arduino Leonardo*[2] was used, which is a powerful, programmable open-source microcontroller board (see Figure 5.2b). Additionally in the final user study a separate pulse sensor[3] was employed (see Figure 5.2c), as to enable controlling a game with a joystick while measuring the user's heart rate through merely applying a finger strap – both hands in this scenario are busy and therefore unable to grip the ergometer handles (see section 5.3).

(a) Ergometer (b) Arduino Leonardo (c) Pulse Sensor

FIGURE 5.2: Hardware used for studies.

[1]The exact model utilized was Ultrasport F-Bike 150/200B (`http://tinyurl.com/jlqnbna`).
[2]Original product information page: `http://tinyurl.com/qc2xzda`.
[3]The sensor was manufactured by *World Famous Electronics* (`http://pulsesensor.com`), originally financed via a kickstarter crowdfunding campaign (`https://www.kickstarter.com`).

Following sections describe both study projects in detail and qualitatively analyze preliminary results (see Chapter 6 for the main evaluations).

5.2 Game Prototype: Flappy Cycling

Flappy Bird has been an unexpected smash-hit casual game by Vietnamese developer Dong Nguyen and deemed as one of the simplest but nevertheless most addicting mobile games of 2013[4]. It is essentially an endless side-scroller[5], giving players control over a bird, which flies through a horizontally scrolling background and must avoid hitting randomly appearing green obstacle pipes (see Figure 5.3 showing three in-game screenshots).

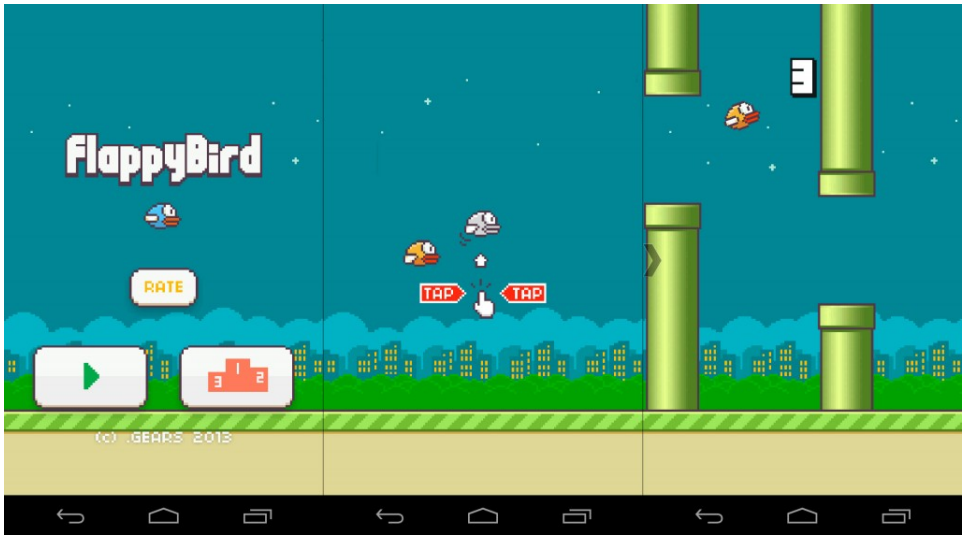


FIGURE 5.3: Original Flappy Bird game.
Source: `http://tinyurl.com/z6sls4s`

The bird in the game is controlled by simply tapping anywhere on the mobile screen, which "makes its wings flap", giving it an upwards boost. Gravity calculations on the other hand are used as a counterforce: not tapping the screen gradually makes the bird loose height, which quickly can result in hitting the ground, triggering a *game over*, i.e. ending the game. Additionally, a collision with any of the obstacle pipes has the same effect. The player's highscore increases whenever the bird passes through any of the small gaps that are formed by pairwise appearing pipes at constantly changing heights.

The simplicity of this game concept makes it perfect for creating an ergometer-controlled prototype: boosting the birds height can be achieved by a full pedal cycle. Thus, a

[4] Brianna Seneca and Emma Klein for Wired Jersey, 2014: "Quitting Flappy Bird cold turkey" (`http://tinyurl.com/z6sls4s`)

[5] Side-scrollers also often called platformers are games, which allow players to take control of one or several characters, which can be moved sideways through virtual 2D environments, while avoiding enemies and obstacles.

game prototype titled "Flappy Cycling" (FC) was designed as well as implemented using creative commons licensed assets – Figure 5.4 shows screenshots from the finished game.

(a) Title screen (b) In-game

FIGURE 5.4: Flappy Cycling protoype.

The purpose of constructing such a prototype was enabling the research of user experiences when playing a specifically created game for the fitness device, much resembling the approach the gaming industry currently takes. A preliminary study was conducted with the goal of collecting opinions for improving the quality of the game in order to further enhance the user experience during the main study.

5.2.1 Preliminary Study

The easiest way to convert the ergometer's signals to appropriate inputs for Flappy Cycling is to send out keystrokes on every pedal cycle to the system running the game. Figure 5.5 shows an architecture sketch for the preliminary Flappy Bird prototype.

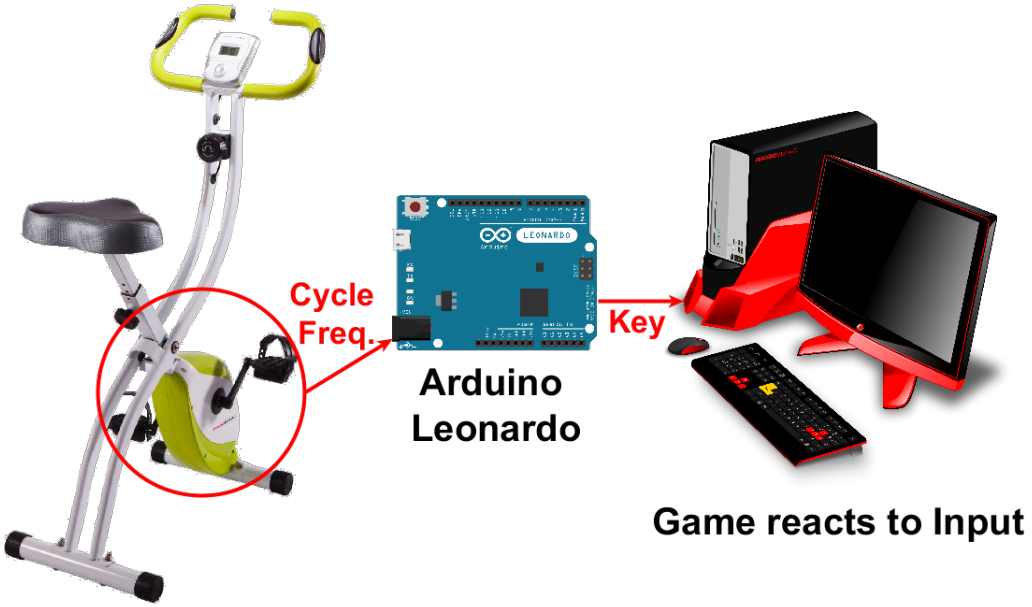


FIGURE 5.5: Architecture of preliminary Flappy Cycling Prototype.

A circuitboard connects the ergometer to the Arduino, which in turn is plugged into a computer. The Arduino acts as an additional keyboard, sending a dedicated keypress for every pedal cycle a participant completes on the bike. In-game the received key is used to give the bird a small upwards boost, counteracting gravity physics. The goal of this game version was to rack up points by keep the bird inside certain vertical screen boundaries – exceeding or falling below the area spanning the middle third of the screen triggers a penalty and visual feedback for the player to react accordingly (see Figure 5.6).

(a) In-game: cycling too slow (b) In-game: cycling too fast

FIGURE 5.6: Flappy Cycling speed control.

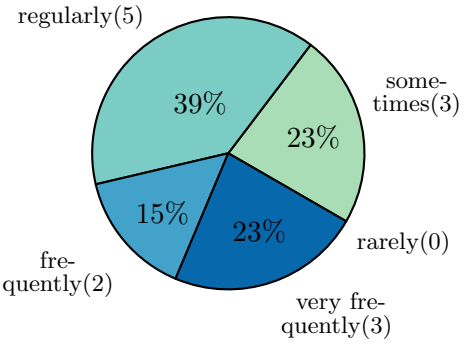
Points are collected over time for successfully staying within the declared area and increase ever more rapidly via a multiplier unless the bird leaves that area, in which case the multiplier is reset. Additionally to this penalty the screen is grayed out and textual feedback urges the players to either "speed up" or "slow down", depending on the bird flying to low or too high. No time limit or other game goals were set – users could work out on the ergometer for as long as they liked.

The study was set up in a supervised lab environment, where 13 participants were asked to try out the game while commenting on it and answering a questionnaire, created with LimeSurvey[6]. The questions asked as well as the complete results can be found in Appendix B, Subsections B.1.1, B.2.1 and B.3.1. In addition, all players were videotaped with their consent in order to be able to reconstruct the situations for subsequent analyses.

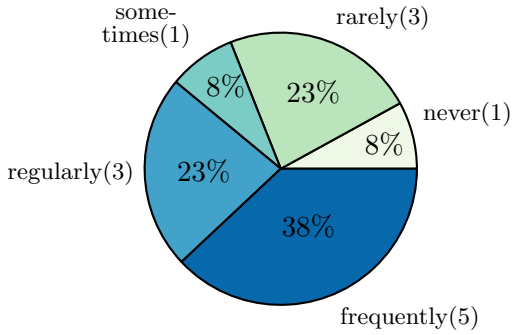
Most of the participants were male students (84%) and the overall average age was 25 (Appendix B, Tables B.7 and B.8). All of them play a variety of video games (Appendix B Table B.11) with varying frequencies as can be seen in Figure 5.7a. Figure 5.7b indicates that well over half of the users practice sports regularly. However, as shown in Figure 5.7c very few play games for fitness.

The participants' oral feedback combined with their answers to free text questions revealed nevertheless that they overall like the idea of combining gaming and sports, as Figure 5.8 depicts, which summarizes most commonly given opinions during the experiments in

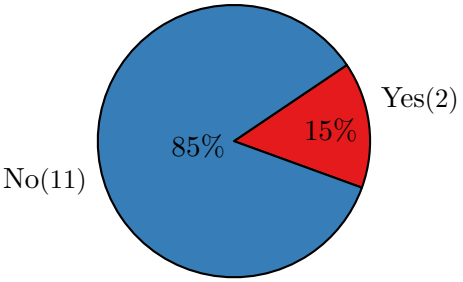
[6]LimeSurvey is a free open-source web software survey tool (`https://www.limesurvey.org`).



(a) B2: How often do you play video games?



(b) B8: How often do you practice sports?



(c) B4: Do you play computer games for fitness?

FIGURE 5.7: Preliminary Flappy Cycling responses: general.

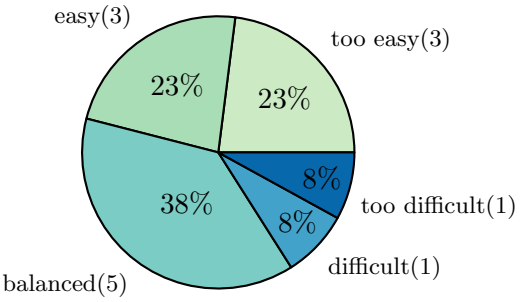
a cloud of phrases: the significance of an expression in terms of letter size and frequency conveys how many people support it.

FIGURE 5.8: User reactions to preliminary study.

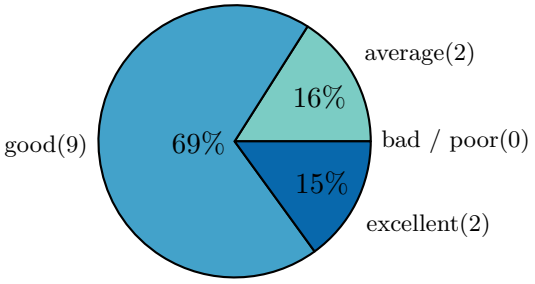
Besides the most prevailing opinion of generally liking the idea of exergaming, there were less significant attitudes, like personally preferring sports without gaming and caring

about the price of a fitness gaming devices. The other two most important notions were specific to the employed prototype game: it lacks game elements and there is no real goal to be accomplished. Indeed, since essentially any highscore could be beat by simply cycling longer, some adjustments had to be made for improving the game and making it ready for a larger scale experiment.

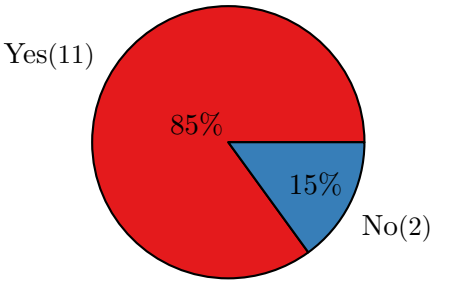
Finally, the users' responses to the survey questions back the observations from their verbal feedback, as can be discovered when regarding Figure 5.9, which shows a collection of selected questions that highlight the peoples' overall interest in fitness games.



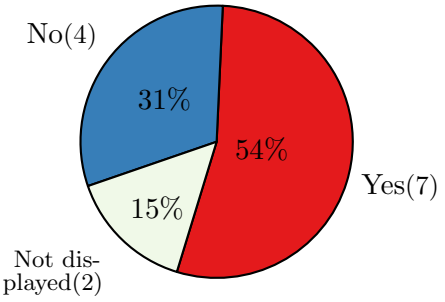
(a) C6: How did you perceive the game's difficulty?



(b) C11: How would you rate your overall experience during the experiment?



(c) C8: Would you play/exercise with games/devices like this at home? (Y→C10)



(d) C10: Would you thus be inclined to work out more often?

FIGURE 5.9: Preliminary Flappy Cycling responses: game specific.

The majority of ratings given for the game's difficulty were 'very easy' up to 'just right' (Figure 5.9a), indicating that the participants were presented with an easy to average challenge, which is great for onboarding, as previously mentioned in Chapter 1. Accordingly, their overall experience was at least average, as Figure 5.9b shows, and in most cases even 'very good'. Also, by analyzing Figures 5.9c and 5.9d, it seems that 85% of the users would be interested in owning exergames, half of which even considered working out more because of it. Since this stands in contrast to the responses given for question B4 (Figure 5.7c), where most people stated that they do not play fitness games, there is room for speculations: either they do not own suitable platforms or equipment, which among other things can be a monetary problem or they were never fully convinced of these games.

All things considered, a bias caused by external factors, such as the personal supervision provided, can certainly not be ruled out: for example, the fact that the game lacked refinement was not reflected at all in the rating scales, which could be explained by considering the study procedure: participants tried out the game, while verbally giving feedback and thus, by already having criticized the game that way, refraining from letting those aspects influence their final verdict. Therefore, additionally to refining the prototype, the questionnaires of the subsequent studies were as well improved. Several optional free text fields have been added, enabling users to submit anonymous feedback (see Appendix B, Section B.2.1). In particular, the subsequent studies did not involve explicit feedback inquiries through supervisor conversations during the game testing phase, attempting to reduce the overall bias and keeping the persons focused on playing.

5.2.2 Refinements

As mentioned in the previous section, Flappy Cycling still lacked game design elements that made it challenging and a more realistic goal than just open-ended pedaling. To address these issues a more elaborate game had to be devised, building on the core mechanics of the already implemented version. First, to be able to give the users more control over the bird, the ergometers touch sensors could be utilized (see Figure 5.10), which is a convenient alternative to adding a gamepad to the setup.

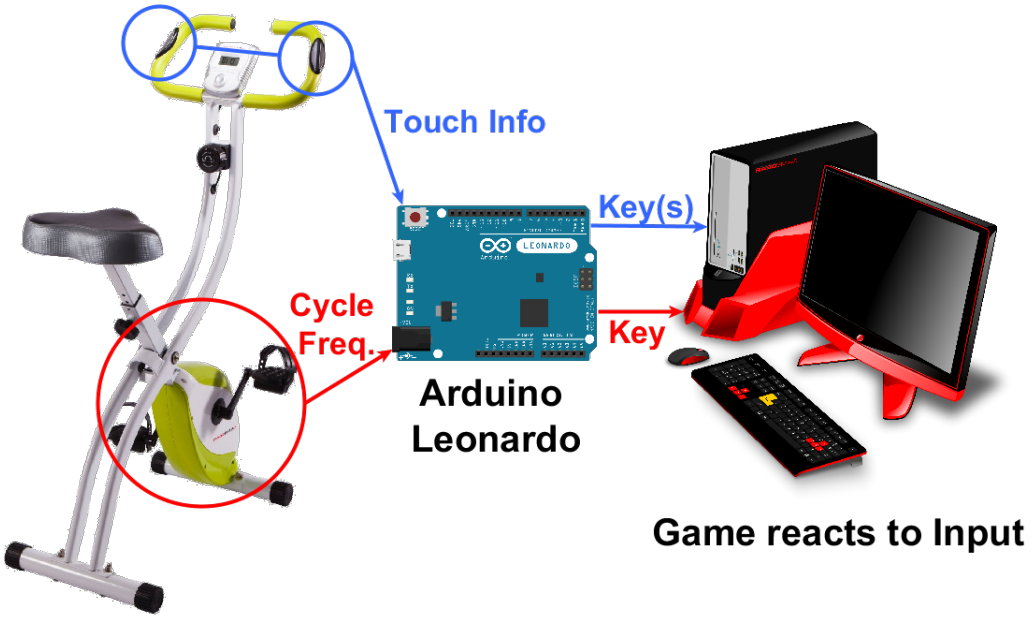


FIGURE 5.10: Architecture of final Flappy Cycling prototype.

By diverting the pulse sensor handles from their intended use – measuring the heart rate – and instead merely detecting when someone is touching either of them, this final Flappy

Cycling hardware configuration adds two more inputs to the game: horizontally flying forwards and backwards. Along with this mechanism enemy birds were added to the game, which enter the game from the right screen border, flying in a wavy pattern towards the controlled bird, which must avoid touching them in order to prevent a penalty. This can be achieved by skillfully steering the bird with the touch handles, while observing the flight pattern of the enemies. Dodging the enemies gradually gets more challenging because they appear ever more frequently, while constantly changing flight pattern and speed. In case of a collision, just like flying out of the virtually set upper or lower bounds, the game is grayed out and the score multiplier reset. In addition to that, also a two second countdown appears in the middle of the screen, which indicates the penalty time before the multiplier is restarted again (see Figure 5.11).

(a) In-game: before hitting an enemy (b) In-game: after hitting an enemy

FIGURE 5.11: Flappy Cycling enemy hit penalty.

The next step towards a better gaming experience was to create a more competitive game goal: a time limit of five minutes was introduced and instead of recording the time spent cycling, a countdown-timer is placed at the upper right screen corner, reminding the players of how much time remains before the game ends. As game over also can be triggered by falling below the lower bounds of the screen, it is as well possible to use the device for less than five minutes and still be presented with the accomplished highscore up until that point. The final measure taken was to increase competition even more by implementing a highscore table, where players can compare their performance to others (see Figure 5.12).

Even though having made all specified adjustments to the game, it must be considered that the game nevertheless remains a prototype, meaning that some aspects of it are still flawed, which, although important to keep in mind, is not particularly game changing for later evaluations, as their main goal, i.e. examining user experience, only partially depends on the quality of the game. Also in a way this can be beneficial because, as mentioned before, the prototype should represent a rudimental fitness game, similar in concept to some actually available on the market, many of which when examined closely are as well flawed and can be cheated[7].

[7]E.g. wikiHow: "How to cheat on Wii Fit" (`http://tinyurl.com/hnxhufv`)

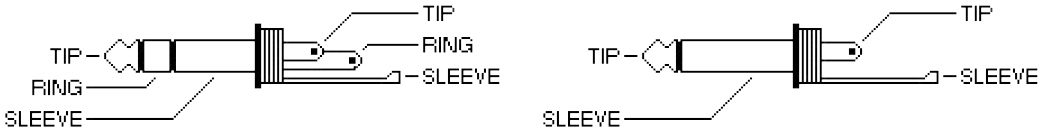
Player	Score	Time
1	338	00:05:00
2	291	00:05:00
3	290	00:05:00
4	264	00:05:00
5	261	00:05:00
6	260	00:05:00
7	259	00:05:00
8	258	00:05:00
9	255	00:05:00
10	251	00:05:00

FIGURE 5.12: Flappy Cycling study highscore (rank 1-10).

5.2.3 Implementation

Flappy Cycling is powered by three devices – an ergometer, an Arduino and a computer. When it comes to connecting all of them, there are two things to consider: how to connect the ergometer to the Arduino and how to communicate with the game. This section will elaborate on both of these aspects and outline important implementation details.

A prototyping solderless breadboard serves as a means for creating custom circuits using the Arduino – the ergometer is plugged into an electronic circuit, which enables the Arduino to read pedaling and touch signals, respectively. Luckily the utilized ergometer already provides two male 3.5mm *Tip-Ring-Sleeve* (TRS) (see Figure 5.13a) connectors – one for measuring the pedal *revolutions per minute* (rpm) and another one intended for calculating the *heart rate* (HR) from the pulse sensors. Both normally connect directly to the device's control unit, which can calculate and display said values running on a 9-volt battery. Conveniently they can just be unplugged and in order to connect them to the breadboard, it is only necessary to solder together two fitting female TRS jacks. Those are then connected to the ergometer's jacks and unisolated litz wire ends plugged into breadboard holes. Subsequently, nothing stands in the way of devising appropriate circuits.



(a) TRS plug (TIP: left signal *L*, RING: right signal *R*, SLEEVE: ground *GND*).

(b) TS plug (TIP: signal *SIG*, SLEEVE: ground *GND*).

FIGURE 5.13: Tip-Ring-Sleeve plugs.
Source: Leads Direct technical library (`http://tinyurl.com/gln8tw4`)

The ergometer's pedal wiring works similarly to a circuit containing a simple switch: electric *voltage* (V) is measured at a certain position in the circuit and the observed signal can either be high or low, depending on whether the switch is on or off. On every pedal cycle the ergometer's circuit is closed for a short time period via such a mechanism and this can easily be measured with the Arduino[8] by constructing a small circuit, shown in Figure 5.14.

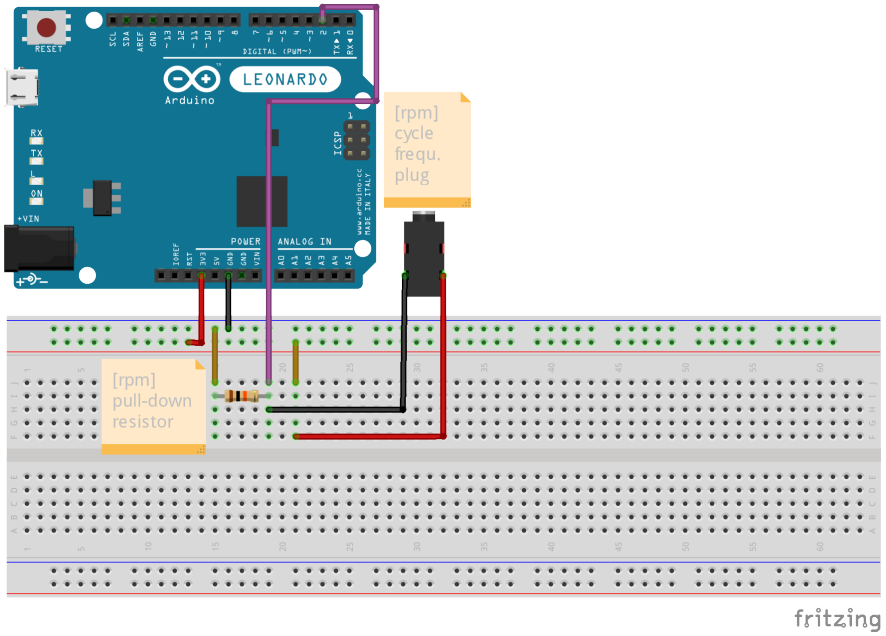


FIGURE 5.14: Arduino circuit for capturing pedal cycles.
Created with Fritzing (`http://fritzing.org`)

Since a plain switch only needs to let *current* (I) flow through or not, it is sufficient to use a simple *Tip-Sleeve* (TS) jack (see Figure 5.13b) for both male and female pedal connectors, commonly also called *mono* 3.5mm plug. Therefore, only two cables emerging from the female jack need to be connected to the circuit: a signal (tip) and a ground (sleeve) wire. The tip wire (red) is plugged into the Arduino's 3.3 volts power supply, while the sleeve (black) wire is used to measure high or low current via the Arduino's digital input pin number 2 (violet cable), depending on the ergometer's internal switch. Additionally, a *pull-down resistor* (10kΩ) needs to be placed after this switch, pulling any stray voltage to ground[9] when the switch is open, i.e. the circuit is disconnected – otherwise the input pin would pick up random noise leading to arbitrary fluctuations between high and low voltage measurements.

[8] For the Arduino Leonardo HIGH means a voltage of greater than 2 volts is present, everything else is considered LOW (`http://tinyurl.com/jzlwp5d`).

[9] In case the ergometer's switch is open, any current caused by noise flows through the only available path: towards ground through the pull-down resistor and the input pin will read low voltage. Otherwise when the switch is closed, 3.3V can flow through the circuit and the input pin reads high voltage. Some current will still flow through the pull-down resistor but according to Ohm's law ($I = \frac{V}{R}$) it will be noneffectively low: $I = \frac{3.3}{10000} = 0.00033A$.

The second circuit uses the ergometer's metal touch handles, which when having skin contact, in their intended function interpret small signals that are passed through the body. The control unit amplifies these, filters out noise like respiration or muscle contractions and displays the HR. In Flappy Cycling only information about which handle is being touched should be recognized. Therefore, the signals of both handles must be made distinguishable, which can be achieved by constructing the circuit shown in Figure 5.15.

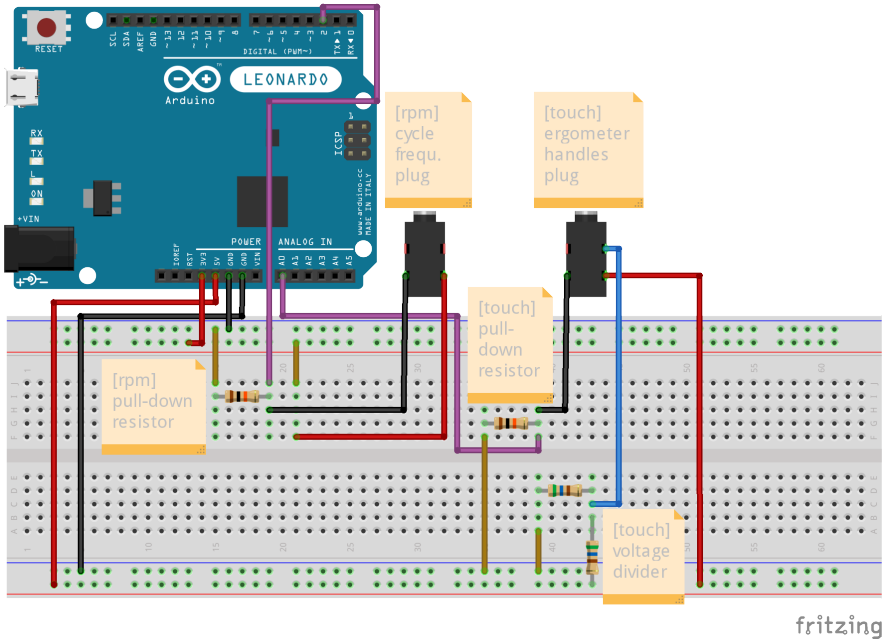


FIGURE 5.15: Arduino circuit refinement for capturing sensor handle touch information. Created with Fritzing (`http://fritzing.org`)

Again, a signal is measured at a dedicated spot in the circuit, but this time it needs to be analog to be able to discern between different voltage states. Similar to the previous pedal circuit, the signal is observed after a pull-down resistor with the only distinction that analog pin A0 reads the voltage. To distinguish the handle signals from each other, they are powered by different voltages: while the right (ring) is plugged into the Arduino's 5V supply, the left only receives 2.5V, which is possible by constructing a *voltage divider* using two equal resistors to half the input voltage[10]. Hence, touching a handle and thereby closing a circuit results in measuring lower voltage for the left handle, higher voltage for the right one. Unfortunately there are downsides to this procedure, which partially were not discovered until the study had already been in progress: the analog signal received depends on the perspiration of the hands touching the handles and is also distorted by noise. Additionally there is no way to determine if both handles are touched simultaneously – the signal simply resembles touching the handle powered by

[10] A special formula, i.e. $V_{out} = \frac{V_{in} * R_2}{(R_1 + R_2)}$, can be utilized to calculate the voltage divider output of two resistors: $V_{out} = \frac{5 * 560}{(560 + 560)} = 2.5$.

the higher voltage, i.e. the right one. These flaws – once uncovered – have henceforth been communicated to every study participant either prior to trying out the game or during an instructional test run (see Section **??** for more details on that matter).

After having built all necessary circuits for Flappy Cycling, the communication with the game must be handled. The Arduino plugs into a computer via an *universal serial bus* (USB) 2 connector and can be programmed by using the open-source Arduino *integrated development environment* (IDE) (`http://tinyurl.com/osnz57o`). The Arduino language is based on C/C++ and links against AVR Libc (`http://tinyurl.com/2wwow8`), which is high quality open-source C library for Atmel AVR microcontrollers[11]. As mentioned above, the ergometer controls Flappy Cycling via keyboard commands, hence an appropriate library must be utilized, which conveniently is already part of the provided software package. Code Listing 5.1 shows the main piece of code[12] that handles capturing a pedal cycle and reacting by virtually pressing an appropriate key.

```
1  /* globals (pedals) */
2  int pedalsState = 0;              // stores pin reading
3  boolean pedalsWereHigh = false;   // previous pedal state
4  const int pedalsPin = 2;          // input pin
5
6  /* init */
7  void setup()
8  {
9     pinMode(pedalsPin, INPUT);    // use pedalsPin as input
10    Serial.begin(115200);         // comm. rate p. second (baud)
11    Keyboard.begin();             // start emulating keyboard
12 }
13
14 /* main loop */
15 void loop()
16 {
17    pedalsState = digitalRead(pedalsPin);   // read pin voltage
18    if (pedalsState == HIGH)
19    {
20       // prevent repeated keypress
21       if (!pedalsWereHigh)
22       {
```

[11] The Arduino environment is based on Atmel Atmega microcontrollers (`http://tinyurl.com/ppch8ns`).

[12] The full source code has been published on GitHub (`http://tinyurl.com/jl6md9r`).

```
23            Keyboard.write(120);              // press and release 'x' key
24            pedalsWereHigh = true;            // remember pedal status
25         }
26      }
27      else
28      {
29         pedalsWereHigh = false;               // reset pedal status
30      }
31   }
```

CODE LISTING 5.1: Arduino sketch for capturing pedal cycles (full code: `http://tinyurl.com/h6e9qdw`).

The sketch starts by defining global variables that help identifying the input source and remembering the state of the pedals. Following `setup` function, beginning at line 6, is called once when powering up or resetting the Arduino, hence containing initializing code: telling the Arduino to use the pedal signal pin in `INPUT` mode, setting data communication rate and starting keyboard emulation. After initialization is completed, the remaining section starting from line 14 defines a `loop` function that at runtime is executed over and over, infinitely interpreting the circuit signal and reacting to it. The Algorithm is as follows:

1. Store the current pin signal (`HIGH` = `1` or `LOW` = `0`) into `pedalState`. If the signal is `HIGH` move on to 3., else continue at 4.
2. Signal is `HIGH`: prevent repeated keypresses by consulting `pedalsWereHigh`, which accounts for the pedal state from the previous loop. If this signal was `LOW`, i.e. `pedalsWereHigh` set to `false`: press and release key 'x'[13] and remember current `HIGH` state for next loop by setting `pedalsWereHigh` to `true`. Skip 3.
3. Signal is `LOW`: reset `pedalsWereHigh` to `false` for next loop traversal.
4. Move on to next loop (1.).

Constructing the code like this has the advantage that only a single keystroke is sent for the entire time period the pedal signal is `HIGH` – for a fresh key press the signal needs to be `LOW` momentarily, which only occurs during a new pedal cycle.

For the second piece of Arduino code, capturing pulse sensor touch information, two main aspects must be taken into account: discerning between left (`L`) and right (`R`) handle,

[13] The *American Standard Code for Information Interchange* (ASCII) decimal character code for 'x' is 120.

while filtering out unwanted noise. Additionally, moving the bird forward or backward involves continually pressing keys for as long as the handles are touched, which implicates the need for properly regulating key presses and releases. Code Listing 5.2 shows the complete Arduino sketch[14] and outlines most important aspects via inline comments.

```
/* globals (touch) */
const int MAX_LOWER_NOISE = 10;     // ignore signals below
const int MIN_UPPER_NOISE = 250;    // ignore signals above
const int MAX_L_SIGNAL = 35;        // weaker, downpowered by v.divider
const int MIN_R_SIGNAL = 45;        // higher (5V)
const int touchPin = 0;             // analog pin reading touch sig
unsigned long lastSignalCheck = 0; // last signal check time
char keyL = 'z';                    // default keyb. layout: en (z,x,c)
char keyR = 'c';
int keyStateL = 0;
int keyStateR = 0;

/* init */
void setup()
{
   Serial.begin(115200);          // data comm. rate p. second (baud)
   Keyboard.begin();              // start emulating keyboard
   lastSignalCheck = millis();    // record current time
}

/* main loop */
void loop()
{
   // check for touch info (every 50ms)
   if (!isIntervalElapsed(lastSignalCheck, 50)) return;
   int a = analogRead(pin);    // read touch pin
   if (a >= MAX_LOWER_NOISE && a <= MAX_L_SIGNAL)
   {
      // signal in L range [MAX_LOWER_NOISE, MAX_L_SIGNAL]
      if (keyStateR > 0)
      {
         Keyboard.release(keyR);    // if R pressed, release key
```

[14] The code for handling pedal cycle capturing, which resides in the same file, is omitted for the sake of simplicity and clarity.

```
33          keyStateR = 0;
34       }
35       if (keyStateL < 1)
36       {
37          Keyboard.press(keyL);     // press L key, if not pressed
38          keyStateL = 1;
39       }
40    }
41    else if (a >= MIN_R_SIGNAL && a <= MIN_UPPER_NOISE)
42    {
43       // signal in R range [MIN_R_SIGNAL, MIN_UPPER_NOISE]
44       if (keyStateL > 0)
45       {
46          Keyboard.release(keyL);   // if L pressed, release key
47          keyStateL = 0;
48       }
49       if (keyStateR < 1)
50       {
51          Keyboard.press(keyR);     // press R key, if not pressed
52          keyStateR = 1;
53       }
54    }
55    else
56    {
57       // signal is lower or upper noise: release all pressed keys
58       // ( signal < MAX_LOWER_NOISE || signal > MIN_UPPER_NOISE)
59       if (keyStateL > 0)
60       {
61          Keyboard.release(keyL);
62          keyStateL = 0;
63       }
64       if (keyStateR > 0)
65       {
66          Keyboard.release(keyR);
67          keyStateR = 0;
68       }
69    }
70    lastSignalCheck = millis();       // update touch check time
71 }
```

```
72
73 /* Check for elapsed interval since time (avoids inserting delays) */
74 bool isIntervalElapsed(unsigned long sinceTime, int interval)
75 {
76    unsigned long currentTime = millis();
77    unsigned long elapsedTime = currentTime - sinceTime;
78    if (elapsedTime >= interval) return true;
79    return false;
80 }
```

CODE LISTING 5.2: Arduino sketch for capturing handle touch information (full code: `http://tinyurl.com/h6e9qdw`).

Beginning in similar fashion to the pedal cycle code, globals, `setup` and `loop` are declared. Line 2 to 5 define several constants, due to the necessity of restricting signal level intervals for `L` and `R`. Limiting the rate of reading the input pin signal (stored in `touchPin`) also reduces the probability of noise interferences and is realized using conditionals rather than introducing forced delays, which would slow down the execution frequency of the `loop` function, thus, interfering with the code for capturing pedaling, omitted in the figure. Therefore, `loop`, in addition to adjusting the key press states (`keyStateL`, `keyStateR`) on every signal check, needs to record the current time as well (`lastSignalCheck`), in order to enable a constant reading delay of `50` milliseconds[15] (ms) by checking for the time elapsed since the last recorded time, which is done in the `isIntervalElapsed` function starting on line 73. The algorithm for handling a signal works in the following manner:

1. Has the last signal check been `50` ms or longer ago? If so, move on to 2., else exit `loop`.
2. Read analog signal and check if it is falling into the value ranges of `L` (`[MAX_LOWER_NOISE, MAX_L_SIGNAL]`) or `R` (`[MIN_R_SIGNAL, MIN_UPPER_NOISE]`). If it is part of range `L` go to 3., for `R` move on to 4., else skip to 5.
3. The left handle is touched: reset `keyStateR`, release `keyR` (`'c'`), then set `keyStateL` and press `keyL` (`'z'`).
4. The right handle is touched: reset `keyStateL`, release `keyL` (`'z'`), then set `keyStateR` and press `keyR` (`'c'`).
5. No handle is touched and signal is either 0 or noise: release all pressed keys and reset key states.

[15] All hard-coded values have been determined by trial and error prior to conducting the study.

6. Update `lastSignalCheck` to current time for successive iterations.

This concludes the description of all microcontroller implementations needed to provide the necessary input for the game. Since the code base of Flappy Cycling is too large to include in its entirety, the remaining part of this section will outline the game's structure and most important parts.

Flappy Cycling is implemented in popular HTML5 game developer's framework Phaser[16], which uses Pixi.js[17] for browser rendering and enables developing using JavaScript as a programming language – in particular, TypeScript[18] was used, a free open source language developed and maintained by Microsoft[19]. Typescript is a superset of JavaScript, aiming at making programming with it more object-oriented and enjoyable: for runtime TypeScript files are source-to-source compiled to plain JavaScript, which is also called transpiling – as an example see Code Listing 5.3 transpiled to Code Listing 5.4.

```
class Student
{
    name : string;
    constructor(public firstname) {this.name = firstname;}
}
function greeter(student : Student)
{
    return "Hello, " + student.name+"!";
}
var user = new Student("Jane");
document.body.innerHTML = greeter(user);
```

Code Listing 5.3: Sample TypeScript code.

```
var Student = (function ()
{
    function Student(firstname)
    {
        this.firstname = firstname;
        this.name = firstname;
    }
```

[16] Phaser HTML5 framework (`http://phaser.io`) is created by company Photon Storm (`http://www.photonstorm.com`).
[17] 2D webGL renderer Pixi.js (`http://www.pixijs.com`) is developed by Goodboy Digital (`http://www.goodboydigital.com`).
[18] `http://www.typescriptlang.org`
[19] `https://www.microsoft.com`

```
8     return Student;
9 })();
10 function greeter(student)
11 {
12     return "Hello, " + student.name + "!";
13 }
14 var user = new Student("Jane");
15 document.body.innerHTML = greeter(user);
```

Code Listing 5.4: Transpiled JavaScript code.

Code Listing 5.5 shows the complete project structure of Flappy Cycling[20], which includes the Phaser library, game assets[21], TypeScript files holding the main code as well of course their transpiled JavaScript counterparts, style sheets and HTML files holding containers for display. The total lines of code (LOC) amount is 952, excluding tools, libraries, `.html`, `.js` and `.css` files.

```
FLAPPY_CYCLING
|   create_dist.cmd
|   highscore.html
|   index.html
|
+---assets
|   +---fonts
|   +---gfx
|   +---sound
|   +---spritesheets
+---css
|       app.css
|       highscore.css
|
+---js
|   +---GameObjects
|   +---States
|   +---Utils
+---phaserLib
+---tools
+---ts
    |   app.ts
    |   config.ts
    |   references.d.ts
    |
    +---GameObjects
    |       Bird.ts
    |       Enemy.ts
    |
    +---States
    |       GameOverScreenState.ts
    |       GameScreenState.ts
    |       TitleScreenState.ts
    |
    +---Utils
            CountdownTimer.ts
            IntervalTimer.ts
            RandomIntervalTimer.ts
            Timer.ts
            UtilFunctions.ts
```

Code Listing 5.5: FC: project structure.

The game consists of three screens or states contained in the corresponding `.ts` files: `TitleScreenState`, `GameScreenState` and `GameOverScreenState`. The `app.ts` file contains the `FlappyCycling` class, abbreviatedly shown in Code Listing 5.6, which marks the

[20] The project's GitHub repository is publicly available: `http://tinyurl.com/ja5qzhm`.

[21] All of Flappy Cycling's game assets were published under some type of creative commons license (`https://creativecommons.org`) that allows non-commercial use – see Appendix C for a detailed listing.

game's entry point, making all necessary preparations like setting the screen resolution and loading all utilized assets, ultimately starting up the title screen.

```
/// <reference path="references.d.ts"/>
/// <reference path="config.ts"/>
class FlappyCycling
{
   game: Phaser.Game;

   constructor(width:number, height:number)
   {
      // create game in 'content' container
      this.game = new Phaser.Game(1920, 1080, Phaser.AUTO, 'content',
 ↪  {create: this.create, preload: this.preload});
   }

   preload()
   {
      var game = this.game;
      // load images, sprites and sounds
      var gfxPath = "assets/gfx/";
      game.load.image('bg_z-2', gfxPath+'bg/z-2.png');
      game.load.image('bg_z-1', gfxPath+'bg/z-1.png');
      game.load.image('bg_z0', gfxPath+'bg/z0.png');
      // sprite sheets
      var spritePath = "assets/spritesheets/";
      game.load.atlasJSONHash("BIRD_FLYING", spritePath+"bird.png",
 ↪  spritePath+"bird_flying.json");
      game.load.atlasJSONHash("BIRD_OUTLINE",
 ↪  spritePath+"bird_outline.png", spritePath"bird_outline.json");
   }

   create()
   {  // add all game screens and show title screen
      var sm: Phaser.StateManager = this.game.state;
      sm.add("TitleScreenState", States.TitleScreenState, true);
      sm.add("GameScreenState", States.GameScreenState, false);
      sm.add("GameOverScreenState", States.GameOverScreenState,
 ↪  false);
```

```
33      }
34  }
```

Code Listing 5.6: FC: abbreviated `app.ts`
(full code: `http://tinyurl.com/hmjvzse`).

As can be guessed by closely regarding the code, `game` of type `Phaser.Game` is the single most important object, capable of accomplishing almost everything within the framework. Therefore, it contains a huge list of objects as well as methods that allow a variety of game manipulations, like loading assets for later use (see `preload` function) and adding different game states (see `create` function), which are of type `Phaser.State`. As hinted above, the game controls these three added states according to the user input received and of course the player's performance. Their main structure is very similar, hence it is sufficient to merely show a short version of the most important one – `GameScreenState.ts` – in Code Listing 5.7.

```
1   module States
2   {
3       export class GameScreenState extends Phaser.State
4       {
5           game: Phaser.Game;
6           bird: Sprites.Bird;
7           // sprite groups (object pools for visible layers)
8           bgGroup: Phaser.Group;
9           midGroup: Phaser.Group;
10          frontGroup: Phaser.Group;
11          // parallax scrolling back- and foreground
12          bgTile0: Phaser.TileSprite;
13          bgTile1: Phaser.TileSprite;
14          bgTile2: Phaser.TileSprite;
15          // timing
16          gameTimeCountDown: Utils.CountdownTimer;
17          enemySpawnTimer: Utils.RandomIntervalTimer;
18          enemyHitTimeout: Utils.CountdownTimer;
19
20          // init
21          create()
22          {
23              var game = this.game;
24              // start arcade physics
```

```
25          game.physics.startSystem(Phaser.Physics.ARCADE);
26          game.physics.arcade.gravity.y = 200;
27          // build scene
28          this.bgGroup = game.add.group();
29          this.midGroup = game.add.group();
30          this.frontGroup = game.add.group();
31          // add bg far
32          this.bgTile0 = game.add.tileSprite(0, 0, game.stage.width,
    ↪ game.cache.getImage('bg_z-2').height, 'bg_z-2');
33          this.bgGroup.add(this.bgTile0);
34          // add bg near
35          this.bgTile1 = game.add.tileSprite(0, 0, game.stage.width,
    ↪ game.cache.getImage('bg_z-1').height, 'bg_z-1');
36          this.bgGroup.add(this.bgTile1);
37          // create bird and add to mid group (bird contains outline)
38          this.bird = new Sprites.Bird(game, 200, game.height -
    ↪ game.height/2);
39          this.midGroup.add(this.bird);
40          // add foreground
41          this.bgTile2 = game.add.tileSprite(0, 0, game.stage.width,
    ↪ game.cache.getImage('bg_z0').height, 'bg_z0');
42          this.frontGroup.add(this.bgTile2);
43          this.frontGroup.add(this.bird.getOutline()); // bird outline
44          // timers
45          this.gameTimeCountDown = new Utils.CountdownTimer(game, 1000
    ↪ * 60 * 5); // 5min
46          this.enemySpawnTimer = new Utils.RandomIntervalTimer(game,
    ↪ 8000, 4000); // max: 8sec, min: 4sec
47          this.enemyHitTimeout = new Utils.CountdownTimer(game, 0);
48      }
49
50      // main loop
51      update()
52      {
53          var game = this.game;
54          this.calcScore(); // calculates score (omitted)
55          this.updateScoreDisplay(); // adjust user info (omitted)
56          // spawn enemy after random interval
```

```
57          if (this.enemySpawnTimer.checkInterval())
    ↪  this.generateEnemy();
58          this.adjustDifficulty(); // difficulty+ after 30s (omitted)
59          // check  for game over
60          if (this.bird.body.bottom == game.world.bounds.bottom ||
    ↪  !gameTimeCountDown.isRunning())
61          {  // death by touching the ground
62             this.bird.die();
63             game.state.start("GameOverScreenState");
64          }
65          // parallax scroll bg tiles
66          this.bgTile0.tilePosition.x -= 4;
67          this.bgTile1.tilePosition.x -= 5;
68          this.bgTile2.tilePosition.x -= 6;
69       }
70
71       // spawns new enemy
72       generateEnemy()
73       {
74          var game = this.game;
75          var rndYPos: number = this.rndGen.realInRange(1.0/3, 2.0/3);
76          var crow: Sprites.Enemy = new Sprites.Enemy(game,
    ↪  game.width+500, game.height * rndYPos-100, this.bird);
77          crow.onHit(this.enemyHit);
78          this.midGroup.add(crow);
79          this.frontGroup.add(crow.getOutline());
80       }
81
82       // called when bird hits enemy (instance arrow function)
83       public enemyHit = () =>
84       {  // 'this' belongs to game state here, (NOT enemy class!)
85          this.enemyHitTimeout.setCountdown(1000 * 2); // 2 sec timeout
86          this.enemyHitTimeout.restart();
87       }
88    }
89 }
```

Code Listing 5.7: FC: abbreviated `GameScreenState.ts` (full code: `http://tinyurl.com/gvkk3ha`).

All states are members of the `module States` and can use certain dedicated functions that are defined in `Phaser.State`. They are called in a specific order, hence, offering many possibilities for customization. Flappy cycling merely overrides its most common functions: `create`, which is called first once a state is activated and `update`, processed whenever a new frame is rendered. The game screen is divided into three distinct layers of varying perceived depth, which is achieved by creating different `Phaser.Group`'s that contain all in-game images and objects, also named *sprites*. Sprites generally are 2D graphics that are placed into a game scene and often can be controlled or interacted with. For instance, the playable character, which will be discussed later, is as well a customized kind of sprite that is animated and has special capabilities (see Code Listing 5.8). To create perception of depth for the player, every layer contains a different background image of the type `Phaser.TileSprite`, horizontally scrolling at its own pace: `bgTile0` for the far background `bgGroup`, `bgTile1` for the near background `midGroup` and `bgTile2` for the foreground `frontGroup` (see Figure 5.16 showing all single layers and their combination). The images are added to `game` in the `create` function and their scrolling speed is set in `update` after line 65, using a technique called *parallax scrolling*.

(a) Flappy Cycling: far background image.

(b) Flappy Cycling: near background image.

(c) Flappy Cycling: foreground image.

(d) Flappy Cycling: combined images.

FIGURE 5.16: Flappy Cycling: parallax scrolling images.
Source: see Appendix C.

Birds and enemies are placed on top of all `midGroup` sprites and therefore are only obstructed by `frontGroup` sprites. To prevent them from completely disappearing behind trees or hills, they possess a foreground outline that is placed on top of everything. Lastly, some timers need to be initialized with the purpose of timing the game (`gameTimeCountDown`), the enemy spawn rate (`enemySpawnTime`) and the penalty timeout for hitting an enemy (`enemyHitTimeout`). Specific timer implementations are omitted, but they work in the same fashion as timing the rate for reading touch handle signals in the associated arduino code (Code Listing 5.2). A typical execution of the `update` function, i.e. the repeating game loop, starts by calculating the score (`calcScore`) and refreshing its display (`updateScoreDisplay`), both of which again are omitted, since they are quite lengthy, considering lots of conditions. Nevertheless, the formula of calculating the score is simple:

- Score (`totalScore`) is increased every 1.3 seconds.

- If an enemy hit timeout is running or the bird flies out of the designated vertical area, `totalScore` is increased by 0.5 and the score multiplier (`scoreMultiplicator`) is set to 1.
- On normal conditions, i.e. no restrictions are violated, `scoreMultiplicator` increases by 1 every 10 seconds, maximally reaching the value 5. The `totalScore` increases according to following formula: `totalScore += (1 * scoreMultiplicator)`.

Final scores are stored into the browser's local storage[22] and a simple highscore table on a separate HTML page (`highscore.html`) lists all accumulated contestants' rankings.

Another task of `update` is to initiate randomly spawning enemies, i.e. calling `generateEnemy` after consulting `enemySpawnTimer`. Similar to how the bird is added to `game` in `create`, a new enemy is placed just outside the right screen border. Its gravity settings, *bounding box*[23] and random flying pattern[24] are calculated in the `Enemy` class – also omitted for the sake of simplicity. A `GameScreenState` callback function (`enemyHit`), triggering the enemy hit timeout, is registered to newly created rival `crow`, which calls the function in case of a collision with `bird` occurs. As mentioned, in order to ensure a greater challenge towards the end of the game, the game difficulty is adjusted every 30 seconds in `adjustDifficulty`, which basically merely lowers the enemy spawn time probability. The final functionality of `update` is initiating a game over by checking if the bird touches the bottom of the screen or the game time of 5 minutes is up, which is done after line 59. The simplified procedure included in the listing tells the bird object to self-destruct and in succession switches to `GameOverScreenState`.

Before concluding this section, a quick look at the bird's implementation is taken in Code Listing 5.8, displaying a simplified version of the `Bird` class.

```
1  module Sprites
2  {
3     export class Bird extends Phaser.Sprite
4     {
5        FLY_BTN: Phaser.Key;
6        LEFT_BTN: Phaser.Key;
7        RIGHT_BTN: Phaser.Key;
8        currentXSpeed: number;       // x-axis speed
```

[22] Local storage key-value pairs can be stored since HTML5 (`http://tinyurl.com/8peqwvd`).
[23] Bounding boxes usually are non-visual squares or rectangles surrounding a sprite, used for collision detection with other objects.
[24] This particular quality uses an easy-to-use Phaser function to manipulate x- and y-axis positions of sprites, called *tween*. The moving curves can be customized and for the y-axis tween a quadratic easing function was used (`Phaser.Easing.Quadratic.InOut`).

```
9         outline: Phaser.Sprite;     // foreground outline
10
11        constructor(game:Phaser.Game,x:number,y:number)
12        {
13            super(game,x,y,"BIRD_FLYING", 0); // frame 0
14            this.outline = new Phaser.Sprite(game,x,y,'BIRD_OUTLINE',0);
15            // keys
16            var kb: Phaser.Keyboard = game.input.keyboard;
17            this.FLY_BTN = kb.addKey(Phaser.Keyboard.X);
18            this.LEFT_BTN = kb.addKey(Phaser.Keyboard.Y);
19            this.RIGHT_BTN = kb.addKey(Phaser.Keyboard.C);
20            this.FLY_BTN.onDown.add(Sprites.Bird.prototype.flap, this);
21            this.currentXSpeed = 0;
22            // arcade physics bird
23            game.physics.enable(this, Phaser.Physics.ARCADE);
24            this.body.collideWorldBounds = true;
25            this.body.bounce.set(0.4);
26            this.body.setSize(78,86, 10, 0);
27            // add bird & outline to game
28            var birdStartHeight = (game.height/2) - (this.height/2);
29            this.y = birdStartHeight ;
30            this.outline.y = birdStartHeight;
31            this.Animate(); // infinitely loop Animation (impl. omitted)
32            game.add.existing(this);
33            game.add.existing(this.outline);
34        }
35
36        flap(): void
37        {
38            this.body.velocity.setTo(0, -70);
39            this.outline.body.velocity.setTo(0, -70);
40        }
41
42        flyTo(direction:number): void
43        {
44            this.x += this.currentXSpeed * direction;
45            this.outline.x += this.currentXSpeed * direction;
46        }
47
```

```
48          getOutline(): Phaser.Sprite
49          {
50              return this.outline;
51          }
52
53          update()
54          {
55              if (this.LEFT_BTN.isDown && this.RIGHT_BTN.isDown)
56              {
57                  this.currentXSpeed = 0;
58              }
59              else if (this.LEFT_BTN.isDown)
60              {
61                  this.currentXSpeed = 15;
62                  this.flyTo(-1);
63              }
64              else if (this.RIGHT_BTN.isDown)
65              {
66                  this.currentXSpeed = 10;
67                  this.flyTo(1);
68              }
69          }
70      }
71  }
```

Code Listing 5.8: FC: abbreviated `Bird.ts` (full code: `http://tinyurl.com/zkb3ut6`).

The bird instance knows which keyboard buttons to listen to (`FLY_BTN`, `LEFT_BTN`, `RIGHT_BTN`), how to react to them and it provides some functions that enable interaction from outside (e.g. the previously mentioned `die`). The class is initialized via `constructor`[25], which adds the bird to the game at a certain position, defines the input keys, starts the flying animation and sets up the physics engine (`Phaser.Physics.Arcade`). Game loop `update`'s sole function here is to react to keystrokes sent by the Arduino:

- 'x' gives the bird a gravity boost, i.e. sets a negative y-axis velocity in the `flap` function.
- 'y' lets the bird fly to the left, i.e. adds a negative value to its current x-axis position in the `flyTo` function.

[25] A constructor must be used whenever it is necessary to pass variables to a class on creation.

- 'c' lets the bird fly to the right, i.e. adds a positive value to its current x-axis position in the `flyTo` function.

Altogether, the development of flappy cycling took roughly two weeks. JetBrains' WebStorm[26] served as an IDE during the implementation process, offering convenient debugging capabilities and TypeScript support. To optimize the prototype's loading performance while conducting the study, Google's closure compiler[27] was used in order to unify all JavaScript output files and minimize their code.

5.2.4 Study Setup

Since both studies aimed at gathering a widespread set of opinions, no specific audience was targeted for conducting them and they simply were set up in a public place: the AAU's aula. Like the preliminary study, this also can be classified as a supervised field study, although inevitably more unpredictable variables are introduced when proceeding in this fashion: the bias of questionnaire responses increases with reducing individual participant isolation due to the fact that bystanders could observe the players at will, potentially communicating with them. This, however, also can be considered a positive aspect because it is quite plausible that this situation resulted in discussions that triggered change of opinions through broadening the user's perception on the topic at hand.

The study equipment used for Flappy Bird was an accurately powerful personal computer, a laptop for filling out the accompanying questionnaire, a *high definition* (HD) beamer projecting the game onto a white wall and the hardware capturing user inputs consisting of ergometer as well as Arduino (see Figure 5.17). The game was hosted on a local web server[28] and run via Google Chrome[29] browser.

All 33 individuals, whose opinions were collected (see Appendix B, Subsection B.3.2), gained interest to participate either via their own curiosity while walking by or through explicitly being addressed by the supervisor. Used questions as well as obtained responses are listed Appendix B, Subsections B.1.1, B.2.1 and B.3.2. The study approximately was conducted within nine hours, including setup and disassembly.

[26] `https://www.jetbrains.com/webstorm/`
[27] `http://tinyurl.com/haconon`
[28] Used web server: Z-Wamp Server Pack (`http://zwamp.sourceforge.net/`).
[29] `https://www.google.com/chrome/browser/`

(a) Front view. (b) Back view.

FIGURE 5.17: Flappy Cycling: study setup.

5.3 AAA Game Integration: Quake 3 Arena Fitness Edition

Quake 3 Arena is the third installment of the popular Quake series[30], which included first-person shooters that have been among the first games featuring true real-time 3D game *engines*[31], developed by John Carmack [132]. Back then, his company named id software[32], having started out by creating controversial cult game series Wolfenstein 3D[33] and Doom[34], has been instrumental in defining the nowadays widespread FPS genre. Early on, shooters were liked for creating a feeling of immersion as single-player experiences, but they came to be loved for taking competition between gamers to the next level as multi-player challenges: serial, local area network (LAN) and Internet connections paved the way for *deathmatch*[35] marathons, where two or more people could hunt each other down through the game's fictional environments with the goal of virtually wiping out as many others as possible, i.e. scoring kills or *frags*. The era of FPSs truly had begun in the 1990s.

This rapidly growing demand for shooters also initiated another interesting development, which greatly influenced the reasoning behind choosing such type of a AAA game for the second study: because FPSs were so popular, people started doing more than just playing them – they created modifications or *mods*, where they altered the games in

[30] The Quake series on steam: `http://store.steampowered.com/sub/434`.

[31] Engines are the cores of video games, usually providing the developers with specialized frameworks to enable focusing on game creation instead of tasks like graphics rendering, physics etc.

[32] The company id ('in demand') sofware (`http://www.idsoftware.com`) was founded by 'rock star' programmers John Romero & John Carmack, game designer Tom Hall, and artist Adrian Carmack, each of whom have left by now.

[33] Wolfenstein 3D on Steam: `http://store.steampowered.com/app/2270`.

[34] Doom series on Steam: `http://store.steampowered.com/sub/18397`.

[35] According to 'Masters of Doom' by Dave Kushner [132], John Romero invented the name deathmatch when trying out multiplayer capabilities of Doom during development.

terms of visuals, mechanics or gameplay. Carmack picked up on this development and started making his engines more modding-friendly [132], which was very well received: passionate modding communities formed on the Internet. This custom of letting the players experiment with their acquired products still prevailed to this day – especially when regarding the shooter genre[36], rendering them the topmost worthwhile games for modding. This ultimatively has been the most influential factor in deciding to modify an FPS and choosing to create Quake 3 Arena Fitness Edition, due to its publicly available source code as well as personal interest.

The game was released in 1999 and thus may be deemed old, but at its core the shooter genre did not change much up until today – at least when considering multiplayer-focused releases like Quake 3: their notion of fun lies in wreaking havoc by shooting targets using a variety of weapons and competing against others in virtual environments. Furthermore, the game nowadays works on practically any computer and therefore retains a much higher portability than modern FPSs that are restricted to running on higher end machines. Lastly, the Quake 3 source code is structured in a good way, modular and, besides online resources, even a well written book is available [133], which is very helpful, since modding still is largely a community-driven activity.

Often called Q3 or Q3A for brevity, the game is largely multiplayer oriented, as the word 'Arena' in its title hints: up to 16 contestants battle each other in one of 30 arenas, also named levels or *maps*[37], much like gladiators did in ampitheatres during the ancient Roman Empire. Every participant controls a virtual character seeing through its eyes, i.e. experiencing a first-person view, being equipped with weapons, ammunition (ammo), items and armor, all of which can be acquired by venturing through a level[38]. Up to 200 health points (HP) determine a player's status – death occurs at 0 HP. HP are reduced by taking damage either from other players, who can inflict it through using the various weapon types[39], or from arena hazards, like water, poisonous gas and so on. As a protection against losing health quickly, a maximum of 200 armor points (AP) can be accumulated by looking for AP increasing items. Power-ups can also be picked up, provided that players are lucky enough to find them, and they give their collectors a variety of advantages for a limited period of time. Just like AP, it is also possible to recover HP during a battle by picking up specific items, most of which exhibiting a first aid cross. There are different game modes[40]: deathmatch or *free for all* (FFA) is the classic everyone-fights-everyone type of game where the goal is to rack up frags by

[36] E.g. Epic Game's cutting edge Unreal 3D engine comes with extensively moddable FPS Unreal Tournament 4 (`https://www.unrealtournament.com/tutorials`).
[37] Q3A map listing: `http://quake3.wikia.com/wiki/Maps`.
[38] Q3A pickup items listing: `http://www.quake2.com/QBitch/items.htm`.
[39] Q3A in-game weapons and ammunition listing: `http://www.quake2.com/QBitch/weapons.htm`.
[40] A more detailed FAQ: `http://tinyurl.com/j65hgbb`

killing others with an optional time limit; *team deathmatch* (TDM) is a slight variation that allows players to form two different teams that compete against each other and the team first reaching the frag limit wins; in *capture the flag* (CTF) both teams possess separately located, very specific starting points – their bases – which also include flags that can be captured by the enemy team – the goal is to fetch a flag, bring it back to your own base and reach the globally set flag capturing limit first.

The mod created for the study only considers FFA mode and since participants can only try out the game one at a time, non-human players also called *bots*[41] need to be added in order to enable playing the game properly. Bots already are part of Q3A and they can individually be set to five levels of difficulty, which allows increasing the game's challenge for more experienced players. Basically before trying out the modification a specific map, the amount of bots, their difficulty and the frag limit must be set. Additionally, several other variables have been integrated like the player's age and workout goal, which will be explained shortly. As was determined in the prestudy phase (see section 5.3.1), settings that offered a good challenge were choosing a small map (q3dm1), three bots with difficulty 2 to 3 and a frag limit of 10.

A new game begins by randomly putting the player as well as all bots on distinct starting spots set throughout the map, which is called *spawning*. Newly spawned players have 150 HP[42], a machine gun with 100 ammo, a close range or *melee* weapon (gauntlet) as a last resort if they run out of ammunition and no protective armor. Weapons, items and power-ups also have dedicated spawn points and they can be collected as soon as players enter the game. Killing an enemy credits one frag to the player and committing suicide, like executing oneself with a personally wielded weapon, i.e. dying through self-caused *splash damage*, decrements the affected person's kill count by one. After having been fragged, a player loses all acquired items and starts over by spawning again at a random spot only bearing the previously described initial equipment. As soon as one combatant reaches the overall frag limit or a previously set play time restriction runs out, the game is over and the top three players are displayed standing on a pedestal. Study participants played one such scenario with the aforementioned settings, while being rewarded for performing well on the ergometer.

Compared to flappy cycling, a lot more buttons are needed to control Quake 3: walking forwards, backwards or sideways (*strafing*), looking around (panning the camera), jumping, ducking, shooting, switching weapons, zooming and even more – the previously described Arduino circuits would not even provide enough input for plainly moving one's character in-game. Therefore, an additional input possibility needs to be provided.

[41] Q3A bots listing: `http://quake.wikia.com/wiki/Quake_3_Bots`.

[42] HP or AP greater than 100 slowly decrease over time until 100 is reached – this way players are encouraged to keep moving and have a lookout for items to improve their conditions.

Unfortunately, the most common input method – a keyboard in combination with a mouse – can not be realized easily in the study's setup because the employed ergometer does not include a suitable surface for placing those devices[43]. Subsequently, a gamepad directly plugged into the PC was employed, providing all the inputs needed to play. As mentioned before, holding such a game controller in turn renders the ergometer's pulse handles useless, as touching them requires letting go of the gamepad, which would be fatal in a fast-paced FPS like Quake 3. Since the pedals were the sole remaining means of capturing fitness input, a finger-attachable pulse sensor has been added, previously depicted in the hardware overview segment (Figure 5.2). The resulting architecture is depicted in Figure 5.18.

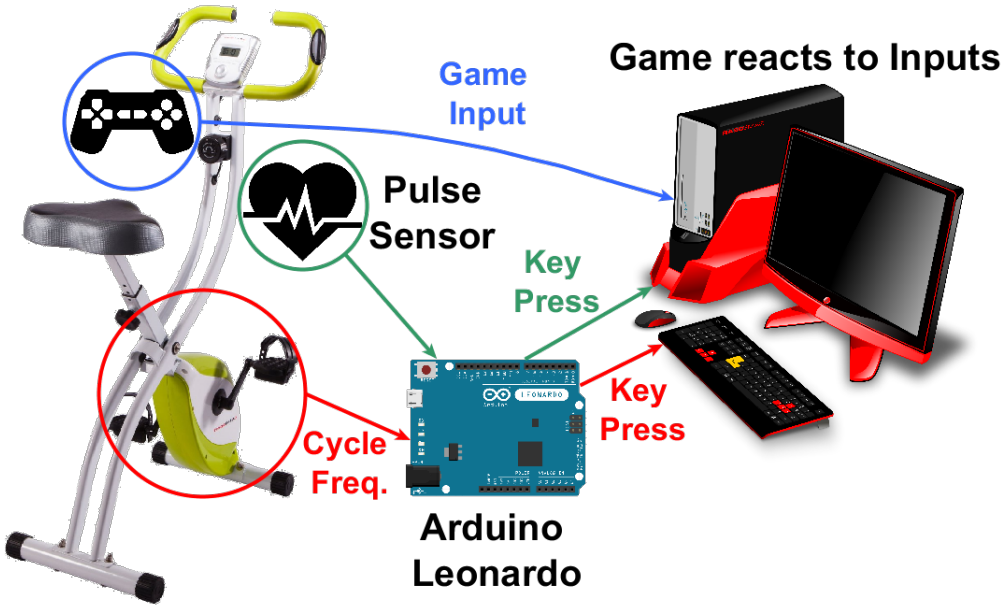


FIGURE 5.18: Architecture of Quake 3 Mod.

Introducing a pulse sensor to measure a participant's heart *beats per minute* (bpm) in addition to capturing the pedal's rpm enables the creation of a fitness experience that can adapt to the person's workout goals: cycling influences the controlled character's running speed[44], while heart rate target zones for the workout can be measured with the pulse sensor, permitting a customizable way of distributing in-game items, weapons and power-ups. For measuring HR zones, the maximum allowed HRs of the participants must be calculated, which requires knowing their age – a necessary inquiry prior to playing the mod. Many formulas[45] exist for determining such an upper pulse limit, but most do

[43] However, possibilities do exist: fitness devices that enable exercising while working would have been viable options for the study (E.g. `http://tinyurl.com/h64ncl2`).

[44] Note that directly controlling the in-game player movement with the pedals would be a bad idea due to the fact that playing an FPS often requires stop-and-go motions – ceasing to cycle would interrupt the workout.

[45] A small selection: `http://www.brianmac.co.uk/maxhr.htm`.

not differ much in their results. The formula used for the study has been proposed by Jackson et al. [134] and is deemed to reflect the HR-age relationship more accurately than the commonly utilized standard formula $(220 - age)$:

$$max_{HR} = 206.9 - (0.67 \times age) \tag{5.1}$$

The target HR zones are calculated in relation to max_{HR} – usually three different regions are identified and for the Q3A mod they have been defined as shown in Table 5.1, which also lists the mod's policy on item reward frequencies or *drop rates* according to preselectable target HR zones, offering a way to set specific goals for the workout.

TABLE 5.1: Quake 3 Mod Policy: pulse bpm in relation to in-game item drops. Source: based on Table 'Classification of Physical Activity Intensity', Fletcher et. al [135] (pg. 1719)

HR (% of max_{HR})	**zone**	**intensity**	**drops per target zone (%)**		
			recovery	aerobic	anaerobic
<45	low	very light to light	0	0	0
45–60	recovery	light to moderate	30	20	15
60–75	aerobic	moderate to hard	20	30	20
75–90	anaerobic	hard to very hard	15	15	30
>90	high	very hard to max.	0	0	0

Per default an aerobic workout was selected for study participants, which provides a moderate up to hard challenge: 60 to 75 percent of the maximum calculated heart rate (max_{HR}). In case a player stays inside this HR zone, items, weapons as well as power-ups are rewarded constantly – at a rate of 30%, which according to the employed Formula 5.2 happens every 3.333 seconds[46].

$$interval_{drops} = \lfloor \frac{1}{rate} \times 1000 \rfloor \tag{5.2}$$

Furthermore, rewards also are dispensed when the pulse rate is within one of the other zones, although less frequently with a pace of 20 or 15 percent. Thus, exercising towards the set goal yields the most benefit for the user. For the remaining cases – falling below the recovery or above the anaerobic zone – no rewards are given. Instead for working out too hard, i.e. having a pulse rate greater than 90% of max_{HR}, punishments are dealt

[46] $\lfloor \frac{1000}{0.3} \rfloor = 3333$ ms

out: with a 50:50 percent chance the player loses armor or health points in increasing intensities.

Since some in-game items are more desirable than others and a simple random reward scheme would most likely overpower the player, a hierarchy of item qualities and their probability of being selected as rewards, i.e. their *drop chances* are shown in Table 5.2.

TABLE 5.2: Q3A Mod Items: types, qualities and drop chances (A: ammo, U: useable).

	quality (drop %)		
type (drop %)	low (60%)	medium (30%)	high (10%)
item (50%)	AP+5	AP+50	AP+100
	HP+5	HP+50	HP+100
	Shotgun (A)	HP+25	Medikit (U)
	Machine Gun (A)	Lightning Gun (A)	BFG (A)
		Grenade Launcher (A)	Rocket Launcher (A)
		Plasma Gun (A)	Railgun (A)
weapon (30%)	Shotgun	Lightning Gun	The BFG
	Machine Gun	Grenade Launcher	Rocket Launcher
	Gauntlet	Plasma Gun	Railgun
power-up (20%)	Regeneration	Haste	Quad Damage
	Teleporter (U)	Invisibility	Battle Suit
			Flight

Items are rewarded with the biggest probability of 50 percent and as a further segregation different qualities are considered: if an item is rewarded, there is a 60 percent chance that it will be a low quality item. After being killed, as in the unmodified Q3A, all rewards are lost except power-ups, which are dropped on the ground and can be picked up by enemies enjoying its active remaining time.

To sum up and giving a complete example of how the reward system works, following step by step guide should be considered:

1. A player discloses his or her age and the Q3A mod is started with optional settings, including choosing a target HR of recovery, aerobic or anaerobic.

2. The mod calculates max_{HR} and decides to reward items according to the fixed drop rates for the selected target HR (Table 5.1) and the measured pulse sensor

input HR. For example, if the target HR is set to recovery and the measured HR is in the anaerobic zone, then the drop rate is 15% or every 6.666 seconds.

3. Whenever a reward is due, the resulting item is selected according to Table 5.2. For instance, in a lucky event a *Big Fucking Gun* (BFG) is chosen as a reward, which would first require the selection of a weapon – an incident happening with a 30% chance. In turn, the game had to pick a high quality item bearing a 10% probability and finally the gun would have to be randomly chosen out of three weapons with a likelihood 33%. When calculating the probability of these independent successive events, the overall likelihood would merely be next to 1%: $P(weapon, high, BFG) = \frac{3}{10} \times \frac{1}{10} \times \frac{33}{100} = \frac{99}{10000} = 0.0099$.

Regarding the second fitness input – the pedaling speed – all in-game calculations are based on the recommended average bike riders' cycling speed or *cadence*, as is shown in Table 5.3, which also describes the measures taken by the Q3A mod depending on the cycling rate.

TABLE 5.3: Quake 3 Mod Policy: ergometer rpm in relation to in-game running speed. Source: based on Ryan Haas for Livestrong, 2013 (`http://tinyurl.com/gqgvtr4`)

pedal (rpm)	**zone**	**cadence**	**run speed (%)**
<50	low	too low	30
50–70	acceptable	just below average	60
70–100	recommended	just right	100
>100	high	higher than average	130

Participants exercising within the recommended pedaling frequency of 70 to 100 rpm will be able to move with full speed (100%) in-game. Accordingly, cycling slower reduces the character's speed by factor 0.6 or even 0.3. The running speed can even be increased beyond 100% if the players pedal faster, producing more than the recommended rpm. In contrast to the high HR zone, there is no penalty for going beyond the recommended zone here because on one hand it is not a good idea to restrict the cycling rate, due to possible personal preferences of the participants and on the other hand the HR cap at 90% of max_{HR} should suffice as a penalty if pedaling becomes too exhausting.

This concludes the basic description of the mod. A prestudy test, which is covered in the next section, was administered at the ITEC institute to determine possible flaws of the game before conducting the study.

5.3.1 Pre-study Tests and Refinements

Pre-study participants exclusively consisted of fellow ITEC colleagues and friends. Their input has been very helpful in deciding whether already implemented game components should be changed or if additional ones were still missing. The tests especially facilitated identifying striking flaws that ought to be corrected before the main study.

A big concern of several testers was the rather naive way of switching weapons automatically: any possibly rewarded weapon has been immediately equipped for the player, which can be very confusing, if not even resulting in a disadvantage – oftentimes weapon changes have occurred when players at the time wielded a more powerful gun than the newly received one[47]. Furthermore, Q3As hidden weapon – the grappling hook[48] – was included in the mod but turned out to be very annoying, especially since it can not cause any damage and when automatically selected, players are more or less defenseless until they manually change weapons again. Subsequently the hook has been removed from the mod and a smarter way of auto-switching weapons was devised: in the final version employed in the main study, weapon types are part of a hierarchy and only the ones considered stronger are switched automatically.

Another shortcoming was calculating the heart rate across every heart beat, which could result in the determined data changing much too quickly, especially when the measured pulse is not steady, i.e. the rate rapidly increases or decreases due to different levels of exertion or premature heart contractions[49] are present. As a countermeasure, the finished mod uses an adjustable amount of heart beat samples for calculating the average pulse, which relaxes the transition between target HR zones. Additionally, as has also been suggested in this test phase, the maximum HR as well has been made adjustable in-game, in order to be able to compensate for sportive persons' advanced capabilities, i.e. their overall higher performance at lower pulse rates.

In contrast to the HR, rpm calculation across every cycle has been left unchanged, since it produces a very responsive feeling of controlling the in-game speed of the character. Only a single small modification has been made in order to dampen the effect of falling below the recommended cadence: as Table 5.3 already reflects, an additional acceptable cycling region (50–70 rpm) prevents a character speed drop from 100% to 30% by introducing a 60% zone in between.

[47] Q3A's policy on picking up weapons that are already owned is simply to add corresponding ammo. Nevertheless, in this particular case a weapon switch still occurred.

[48] A grappling hook was once intended to be part of Q3A but was removed before the release. It allows players to shoot a hook against any wall and in succession pull themselves towards it, e.g. to reach higher levels of a map.

[49] Premature heart contractions are called *extrasystoles*. They cause the HR to pause for a short period of time, usually rendering the first following beat much stronger.

An option to disable all item spawns in levels has been added to the mod, which, as no items can be picked up any more, creates the need for giving the bots a small chance of getting rewards (10% or every 10 seconds) and by doing so, ensuring a less unfair fight. However this majorly influences the bots' behavior: they remain motionless until either they are attacked or see a target, which often causes them to apathetically stare into walls and only reacting normally when attacked. Apparently, their *artificial intelligence* (AI) routine uses collectable items scattered throughout the map as waypoints. Though this can nevertheless be challenging, lifeless bots never paint a good picture and hence the item drops were always turned on during the main study.

Finally, as mentioned above, a good mod configuration for the main study has been determined: 3 bots on difficulty 2 to 3, depending on the participants' experience with FPSs and a frag limit of 10.

5.3.2 Implementation

Plugging the pulse sensor into the Arduino is straightforward, as shown in Figure 5.19.

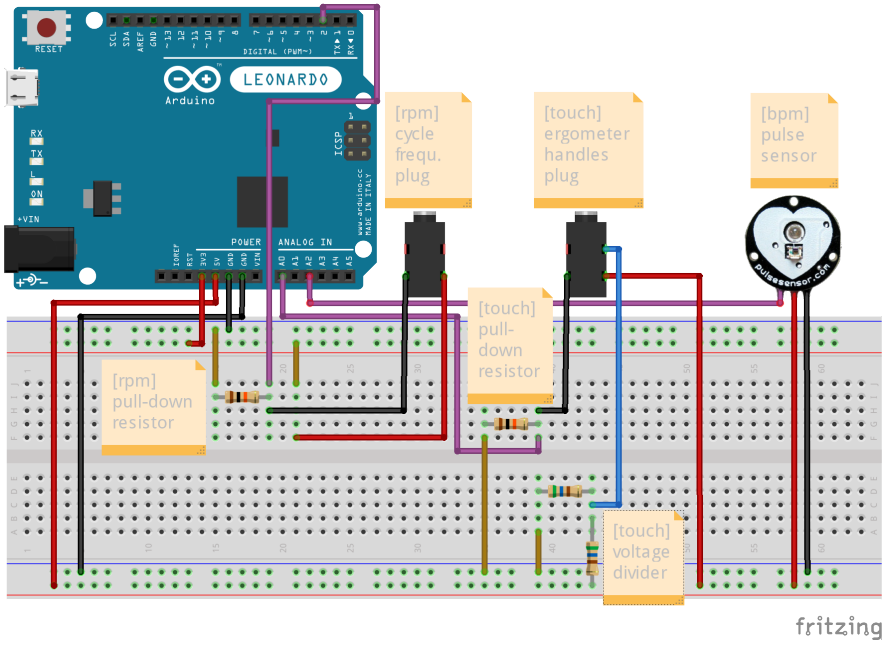


FIGURE 5.19: Arduino circuit refinement for capturing pulse information. Created with Fritzing (`http://fritzing.org`)

For completeness, this final fritzing sketch includes all circuits utilized throughout the thesis[50]. The HR sensor is attached to the microcontroller's 5V power supply (red cable), GND (black cable) and its plug for passing captured HR data (violet) is connected to the analog A3 input pin.

[50]Since the Q3A did not make use of the ergometer handles circuit, its functionality has simply been disabled in the Arduino code.

Internally, for measuring the heart rate, the sensor uses two *diodes*[51], one being a *light-emitting diode* (LED) producing green light, which is reflected by the blood stream running through a limb in direct contact, i.e. a finger or an ear. The other one, called *protodiode*, is sensitive to these returning light waves and their amount varies depending on the volume of blood flowing. Thus, by additionally filtering out unwanted noise, i.e. electromagnetic interference etc., the pulse rate can be determined fairly accurately by analyzing the analog voltage fluctuation that is passed to the microcontroller, identifying a heart beat's distinct wave shape. Devices using this technique for determining a person's HR are called *photoplethysmographs*, generating *photoplethysmograms* (PPGs) when plotting the magnitude of their produced signal for a specific amount of time.

The accompanying Arduino code is provided by the sensor's developers[52] and comprises several, well-documented files (`AllSerialHandling.ino`, `Interrupt.ino` and `Timer_Interrupt_Notes.ino`), which can easily be added to an existing project by placing them in the corresponding directory. To ensure a steady, high resolution sampling rate of 500 MHz (every 2 ms) the implementation operates by periodically throwing interrupts through a specifically set up hardware timer[53], which needs to be handled individually for different microcontrollers and hence as well requires particular settings for the Arduino Leonardo's ATmega32u4 processor running with a 16MHz clock[54].

Code Listing 5.9 contains the necessary modifications made to the main program, omitting the implementations for the other circuits and outlining most important sections of the newly added HB detection algorithm[55], contained in `Interrupt.ino`.

```
1 /* globals (heart rate) */
2 int pulsePin = 2;                 // P.Sensor purple wire plugged to A2
3 volatile int Signal;              // holds incoming raw data
4 volatile int IBI = 600;           // time interval betw. beats (ms)
5 volatile boolean Pulse = false; // true: live HB. false: not live
6 volatile boolean QS = false;     // true when beat found
7
8 /* init */
9 void setup()
```

[51] Diodes conduct electric current in one direction and can therefore be used to modulate, rectify, mix or emit signals, regulate voltage etc.

[52] World Famous Electronics Github: `http://tinyurl.com/h7lbs8p`.

[53] This is called describing an Arduino *interrupt service routine* (ISR): `http://tinyurl.com/o6lxjyn`.

[54] This can be achieved by modifying the `interruptSetup` function in `Interrupt.ino` as documented in the additional notes file `Timer_Interrupt_Notes.ino`.

[55] Please note that in this form the code is not runnable because all interrupt setup implementations and sections have been omitted for brevity – refer to the developers tutorial for more elaborate explanations of the code (version `V1.2`) as well as specifics about the sensor's mode of operation (`http://tinyurl.com/za8ucjk`).

```
{
    Serial.begin(115200);        // comm. rate p. second (baud)
    interruptSetup();            // setup P.Sensor sig (read every 2mS)
    Keyboard.begin();            // start emulating keyboard
}

/* main loop */
void loop()
{
    if (QS == true)              // HB found!
    {
        Keyboard.write(104);     // press and release 'h' key
        QS = false;              // reset Quantified Self flag
    }
}

/* Interrupt.ino contents */
volatile unsigned long sampleCounter = 0; // for pulse timing
volatile unsigned long lastBeatTime = 0;  // used to find IBI
volatile int P = 512;                     // peak in pulse wave
volatile int T = 512;                     // trough in pulse wave
volatile int thresh = 512;                // find instant moment of HB

/* interruptSetup() - omitted: sets ISR Timer1 */
/* ISR routine: called after Timer1 expires (every 2ms) */
ISR(TIMER1_COMPA_vect)
{
    cli();                                  // tmp. disable interrupts
    Signal = analogRead(pulsePin);          // read the Pulse Sensor
    sampleCounter += 2;                     // track time
    int N = sampleCounter - lastBeatTime;   // time since last HB
    //  find the peak and trough of the pulse wave
    if(Signal < thresh && N > (IBI/5)*3)    // wait 3/5 of IBI (noise)
    {
        if (Signal < T) T = Signal;         // lowest wave point
    }
    // thresh condition helps to avoid noise
    if(Signal > thresh && Signal > P)  P = Signal; // highest point
    // Look for HB: signal surges up in value on every pulse
```

```
49      if (N > 250) // wait 250ms filtering noise (limit: 240 bpm)
50      {
51          if ((Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3))
52          {
53              Pulse = true;                       // flag occurred pulse
54              IBI = sampleCounter - lastBeatTime; // ms betw. beats
55              lastBeatTime = sampleCounter;       // log pulse time
56              QS = true;                          // signal HB
57              // QUANTIFIED SELF FLAG NEEDS TO BE CLEARED IN MAIN LOOP
58          }
59      }
60      if (Signal < thresh && Pulse == true) // beat over if vals drop
61      {
62          Pulse = false;          // reset Pulse flag to start over
63          int amp = P - T;        // get amplitude of the pulse wave
64          thresh = amp/2 + T;     // set thresh at 50% of amplitude
65          P = thresh;             // reset these for next time
66          T = thresh;
67      }
68      /* reset vars to default vals if HBs stop (2.5s) - omitted */
69      sei();                      // enable interrupts again
70  }
```

Code Listing 5.9: Arduino code segment for capturing pulse information (full code: `http://tinyurl.com/h6e9qdw`, `http://tinyurl.com/zho3we9`).

The originally provided implementation actually determines the bpm based on several detected HB samples, which the Q3A mod does not require – for sending keypresses it suffices to merely recognize occurring beats. In order to find such events, the pulse wave needs to be analyzed measuring a short, rapid rise in the signal – an instantaneous moment of a beat occurs at some point[56] during this growth in signal strength, identified by crossing a threshold (`thresh`) in between the corresponding lowest (trough, `T`) and highest (peak, `P`) amplitude level. The first part of the code simply deals with sending a keyboard signal to the attached PC whenever an HB is found, i.e. the *Quantified Self*[57] (`QS`) variable is set to `true`. When attaching an interrupt routine like shown in the listing, its modifiable variables must be declared `volatile`, which tells the compiler

[56] The notion of when exactly an HB occurs is much argued over – the sensor developers use 50% of the pulse amplitude.

[57] Quantified Self is a movement accompanying the rising desire of individuals to log and analyze their daily life with devices like fitness bracelets, smartphone apps or even clothes enhanced with sensors (`http://quantifiedself.com/`).

to load them from memory instead of storage registers, preventing possible inaccuracies from happening[58]. Additionally to the input pin number 2 (`pulsePin`), several other variables are initialized: `Signal` for holding the read signal , the *inter beat interval* (`IBI`) storing the time between pulses, an indicator for an ongoing HB (`Pulse`) and `QS`. The main loop basically waits for the interrupt routine `ISR` to signal HB, resetting `QS` afterwards to be able to detect a subsequent beat – `ISR` does not depend on the flag, it merely is used as one-way communication with `loop`.

Before defining `ISR` it is necessary to initialize additional variables and set up the timer acting as the 2 ms delay between periodical routine calls, which happens in the `interruptSetup` function executed at startup – its implementation is omitted here because it only involves setting a few HW registers. The `sampleCounter` variable counts the time passed, while `lastBeatTime` remembers the moment of the last HB, initially set to `0`. `P` and `T` are initialized at half of the maximum signal strength ($\frac{1024}{2} = 512$). The threshold (`thresh`) takes the same value, but, ultimately, after every HB, it is dynamically adjusted to a value resembling 50% of the last amplitude between `P` and `T` (line 64). While doing its calculations, the interrupt routine `ISR` temporarily disables all further interrupts with the function `cli`, afterwards re-enabling them again by calling `sei`. The signal is read and the elapsed milliseconds since the last HB are calculated (`N`), which enables filtering out high frequency noise by delaying the beat detection routine (line 49). Subsequently, `T` and `P` are determined, which involves introducing another delay ($\frac{1}{5}$ of `IBI`) removing possible noise[59]. Since the threshold should represent a signal level within the interval [`T`,`P`], the trough can never be higher than `thresh`, just as the peak must not be lower than that value.

Consequently, an HB check occurs after every 250 ms, limiting the maximum possible rate to 240 bpm. Beats are found if the signal crosses `thresh` and $\frac{5}{3}$ of the previously measured IBI (initially `600`) are bypassed, which carries the same functionality as it does when determining `T` (line 44) – filtering out noise and false readings. To prevent detecting multiple HBs, `Pulse` is set to `true`. Then, the interval between last and current beat is stored in `IBI`, `lastBeatTime` is set to the present time, only leaving one final action – raising `QS` in order to notify the main loop. When the signal falls below `thresh` again (line 60), the HB is considered over and preparations for the next one can commence: `Pulse` is turned off, while half the amplitude of [`T`,`P`] determines the new values of `thresh`, `P` and `T`, requiring the algorithm to start over. A few cleanup actions are needed in case no HB occurs for long time period (>2.5 s), which simply implies resetting all variables to their initial state – a set of instructions that as well has been omitted for abbreviation.

[58] See Arduino reference for more information: `http://tinyurl.com/hhga35h`.

[59] Besides noise, after every pulse there is a steep decline of the signal falling below the normal signal level between beats, which is called the *dichrotic notch* and it can lead to false readings if no delay is set.

The preceding pulse sensor code description finalizes all Arduino related implementation aspects. This section's remainder is dedicated to outlining technical details of Quake 3 Arena and mod implementation specifics.

At its core, Q3A works with the idTech3 engine, which, for a large part, builds on the previous Quake 2 engine (idTech2). Throughout developing the three games, John Carmack's engines went through big evolutions in order to adapt to changing hardware and software [133]: especially the transition from Microsoft's *Disk Operating System* (DOS) to 32-bit Windows (Win32), which already started after id had released Quake (idTech1) around 1996, and the adaptation to upcoming GPUs[60], speeding up Quake 2's CPU[61]-processing using *OpenGL*[62], required large code adjustments. As already mentioned, like all id's shooters, the Quake series as well was made modding-friendly from the beginning, which did not imply that the entire source code had to be released: merely the isolated game logic needs to be publicly available, while the engine itself can remain proprietary. This concept assured id's success over the years, although eventually, they always published all source code of their games[63].

Accordingly, modding demanded modularity and therefore Win32 compatible Quake releases[64] make use of *dynamic-link libraries* (DLLs), which allow dynamically integrating interchangeable code into the engines. Thus, after modifying the freely obtainable game logic code, users can create such DLLs, place them in a new subfolder below the game directory and point the executable file to it by setting startup parameters. This, however, also poses a security threat to the *operating system* (OS): DLLs can contain any kind of code, hence, it is also possible to include malicious instructions like formatting the hard drive, leaving people with anything but trust when downloading a brand new mod from the Internet. Therefore, Q3A offers a safer approach: the engine is running *quake virtual machines* (QVMs) that essentially contain the same code that could reside in a DLL[65] file but are only interpretable by the game, lacking the ability to call Win32 functions. For this cause Carmack made a custom compiler that translates game code, written in *American National Standards Institute* (ANSI) C, into QVM bytecode: the

[60] *Graphics Processing Units* (GPUs) are highly parallel, multi-cored units that are added to a system in order to speed up graphic-intensive calculations.

[61] *Central Processing Units* (CPUS) are the heart of a computer, capable of performing instructions given in machine code at a certain rate.

[62] *Open Graphics Library* (OpenGL), an open programming *Application Programming Interface* (API) for graphics rendering (`https://www.opengl.org`)

[63] John Carmack always has been of the opinion that anybody should be given the opportunity to learn from the work of others, as he himself grew up living by the *hacker ethic*, a community driven movement above all promoting freedom of information [132].

[64] In 1997 WinQuake was released, making the whole series Windows compatible.

[65] Note that it is still possible just to create DLLs for Q3A and ignore QVMs, which is important for mod development, as creating QVMs is not integrated into the IDE projects (Microsoft's Visual Studio) and must be done via command line.

cross-platform *Little C Compiler*[66] (LCC) creates text-based assembly files that are merged by id's `q3asm` tool and transformed into binary format to gain runtime speed. The executable file (`quake3.exe`) uses a bytecode interpreter to integrate the generated QVM files – three of them are loaded at any time: one acting as a server (`game`), another operating as a client (`cgame`) and a third realizing the in-game menu (`q3_ui`). The complete engine architecture is depicted in Figure 5.20.

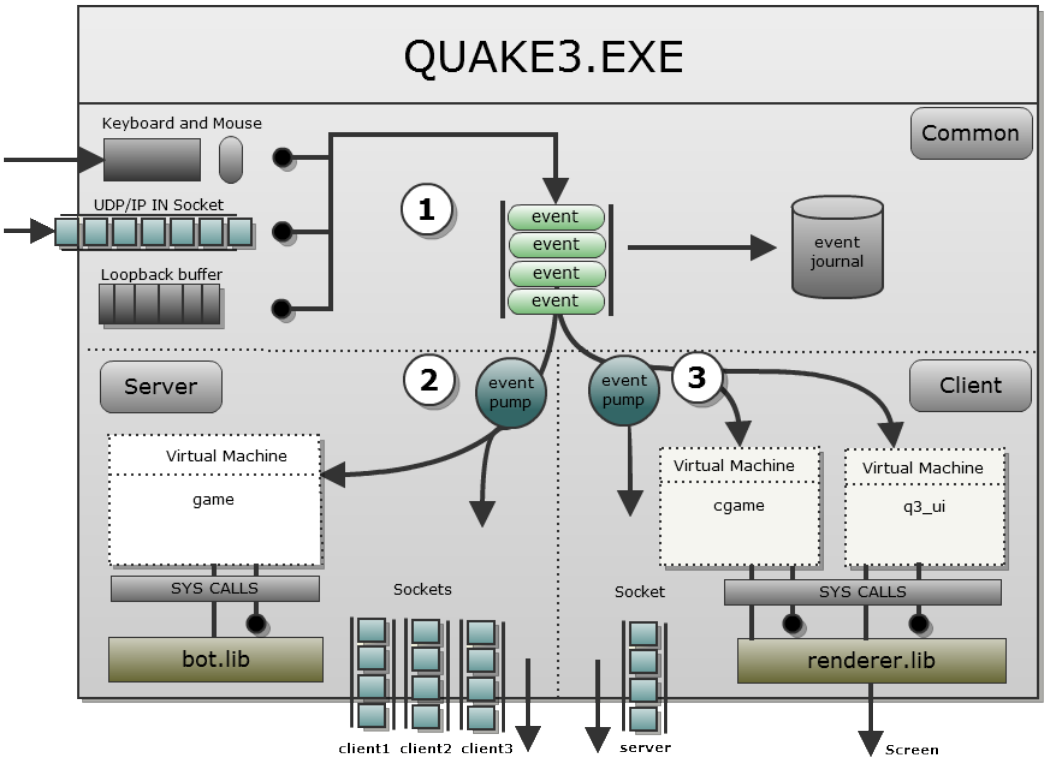


FIGURE 5.20: Q3A game architecture.
Source:from Fabien Sanglard, 2012 (`http://tinyurl.com/8ay5f5z`),
used with permission.

Looking at its structure, idTech3 can be viewed as a tiny OS (`quake3.exe`), offering *system calls*[67] (sys calls) to three processes[68] (`game`, `cgame`, `q3_ui`). All inputs, i.e. keystrokes, mouse moves and network packets (structures of type `event_t`) are part of a single system event queue (`sysEvent_t eventQue[256]`), which creates the possibility of keeping a chronological journal, logging all occurring events in order. Among other benefits, this feature adds a big advantage for debugging during development because it greatly facilitates recreating and analyzing not intended behavior step by step[69]. Q3A's client and server components are strictly separated and only communicate with each

[66] LCC, a retargetable compiler for ANSI C (`http://tinyurl.com/d9cbelo`).
[67] System calls are important functions for essential tasks and they are provided by the core of an operating system (kernel) for programs to use.
[68] Fabien Sanglard's Q3A code review, 2012 (`http://tinyurl.com/8ay5f5z`).
[69] John Carmack's public Q3A `.plan` file: "Oct 14, 1998" (`http://tinyurl.com/hv6cwe5`).

other via *User Datagram Protocol*[70] (UDP) *sockets*[71], which suits the fast-paced shooter for most of the exchanged data because no expensive retransmissions of information that might already be outdated can occur: servers keep track of the game state that is propagated to clients via *delta compression*, i.e. only the difference to the last state known by a client is transferred, reducing the transmission overhead. Regardless of a running game's purpose – hosting a session, taking part in it (multiplayer) or both (singleplayer) –, the game's executable receives and dispatches all occurring events either to the locally loaded virtual machines (VMs) or to other game instances via network sockets. The event receiving VMs react accordingly and are able to execute sys calls that `quake3.exe` receives, before determining further proceedings like calling functions in `renderer.lib`, the client-side OpenGL rendering library. The server-side linked library `bot.lib` contains the bot AI controlled by `game`.

The full source code of Quake 3 Arena can be found on id's GitHub account[72]. Besides LCC and q3asm, it comprises 559 files containing 233 952 LOC. A project file for Microsoft's Visual Studio 7 (VS7) is included and, when opened in the IDE, eight sub-projects are displayed, the purposes of which are roughly outlined in Table 5.4.

TABLE 5.4: Q3A: Visual Studio 7 Projects
(DLL: dynamic library, LIB: static library, QVM: bytecode).
Source: based on "Quake 3 Source Code Review: Architecture", Fabien Sanglard (`http://tinyurl.com/8ay5f5z`)

project	type	description
`botlib`	LIB	Bots and AI. Used by server.
`cgame`	DLL / QVM	Client – predicts positions of objects (*entities*) for latency compensation and handles display using renderer library.
`game`	DLL / QVM	Server – maintains game state and sends updates to clients. Administers bots.
`q3_ui`	DLL / QVM	Client user interface (UI) – active when in-game menus are displayed, used to start new sessions, change settings etc.
`quake3`	Executable	Main process.
`renderer`	LIB	OpenGL based rendering library. Used by client.

[70] UDP is a connectionless transfer protocol that does not retransmit lost packets, i.e. is unreliable.
[71] Sockets are network endpoints, addressed with a combination of *Internet Protocol* (IP) address and *port* number.
[72] id Software's GitHub: `http://tinyurl.com/zgqknfv`.

`Splines`	LIB	Not used – implemented experimental scripted cameras in very early versions.
`ui`	DLL / QVM	Additional UI code for Q3A expansion Team Arena (TA) (unless TA explicitly activated, not used).

As can be guessed, the most important projects for modding are `game`, `cgame` and `q3_ui`. The server (`game`) knows the true state of a running game and all its rules, while clients (`cgame`), whenever possible, can make predictions about events: for example, knowing a fired missile's origin and angle enables the client to calculate where its impact is going to be, which is called *interpolation*. This way no bandwidth is wasted for transmitting unnecessary data and the client can piece together its own view of the world, which, due to its capabilites, can be updated less regularly by the servers absolute knowledge, delivered via omniscient game state *snapshots* in so-called *authoritative updates*. Besides predicting the future, the `cgame` VM also controls rendering during combat, including the display of a map leaderboard, loading screens and the *heads-up display* (HUD), which comprises all on-screen icons, text and notifications. Lastly, the user interface VM (`q3_ui`) handles the menu display, providing clients with options like changing game settings, customizing controls or joining servers. Therefore, it can be modified to create custom menus or change the existing ones.

For the most part, the study related modifications are rules that can be applied to the server project. With the exception of one small adjustment, namely smart automatic weapon switching, the client has not been touched, neither the user interface. Table 5.5 lists all 14 modded files with corresponding links to their locations in the mod project's GitHub repository[73], the amount of code added or changed and a description of their in-game purpose.

TABLE 5.5: Q3A mod: modified files (full project: `http://tinyurl.com/hrdhh5l`). Sources: based on Q3A inline comments and [133]

file (URL)	+LOC	description
server (`game`)		
`bg_public.h` (`tinyurl.com/zqoymyp`)	19	Definitions shared by both server and client modules.

[73]Changes of any kind to the original files have been enclosed within [ERGO MOD START] and [ERGO MOD END], or commented via [ERGO MOD] (`http://tinyurl.com/jhas5hl`).

q_shared.h (tinyurl.com/h8qlz5m)	2	Included first by nearly all program modules.
g_local.h (tinyurl.com/zhh7r6p)	100	Local definitions for server module.
bg_pmove.c (tinyurl.com/jte7j8l)	1	Controls player movement like running, walking or swimming for both modules.
g_active.c (tinyurl.com/gqtvbco)	341	Processes player think events like taking damage, using items or activating triggers.
g_client.c (tinyurl.com/hku2q2k)	79	Handles player spawn and client's connections to the server.
g_cmds.c (tinyurl.com/jk8kbmh)	113	Regulates server-side console commands like sending messages to the client, such as giving the player an item or calling for a vote.
g_items.c (tinyurl.com/jncop77)	16	Specifies how items like weapons or power-ups are picked up, dropped and respawned in the game.
g_session.c (tinyurl.com/zbk4t5u)	26	Stores as well as retrieves information that needs to be persistant across multiple level loads and tournament restarts.
g_spawn.c (tinyurl.com/z4uwh4c)	14	Controls spawning entities.
g_utils (tinyurl.com/gtdgbjd)	26	Contains utility functions used in many parts of the server module.
client (cgame)		
cg_local.h (tinyurl.com/hf5pxf8)	2	Local definitions for client module.
cg_consolecmds.c (tinyurl.com/h9ncmx2)	3	Regulates client-side console commands that are either typed-in or bound to keys.
cg_event.c (tinyurl.com/hlwjdjs)	18	Handles entity events at snapshot or player state transitions.
total +LOC	760	

In general, every object added to a Q3A map is an entity (`gentity_t`[74]), including players, bots, weapons or even explosions [133]. For instance, the launch of a grenade involves picking a free slot[75] in the level's entity array (`g_entities`) and turning it into a grenade by adding all corresponding properties[76]. This also implies setting its behavior, i.e. how it "thinks". Entites periodically think and, to regulate the way they think, two variables are set whenever they are created: the `think` function pointer defines the object's behavior, while `nextthink` determines its consecutive thinking interval. In case of the grenade, `nextthink` is set to 2.5 seconds and the `think` function called is `G_ExplodeMissile`, which, as its name suggests, destroys the entity, while dealing out damage to other entities in the proximity. Of course the projectile can also be destroyed prematurely, i.e. before `nextthink` runs out: for example, the collision with another player would as well trigger `G_ExplodeMissile`.

Perhaps the most heavily used entity in any Q3A mod is the player or client (`gclient_t`), since everything else revolves around it – even the bot AI internally controls the same entity instead of having its own. Evidently, judging by the considerable amount of code altered in client-related files, also the modifications proposed for the study are tightly connected with this entity. However, since an elaborated reward system is part of the mod, some thought had to be put as well into working with another object: the in-game item (`gitem_t`). Luckily, handing out rewards is a functionality that is already implemented as part of game's developer mode[77] and hence is reusable without too much effort. The mod implementation can be split into four parts, which will be outlined in the rest of this section by describing selected code passages: changing the client's properties, introducing new commands for keypresses, reacting to the ergometer's input and realizing smart automatic weapon switching.

As can be expected, Q3A's player contains very simple attributes like health, armor and an array of weapons that are currently held. Yet, its duties reach far beyond those elementary aspects and what it is composed of depends on whether regarded by server or client: a client's purpose is to render the models of players as well as predicting possible states, while the server, as being the coordinator, always knows the real player states and carries additional information like tournament statistics, chosen CTF teams or map voting counts. Their interests overlap in a structure named `playerState_s`, which is constantly communicated between the VMs and defined in the shared header

[74] The developers decided to add '`_s`' suffixes to all created structures, called *structs*, and chose to define corresponding aliases using '`_t`' suffixes via `typedef`.

[75] Note that at maximum 1024 entities are allowed in a level, which is appropriate, since a packed game with 16 players, 50 item spawn points and flying projectiles only make for close to 100 entites.

[76] see `fire_grenade` in `g_missile.c` (`http://tinyurl.com/hufed2q`).

[77] Cheating in this way can be activated by starting a game with the "-devmap [MAPNAME]" option. Given the exact name string, an item can be rewarded to a player by typing "\give [ITEMNAME]" into the in-game console.

file `q_shared`. Since the implemented mod only slightly is concerned with the client side, all further references to the player will relate to the server's point of view.

Apart from being itself contained in `gentity_s`, the server's client structure `gclient_s`, defined in `g_local.h`, comprises three very important structs discerning between different states of the game: `playerState_s`, `clientPersistant_s` and `clientSession_s`. Variables belonging to these structures are only valid for very specific amounts of time before they are reset, which either happens on every player respawn, map change or reconnect. Figure 5.21 illustrates the simplified client structure and also lists most important variables used by the mod.

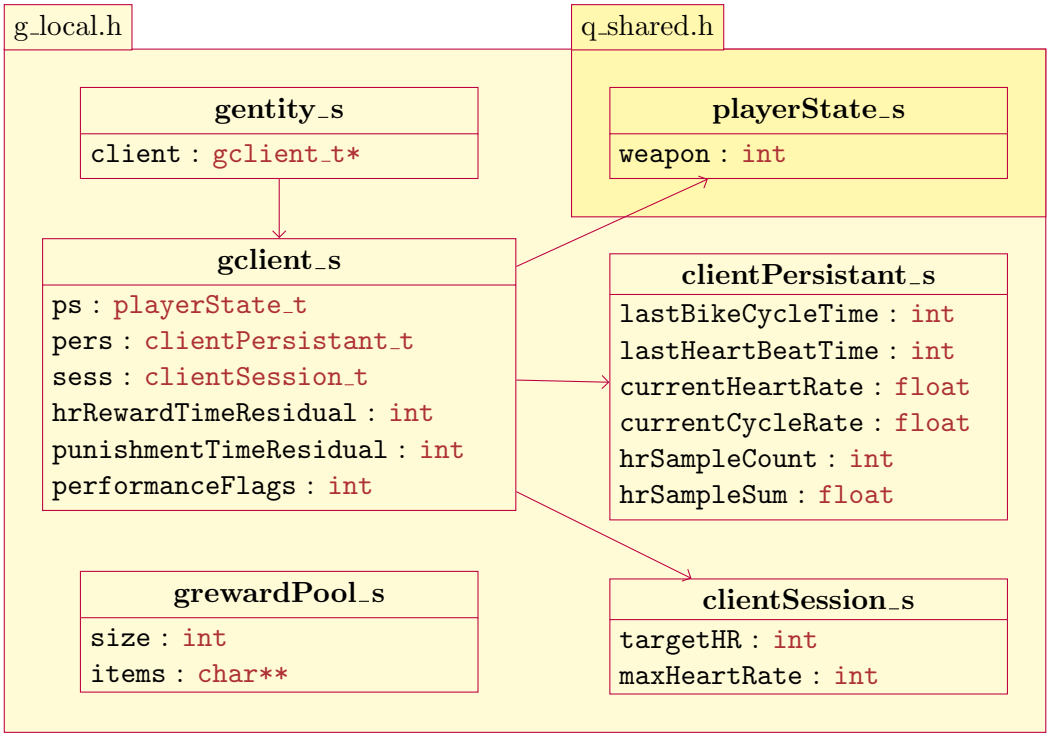


FIGURE 5.21: Q3A mod: server-side client structure.

Since `playerState_s` is communicated between server and client, changes to the struct should not be made lightly – doing so entails having to adjust the communication definitions (`msg.c`) inside the project of the main executable (`quake3`), which is not desirable. Therefore, an easier way to define variables that are cleared on every respawn is to simply add them to `gclient_s`. Thus, the typical mod aspects that can be added to this struct are rewards and punishments schedules as well as current fitness-related performance feedback because none of them necessarily need to be remembered when the in-game character dies. The residual time variables (`hrRewardTimeResidual`, `punishmentTimeResidual`) are counters for determining when exactly to issue the next reward or punishment, given the right prerequisites (see previous Table 5.1).Because the mod requires having different reward categories (items, weapons, power-ups), each of them being further segregated

into quality pools (low, medium, high), three arrays have been defined for each item type with their content – strings of item names[78] – classified according to Table 5.2. Additionally, to ease the mod's process of handing out rewards, a struct named `grewardPool_s` has been created, which contains a pointer to a single item array and its size[79]. Hence, a nine elements array of `grewardPool_s` structs (`g_rewards`) conveniently enables access to the entire set of rewards for distribution. To make them usable server-wide, the arrays are declared in `g_local.h` and their definitions can be found in `g_items.c`, shown in Code Listing 5.10.

```
55 // item pool, sorted by qualities
56 char *g_rewardItemsLQ[] =
57   {"5 Health","Armor Shard","Bullets","Shells"};
58 char *g_rewardItemsMQ[] =
59   {"25 Health","50 Health","Armor","Grenades","Cells","Lightning"};
60 char *g_rewardItemsHQ[] =
61   {"Mega Health","Medkit","Heavy Armor","Rockets","Slugs","Bfg Ammo"};
62 // weapon pool, sorted by qualities
63 char *g_rewardWeaponsLQ[] = {"Machinegun","Shotgun"};
64 char *g_rewardWeaponsMQ[] =
65   {"Plasma Gun","Grenade Launcher","Lightning Gun"};
66 char *g_rewardWeaponsHQ[] = {"Rocket Launcher","Railgun","BFG10K"};
67 // powerup pool, sorted by qualities
68 char *g_rewardPowerupsLQ[] = {"Regeneration","Personal Teleporter"};
69 char *g_rewardPowerupsMQ[] = {"Speed","Invisibility"};
70 char *g_rewardPowerupsHQ[] = {"Quad Damage","Battle Suit","Flight"};
71 // server-global rewards pool (sizes & pointers to all quality pools)
72 grewardPool_t g_rewards[] =
73 {
74   {4,g_rewardItemsLQ},{6,g_rewardItemsMQ},{6,g_rewardItemsHQ},
75   {2,g_rewardWeaponsLQ},{3,g_rewardWeaponsMQ},{3,g_rewardWeaponsHQ},
76   {2,g_rewardPowerupsLQ},{2,g_rewardPowerupsMQ},{3,g_rewardPowerupsHQ}
77 };
```

Code Listing 5.10: Q3A mod: `g_items.c` – items, weapons and power-up definitions, quality-based (full code: `http://tinyurl.com/jncop77`).

[78] All name strings for the arrays have been taken from a globally shared item list (`bg_itemlist[]`) in the `bg_misc.c` file (`http://tinyurl.com/zx968ln`).

[79] Since in ANSI C an array label simply is a pointer to the first element, the size must either be explicitly remembered or a stop mark set at the array's end, e.g. `NULL`.

The variable `performanceFlags` is used to store the current fitness-performance state by setting bits according to the definitions in `bg_public.h`, shown in Code Listing 5.11.

```
71 // gentity->performanceFlags (br and hr zones)
72 #define BR_ZONE_LOW        1
73 #define BR_ZONE_POOR       2
74 #define BR_ZONE_NORMAL     4
75 #define BR_ZONE_HIGH       8
76 #define HR_ZONE_LOW        16
77 #define HR_ZONE_RECOVERY   32
78 #define HR_ZONE_AEROBIC    64
79 #define HR_ZONE_ANAEROBIC  128
80 #define HR_ZONE_HIGH       256
```

Code Listing 5.11: Q3A mod: `bg_public.h` – performance flags bit definitions (full code: `http://tinyurl.com/zqoymyp`).

The `playerState_s` struct contains a Q3A original variable named `weapon`, which is useful for the later described client programming part because it indicates a player's currently held weapon. Data concerning the calculation of heart rate and pedal cycle zones is persistently stored across respawns, therefore, contained in `clientPersistant_s`. The struct includes time records of the last recognized keypresses (`lastBikeCycleTime`, `lastHeartBeattime`), current rpm as well as bpm information (`currentHeartRate`, `currentCycleRate`) and HB sampling related variables (`hrSampleCount`, `hrSampleSum`) that are used to calculate the average HR across a default of 6 samples[80]. Finally, the last two variables belong to the session struct (`clientSession_s`): `targetHR` simply stores the maximum allowed heart rate, calculated by using the player's age according to Formula 5.1 and `maxHeartRate` holds the desired target HR zone[81].

Since `gclient_s`'s variables have different lifespans, they are initialized in separate functions. `ClientSpawn` in `g_client.c`, as its name suggests, spawns the client into a map and therefore is a good spot to initialize all non-persistent data. `ClientConnect` belonging to the same file, as one might guess, is executed when a client connects, but it is named deceivingly because it is as well called every time a new level is loaded, rendering it a perfect place to deal with persistent variables. It is also suited for restoring session data (`G_ReadSessionData`) using `g_session.c`, which is necessary because `clientSession_s`'s variables are stored to `cvars` (`G_WriteSessionData`) on each level change or tournament

[80] As mentioned above, in the final version the number of samples has been made adjustable – this process is straightforward and hence is omitted, as it would only clutter the presented code.

[81] Initializing both of these variables requires specific input that can configured via server console variables (`cvars`), utilizing the '`+set [VAR] [VALUE]`' directive when starting `quake3.exe`.

restart[82]. All initialization code segments will be omitted here as they basically deal with adjusting the mod to a variety of settings provided by a custom launch script (`runmod.bat`) in a straightforward manner (see Appendix B.2.2, Code Listing B.1 for the complete list of options). Having set up all needed variables, it is now time to think a about how to detect Arduino-generated buttonpresses.

Keyboard inputs in Q3A are actually console commands that are bound to keys via a config file, like the at startup automatically executed `autoexec.cfg`[83]. Recognizing the ergometer's input keys, therefore, requires adding two new commands and also an easy way to record occurring buttonpresses for communicating them to all parts of the game that need to be informed about those events. The desired effects can be achieved by altering `q_shared.h` (Code Listing 5.12) as well as `g_cmds.c` (Code Listing 5.13).

```
1222 // usercmd_t->button bits
1223 #define BUTTON_ATTACK          1
1224 #define BUTTON_TALK            2     // talk balloon & disable actions
1225 #define BUTTON_USE_HOLDABLE    4
1242 /* ---------------------- Skip lines 1226-1242 ---------------------- */
1243 #define BUTTON_BIKE_CYCLE     4096  // button for received bike cycle
1244 #define BUTTON_HEART_BEAT     8192  // button for received heart beat
```

Code Listing 5.12: Q3A mod: `q_shared.h` – button bits definitions (full code: `http://tinyurl.com/h8qlz5m`).

As hinted in the comment left by id software, keypresses are propagated by setting single bits of `usercmd_t->buttons`, which is a common integer variable. For the mod two new bit positions were defined: 4096 (2^{12}) signaling an HB and 8192 (2^{13}) a pedal cycle.

```
1570 void Cmd_BikeCycle_f(gentity_t *ent)
1571 {
1572     ent->client->buttons |= BUTTON_BIKE_CYCLE; // record this cycle
1573 }
1587 /* ---------------------- Skip lines 1574-1587 ---------------------- */
1588 void Cmd_HeartBeat_f(gentity_t *ent)
1589 {
1590     ent->client->buttons |= BUTTON_HEART_BEAT; // record this beat
1591 }
1725 /* ---------------------- Skip lines 1592-1725 ---------------------- */
```

[82] Session data is initialized (`G_InitSessionData`) by simply setting a boolean variable to indicate a client connecting for the first time.

[83] The complete set of configuration files is part of the mod's GitHub repository: `autoexec.cfg` (`http://tinyurl.com/hrqn6vj`) and `ergomod.cfg` `http://tinyurl.com/z8qj58y`.

```
1726 void ClientCommand( int clientNum )
1727 {
1728     gentity_t    *ent;
1729     char         cmd[MAX_TOKEN_CHARS];
1730     ent = g_entities + clientNum;
1731     if ( !ent->client ) return; // not fully in game yet
1732     trap_Argv( 0, cmd, sizeof( cmd ) );
1783 /* ---------------------- Skip lines 1733-1783 ---------------------- */
1784     if (Q_stricmp(cmd, "heart") == 0)
1785     {
1786         Cmd_HeartBeat_f(ent);
1787         return;
1788     }
1789     if (Q_stricmp(cmd, "bikecycle") == 0)
1790     {
1791         Cmd_BikeCycle_f(ent);
1792         return;
1793     }
1794 }
```

Code Listing 5.13: Q3A mod: `g_cmds.c` – HB & pedal cycle commands. (full code: `http://tinyurl.com/gwb3chq`).

`ClientCommand` is called whenever the in-game console is brought up via the '~' key and a command prepended with a backslash is entered – for example '\god' turns on the god mode, making the player invulnerable, which of course is only allowed when starting a map with developer options enabled. The command string, after being retrieved via a system call trapping into the main executable (`trap_Argv`), is stored in `cmd`, which subsequently is compared against known command strings using `Q_stricmp`. Hence, if entered commands are valid, corresponding functions can be called, which have been labeled `Cmd_BikeCycle_f` for pedal cycles and `Cmd_HeartBeat_f` for heart beats. The associated functions simply turn on previously mentioned button bits in `ent->client->buttons`. All that remains to be done now is binding the commands to Arduino produced keypresses, which is accomplished via the mod's `autexec.cfg` file using the Q3A bind instruction: `bind x "bikecycle"` and `bind h "heart"`. Several additional command definitions have been omitted here for abbreviation: displaying the mod's settings, altering the maximum heart rate and adjusting the amount of HB samples taken. Subsequently, the main part of mod is outlined, dealing with appropriate reactions to the inputs.

Roughly once every animation frame the server calls the CLientThink_real function, which is contained in g_active.c and is essential for updates to some of gclient_s's variables – it especially adjusts many of the movement-related playerState_s attributes. Since this happens very quickly because the game by default can run with up to 90 *frames per second* (fps), the function is perfectly suited for reacting to ergometer and pulse sensor inputs. Code Listings 5.14, 5.15 and 5.16 show the most significant changes to g_active.c in order to accomplish desired effects for the mod.

```
1194 // called once for each client frame
1195 void ClientThink_real( gentity_t *ent )
1196 {
1197     clientPersistant_t    *clPers;
1198     int                   msec;
1199     clPers = &ent->client->pers;
1200     msec = ucmd->serverTime - ent->client->ps.commandTime;
1287 /* ---------------------- Skip lines 1201-1287 ---------------------- */
1288     // react to pedal cycle & HB input
1289     if(!(ent->r.svFlags & SVF_BOT)) // only allow human players
1290     {
1291         if (ent->client->buttons & BUTTON_BIKE_CYCLE) // check BC bit
1292         {   // handle a received bikecycle cmd
1293             clPers->currentCycleRate = 60000.0 / (ucmd->serverTime -
    ↪ clPers->lastBikeCycleTime); // rpm
1294             clPers->lastBikeCycleTime = ucmd->serverTime;
1295             ent->client->buttons ^= BUTTON_BIKE_CYCLE; // reset BC bit
1296         }
1297         if (ent->client->buttons & BUTTON_HEART_BEAT) // check HB bit
1298         {   // handle a received heart cmd
1299             clPers->hrSampleCount++; // collect 6 HB samples
1300             clPers->hrSampleSum += 60000.0 / (ucmd->serverTime -
    ↪ clPers->lastHeartBeatTime); // bpm
1301             if (clPers->hrSampleCount >= 6)
1302             {   // calc avg. HR & reset counters
1303                 clPers->currentHeartRate =
1304                   clPers->hrSampleSum / (float) clPers->hrSampleCount;
1305                 clPers->hrSampleCount = 0;
1306                 clPers->hrSampleSum = 0.0;
1307             }
1308             clPers->lastHeartBeatTime = ucmd->serverTime;
```

```
1309                ent->client->buttons ^= BUTTON_HEART_BEAT; // reset HB bit
1310            }
1311            // handle cycling / HR zones & rewards
1312            if (!ent->client->ps.powerups[PW_HASTE])
1313              handleErgometerSpeeds(ent);
1314            handleHeartRate(ent, msec);
1315        }
1316    }
```

CODE LISTING 5.14: Q3A mod: `g_active.c` input reactions 1/3, handle inputs (full code: `http://tinyurl.com/gqtvbco`).

First, `ClientThink_real` deals with reacting to detected inputs by calculating and setting the persistent cycle and heart rate variables (`currentCycleRate`, `currentHeartRate`). This should not be done for bots and therefore it is necessary to check if a specific flag belonging to the client's entity is set in order to differentiate between human and AI controlled players (`SVF_BOT` in `ent->r.svFlags`). Then `ent->client-buttons` can be checked for activated pedal cycle and heart rate bits. In case any of the two have been received, the corresponding current rate is set and the bit is reset again to prepare for the next input. Now that the rates are set everything else related to issuing rewards can be attempted, which has been put into separate functions: `handleErgometerSpeeds` and `handleHeartRate`. Dispensing rewards happens time-based and therefore it can make use of the Q3A `msec` variable, which stores the amount of milliseconds passed since the last call of `ClientThink_real`, a very useful value to keep track of time. Lastly, the in-game running speed is not updated for players with an active 'haste' power-up – the sole purpose of this item is to increase the characters velocity by a factor of 1.5 for 30 seconds and applying the mod's regulations would render it useless.

```
958  // set performanceFlags bits & reset residual time on HR zone change
959  void setPerformanceZone(gentity_t *ent, int zone); // impl. omitted
1018 /* --------------------- Skip lines 960-1018 --------------------- */
1019 // punish the player by periodically removing HP & AP
1020 void handlePunishments(gentity_t *ent, int msec); // impl. omitted
1074 /* --------------------- Skip lines 1021-1074 --------------------- */
1075 void handleErgometerSpeeds(gentity_t *ent)
1076 {
1077     clientPersistant_t    *clPers;
1078     clPers = &ent->client->pers;
1079     if (clPers->currentCycleRate < 50)
1080     {   // cycle speed too low (reduce speed)
```

```
1081            ent->client->ps.speed *= 0.3;
1082            setPerformanceZone(ent, BR_ZONE_LOW);
1083        }
1084        else if ((clPers->currentCycleRate >= 50) &&
    ↪   (clPers->currentCycleRate < 70))
1085        {   // cycle speed acceptable (reduce speed)
1086            ent->client->ps.speed *= 0.6;
1087            setPerformanceZone(ent, BR_ZONE_POOR);
1088        }
1089        else if ((clPers->currentCycleRate >= 70) &&
    ↪   (clPers->currentCycleRate <= 100))
1090        {   // cycle speed just right (normal speed)
1091            ent->client->ps.speed *= 1.0;
1092            setPerformanceZone(ent, BR_ZONE_NORMAL);
1093        }
1094        else
1095        {   // cycle speed very high (increase speed)
1096            ent->client->ps.speed *= 1.3;
1097            setPerformanceZone(ent, BR_ZONE_HIGH);
1098        }
1099    }
1140    /* ---------------------- Skip lines 1100-1140 ---------------------- */
1141    void handleHeartRate(gentity_t *ent, int msec)
1142    {
1143        clientPersistant_t    *clPers;
1144        clientSession_t       *clSess;
1145        clPers = &ent->client->pers;
1146        clSess = &ent->client->sess;
1147        if (clPers->currentHeartRate < 0.45 * clSess->maxHeartRate)
1148        {   // HR too low (no rewards)
1149            setPerformanceZone(ent, HR_ZONE_LOW);
1150        }
1151        else if (clPers->currentHeartRate < 0.6 * clSess->maxHeartRate)
1152        {   // recovery training (reward rate: 0.2)
1153            setPerformanceZone(ent, HR_ZONE_RECOVERY);
1154            handleRewards(ent, 0.2, msec);
1155        }
1156        else if (clPers->currentHeartRate < 0.75 * clSess->maxHeartRate)
1157        {   // aerobic training (reward rate: 0.3)
```

```
1158            setPerformanceZone(ent, HR_ZONE_AEROBIC);
1159            handleRewards(ent, 0.3, msec);
1160        }
1161        else if (clPers->currentHeartRate < 0.9 * clSess->maxHeartRate)
1162        {   // anaerobic training (reward rate: 0.15)
1163            setPerformanceZone(ent, HR_ZONE_ANAEROBIC);
1164            handleRewards(ent, 0.15, msec);
1165        }
1166        else
1167        {   // HR too high (punish player)
1168            setPerformanceZone(ent, HR_ZONE_HIGH);
1169            handlePunishments(ent, msec);
1170        }
1171    }
```

Code Listing 5.15: Q3A mod: `g_active.c` input reactions 2/3, handle performance zones (full code: `http://tinyurl.com/gqtvbco`).

Both functions, `handleErgometerSpeeds` and `handleHeartRate`, are used to determine the current workout zones, which is done by comparing the previously set rates to the values[84] from Tables 5.1 and 5.3 – in-game player speeds are set via adjusting the value of `ent->client->ps.speed`. The omitted `setPerformanceZone` function is used to set all necessary flags in `ent->client->performanceFlags` and to reset `ent->client->hrRewardTimeResidual` when the player switches HR zones, which is important for the upcoming `handleRewards` function. In case the HR gets too high, the player is punished by losing health and armor points – the implementation of `handlePunishments` is very similar to `handleRewards`, hence, is also omitted here.

```
811 // picks an index based on percentages in array
812 int pickFromArray(float *array, int size)
813 {
814     int      rndSeed, pickedIndex, counter;
815     float    rnd;
816     rndSeed = rand() % INT_MAX;
817     rnd = Q_random(&rndSeed); // rnd number betw. 0 and 1
818     pickedIndex = -1.0;
819     for (counter = 0; counter < size; counter = counter + 1)
820     {
```

[84] For convenience, all values are hard-coded here – a measure taken to avoid listing all variable and constant definitions.

```
821         if (rnd < array[counter])
822         {   // index found - stop
823             pickedIndex = counter;
824             break;
825         }
826         rnd -= array[counter];
827     }
828     return pickedIndex;
829 }
840 /* ---------------------- Skip lines 830-840 ---------------------- */
841 // spawn item directly above entity
842 void giveItemTo(gentity_t *ent, gitem_t *item); // impl. omitted
860 /* ---------------------- Skip lines 843-860 ---------------------- */
861 // reward items according to rate
862 void handleRewards(gentity_t *ent, float rate, int msec)
863 {
864     gitem_t           *item;
865     grewardPool_t     *qualityPool;
866     int interval, pickedItem, pickedQuality, rewardIndex, rndIndex;
867     float itemDropChances[] = {0.5, 0.3, 0.2};
868     float qualityDropChances[] = {0.6, 0.3, 0.1};
869     ent->client->hrRewardTimeResidual += msec;
870     interval = (int)((1.0 / rate) * 1000.0); // reward frequency in ms
871     while (ent->client->hrRewardTimeResidual >= interval)
872     {   // interval has passed - reward item based on dropchances
873         ent->client->hrRewardTimeResidual -= interval; // reset timer
874         pickedItem = pickFromArray(itemDropChances, 3);
875         pickedQuality = pickFromArray(qualityDropChances, 3);
876         rewardIndex = (pickedItem * 3) + pickedQuality; // pool index
877         qualityPool = &g_rewards[rewardIndex];
878         rndIndex = rand() % qualityPool->size; // rnd item index
879         item = BG_FindItem(qualityPool->items[rndIndex]);
880         giveItemTo(ent, item); // give item to player or bot
881     }
882 }
```

Code Listing 5.16: Q3A mod: `g_active.c` input reactions 3/3, handle rewards (full code: `http://tinyurl.com/gqtvbco`).

Given all requirements are met, the next step is to decide which reward to distribute and when this should happen. These tasks are solved by the handleRewards and pickFromArray functions. First it is necessary to determine the reward frequency, i.e. the interval in which items are handed out. Since the reward rate has already been set in handleHeartRate, the interval can be calculated by using Formula 5.2. The hrRewardTimeResidual variable is used to determine if the interval has passed and a reward is due, which is achieved by regularly summing up msec, the amount of time since the last frame. This does not pose a problem for the case a player changes HR zones, as setPerformanceZone resets hrRewardTimeResidual when such an event occurs, effectively restarting the timer. Next, as soon as the interval has passed, the set of rewards has to be narrowed down to a specific category and quality, which is done by calling pickFromArray twice. The function selects an array index based on the percentages of its individual elements by calculating a random number between [0,1] and comparing it to the portions of 100% that the probabilities cover. Using this function with itemDropChances and qualityDropChances yields two indices that can be used to determine the corresponding reward pool precisely: $poolIndex = 3 \times itemIndex + qualityIndex$. At last, an item is randomly chosen from the quality pool and given to the player via giveItemTo, which spawns the item directly above the in-game character, a standard procedure in Q3A and therefore omitted here.

The final piece of code deals with introducing a smarter automatic weapon switching system. Issuing commands for changing weapons is a task associated with the client VM and it is usually done via keyboard input. However, as already mentioned, Q3A provides the option of automatically switching weapons on pick up, which does not differentiate between their qualities. This has been improved by defining a weapon hierarchy based on the ratings in Table 5.2. Related changes mostly have been made to cg_events.c, which is displayed in Code Listing 5.17.

```
33  weapon_t cg_weaponHierarchy[] = { WP_NONE, WP_GRAPPLING_HOOK,
    ↪  WP_GAUNTLET, WP_MACHINEGUN, WP_SHOTGUN, WP_PLASMAGUN,
    ↪  WP_GRENADE_LAUNCHER, WP_LIGHTNING, WP_ROCKET_LAUNCHER, WP_RAILGUN,
    ↪  WP_BFG };
409 /* -------------------- Skip lines 34-409 -------------------- */
410 // description
411 qboolean isWeaponBetter(weapon_t weapon1, weapon_t weapon2)
412 {
413     int currentWeaponIndex, newWeaponIndex, i;
414     for (i = 0; i < NUM_HIERARCHY_WEAPONS; i++)
415     {
416         if (weapon1 == cg_weaponHierarchy[i]) currentWeaponIndex = i;
```

```
417         if (weapon2 == cg_weaponHierarchy[i]) newWeaponIndex = i;
418     }
419     if (currentWeaponIndex < newWeaponIndex) return qtrue;
420     else return qfalse;
421 }
409 /* ---------------------- Skip lines 422-409 ---------------------- */
410 // a new item was picked up in this frame
411 static void CG_ItemPickup(entityState_t *es, int itemNum)
412 {
413     cg.itemPickup = itemNum;
414     cg.itemPickupTime = cg.time;
415     cg.itemPickupBlendTime = cg.time;
416     if ( bg_itemlist[itemNum].giType == IT_WEAPON &&
    ↪ cg_autoswitch.integer )
417     {   // item is a weapon & autoswitch is on (disable in mod cfg)
418         if (!isWeaponBetter(es->weapon, bg_itemlist[itemNum].giTag))
419           return;
420         cg.weaponSelectTime = cg.time;
421         cg.weaponSelect = bg_itemlist[itemNum].giTag;
422     }
423 }
```

Code Listing 5.17: Q3A mod: `cg_event.c` – smart weapon autoswitching (full code: `http://tinyurl.com/hlwjdjs`).

The hierarchy array is of the *enumeration* (`enum`) type `weapon_t`, which provides identifiers for the game's entire weapon arsenal. `CG_ItemPickup` is a function that is called whenever an item is picked up by the player. It checks whether the acquired object is a weapon and if the `cg_autoswitch` option is turned on. If these requirements are met, instead of just blindly equipping the weapon, `isWeaponBetter` to consult the defined hierarchy: a simple loop through the array allows for determining if the currently held weapon is weaker than the retrieved one, in which case a switch is ordered for the next update that is sent to the server.

As repeatedly outlined above, lots of features have been omitted in order to abbreviate this section. An important disregarded mod element is giving players feedback about their performance, which is essential for improving their understanding of the mod's goals. Informing the users has been kept simple – based on Tables 5.1 and 5.3, in-game debug printouts were given via the server console `print` command, the format of which is listed below, concluding this section:

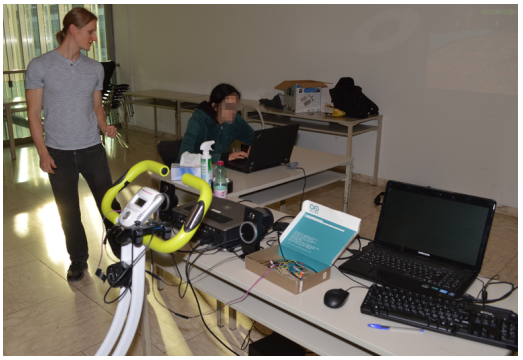
```
BIKING: [BIKE FEEDBACK]
PULSE:  [PULSE FEEDBACK] ZONE: [ACTUAL ZONE] TARGET: [TARGET ZONE]

[BIKE FEEDBACK] ........ bad / acceptable / good / high
[PULSE FEEDBACK] ....... bad / acceptable / good / high / too high
[ACTUAL/TARGET ZONE] ... recovery / aerobic / anaerobic
```

5.3.3 Study Setup

As already mentioned, to retain comparability between both the FC and the Q3A mod study, they have been set up very similarly. Hence, for this final supervised field study, again the ergometer as well as the Arduino, an HD projector and two laptops – one running the mod and the other providing the survey – have been placed in the AAU's aula. Figure 5.22 shows the setup from different angles.

(a) Back view.

(b) User view.



(c) Front view.

FIGURE 5.22: Q3A mod: study setup.

Concerning voluntary participants, the same procedure that has previously been applied in the first study was employed: a total of 34 passers-by either were invited by the supervisor or took part on their own accords. All survey questions asked can be found in B, Subsections B.1.1 and B.2.1, given answers in Subsection B.3.2, where they are listed in comparison to responses from the FC study. Altogether the approximate duration of this study has been ten hours, including preparing and disassembling the hardware.

5.4 Summary

In order to evaluate proposed hypotheses from Chapter 4, two supervised user studies were conducted in the AAU's aula during normal working day hours. Both of them utilized an ergometer device as well as other fitness-related technologies connected to PC games via an Arduino microcontroller. One game, a Flappy Bird clone titled "Flappy Cycling", has been created from scratch using HTML5-based game framework Phaser and resembled a common casual exergame, while the other, a modified version of the classic first-person shooter Quake 3 Arena, developed in ANSI C, represented a fairly new experience to voluntary participants: a fitnessified AAA game. Respectively, 33 and 34 test subjects were asked to try out the games for as long as they liked, provided that they afterwards portray their experiences in two identical online surveys. Prior to both experiments, preliminary user evaluations aimed at identifying major flaws in the systems and yielded invaluable feedback for improving the final study setups.

Chapter 6

Evaluations

After having conducted both studies and as the final component of this thesis, the collected data needs to be verified and evaluated. As mentioned, all survey questions as well as statistics about given responses can be found in Appendix B, Sections B.1.1, B.2.1 and B.3.2. This chapter is subdivided into three parts. Section 6.1 qualitatively analyses the results in order to establish a big picture and discusses given free text answers. Finally, Section 6.2 outlines all assumptions and analyzes the data quantitatively.

6.1 Qualitative Analysis

Overall, 33 people participated in the FC and 34 in the Q3A mod study – a very balanced basis for analyses. Furthermore, since every test person partook voluntarily and the experiments were supervised, from a subjective point of view, it seems that the survey questions have been answered to the best of everyone's knowledge and beliefs. The questionnaires' inquiries can roughly be grouped into five categories[1], which are listed and described in Table 6.1.

Table 6.1: Categories for survey questions.

short	**category**	**description**
`C_GEN`	General	Demographic data like gender, age etc.
`C_SPT`	Sports	Information about exercise habits.
`C_GAM`	Games	Information about gaming habits.

[1] See Section 6.2 for a more detailed listing of variables.

C_COM	Combination of Sports & Games	Experiences and opinions about combining games and sports.
C_PRO	Prototypes	Questions specifically regarding the study prototypes.

Following paragraphs discuss noteworthy statistics and responses given for each of the question categories.

Collected data for C_GEN type questions for both studies reveals a predominant male participation ($> 70\%$). Therefore, it is not feasible to conduct any gender-specific analyses on the accumulated survey records. The average specified ages range from 25 to 30 and in particular, almost all participants were below 35 years old with altogether five exceptions. Close to half of the volunteers, in equal proportions for both studies, were students and the majority of the remaining people were employed in a technical field. This is expectable, since study setups, such as the ones conducted for this thesis, especially appeal to people interested in technology. Despite the AAU's multicultural student base[2], most of the test subjects have been from Austria (FC: 76%, Q3A: 70%) and in general, almost all participants were from Europe with two exceptions.

C_SPT results indicate that 94% of both studies' respondents practice sports, many of them on a regular basis (FC: $> 50\%$, Q3A: $> 33\%$). Participants like a variety of sportive activities, the most popular of which were: water sports, bicycling, jogging and ball games (in descending order). Biking has been mentioned about twice as much in the FC study (61%) than in the Q3A one (29%). However, when asked about their experience in handling the ergometer, Q3A study attendants rated above 'average' twice as much (77%) as FC study volunteers. This partially could be due to FC being physically more exhausting, but also is the result of game control issues (see C_PRO paragraph below). In both studies responses were given suggesting that the games were challenging: "*Need a shower now.*" (FC), "*Training.*" (FC), "*After some time it can get quite demanding ...*" (Q3A), "*The ergometer level was set to high.*"[3] (Q3A).

C_GAM related responses show that the majority of both experiments' attendees were gamers (FC: 67%, Q3A: 79%) playing a variety of game genres: Stategy & RPG, FPS & Adventures and Action (in descending order). It is unusual that among all participants only five play casual games (10% of gamers), while expectedly a fair amount play FPSs – 22 in total (45% of gamers). Many gamers that participated play regularly, albeit this

[2] cf. AAU report 2015/16 (http://tinyurl.com/jqq2c7h).

[3] The participants were able to adjust this level themselves – here explicit instructions should have been given.

percentage is higher for Q3A mod testers (FC: 41%, Q3A: 59%). Even though quite some people trying out the prototypes do not usually play games, it is interesting to see that even persons specifically expressing their hate towards video games partook in the studies, mostly even enjoying themselves.

Answers to `C_COM` questions are significant: almost all like the idea of exergames ($> 90\%$), yet only two persons play them, which could have several reasons: current fitness games are rather boring, people do not have access to them or simply laziness. Another reason can be discovered by reading given comments, as they suggest that fitness games often are considered social experiences and therefore less likely to be played alone: "*(Exergames allow) direct comparison with others ...*" (FC), "*Sports games to me are social games that I cannot and do not want to play alone.*" (Q3A). Generally, participants disclosed many opinions on their conception of exergaming, most of which were similar to the following: it is meaningful/healthy, interesting/entertaining, motivating/challenging and something new. Some skeptical responses claim that games rather should be played for relaxation, a disposition that seems to be shared by other people as well[4]. Indeed, escaping reality might be considered relaxing but, other than passively enjoying a beautiful scenery, actively playing a game more aptly can be characterized as *engaging*, often requiring players to stay focused and muster a lot of effort. Therefore, more solid arguments against combining sports with gaming would either be that it can be distracting or too exhausting, the latter of which having been remarked a few times. Also, several comments suggest that sports should be practiced outside, which of course will always be a matter of attitude and opinion. Lastly, after trying out the prototypes, about 73% of both studies' attendees would consider playing casual exergames like FC or fitness-enhanced versions their favorite games, i.e. in many cases AAA games, at home if they were available. Over half of those people would even be inclined to work out more, which slightly differed for each experiment (FC: 52%, Q3A: 65%).

The `C_PRO` category questions constituted a large part of the main evaluations, hence, they have been more specific than the others. Judging by the overall study ratings, the Q3A mod experiment seems to have been received better than the FC study (FC: $64\% \geq$ 'good', Q3A: $86\% \geq$ 'good'). Still, in order to give an absolute verdict, several factors must be taken into account. First, as already mentioned in Section 5.2.3, FC controls were not optimal, as the pulse handles did not always work properly. This heavily influenced some of the ratings and was mentioned many times via questionnaire comments. However, various participants generally seemed to have difficulties with the employed unconventional game controls, as can be observed when regarding comments for any of the studies: "*It doesnt work at the beginning.*", (FC) "*Pulse sensors werent always detected*

[4] cf. [136] or Andy Kelly for PC Gamer, 2015: "PC's most relaxing games" (`http://tinyurl.com/qgmc5ab`).

at the beginning." (FC), "*Would need more time to get used to the controls ...*" (Q3A), "*Controls are hard.*" (Q3A). In contrast to these reports, noticeable improvements in handling the controls could be observed, as some volunteers tried out the games multiple times. Moreover, altogether pedaling on the ergometer has been received quite well, as is also reflected in many user comments. Specific gaming experience ratings are somewhat in line with the overall study scores (FC: 42% $\geq$ 'good', Q3A: 62% $\geq$ 'good') and as well the game difficulty ratings seem to reflect the previous observations: FC has been considered slightly harder than Q3A (FC: 45% $\geq$ 'difficult', Q3A: 35% $\geq$ 'difficult'). A noteworthy difference can be discovered in the ratings of the games themselves: FC got a better score than Q3A (FC: 70% $\geq$ 'good', Q3A: 59% $\geq$ 'good'). This could be due to FC's calmer setting and the fact that several Q3A study participants generally do not like shooters, which they expressed in various comments. Surprisingly, a few of these participants also disclosed that they enjoyed themselves despite their negative disposition towards FPS: "*I usually dont like shooters, but combined with exercise it is fun.*" (Q3A).

In conclusion, both studies influenced user experience positively as well as negatively, although Q3A seems to have been better accepted. The reason both experiments overall received fairly high ratings could be due to their novelty effect: in both cases only 15% claimed to have partaken in similar studies and comparable home devices are still rare, as already has been mentioned in Section 3.5.2. Nevertheless, the biggest issues reported were control-specific, i.e. difficulties with FC's pulse handles and Q3A's use of a gamepad instead of keyboard/mouse, which in future versions should be improved as to prevent any biased opinions caused by these inconveniences. Furthermore, the most important future aspect should be testing long-term user engagement in playing these games. Outcomes for such experiments probably depend on how regularly users play games, but judging by several user remarks FC might not perform as well as Q3A: "*Game seems a bit random.*" (FC), "*It is a fun game, but i think it will get boring after a while.*" (FC).

6.2 Quantitative Analysis

The second and final part in this thesis' hybrid evaluation approach is to conduct a quantitative analysis on the collected data. For this purpose, measurable variables need to be developed out of relevant survey questions, which is performed in the following Section 6.2.1. Afterwards, Section 6.2.2 lists all assumptions that have to be made about the data to ensure study validity. The subsequent Section 6.2.3 verifies the null hypotheses proposed in Section 4.2 via significance tests. Finally, Section 6.2.4 investigates variable correlations for further evaluations.

6.2.1 Relevant Variables

Since several questions did not achieve a minimum response rate of 30%, they have been excluded from variable development. The remaining questionnaire allows for defining a total of 23 variables, which are listed in Table 6.2.

TABLE 6.2: Variables derived from questions (D: depends on other variable).

variable (D)	values	questions (# values)
C_GEN		
gender	male, female	A1: Please select your gender (2)
age	18-24, 25-34, 35-44, 45-54, 55+	A1: Please enter your age (112)
isStudent	false, true	A4: Please select your occupational field (24)
isTechnician	false, true	A4: Please select your occupational field (24)
C_SPT		
isActive	$<$ rarely, $\geq$ rarely	B8: How often do you practice Sports? (5)
isRegActive (*isActive*)	$<$ regularly, $\geq$ regularly	B8: How often do you practice Sports? (5)
frequencyActive (*isActive*)	never, rarely, sometimes, regularly, frequently	B8: How often do you practice Sports? (5)
isCyclist (*isActive*)	false, true	B9: Which sportive activities do you like? (13)
C_GAM		
isGamer	false, true	B1: Do you play video games? (2)
isRegGamer (*isGamer*)	$<$ regularly, $\geq$ regularly	B2: How often do you play video games? (5)
frequencyGames (*isGamer*)	rarely, sometimes, regularly, frequently, very frequently	B2: How often do you play video games? (5)
isFPSGamer (*isGamer*)	false, true	B3: Which types of games do you play? (10)

isCasualGamer (*isGamer*)	false, true	B3: Which types of games do you play? (10)
C_COM		
likesEGames	false, true	B9: Do you like the idea of combining gaming and exercising? (2)
isEGamer (*isGamer*)	false, true	B4: Do you play games for fitness? (2)
considersEGames	false, true	C9: If possible, would you consider playing (FC) such games / (Q3A) your favorite games with similar devices at home? (2)
$frequencyEGames$ (*considersEGames*)	rarely, sometimes, regularly, frequently, very frequently	C10: How often? (5)
increaseActivity (*considersEGames*)	false, true	C11: Would you thus be inclined to work out more often? (2)
C_PRO		
$ergoXP$	bad, poor, average, good, excellent	C1: How was your experience regarding handling the ergometer? (5)
$gameXP$	bad, poor, average, good, excellent	C3: How was your experience regarding playing the game? (5)
gameRating	bad, poor, average, good, excellent	C5: How would you rate the game? (5)
$gameDifficulty$	too easy, easy, balanced, difficult, too difficult	C7: How did you perceive the game's difficulty? (5)
$overallXP$	bad, poor, average, good, excellent	C13: How would you rate your overall experience during the experiment? (5)

A majority of the chosen variables (15) are bi-valued (dichotomous) or describe categories – properties which can be utilized to partition the data. The remaining variables (8) describe ordered values that are comparable to each other but there is no exactly defined

interval between them, which stands in contrast to precisely measurable variables like, for example, a person's height. Observed ordinal responses will be used to evaluate tendencies in selected groups.

6.2.2 Premises and Assumptions

As already mentioned, both supervised field studies were conducted on two arbitrary working days in the AAU's aula using voluntary participants. The initial intention for verifying proposed hypotheses on the collected data has been to conduct parametric t-tests [137] with a significance level of $\alpha = 0.05$. Because this kind of evaluation assumes the results to be normally distributed ($X \sim N(\mu, \sigma^2)$), the data needs to be analyzed accordingly. Unfortunately, visual inspections via quantile-quantile (q-q) plots [138] as well as numerical evaluations with Shapiro-Wilk tests [139] do not indicate normal data distributions in any of the variables. This could be due to the fact that their values are either dichotomous, categorical or ordinal rather than describing intervals, which arguably is better suited for parametric tests [140]. Thus, a non-parametric evaluation approach is chosen and assumptions made are updated to the following:

- The experiments' considered population (N_P) is restricted to all interested or voluntary passersby over 18 years of age.
- $n \geq 30$ samples are taken in two independent situations, i.e. individual studies.
- There should not be dependencies between individual samples other than the experience using corresponding prototypes.
- The variables for significance testing are ordinal, i.e. their values are comparable on a scale (e.g. 1-5).
- Given Hypotheses H1-H6 are evaluated using Mann-Whitney-Wilcoxon tests[5] with a significance level of $\alpha = 0.05$.

Given the methodology outlined throughout Chapter 5, possible introduced biases are:

- Conducting the experiments in a public place, allowing external influences during the tests.
- Supervisor asking people to volunteer.
- Supervisor speaking to participants, while they played the games.

[5]The MWW test has originally been developed by Frank Wilcoxon in 1945 [141] (MWW) tests, before being refined for variable sample sizes by Henry Mann and Donald Whitney in 1947 [142].

- Difficulties in handling the inputs producing negative impacts on user experience.

Given these premises together with the study observations, the pre-evaluation parameters can be summarized: verifying hypotheses H1-H6 on $N_{total} = N_{FC} + N_{Q3A} = 33 + 34 = 67$ independent samples of population N_P using MWW tests with $\alpha = 0.05$.

6.2.3 Hypotheses Evaluations

The non-parametric MWW test is used to determine if two populations (A, B) of certain sizes (n_A, n_B) differ in their distributions, i.e. the assumed null hypothesis H_0: $A = B$ can be rejected in favor of an alternative hypothesis H_1: $A \neq B$, $A < B$ or $A > B$. It accomplishes this by ranking the elements of both sample sets (lower value are assigned smaller ranks), summing up the ranks for A in T_A (tied ranks are averaged) and calculating the U value via following formula:

$$U = n_A \cdot n_B + \frac{n_A \cdot (n_A + 1)}{2} - T_A \tag{6.1}$$

This value informally is a sum of n_A elements, counting how many ranks of B are below each rank of A or in other words: how often does any rank of A exceed the ranks of B. If H_0 holds, no difference can be detected by means of the sampled data and the expected U value is:

$$\mu_U = \frac{n_A \cdot n_B}{2} \tag{6.2}$$

In this case, all U values are symmetrically distributed around μ_U with a standard error of:

$$\sigma_U = \sqrt{\frac{n_A \cdot n_B \cdot (n_A + n_B + 1))}{12}} \tag{6.3}$$

If there several values sharing the same rank, σ_U needs to be corrected and above formula is replaced by the subsequent one ($n = n_A + n_B$, t_i = number of values that share rank i, k = number of rank ties):

$$\sigma_{U_{corr}} = \sqrt{\frac{n_A \cdot n_B}{n \cdot (n-1)}} \times \sqrt{\frac{n^3 - n}{12} - \sum_{i=1}^{k} \frac{t_i^3 - t_i}{12}} \tag{6.4}$$

For sample sizes of $n_A > 10$ or $n_B > 10$ the U values are approximately normally distributed, which allows for calculating a test statistic by the means of which H_0 can be evaluated:

$$z = \frac{U - \mu_U}{\sigma_U} \tag{6.5}$$

The test statistic z indicates how much the observed U value deviates from the expected value μ_U as a factor of the standard error σ_U. It can be used to calculate a p-value via the cumulative distribution function for the standard normal distribution[6]:

$$\Phi(x) = \frac{1}{2}\left[1 + \frac{1}{\sqrt{\pi}} \int_{-\frac{x}{\sqrt{2}}}^{\frac{x}{\sqrt{2}}} e^{-t^2} dt\right] \tag{6.6}$$

Finally, the resulting p-value determines the statistical significance of the evaluation as it conveys how likely the collected data occurs, given that H_0 is true, i.e. the distributions of A and B are equal ($p = 0.5$). Accordingly, if this probability is below the chosen significance level α, H_0 can be rejected and H_1 concluded. For lower sample sizes ($n_A \leq 8$, $n_B \leq 8$) several tables containing exact p-values have been created by the test originators [142], since using the normal distribution approximation in this case would be too imprecise[7]. Additionally, tables listing critical upper U values are available for average sample sizes ($n_A \leq 20$, $n_B \leq 20$) and typical alpha values (see Appendix B, Section B.4 containing both kinds of tables). All of the hypotheses have been tested with the statistics software environment R[8] using the `wilcox_test` function of the coin package[9] as well as individual calculations utilizing above formulas and employing the methodology suggested by Bortz et. al [143]. Table 6.3 lists an overview of the retrieved results.

TABLE 6.3: Evaluation of Hypotheses: MWW results overview (u = observed U value).

hypothesis (H_1)	n_A	n_B	p	z	u	σ_u	**reject** H_0
H1: $A_{\subseteq FC} < B_{\subseteq Q3A}$	16	21	0.038	-1.774	118	168	for $\alpha = 0.05$
H2: $A_{\subseteq FC} < B_{\subseteq Q3A}$	23	23	0.087	-1.360	211.5	264.5	for $\alpha = 0.10$
H3: $A_{\subseteq FC} < B_{\subseteq Q3A}$	4	10	0.243	-0.698	17	20	no

[6] Note that $\Phi(x)$ always returns standard normal curve region from negative infinity to x, which can be used to test $A < B$. For $A > B$ the inverse probability $1 - \Phi(x)$ must be used and for $A = B$ the result needs to be doubled ($2 \cdot \Phi(x)$) since the H_0 rejection region is then two-tailed.

[7] It is recommended to only use $\Phi(x)$ if either sample size, na_A or n_B, is larger than 10 [143].

[8] `https://www.r-project.org`

[9] The coin package (`http://tinyurl.com/hntzarh`) `wilcox_test` function is superior to the R default `wilcox.test` function because it supports calculating exact p-values when rank ties are present.

$H4_a$: $A_{1\subseteq FC} < B_{1\subseteq FC}$	5	19	0.349	-0.388	43	47.5	no
$H4_b$: $A_{2\subseteq Q3A} < B_{2\subseteq Q3A}$	9	16	0.437	-0.159	69.5	72	no
$H5_a$: $A_{1\subseteq FC} > B_{1\subseteq FC}$	13	11	0.057	1.581	50	71.5	no ($U_{\alpha=0.10} < u$)
$H5_b$: $A_{2\subseteq Q3A} > B_{2\subseteq Q3A}$	17	8	0.089	1.344	55	68	no ($U_{\alpha=0.10} < u$)
H6: $A_{\subseteq Q3A} > B_{\subseteq FC}$	3	8	0.882	-1.186	7.5	12	no

The remaining part of this section accounts for how the data sets (FC, $Q3A$) have been partitioned for each hypothesis and discusses observed results.

H1: Gamers would rather play AAA exergames than casual exergames at home.

Test variable: $frequencyEGames$, partitions ($FC \cap Q3A = \emptyset$):

$$A = \{x \in FC | x.isGamer \wedge x.considersEGames\},$$
$$B = \{y \in Q3A | y.isGamer \wedge y.considersEGames\},$$
$$H1_0 : A = B,\ H1_1 : A < B\ (P(U_{A\subseteq FC} \leq u_{A\subseteq FC}))$$

H1 addresses gamers that would consider exergames similar to the ones they were trying out and inspects their estimated playing frequency if they were to own such systems. The results for the corresponding MWW test, yielding a p-value of 0.038, are significant for $\alpha = 0.05$ and the conclusion that gamers would indeed rather play AAA fitness games seems reasonable, since enough data records satisfy the above selection criteria ($n_A = 16, n_B = 21$).

H2: Active people would rather play AAA exergames than casual exergames at home.

Test variable: $frequencyEGames$, partitions ($FC \cap Q3A = \emptyset$):

$$A = \{x \in FC | x.isActive \wedge x.considersEGames\},$$
$$B = \{y \in Q3A | y.isActive \wedge y.considersEGames\},$$
$$H2_0 : A = B,\ H2_1 : A < B\ (P(U_{A\subseteq FC} \leq u_{A\subseteq FC}))$$

This hypothesis as well requires investigating people's speculative exergaming frequency with the distinction of only regarding persons that at least rarely practice sports. Compared to H1, H2 test results are slightly less significant ($p = 0.087$) and $H2_0$ can only be rejected if $\alpha = 0.10$, which could nevertheless be interpreted as a strong tendency towards the hypothesis' assumptions, as it conveys that chances for active people equally

preferring to play AAA and casual fitness games are less than 10%, which contradicts $H2_0$ assuming a 50% likelihood of such events to occur. However, *isActive* seems to describe a very skewed distribution, as it excludes only two persons in each data set. Therefore, the results for this hypothesis can be purely coincidental and further investigations should be conducted.

H3: Regular gamers that are less active would rather consider working out more playing AAA exergames than casual exergames.

Test variable: $increaseActivity$, partitions ($FC \cap Q3A = \emptyset$):

$$A = \{x \in FC | x.isRegGamer \wedge \neg x.isRegActive \wedge x.considersEGames\},$$
$$B = \{y \in Q3A | y.isRegGamer \wedge \neg y.isRegActive \wedge y.considersEGames\},$$
$$H3_0 : A = B,\ H3_1 : A < B\ (P(U_{A \subseteq FC} \leq u_{A \subseteq FC}))$$

The data for H3 is grouped into regular gamers that work out sometimes or even less, but consider exergaming at home. The test criterion *increaseActivity* is those people's opinion on whether they, using fitness games, would be inclined to exercise more or not. MWW results are even less significant than those of the preceding hypotheses ($p = 0.243$) and although A generally seems to be lower valued than B, there is no profound evidence against the assumption that the distributions are equal. Additionally, the low sample sizes ($n_A = 4$, $n_B = 10$) allow for a comparison with critical U values: Table B.43 for the chosen $\alpha = 0.05$ yields 7, while Table B.42 for $\alpha = 0.10$ lists 10 – both are smaller than the observed $u = 17$ suggesting to keep the assumption $H3_0$. Therefore, while also keeping in mind the rather small sample sizes ($n_A = 4$, $n_B = 10$), it can not be concluded that physically less active gamers would prefer any of the two fitness games.

H4: Active people would rather play exergames at home than rarely active people.

Test variable: $frequencyEGames$, partitions ($A_1 \cap B_1 = \emptyset, A_2 \cap B_2 = \emptyset$):

$$A_1 = \{x \in FC | x.frequencyActive \leq 2 \wedge x.considersEGames\},$$
$$B_1 = \{y \in FC | y.frequencyActive > 2 \wedge y.considersEGames\},$$
$$H4_{a_0} : A_1 = B_1,\ H4_{a_1} : A_1 < B_1\ (P(U_{A_1 \subseteq FC} \leq u_{A_1 \subseteq FC})),$$
$$A_2 = \{x \in Q3A | x.frequencyActive \leq 2 \wedge x.considersEGames\},$$
$$B_2 = \{y \in Q3A | y.frequencyActive > 2 \wedge y.considersEGames\},$$
$$H4_{b_0} : A_2 = B_2,\ H4_{b_1} : A_2 < B_2\ (P(U_{A_2 \subseteq FC} \leq u_{A_2 \subseteq FC}))$$

Hypothesis H4 also examines the estimated frequency, in which exergame-interested people would consider working out, but in a more general sense: it tries to prove that physically active people would more often play fitness games than rarely or non-active

persons. Since this test does not compare FC with Q3A results directly, it can be conducted on both data sets and therefore two MWW evaluations ($H4_a$, $H4_b$) have been applied. The resulting p-values ($p_a = 0.349$, $p_b = 0.437$) are the least significant in all of the tested hypotheses. Critical U table values[10] for $H4_a$ ($U_{5,19,0.05} = 23 < U_{5,19,0.10} = 28 < u_a = 43$) and $H4_b$ ($U_{9,16,0.05} = 42 < U_{9,16,0.10} = 48 < u_b = 69.5$) as well suggest that the p-values have been calculated correctly. Again, the null hypotheses are concluded indicating that both sportive and rarely active people that consider exergames would play them in equal frequencies. This assumption can be regarded even stronger due to the fact that both studies show similar results for H4.

H5: Less active people would rather consider working out more by playing exergames than regularly active persons.

Test variable: *increaseActivity*, partitions ($A_1 \cap B_1 = \emptyset, A_2 \cap B_2 = \emptyset$):

$$A_1 = \{x \in FC | \neg x.isRegActive \wedge x.considersEGames\},$$
$$B_1 = \{y \in FC | y.isRegActive \wedge y.considersEGames\},$$
$$H5_{a_0} : A_1 = B_1,\ H5_{a_1} : A_1 > B_1\ (P(U_{A_1 \subseteq FC} \geq u_{A_1 \subseteq FC})),$$
$$A_2 = \{x \in Q3A | \neg x.isRegActive \wedge x.considersEGames\},$$
$$B_2 = \{y \in Q3A | y.isRegActive \wedge y.considersEGames\},$$
$$H5_{b_0} : A_2 = B_2,\ H5_{b_1} : A_2 > B_2\ (P(U_{A_2 \subseteq FC} \geq u_{A_2 \subseteq FC}))$$

For H5 the data has been subdivided into participants considering exergames, while either being regularly active or working out more less consistently. The hypothesis assumes that participants belonging to the latter group are more willing to increase the amount of time they spend on exercising than persons that are already practicing sports on a regular basis. This at first seems to partially contradict H4, but the key difference here is that H5, in contrast to the general estimated exergaming frequency conveyed by $frequencyEGames$, really observes the people's inclination to improve upon their exercising habits indicated by *increaseActivity*. Similarly to the previous procedure, H5 also allows for MWW tests on both data sets yielding again more significant results ($p_a = 0.057$, $p_b = 0.089$). Although not quite enough for rejecting $H5a_0$ and $H5b_0$ at a significance level of $\alpha = 0.05$, the values are seemingly enough if α were set to 0.10. However, under these particular circumstances the critical U value listings suggest otherwise: $U_{13,11,0.05} = 42 < U_{13,11,0.10} = 48 < u_a = 50$ ($H5a_0$) and $U_{5,19,0.05} = 23 < U_{5,19,0.10} = 28 < u_b = 55$ ($H5_b$). Since u_a is very close to $U_{13,11,0.10}$, it might be reasonable to reject $H5a_0$, but $u_b = 55$ is not as close to $U_{5,19,0.10}$, which gives reason to believe that the actual likelihood of H_0 lies somewhere between $10 - 20\%$. In any case, this hypothesis should be investigated further in future studies, preferably with larger sample sizes that are sufficiently distributed between the group partitions.

[10] For abbreviation all subsequent U table values will be denoted via $U_{n_A,n_B,\alpha}$.

H6: Regularly active occasional gamers would rather play AAA exergames than casual exergames at home.

Test variable: $considersEGames$, partitions ($FC \cap Q3A = \emptyset$):

$$A = \{x \in Q3A | x.isRegActive \wedge \neg x.isRegGamer\},$$
$$B = \{y \in FC | y.isRegActive \wedge \neg y.isRegGamer\},$$
$$H6_0 : A = B,\ H6_1 : A > B\ (P(U_{A \subseteq Q3A} \geq u_{A \subseteq Q3A}))$$

The final hypothesis addresses regularly exercising participants that merely play games sometimes or even less. A MWW test is again conducted on both FC and Q3A participants, examining if they rather would consider playing AAA or casual exergames at home. Apparently, only very few people satisfy these criteria ($n_A = 3$, $n_B = 8$) making this hypothesis hard to evaluate: results yield a p-value of 0.882, supporting $H6_0$ and even suggesting a distribution shift in the opposite direction of $H6_1$'s assumptions. The low sample size allows for looking up the exact p-value in Table B.41, which considering the hypothesis' direction and the observed u value of 7.5 amounts to a probability somewhere between $[1 - 0.248, 1 - 0.188] = [0.752, 0.812]$, slightly less than the calculated p but still very insignificant. As expected the critical U tables as well list lesser values: $U_{3,8,0.05} = 3 < U_{3,8,0.10} = 5 < u_a = 7.5$. Therefore, $H6_0$ can not be rejected and again, a larger set of samples should paint a clearer picture on the given assumptions.

6.2.4 Correlations and Further Hypothesis Development

Statistical correlation measures are useful for determining whether two or more variables are dependent on each other or not. An encountered correlation between x and y can indicate a one- as well as a two-way causal influence ($x \rightarrow y$, $y \rightarrow x$, $x \leftrightarrow y$), but also could be triggered by an additional variable z or even occurred entirely by chance. Such causalities are expressed by means of a correlation coefficient r, which typically is a real number within the interval $[-1.0, 1.0]$, where both extremes denote perfect positive and negative correlation respectively. That is to say, for example, when applying a specific correlation test on two ordinal variables x and y, i.e. calculating `corr(`x`, `y`)`, the outcome $r = 1$ would imply that they are agreeing on each rank, containing exactly the same ratings, while $r = -1$ on the other hand were to specify their total disagreement – given rankings oppose each other in every way. Since the goal is to detect dependencies (H_1), the null hypothesis (H_0), as usual, makes the opposite assertion: the variables are *not* correlated ($r = 0$). Several methods have been developed for determining r and three of them have been selected, due to their capabilities of handling all kinds of the previously defined data set variables: the ϕ-coefficient [144] for two dichotomous features, the Kendall rank correlation (KRC) with tie correction [145, 146] used for ordinal data

comparison and the rank-biserial correlation (RBC) [147] applied to a pair consisting of both variable types.

The ϕ-coefficient, a dichotomous special case of the Pearson product-moment correlation coefficient [148], analyzes a contingency table, i.e. a matrix containing the frequency distribution of variable values, formed out of the bi-valued input data. This creates a 2×2 tabular containing frequencies a, b, c and d, as outlined in Table 6.4.

TABLE 6.4: 2×2 contingency table.

	$v2_x$	$v2_y$	rows total
$v1_x$	a	b	$a+b$
$v1_y$	c	d	$c+d$
cols total	$a+c$	$b+d$	$n=a+b+c+d$

A pair of dichotomous variables $v1$ and $v2$, used to categorize one or two particular data sets, are cross tabulated against each other via listing the individual occurrence rates of x and y values. For example, $v1$ could separate a sample population by gender and $v2$ might indicate smoking or nonsmoking test subjects. Rows and columns summations are called marginal totals, which, when added up, amount to a grand total of n. Having calculated all these values, following formula can be used to determine the correlation r_ϕ, measuring the deviation between observed and expected table values:

$$r_\phi = \frac{a \cdot d - b \cdot c}{\sqrt{(a+c) \cdot (b+d) \cdot (a+b) \cdot (c+d)}} \tag{6.7}$$

Under the assumption that the frequency counts are approximately normally distributed, a significance test may be conducted via the statistic $\chi^2_r = n \cdot r^2_\phi$, yielding a χ^2 distributed value. The occurrence probability of χ^2_r can be determined by inserting it into the cumulative χ^2 distribution function with 1 degrees of freedom[11] ($x = \chi^2_r, k = 1$):

$$\chi^2(x,k) = \frac{\int_0^{\frac{x}{2}} t^{\frac{k}{2}-1} e^{-t} dt}{\int_0^{\infty} t^{\frac{k}{2}-1} e^{-t} dt} \tag{6.8}$$

Cumulative distribution functions always yield probabilities (p-values) of observing values at most as large as their input parameters ($p(\chi^2 \leq \chi^2_r | H_0)$). However, the hypotheses

[11] The degrees of freedom characteristic (k) determines which χ^2 distribution is used – it is calculated using the formula $k = (|v_x| - 1)(|v_y| - 1)$, where $|v_x|$ and $|v_y|$ symbolize the amount of values each variable can take, thus for 2×2 contingency tables resulting in $k = 1$.

in this case are interested in values at least as extreme ($p(\chi^2 \geq \chi_r^2|H_0)$) because bigger deviations of observed from expected values indicate larger correlations. Therefore, the opposite probability ($1 - p(\chi^2 \leq \chi_r^2|H_0)$) needs to be calculated in order to determine the significance of the variables' mutual dependency: values lower than a small chosen α convey a slim likelihood of H_0 being true, thus H_1 can be assumed.

The KRC can be calculated for data sets partitioned by two ordinal variables (A, B) and is based on ranking any possible relation between cross variable observations ($i \neq j$, $x \in A$, $y \in B$): the sum of all pairs with the characteristics $(x_i < y_i) \wedge (x_j < y_j)B$ or $(x_i > y_i) \wedge (x_j > y_j)$ is declared the number of concordant pairs (n_c), while $(x_i < y_i) \wedge (x_j > y_j)$ or $(x_i > y_i) \wedge (x_j < y_j)$ identifies the sum of discordant pairs (n_d). Pairs with equal values, i.e. ties, are counted separately for each variable and summed up resulting in n_0 as well as n_1. With $n_0 = \frac{n \cdot (n-1)}{2}$ denoting the maximum potential amount of pair combinations, the Kendall coefficient[12] is determined by following formula:

$$r_{KRC} = \frac{n_c - n_d}{\sqrt{(n_0 - n_1) \cdot (n_0 - n_2)}} \tag{6.9}$$

Ordinarily, it would simply suffice to divide the difference between both kinds of pair combinations by n_0 in order to retrieve the normalized balance between them, but the expression in the denominator also accounts for omitting the total amount of ties in the numerator. Thus, the measure can take any value within $[-1.0, 1.0]$, 0 again identifying no correlation at all, which corresponds to H_0 for the significance test. The approximately normally distributed evaluation statistic z_{KRC} as well needs to handle rank ties, but its calculation then becomes rather extensive and therefore only parts of it are shown here in order to outline the general strategy:

$$z_{KRC} = \frac{n_c - n_d}{\sqrt{\frac{n(n-1)(2n+5) - v_0 - v_1}{18} + v_2 + v_3}} \tag{6.10}$$

The variables v_0, v_1, v_2 and v_3 in the denominator are used for tie correction by using their occurrence counts for specific variable values [149] and the p-value for accepting or rejecting H_0 is determined by inserting $-|z_{KRC}|$ into the cumulative normal distribution function (Formula 6.6), doubling the result.

[12] The Kendall coefficient with tie correction is usually identified with τ-b, but here has been altered for keeping a consistent naming scheme for all presented coefficients (r_ϕ, r_{KRC}, r_RBC).

RBC coefficients between dichotomous and an ordinal variables[13] are calculated very similarly to the MWW test described in Section 6.2.3. Formula 6.1 with some small modifications may also be used to calculate the U value for the second population B or an even simpler shortcut via the relation $U' = n_A \cdot n_B - U$ can be taken. The correlation r_{RBC} is then determined by:

$$r_{RBC} = \frac{U - U'}{n_A \cdot n_B} \tag{6.11}$$

The denominator accounts for the maximum possible U value, rendering the result again a value between $[-1.0, 1.0]$. Testing for significance largely remains equivalent to the already described procedure: Formulas 6.2, 6.4 and 6.5 are used to calculate the test statistic z for a two-tailed evaluation, which once more makes use of the cumulative standard normal distribution function (Formula 6.6), finally yielding $p = 2 \cdot \Phi(-|z|)$.

Similarly to the previous analyses, all correlations have been determined with R utilizing `phi` from the `psych` package (r_ϕ), the standard `cor.test` function with the "kendall" method (r_{KRC}) and a custom implementation of the RBC (r_{RBC}). The coefficients were calculated for every possible variable combination partitioning both the FC and the Q3A data set respectively[14], which, given that they only have to be paired once[15], amounts to $\frac{23 \cdot (23-1)}{2} = 276$ coefficients each, all of them fully listed in Tables 6.5 and 6.6.

General guidelines for interpreting r values have proposed by Jacob Cohen [150], also known as measures for the effect size:

$$\begin{aligned} \text{small correlation:} &\quad 0.1 < |r| < 0.3 \\ \text{medium correlation:} &\quad 0.3 < |r| < 0.5 \\ \text{large correlation:} &\quad 0.5 < |r| \end{aligned}$$

However, these reference points are not necessarily always suitable for interpreting the observations at hand, as becomes apparent when regarding the correlation matrices for both studies' variables: sample groups with very skewed value distributions correlate to each other, simply because they agree or disagree very often. For example, $isCasualGamer$ in the Q3A coefficient table shows a highly negative correlation to $isActive$ ($r = -0.89$, $\alpha = 0.01$), which would implicate that the number of casual gamers inversely relates

[13] Note that the RBC can only be calculated in following specific fashion: the bi-valued variable partitions the data set, while the ordinal variable is used to determine the ranks (T_A,T_B) of both resulting subsets (A, B).

[14] Note that correlations between the data sets can only be computed when their sizes are the same, which would be achievable by removing one record from the Q3A set but in order to retain comparability to previous analyses this has not been attempted.

[15] Since coefficient calculations have been conducted in a non-directed manner, the functions always yield the same results regardless of the variable pairing, rendering a correlation matrix symmetrical.

TABLE 6.5: Correlation coefficients (r_ϕ, r_{KRC}, r_{RBC}) for FC.

significance
bold $\alpha = 0.01$
italic $\alpha = 0.05$
ttype $\alpha = 0.10$

symbols
[n] # values
x # values observed < 2 in $v1$ or $v2$

	[2]gender	[5]age	[2]isStudent	[2]isTechnician	[2]isActive	[2]isRegActive	[5]frequencyActive	[2]isCyclist	[2]isGamer	[2]isRegGamer	[5]frequencyGames	[2]isFPSGamer	[2]isCasualGamer	[2]likesEGames	[2]isEGamer	[2]considersEGames	[5]frequencyEGames	[2]increaseActivity	[5]ergoXP	[5]gameXP	[5]gameRating	[5]gameDifficulty	[5]overallXP
[2]gender	**1**																						
[5]age	0.005	**1**																					
[2]isStudent	*-0.277*	-0.174	**1**																				
[2]isTechnician	**-0.336**	0.035	-0.061	**1**																			
[2]isActive	*0.285*	-0.097	**0.528**	**0.579**	**1**																		
[2]isRegActive	*-0.248*	0.037	0.03	0.092	**-0.502**	**1**																	
[5]frequencyActive	0.06	0.066	-0.256	0.292	*1*	**1**	**1**																
[2]isCyclist	-0.105	0.232	0.194	0.226	**0.69**	0.131	**-0.568**	**1**															
[2]isGamer	-0.066	**-0.562**	0.214	*0.273*	**-0.343**	0.184	0.07	0.069	**1**														
[2]isRegGamer	*-0.321*	-0.06	0.047	-0.091	**0.507**	-0.091	-0.197	-0.182	**0.507**	**1**													
[5]frequencyGames	*0.562*	-0.102	0	0.017	x	-0.033	0.017	-0.162	x	**1**	**1**												
[2]isFPSGamer	*-0.365*	-0.062	0	-0.138	**0.471**	-0.138	0.196	-0.228	**0.471**	-0.047	-0.134	**1**											
[2]isCasualGamer	**-0.596**	0.368	-0.262	**-0.39**	0.27	**-0.39**	-0.263	**-0.472**	0.27	*-0.306*	0.404	-0.262	**1**										
[2]likesEGames	x	x	x	x	x	x	x	x	x	x	x	x	x	x									
[2]isEGamer	**-0.596**	0	-0.262	**-0.39**	0.27	**-0.39**	0.614	**-0.472**	0.27	*-0.306*	-0.263	-0.262	0	x	**1**								
[2]considersEGames	0	0.144	*0.277*	**0.336**	*-0.285*	*0.248*	0.037	0.105	0.066	*0.321*	0.271	*0.365*	**0.596**	x	**0.596**	**1**							
[5]frequencyEGames	-0.176	0.163	-0.077	0.189	*-1*	-0.133	-0.094	0.083	-0.281	0.317	0.197	-0.073	0.286	x	-0.308	x	**1**						
[2]increaseActivity	0	0.109	0.254	0.254	*-0.335*	0.254	-0.361	0.046	0.045	0.318	-0.167	*0.438*	**0.63**	x	**0.564**	**0.741**	*0.471*	**1**					
[5]ergoXP	-0.139	0.273	-0.189	0.192	0.661	0.294	0.28	0.023	-0.004	-0.12	-0.041	0.062	0	x	0	0.056	-0.148	-0.109	**1**				
[5]gameXP	-0.185	-0.019	-0.33	0.158	0.597	0.051	0.128	-0.114	0.298	-0.179	-0.045	0.071	0.228	x	-0.123	0.032	-0.059	-0.118	0.242	**1**			
[5]gameRating	0.032	0.135	0.137	0.173	0.258	-0.114	-0.104	0.005	0.041	-0.239	-0.123	-0.071	0.491	x	0.211	*0.468*	0.196	0.294	-0.019	0.184	**1**		
[5]gameDifficulty	0.296	0.059	-0.204	0.312	-0.048	0.088	0.04	0.123	0.07	0.214	0.161	0.223	0.579	x	-0.158	0.009	0.168	-0.059	0.1	-0.138	0.091	**1**	
[5]overallXP	-0.014	-0.091	*-0.411*	**0.662**	*0.806*	0.239	0.241	-0.209	*0.483*	0.034	0.078	0.188	0.158	x	0.158	0.31	0.18	0.059	0.226	*0.332*	*0.381*	0.115	**1**

TABLE 6.6: Correlation coefficients (r_ϕ, r_{KRC}, r_{RBC}) for Q3A.

significance
bold $\alpha = 0.01$
italic $\alpha = 0.05$
ttype $\alpha = 0.10$

symbols
[n] # values
x # values observed < 2 in $v1$ or $v2$

	[2]gender	[5]age	[2]isStudent	[2]isTechnician	[2]isActive	[2]isRegActive	[5]frequencyActive	[2]isCyclist	[2]isGamer	[2]isRegGamer	[5]frequencyGames	[2]isFPSGamer	[2]isCasualGamer	[2]likesEGames	[2]isEGamer	[2]considersEGames	[5]frequencyEGames	[2]increaseActivity	[5]ergoXP	[5]gameXP	[5]gameRating	[5]gameDifficulty	[5]overallXP
[2]gender	**1**																						
[5]age	-0.101	**1**																					
[2]isStudent	**-0.359**	-0.193	**1**																				
[2]isTechnician	**-0.387**	0.297	-0.03	**1**																			
[2]isActive	*0.249*	0.047	**0.566**	**0.591**	**1**																		
[2]isRegActive	**-0.443**	0.071	-0.092	-0.062	**-0.64**	**1**																	
[5]frequencyActive	0.01	-0.008	-0.054	0.125	*1*	**1**	**1**																
[2]isCyclist	**-0.471**	0.073	-0.098	-0.098	**0.43**	-0.033	0	**1**															
[2]isGamer	0.035	-0.032	**0.391**	**0.418**	-0.217	**0.474**	-0.101	**0.504**	**1**														
[2]isRegGamer	*-0.338*	0.136	0.148	0.148	**-0.445**	*0.298*	-0.085	*0.309*	**0.649**	**1**													
[5]frequencyGames	-0.194	0.224	0.039	0.111	0.423	-0.046	-0.155	0.035	x	**1**	**1**												
[2]isFPSGamer	**-0.406**	0.066	0.074	0.074	**-0.507**	0.226	-0.077	0.196	**0.592**	-0.075	*0.445*	**1**											
[2]isCasualGamer	**-0.815**	*0.76*	**-0.423**	**-0.423**	**-0.89**	*-0.286*	-0.06	*-0.293*	0.196	**-0.55**	0.04	**-0.487**	**1**										
[2]likesEGames	0.2	0.226	**0.528**	**0.554**	-0.056	**0.605**	0.269	**0.608**	0.166	**0.39**	0.1	**0.455**	**0.852**	**1**									
[2]isEGamer	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								
[2]considersEGames	-0.034	0.133	**0.327**	**0.355**	*-0.28*	**0.412**	-0.089	**0.406**	-0.069	0.199	0.429	*0.271*	**0.712**	-0.231	x	**1**							
[5]frequencyEGames	-0.167	-0.058	0.039	-0.113	0.022	-0.103	-0.068	0.056	0.298	0.204	0.083	-0.074	-0.079	x	x	x	**1**						
[2]increaseActivity	0.2	-0.091	**0.464**	**0.5**	-0.067	**0.572**	-0.621	**0.495**	0.058	0.224	0.056	*0.316*	**0.763**	**0.886**	x	**0.886**	0.545	**1**					
[5]ergoXP	0.327	0.142	-0.014	0.081	0.375	0.174	-0.013	-0.082	0.233	0.375	0.289	0.121	0.44	-0.065	x	-0.062	0.262	-0.091	**1**				
[5]gameXP	0.274	-0.16	0.3	0.018	0.094	-0.375	-0.165	0.045	*0.471*	-0.045	-0.061	0.187	**-1**	-0.054	x	-0.067	0.223	-0.076	0.237	**1**			
[5]gameRating	0.351	-0.092	0.071	0.055	0.125	0.119	-0.008	-0.118	0.185	0.403	0.333	0.22	-0.04	-0.022	x	0.009	0.174	0.288	*0.382*	0.182	**1**		
[5]gameDifficulty	0.029	0.029	*-0.357*	0.238	0.312	-0.032	0.019	0.259	-0.185	-0.273	-0.22	-0.049	0.22	0.323	x	-0.151	-0.149	-0.061	-0.199	-0.212	**-0.419**	**1**	
[5]overallXP	0.058	-0.228	0.189	-0.147	0.484	0.016	0.019	0.223	0.354	*0.398*	0.271	0.06	-0.62	0.366	x	0.102	*0.459*	0.136	**0.493**	**0.464**	*0.348*	-0.146	**1**

to the amount of active people. Since merely two Q3A testers were casual gamers and only two physically inactive people participated, common sense conveys that this "significant" result rather occurred by chance than through an actual relation. This notion can be confirmed by consulting the FC data set, which exhibits similar distributions for these variables (3 casual gamers, 2 inactive persons): the corresponding table lists a weak positive correlation ($r = 0.27$, $\alpha = 0.10$), very much contradicting the Q3A result. Several variables with skewed value distributions can be identified and therefore will subsequently be excluded from any further analyses: *isCasualGamer*, *isActive*, *isEGamer* and *likesEGames*.

Furthermore, it is possible to draw meaningless conclusions when failing to identify variables inherently dependent on each other, as they of course will show high correlations. This can, for instance, be observed when inspecting the relations between *isRegActive* and *frequencyActive*, exhibiting perfect positive coefficients in both tables. All such dependencies have already been pointed out along with first introducing the variables in Section 6.2.1, Table 6.2.

Lastly, a closer inspection of the listed values suggests that the correlation measurements vary in their definitions of r: $r_{RBC} = 0.361$ resulting from `corr(`*increaseActivity*`,` *frequencyActive*`)` in the FC table is not significant at $\alpha = 0.10$, whereas these conditions are met by `corr(`*ergoXP*`,` *frequencyActive*`)` yielding a lower value of $r_{KRC} = 0.28$. The ϕ-coefficient seems to have the lowest threshold of around 0.2 for the same significance level, indicated by `corr(`*likesEGames*`,` *gender*`)` in the Q3A table. Therefore, a general margin of about 0.1 should be considered when comparing these different coefficients. Also, p-values seem to depend on the underlying data, as is shown in the FC table coefficient between the skewed variable *isActive* and *gameXP*, where $r_{RBC} = 0.597$ is still not enough for significance level $\alpha = 0.10$.

Such discoveries leave much room for misinterpretations, hence, analyses should always reach beyond merely calculating data correlation. The rest of this section is concerned with outlining promising variable relations in both tables, which can be utilized to devise additional hypotheses – a summary of the findings is listed in Table 6.7.

TABLE 6.7: Evaluation of Hypotheses: MWW results overview (u = observed U value).

hypothesis (H_1)	n_A	n_B	p	z	u	σ_u	**reject** H_0
H7: $A_{\subseteq FC} < B_{\subseteq Q3A}$	11	8	0.187	-0.888	35	44	no
H8: $A_{\subseteq FC} < B_{\subseteq Q3A}$	20	10	0.171	-0.952	85	100	no
$H9_a$: $A_{2\subseteq FC} < B_{2\subseteq Q3A}$	15	14	0.001	-2.980	42	105	for $\alpha = 0.01$

$H9_b$: $A_{1\subseteq FC} < B_{1\subseteq Q3A}$	22	27	0.030	-1.880	219.5	297	for $\alpha = 0.01$ ($u < U_{\alpha=0.01}$)
$H9_c$: $A_{3\subseteq FC} < B_{3\subseteq Q3A}$	33	34	0.006	-2.500	385.5	561	for $\alpha = 0.01$
H10: $A_{\subseteq FC} < B_{\subseteq Q3A}$	22	27	0.003	-3.481	140.5	297	for $\alpha = 0.01$
$H11_a$: $A_{1\subseteq FC} < B_{1\subseteq FC}$	21	12	0.145	-1.058	101	126	no
$H11_b$: $A_{2\subseteq Q3A} < B_{2\subseteq Q3A}$	8	26	0.010	-2.320	54.5	104	for $\alpha = 0.01$
H12: $A_{\subseteq FC} < B_{\subseteq Q3A}$	9	16	0.167	-0.967	56.5	72	no
$H13_a$: $A_{1\subseteq FC} < B_{1\subseteq FC}$	17	16	0.051	-1.639	96	136	for $\alpha = 0.10$
$H13_b$: $A_{2\subseteq Q3A} < B_{2\subseteq Q3A}$	23	11	0.179	-0.918	104.5	126.5	no

H7: Regularly active people would rather consider working out more playing AAA exergames than casual exergames.

Test variable: *increaseActivity*, partitions ($FC \cap Q3A = \emptyset$):

$$A = \{x \in FC | x.isRegActive\},$$
$$B = \{y \in Q3A | y.isRegActive\},$$
$$H7_0 : A = B,\ H7_1 : A < B\ (P(U_{A\subseteq FC} \leq u_{A\subseteq FC}))$$

This hypothesis is based on the observation that `corr(`*isRegactive*`,` *incActivity*`)` yields significant variable dependencies for both data sets, but on different α levels ($\alpha_{FC} = 0.10$, $\alpha_{Q3A} = 0.01$). In contrast to H5[16], which focuses on *increaseActivity* for less regularly active people within the FC and the Q3A samples, H7 concentrates on the other regularly active portion to evaluate their tendencies towards AAA exergames. An MWW test shows that the given responses generally are in favor of $H7_1$ ($p = 0.187$), but no concrete evidence to be able to reject $H7_0$ can be discovered. This is confirmed when examining the critical U value tables for the rather small subset partition sizes of $n_A = 11$ and $n_B = 8$ ($U_{11,8,0.05} = 23 < U_{11,8,0.10} = 27 < u_a = 35$). Thus, aside from the fact that the addressed persons already work out regularly, it remains unclear whether they would rather consider exercising even more frequently using casual or AAA games.

H8: Cyclists would rather consider playing AAA exergames than casual exergames.

Test variable: *considersEGames*, partitions ($FC \cap Q3A = \emptyset$):

$$A = \{x \in FC | x.isCyclist\},$$

[16] H5: Less active people would rather consider working out more by playing exergames than regularly active persons.

$$B = \{y \in Q3A | y.isCyclist\},$$
$$H8_0 : A = B,\ H8_1 : A < B\ (P(U_{A \subseteq FC} \leq u_{A \subseteq FC}))$$

The *isCyclist* variable shows a highly significant correlation with *considersEGames* in the Q3A data set ($r = 0.406$, $\alpha = 0.01$), but no remarkable relation in the FC samples ($r = 0.105$). H8 inspects if this lack of correlation can be attributed to those people's exergaming preferences. Again, a general tendency against the null hypothesis can be observed ($p = 0.171$), but as well no significant evidence is found ($U_{20,10,0.05} = 62 < U_{20,10,0.10} = 70 < u_a = 85$). Therefore, bike riding as a particular exercising habit is not necessarily an indicator for a person to favor AAA over casual fitness games.

H9: Students/gamers/all participants enjoy playing AAA exergames more than they do playing casual exergames.

Test variable: $overallXP$, partitions ($A_1 \cap B_1 = \emptyset, A_2 \cap B_2 = \emptyset, FC \cap Q3A = \emptyset$):

$$A_1 = \{x \in FC | x.isStudent\},$$
$$B_1 = \{y \in Q3A | y.isStudent\},$$
$$H9_{a_0} : A_2 = B_2,\ H9_{a_1} : A < B\ (P(U_{A_2 \subseteq FC} \leq u_{A_2 \subseteq FC}))$$
$$A_2 = \{x \in FC | x.isGamer\},$$
$$B_2 = \{y \in Q3A | y.isGamer\},$$
$$H9_{b_0} : A_1 = B_1,\ H9_{b_1} : A < B\ (P(U_{A_1 \subseteq FC} \leq u_{A_1 \subseteq FC}))$$
$$H9_{c_0} : FC = Q3A,\ H9_{c_1} : FC < Q3A\ (P(U_{FC} \leq u_{FC}))$$

Applying `corr(`$isStudent$`,` $overallXP$`)` reveals a significantly negative coefficient for FC data ($r = -0.411$, $\alpha = 0.05$), yet, a less remarkable but positive result for Q3A ($r = 0.189$). MWW tests on $H9_a$ evaluate if students overall had a better experience participating in the Q3A study than attending the FC study. This hypothesis can be further expanded to inspecting all gamers experience accordingly ($H9_b$), as they show high correlations in both cases ($\alpha_{FC} = 0.05$, $\alpha_{Q3A} = 0.10$), and even more extended to considering the whole data sets ($H9_c$) in order to confirm observations made in the qualitative analysis, while as well computing their significance. All three hypotheses show significant results for $\alpha = 0.01$, which is not obvious for $H9_a$ at first because the p-value of 0.030 suggests a different α level, but due to its smaller data subset ($n_A = 15$, $n_B = 14$), the exact critical U values can be found in Table B.44 and B.43: $u_a = 45 < U_{15,14,0.01} = 51 < U_{15,14,0.05} = 66$. Hence, all previous speculative findings about the overall user experience seem accurate and the AAA game experiment has been more enjoyable.

H10: Gamers/all participants have better experience with the ergometer when playing a AAA exergame instead of a casual exergame.

Test variable: $ergoXP$, partitions ($FC \cap Q3A = \emptyset$):

$$A_1 = \{x \in FC | x.isGamer\},$$
$$B_1 = \{y \in Q3A | y.isGamer\},$$
$$H10_{a_0} : A = B,\ H10_{a_1} : A < B\ (P(U_{A \subseteq FC} \leq u_{A \subseteq FC}))$$
$$H10_{b_0} : FC = Q3A,\ H10_{b_1} : FC < Q3A\ (P(U_{FC} \leq u_{FC}))$$

While *isGamer* seems to be completely unrelated to $ergoXP$ in the FC data, a small insignificant correlation can be discovered in the Q3A table. This hypothesis' MWW test again compares both datasets regarding gamers and more generally all participants to determine if ergometer handling has been perceived better in the second study. $H10_a$ as well as $H10_b$ outcomes are very significant ($\alpha = 0.01$), confirming the assumptions of the hypothesis and leading to another pressing question: does ergometer experience influence the overall experience? This subject is addressed in the next hypothesis H11.

H11: Lower ergometer experience influences overall experience negatively.

Test variable: $overallXP$, partitions ($A_1 \cap B_1 = \emptyset, A_2 \cap B_2 = \emptyset$):

$$A_1 = \{x \in FC | x.ergoXP \leq 3\},$$
$$B_1 = \{y \in FC | y.ergoXP > 3\},$$
$$H11_{a_0} : A = B,\ H11_{a_1} : A < B\ (P(U_{A \subseteq FC} \leq u_{A \subseteq FC}))$$
$$A_2 = \{x \in Q3A | x.ergoXP \leq 3\},$$
$$B_2 = \{y \in Q3A | y.ergoXP > 3\},$$
$$H11_{b_0} : A = B,\ H11_{b_1} : A < B\ (P(U_{A \subseteq Q3A} \leq u_{A \subseteq Q3A}))$$

Both data sets feature positive correlations between $ergoXP$ and $overallXP$, Q3A even on a highly significant level ($\alpha = 0.01$). H11 regards the studies individually and can therefore be seen as a one-tailed alternative test for verifying the direction of the KRC determined measures. Subsets are created for people that had an average or worse ergometer experience and the opposite kind of persons. The results reflect the discovered coefficients, although not as significantly (FC: $p = 0.145$, Q3A: $p = 0.051 / U_{17,16,0.05} = 83 < u_a = 96 < U_{17,16,0.10} = 99$). These findings together with the KRC discoveries give reasons to believe that Q3A participants' ergometer experiences influenced their overall judgements more severely than it has been the case in the FC study.

H12: Regular Gamers rate AAA exergames higher than casual exergames.

Test variable: $gameRating$, partitions ($FC \cap Q3A = \emptyset$):

$$A = \{x \in FC | x.isRegGamer\},$$
$$B = \{y \in Q3A | y.isRegGamer\},$$
$$H12_0 : A = B,\ H12_1 : A < B\ (P(U_{A \subseteq FC} \leq u_{A \subseteq FC}))$$

A fairly high correlation resulting from `corr(`*isRegGamer*`,`*gameRating*`)` can be discovered in the Q3A table ($r = 0.403$, $\alpha = 0.10$), while the corresponding FC coefficient exhibits a negative relation ($r = -0.239$). An MWW test is used determine whether these opposing correlations are game related. FC and Q3A are partitioned into subsets consisting of regular gamers and their comparison should verify that overall they rated the AAA game better than the casual one. The test results are not significant enough for rejecting $H12_0$ ($p = 0.167$, $U_{9,16,0.05} = 42 < U_{9,16,0.10} = 48 < u_a = 56.5$), but suggest the gamers' tendency to favor Q3A over FC.

H13: Less regular active people are less comfortable handling the ergometer than regularly active persons.

Test variable: $ergoXP$, partitions ($FC \cap Q3A = \emptyset$):

$$A_1 = \{x \in FC | \neg x.isRegActive\},$$
$$B_1 = \{y \in FC | y.isRegActive\},$$
$$H13_{a_0} : A = B,\ H13_{a_1} : A < B\ (P(U_{A \subseteq FC} \leq u_{A \subseteq FC}))$$
$$A_2 = \{x \in Q3A | \neg x.isRegActive\},$$
$$B_2 = \{y \in Q3A | y.isRegActive\},$$
$$H13_{b_0} : A = B,\ H13_{b_1} : A < B\ (P(U_{A \subseteq Q3A} \leq u_{A \subseteq Q3A}))$$

As the final hypothesis, H13 inspects $ergoXP$ for more and less regularly active people in both data sets. It is based on observing moderately positive agreement for `corr(`*isRegActive*`,` *ergoXP*`)` in the correlation matrices. To evaluate the direction of these relations, H11 assumes that less active people rate their ergometer experience worse than their counterparts. Outcomes show that this assumption only holds for FC participants on a significance level of $\alpha = 0.10$ ($U_{17,16,0.05} = 89 < u_a = 96 < U_{17,16,0.10} = 99$). Although less regularly active Q3A testers as well lean towards being more uncomfortable riding the ergometer, the results are too inconclusive ($p = 0.179$), rendering this hypothesis as well as H5, H7, H8, H11 and H12 subjects for future studies.

6.3 Summary

A hybrid approach has been chosen for evaluating survey responses produced by 67 study participants. The initial qualitative data inspection suggests that the volunteers – predominantly young male students or technicians – mostly were committed to giving honest and constructive answers. Generally, the studies were very well received, although overall the Quake 3 Arena mod has been better accepted. Most issues reported during the experiments were regarded control-specific, which could have introduced a bias, hence, should appropriately be addressed in future experiments. Twofold quantitative

analyses follow a confirmatory approach as well as investigating data correlations in an explorative manner. Since most of the questionnaire variables are ordinal, a Mann-Whitney-Wilcoxon test is applied for verifying research-relevant hypotheses. Findings include strong tendencies towards preferring AAA exergames over casual fitness games when examining students, gamers and active people. Moreover, controlling Quake 3 Arena with an ergometer seems to have been received far better than using it to operate Flappy Cycling. However, it can not be confirmed that the people's fitness level nor their specific interest in bike riding are indicators for them to consider using fitness gaming devices at home. Nevertheless, clear tendencies can be discovered and thus such assumptions should be of interest for further investigations.

Chapter 7

Conclusion

This thesis has introduced fitnessification as a way of improving upon current exergaming concepts. Its major scientific contributions, namely prototype implementations, studies and their evaluations will be laid out in this final chapter. Additionally, research questions will be answered in Section 7.1, limitations presented in 7.2 and future work outlined in 7.3.

Seemingly both **studies** represented two prototypes on trial to find out which one of them is the superior. In truth, however, they resemble two different worlds – one being immersive and addicting, the other representing a fun pastime. Flappy Cycling was modeled after "Flappy Bird", a casual smartphone game simply played at a bus stop or any other situation boring enough for wanting to kill time. Quake 3 Arena is, or at least has been, a challenging FPS played all night and probably throughout the next day if those plans were not thwarted by the lame old job. The prototypes were created with these considerations in mind: Flappy Cycling is a gamified workout solution that features a game for additional challenge, while the Quake 3 Arena mod is a finessified game featuring an ergometer as an additional input method.

This is as well reflected in their **implementations**, as FC had to be created from scratch using HTML5 based game framework Phaser.js, keeping the introduced fitness elements in mind, whereas it has been necessary to modify the Q3A ANSI C code to accomodate those elements, diverting the game from its original intention.

A difference in the two approaches is as well perceivable when regarding the **evaluations** of the 67 participants' questionnaire responses. The overall collected study ratings for Q3A are significantly higher than the ones for FC. Despite having identified that physically active volunteers would rather play AAA exergames than casual fitness games,

no clear further connection between people's exercising habits and fitness gaming preferences could be determined. This notion can be confirmed by regarding the data sets' correlation tables and discovering that the variable corresponding to "physically active" is very skewed, describing almost all participants and exhibiting high correlations to many other variables.

7.1 Research Questions Answered

Q1 How do (sportive) people feel about exergaming?

Although only very few participants actually play exergames, responses are overly positive and the majority of persons view exergames as healthy, entertaining, motivating and a novelty. Given that many would consider working out with either of the studies' devices and games, it can be presumed that there is still potential in fitness gaming.

Q2 Can exergaming be more fun?

Judging by the significant results (H1, H2, H9, H10) this seems to be the case. However biases can not be ruled out, as control issues seemed to be disturbing factors in both studies.

Q3 Given the AAA game integration, would it be possible to motivate people to work out more?

It appears that less sportive regular gamers are not very much inclined to do so (H3), however, a tendency can be found when regarding both collected data sets separately (H5), which is definitely appropriate for justifying future research.

Q4 Which conditions must be satisfied for gamers to consider exergaming at home?

The overall positive results convey that the most important element is fun, which is also reflected by the participants' comments. Furthermore, the games must be responsive in terms of controls and their level of challenge must should adopt to the players. Lastly, volunteers' feedback conveys that they must be affordable, which as well underlines the importance of fun, because hardly anybody would want to buy expensive devices that are boring to operate.

7.2 Limitations

Although 67 persons are seemingly enough to conduct a thorough study, it certainly must be considered that they are equally split between two separate studies, which renders the samples less comparable and through partitioning both data sets often very few samples remain, occasionally not enough for a meaningful evaluation.

Also, the typically short test durations are not necessarily beneficial for getting accurate ratings – as expressed in various comments, often more time is needed in order to adjust to the controls. Additionally, longer test periods should reduce the presumed novelty effect when filling out the questionnaire.

7.3 Future Work

Several hypotheses suggest that there nevertheless seems to be some kind of relation between persons' exercising habits and their ratings for fitness games. It would be interesting to see if longer test periods could help improving the data collection process in order to discover more significant evidence in this regard. It further would be beneficial to setup several ergometer stations, maybe even letting volunteers compete against each other to increase the people's experienced social factor. Finally, long-term engagement should be investigated since this would more accurately describe peoples' preferences and tendencies towards exergaming at home.

Appendix A: Games and Game Elements

A.1 History of social Games

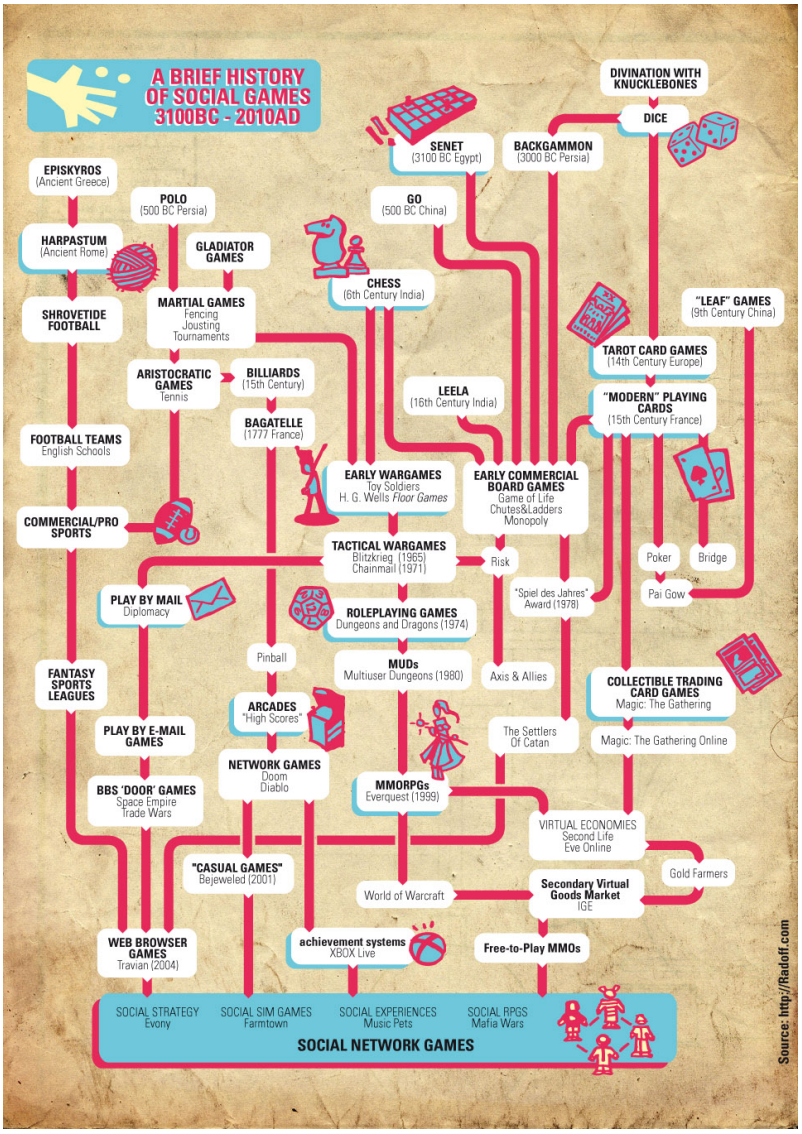


FIGURE A.1: History of social gaming by Jon Radoff (`http://tinyurl.com/29344bt`).

A.2 Steven Reiss' Basic Human Desires

Table A.1: Steven Reiss' Basic Human Desires.
Source: Reiss Profile, 2015, "A new perspective on personality"
(`http://tinyurl.com/jyg6qej`).

Basic Desire	Personality Trait
Power	Power tells us whether an individual craves leadership or responsibility, or would rather work in a service capacity.
Independence	Independence reveals how a person forms their relationships with regard to autonomy or in association with other people.
Curiosity	Curiosity reveals the importance of 'knowledge' in a person's life, and why they want to acquire knowledge.
Acceptance	Acceptance shows who, or what a person uses to construct a positive self-image.
Order	The strength of the basic desire of Order shows how much structure or flexibility a person needs in their life.
Saving	Saving has its evolutionary origin in the storing of supplies. The strength of this basic desire shows how important it is emotionally for a person to have possessions.
Honor	Honor reveals whether an individual seeks to remain true to their principles, or is goal-oriented.
Idealism	Idealism considers the altruistic element of morality, and reveals the importance of responsibility with respect to fairness and social justice.
Social Contact	Social Contact shows the importance of social acquaintances. Here, the quantity of contacts is significant.
Family	Family reveals how strong the desire to care for others is (with regard to a person's own children).
Status	Status is the desire wither to be 'conspicuously different' from others in an elitist sense, or to be ordinary and like everyone else.

Vengeance	Vengeance is chiefly about comparing oneself to others. It includes on the one hand aggression and retaliation, and harmony and conflict-avoidance on the other.
Romance	Romance reveals the importance of sensuality in an individual's life. Besides sex, this desire includes all other aspects of sensuality, e.g. design, art, and beauty.
Eating	Eating seeks to determine the importance of eating in a person's life. How much does the pleasure of eating well contribute to a satisfying life?
Physical Activity	Physical Activity reveals the importance of physical activity (at work or playing sport) for a satisfying life.
Tranquility	Tranquility can also be described as emotional stability. It identifies the importance of stable emotional relations for a satisfying life.

A.3 Jon Radoff's 42 Fun Factors

Table A.2: Jon Radoff's 42 Fun Factors . Source: [39].

	Motivation																
Fun Factor	**Power (a)**	**Independence (b)**	**Curiosity (c)**	**Acceptance (d)**	**Order (e)**	**Saving (f)**	**Honor (g)**	**Idealism (h)**	**Social Contact (i)**	**Family (j)**	**Status (k)**	**Vengeance (l)**	**Romance (m)**	**Eating (n)**	**Physical Activity (o)**	**Tranquility (p)**
Recognizing Patterns			c		e											
Collecting	a				e	f					k					
Finding Random Treasures	a					f					k					
Achieving a Sense of Completion	a	b			e											p
Gaining Recognition for Achievements				d					i		k					

Creating Order out of Chaos				e										p
Customizing Virtual Worlds		b						i		k				
Gathering Knowledge			c					i		k				
Organizing Groups of People				e				i	j	k				
Noting Insider References				d				i						
Being the Centre of Attention	a									k		m		
Experiencing Beauty and Culture				e								m		p
Romance						g			j			m		
Exchanging Gifts				d		g			j			m		
Being a Hero	a	b				g	h	i		k	l	m		
Being a Villain	a	b									l			
Being a Wise Old Man	a	b		d		g			j	k				
Being a Rebel	a	b						i		k	l	m		
Being the Ruler	a			e		g			j	k		m		
Pretending to Live in a Magical Place			c				h					m		p
Listening to a Story			c					i						
Telling Stories	a			d						k				
Predicting the Future				e										
Competition	a									k	l		o	
Psycho-analyzing			c		e									
Mystery			c									m		
Mastering a Skill	a	b											o	
Exacting Justice and Revenge	a										l	m		
Nurturing		b				g			j	k		m		
Excitement													o	
Triumph over Conflict	a										l			

Relaxing															p
Experiencing the Freakish or Bizarre			c												
Being Silly				d											p
Laughing														o	p
Being Scared														o	
Strengthening a Family Relationship						f		h	i						
Improving Ones Health														o	
Imagining a Connection with the Past			c		e										
Exploring a World	b		c												
Improving Society							g	h			k				
Enlightenment	b		c												p

Appendix B: Studies

B.1 Common Elements

Common study elements for Flappy Cycling, its preliminary evaluation and Quake, shared are contained in this section.

B.1.1 Shared Questionnaires

All used LimeSurvey questionnaires contain common sections, i.e. questions categorized with A,B, and D, which will be listed in following Tables B.1, B.2 and B.3.

The questionnaires have been titled 'Flappy Bird Ergometer Edition', 'Flappy Cycling' and 'Quake 3 Arena Fitness Edition'. Some questions trigger other questions, which has been annotated accordingly: for example, Y $\rightarrow$ B7 means if the current question is answered with 'Yes', question B7 is presented to the participant or >1 $\rightarrow$ B9 signals that any selected value greater than '1' enables B9. Mandatory questions are marked with an '*'.

Common

Pre-Questionnaire

Section A: Pre-Questionnaire General information about yourself.
A1: Please select your gender * Please choose only one of the following: ○ Female ○ Male

Section A: Pre-Questionnaire (continued)

A2: Please enter your age *

Your answer must be between 18 and 130.

Only an integer value may be entered in this field.

A3: Please select your nationality

Please choose only one of the following:

○ Afghanistan	○ Albania	○ Algeria	○ American Samoa
○ Andorra	○ Angola	○ Anguilla	○ Antarctica
○ Antigua & Barbuda	○ Argentina	○ Armenia	○ Aruba
○ Australia	○ Austria	○ Azerbaijan	○ Bahamas
○ Bahrain	○ Bangladesh	○ Barbados	○ Belarus
○ Belgium	○ Belize	○ Benin	○ Bermuda
○ Bhutan	○ Bolivia	○ Bosnia & Herzegovina	○ Botswana
○ Bouvet Island	○ Brazil	○ British Indian Ocean Territory	○ Brunei Darussalam
○ Bulgaria	○ Burkina Faso	○ Burundi	○ Cambodia
○ Cameroon	○ Canada	○ Cape Verde	○ Cayman Islands
○ Central African Republic	○ Chad	○ Chile	○ China
○ Christmas Island	○ Cocos Islands	○ Colombia	○ Comoros
○ Congo	○ Cook Islands	○ Costa Rica	○ Cote d'Ivoire
○ Croatia	○ Cuba	○ Cyprus	○ Czech Republic
○ Denmark	○ Djibouti	○ Dominica	○ Dominican Republic
○ Ecuador	○ Egypt	○ El Salvador	○ Equatorial Guinea
○ Eritrea	○ Estonia	○ Ethiopia	○ Falkland Islands
○ Faroe Islands	○ Fiji	○ Finland	○ France

Section A: Pre-Questionnaire (continued)

○ French Guiana	○ French Polynesia	○ Gabon	○ Gambia
○ Georgia	○ Germany	○ Ghana	○ Gibraltar
○ Greece	○ Greenland	○ Grenada	○ Guadeloupe
○ Guam	○ Guatemala	○ Guinea	○ Guinea-Bissau
○ Guyana	○ Haiti	○ Heard Island & McDonald Islands	○ Honduras
○ Hong Kong	○ Hungary	○ Iceland	○ India
○ Indonesia	○ Iran	○ Iraq	○ Ireland
○ Israel	○ Italy	○ Jamaica	○ Japan
○ Jordan	○ Kazakhstan	○ Kenya	○ Kiribati
○ Kuwait	○ Kyrgyzstan	○ Laos	○ Latvia
○ Lebanon	○ Lesotho	○ Liberia	○ Libya
○ Liechtenstein	○ Lithuania	○ Luxembourg	○ Macao
○ Madagascar	○ Malawi	○ Malaysia	○ Maldives
○ Mali	○ Malta	○ Marshall Islands	○ Martinique
○ Mauritania	○ Mauritius	○ Mayotte	○ Mexico
○ Micronesia	○ Moldova	○ Monaco	○ Mongolia
○ Montenegro	○ Montserrat	○ Morocco	○ Mozambique
○ Myanmar	○ Namibia	○ Nauru	○ Nepal
○ Netherlands	○ Netherlands Antilles	○ New Caledonia	○ New Zealand
○ Nicaragua	○ Niger	○ Nigeria	○ Norfolk Island
○ North Korea	○ Norway	○ Oman	○ Pakistan
○ Palau	○ Palestinian Territory	○ Panama	○ Papua New Guinea
○ Paraguay	○ Peru	○ Philippines	○ Pitcairn
○ Poland	○ Portugal	○ Puerto Rico	○ Qatar
○ Romania	○ Russian Federation	○ Rwanda	○ Saint Helena

Section A: Pre-Questionnaire (continued)			
○ Saint Kitts & Nevis	○ Saint Lucia	○ Saint Pierre & Miquelon	○ Saint Vincent & the Grenadines
○ Samoa	○ San Marino	○ Sao Tome & Principe	○ Saudi Arabia
○ Senegal	○ Serbia	○ Seychelles	○ Sierra Leone
○ Singapore	○ Slovakia	○ Slovenia	○ Solomon Islands
○ Somalia	○ South Africa	○ South Georgia	○ South Korea
○ Spain Spain	○ Sri Lanka	○ Sudan	○ Suriname
○ Svalbard & Jan Mayen	○ Swaziland	○ Sweden	○ Switzerland
○ Syrian Arab Republic	○ Taiwan	○ Tajikistan	○ Tanzania
○ Thailand	○ Former Yugoslav Republic of Macedonia	○ Timor-Leste	○ Togo
○ Tokelau	○ Tonga	○ Trinidad & Tobago	○ Tunisia
○ Turkey	○ Turkmenistan	○ Tuvalu	○ Uganda
○ Ukraine	○ United Arab Emirates	○ United Kingdom	○ United States of America
○ United States Minor Outlying Islands	○ Uruguay	○ Uzbekistan	○ Vanuatu
○ Vatican City	○ Venezuela	○ Vietnam	○ Wallis & Futuna
○ Western Sahara	○ Yemen	○ Zambia	○ Zimbabwe
A4: Please select your occupational field			
Please choose only one of the following:			
○ Architecture & Engineering		○ Arts, Design, Entertainment, Sports & Media	
○ Business & Financial Operations		○ Building, Grounds Cleaning & Maintenance	
○ Construction & Extraction		○ Computer & Mathematical	

Section A: Pre-Questionnaire (continued)	
○ Community & Social Service	○ Education, Training & Library
○ Farming, Fishing & Forestry	○ Food Preparation & Serving Related
○ Healthcare Practitioners & Technical	○ Healthcare Support
○ Installation, Maintenance & Repair	○ Legal
○ Life, Physical & Social Science	○ Management
○ Office & Administrative Support & Repair	○ Personal Care & Service
○ Production	○ Protective Service
○ Sales & Related	○ Student
○ Transportation, Material Moving & Repair	○ Other:

TABLE B.1: Common Pre-Questionnaire.

Common

Main Evaluation: General

Section B: Main Evaluation: General

This is the general part of the main evaluation.

B1: Do you play video games? * (Y → B2,B3,B4)

Please choose only one of the following:

○ Yes ○ No

B2: How often do you play video games? *

Please choose only one of the following:

○ 1 rarely ○ 2 sometimes ○ 3 regularly ○ 4 frequently ○ 5 very frequently

B3: Which types of games do you play? *

Please choose all that apply:

☐ Action ☐ Adventure ☐ FPSs[1] ☐ RPGs[2] ☐ Simulation

[1]First Person Shooters
[2]Role-Playing Games

Section B: Main Evaluation: General (continued)

☐ Strategy ☐ Sports ☐ Casual ☐ MMOs[3] / MOBAs[4] ☐ Other: ☐

B4: Do you play computer games for fitness? * (Y → B5)

Please choose only one of the following:

○ Yes ○ No

B5: Do you find those games satisfying? * (Y → B7, N → B6)

Please choose only one of the following:

○ Yes ○ No

B6: Why not?

Please write your answer here:

B7: Why?

Please write your answer here:

B8: How often do you practice sports? * (>1 → B9)

Please choose only one of the following:

○ 1 never ○ 2 rarely ○ 3 sometimes ○ 4 regularly ○ 5 frequently

B9: Which sportive activities do you like?

Please choose all that apply:

☐ Ball Sports[5] ☐ Bicycling / Bike Riding ☐ Dancing / Dance Fitness / Aerobics[6] ☐ Fitness Center / Weight Training ☐ Gymnastics[7]

[3]Massively Multiplayer Online Games
[4]Multiplayer Online Battle Arenas
[5]e.g. Soccer, Table Tennis, Snooker
[6]e.g. Ballet, Modern, Zumba
[7]e.g. Stretching, Yoga

Section B: Main Evaluation: General (continued)				
☐ Hiking / Mountaineering	☐ Martial Arts / Self Defense	☐ Rock Climbing	☐ Roller Skating / Skateboarding / Ice Skating	☐ Running / Jogging / Walking
☐ Swimming / Water Sports	☐ Track & Field	☐ Other:		
B10: Do you like the idea of combining video gaming and exercising? * (Y → B11, N → B12)				
Please choose only one of the following:				
○ Yes	○ No			
B11: Why?				
Please write your answer here:				
B12: Why not?				
Please write your answer here:				

TABLE B.2: Common General Questionnaire.

Common

Post-Questionnaire

Section D: Post-Questionnaire
Please take the time to answer some post-evaluation questions.
D1: Have you ever participated in an experiment similar to this one? *
Please choose only one of the following:
○ Yes ○ No

Section D: Post-Questionnaire (continued)
D2: Any other comments about what you liked or did not like, or things that should be changed during the course of this experiment?
Please write your answer here:

TABLE B.3: Common Post-Questionnaire.

B.2 Individual Elements

All study specific elements are listed in this section.

B.2.1 Individual Questionnaires

The section C questionnaire differs for Preliminary Flappy Cycling (see Table B.4) and both main studies (see Table B.5).

Preliminary Flappy Cycling

Main Evaluation: Game

Section C: Main Evaluation: Game
This is the game part of the main evaluation.
C1: How comfortable did you feel handling the ergometer? *
Please choose only one of the following:
1 not comfortable .. 5 very comfortable
C2: Did exercising affect your mood? * (Y → C3)
Please choose only one of the following:
○ Yes ○ No
C3: How? *
Please choose only one of the following:
1 very badly .. 5 very pleasantly
○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Section C: Main Evaluation: Game (continued)

C4: Did the game's setting affect your mood? * (Y → C5)

Please choose only one of the following:

○ Yes ○ No

C5: How? *

Please choose only one of the following:

1 very badly .. 5 very pleasantly

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

C6: How did you perceive the game's difficulty? *

Please choose only one of the following:

1 too easy .. 5 too difficult

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

C7: Why?

Please write your answer here:

C8: Would you play & exercise with games & devices like this at home? * (Y → C9,C10)

Please choose only one of the following:

○ Yes ○ No

C9: How often? *

Please choose only one of the following:

1 occasionally .. 5 frequently

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

C10: Would you thus be inclined to work out more often? * (N → C11)

Please choose only one of the following:

○ Yes ○ No

Section C: Main Evaluation: Game (continued)				
C11: How would you rate your overall experience during the experiment? *				
Please choose only one of the following:				
1 poor .. 5 excellent				
○ 1	○ 2	○ 3	○ 4	○ 5

Table B.4: Preliminary Flappy Cycling Main Questionnaire.

Flappy Cycling and Quake 3 Arena Mod

Main Evaluation: Game

Section C: Main Evaluation: Game				
This is the game part of the main evaluation.				
C1: How was your experience regarding handling the ergometer? *				
Please choose only one of the following:				
○ 1 bad	○ 2 poor	○ 3 average	○ 4 good	○ 5 excellent
C2: Any comments regarding your choice?				
Please write your answer here:				
C3: How was your experience regarding playing the game? *				
Please choose only one of the following:				
○ 1 bad	○ 2 poor	○ 3 average	○ 4 good	○ 5 excellent
C4: Any comments regarding your choice?				
Please write your answer here:				
C5: How would you rate the game? *				
Please choose only one of the following:				
○ 1 bad	○ 2 poor	○ 3 average	○ 4 good	○ 5 excellent

Section C: Main Evaluation: Game (continued)

C6: Why?

Please write your answer here:

C7: How did you perceive the game's difficulty? *

Please choose only one of the following:

○ 1 too easy ○ 2 easy ○ 3 balanced ○ 4 difficult ○ 5 too difficult

C8: Why?

Please write your answer here:

C9: If possible, would you consider playing (FC) such games / (Q3A) your favorite games with similar devices at home? * (Y → C10,C11)

Please choose only one of the following:

○ Yes ○ No

C10: How often? *

Please choose only one of the following:

○ 1 rarely ○ 2 sometimes ○ 3 regularly ○ 4 frequently ○ 5 very frequently

C11: Would you thus be inclined to work out more often? * (N → C12)

Please choose only one of the following:

○ Yes ○ No

C12: Why not?

Please write your answer here:

Section C: Main Evaluation: Game (continued)				
C13: How would you rate your overall experience during the experiment? *				
Please choose only one of the following:				
○ 1 bad	○ 2 poor	○ 3 average	○ 4 good	○ 5 excellent

TABLE B.5: Flappy Cycling and Q3A Main Questionnaire.

B.2.2 Quake 3 Mod Configuration

```
Program Usage: runmod -age PLAYERAGE [-hrtarget WORKOUTGOAL] [-disable
↪ DOPTIONS] [-map MAPNAME] [-bots NUMBOTS] [-diff DIFFICULTY] [-frag
↪ FRAGLIMIT] [-time TIMELIMIT] [-joy JOYSTICKMODE]
[PLAYERAGE] .......... Player's age (required)
[WORKOUTGOAL] ........ HR Target (Default: aerobic), Available:
                       recovery, aerobic, anaerobic
[DOPTIONS] ........... Disable options (Default: all on), Available:
                       hrmode, items, powerups, weapons, worlddrops
[MAPNAME] ............ Quake Map (Default: q3dm1), Available:
                       q3dm[0-19], q3tourney[1-6], q3ctf[1-4]
[NUMBOTS] ............ Number of Bots  (Default: 3), Bot Pool:
                       crash   ranger    phobos mynx    orbb
                       sarge   bitterman grunt  hossman daemia
                       hunter  klesk     angel  wrack   gorre
                       slash   anarki    biker  lucy    patriot
                       tankjr
[DIFFICULTY] ......... Bot difficulty (1-5, easy-hard), Default: 2
[FRAGLIMIT] .......... Frag Limit (Default: 10)
[TIMELIMIT] .......... Time Limit (Default: none)
[JOYSTICKMODE] ....... Joystick Config (Default: regular), Available:
                       regular, inverted
-------------------------------------------------------------------------
Ingame debug options (enter in '~' console, prepend with a '\'):
debugmod (NUM *) ........... debugging on/off
inchr (NUM +) .............. increase max HR by 5 (max: 300)
dechr (NUM -) .............. decrease max HR by 5 (min: 20)
incnumhrsamples (NUM up) ... increase # HB samples f. calc. (max: 60)
```

```
decnumhrsamples (NUM down) . decrease # HB samples f. calc. (min: 1)
ergostatus ................. settings information
```

Code Listing B.1: Q3A mod: configuration settings, printout from `runmod.bat` (`http://tinyurl.com/gqxgkcr`).

B.3 Statistics and Responses

This section lists statistics about study participation and the data collected from all completed surveys. The data has originally been exported using LimeSurvey's statistics generation tools and was arranged individually in order to improve its significance. Since often free text answers were given in German, they have been translated to English for uniformity.

B.3.1 Preliminary Flappy Cycling

General Statistics
Total number of records: 13

Table B.6: Pre-FC: General.

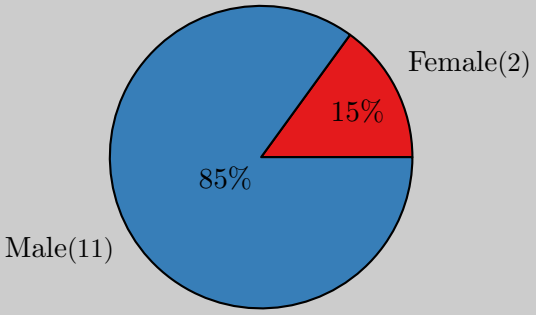
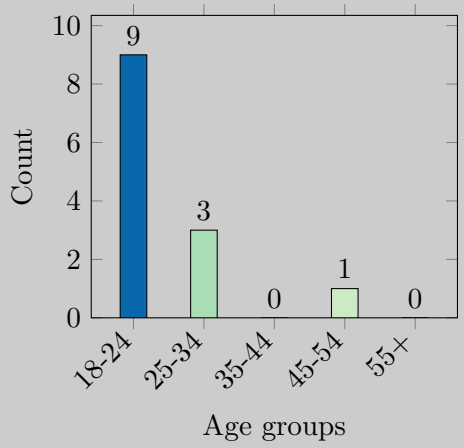
A1: Please select your gender *


Table B.7: Pre-FC: A1.

A2: Please enter your age *



Statistics

Calculation	**Result**
Count	13
Sum	333
Standard deviation	6.95
Average	25.62
Minimum	19
1st quartile (Q1)	22
2nd quartile (Q2)	23
3rd quartile (Q3)	27
Maximum	47

Table B.8: Pre-FC: A2.

A3: Please select your nationality[8]

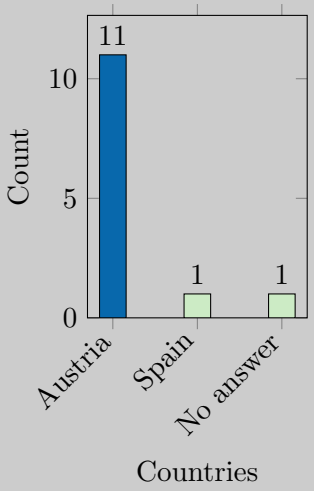


Table B.9: Pre-FC: A3.

A4: Please select your occupational field[9]

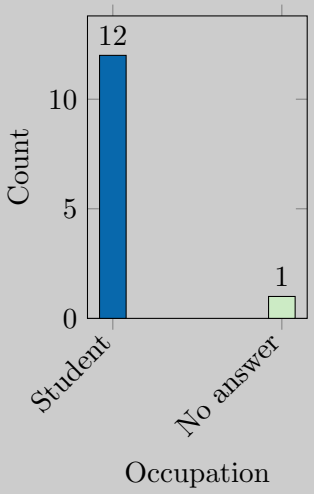
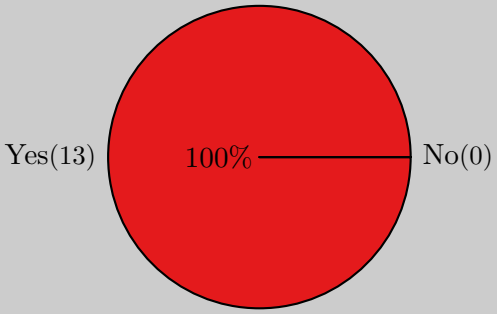


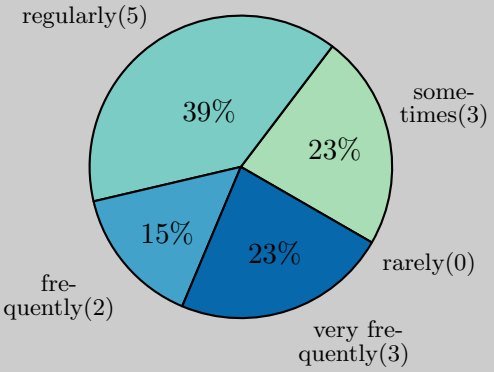
Table B.10: Pre-FC: A4.

[8]All not selected values omitted for abbreviation and increasing visibility.
[9]Labels abbreviated and all not selected values omitted (refer to Table B.1).

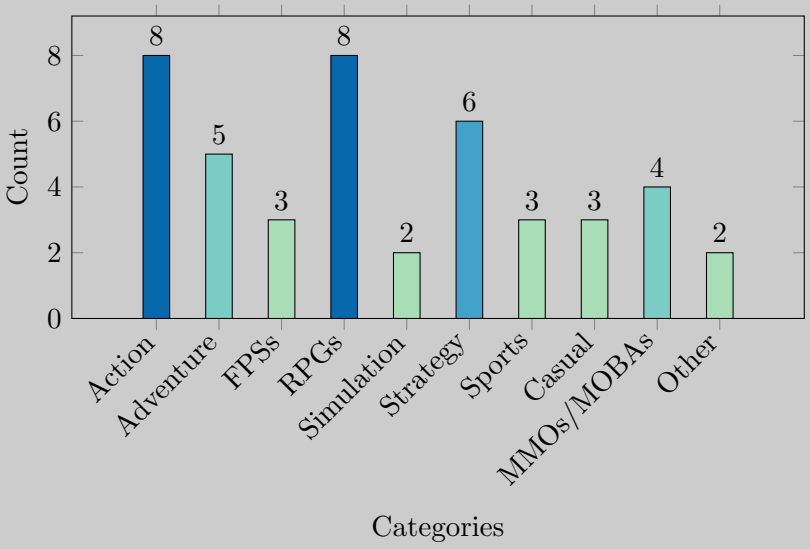
B1: Do you play video games? *



B2: How often do you play video games? * (B1: Yes)



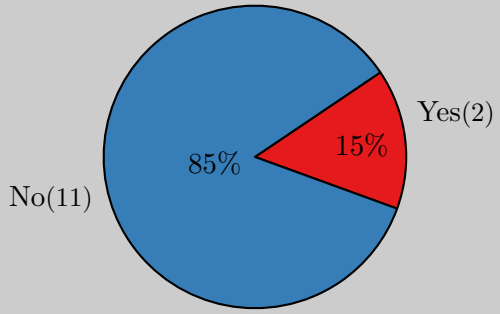
B3: Which types of games do you play? * (B1: Yes)



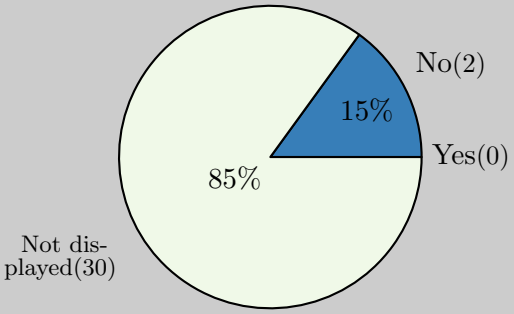
Other responses

Jump N Run
Minecraft

B4: Do you play computer games for fitness? * (B1: Yes)



B5: Do you find those games satisfying? * (B4: Yes)

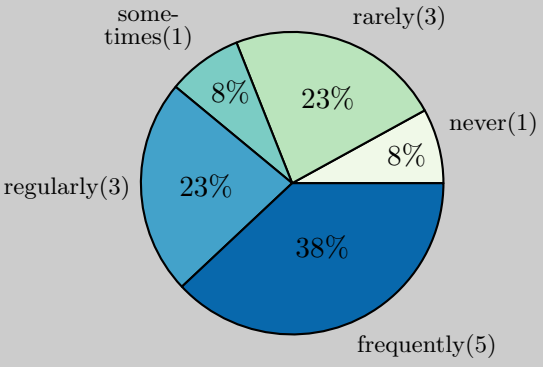


B6: Why not? * (B5: No)

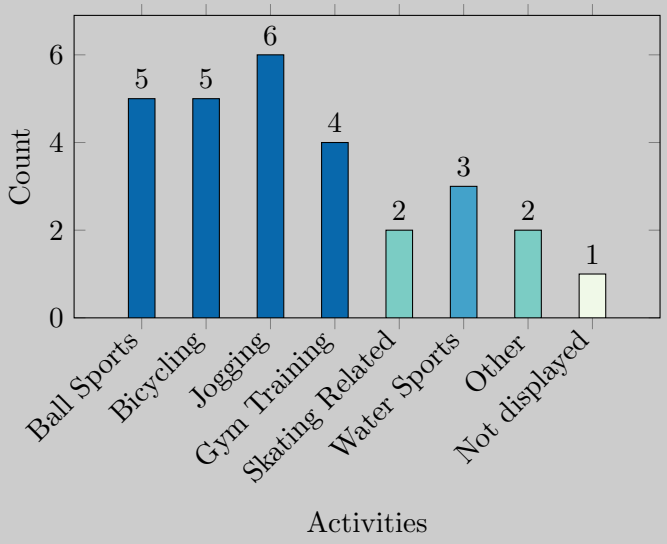
Often aren't challenging enough.
Can easily be outsmarted.

TABLE B.11: Pre-FC: B1–B7.

B8: How often do you practice sports? *



B9: Which sportive activities do you like?[10] (B8 > 1 never)

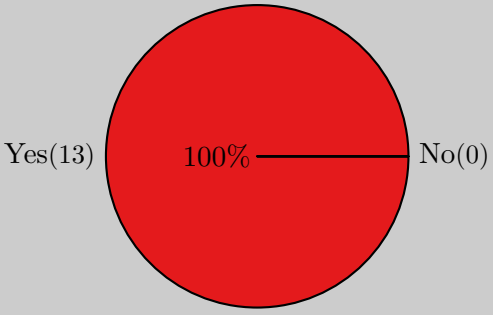


Other responses

Brazilian Jiu Jitsu
Brazilian Jiu Jitsu, Firefighters' contest

TABLE B.12: Pre-FC: B8–B9.

B10: Do you like the idea of combining video gaming and exercising? *



B11: Why? (B10: Yes)

Combines fun and exercise.
In general good but nature is better.
Many possibilities: Google Ingress is an ingenious idea.
Possibilities of using virtual reality: Google Ingress.
Staying fit is more fun this way.

[10] Labels abbreviated (refer to Table B.2).

Because it urges lazy people to exercise.
Best idea to spice up exercise, I do it myself once in a while.
Games make sports more enjoyable.
Is fun if implemented properly.
Fun if not too exhausting.
Motivation.
Exercise is boring – needs some fun for motivation.

Table B.13: Pre-FC: B10–B12.

C1: How comfortable did you feel handling the ergometer? *

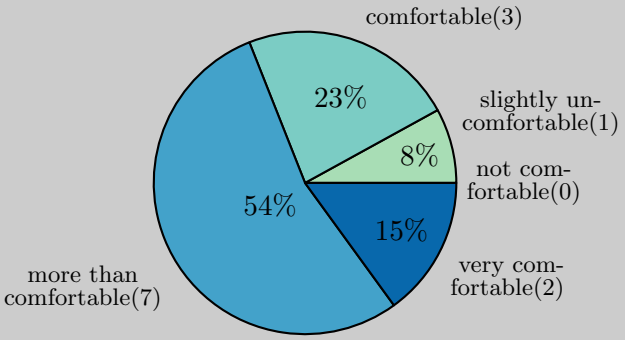
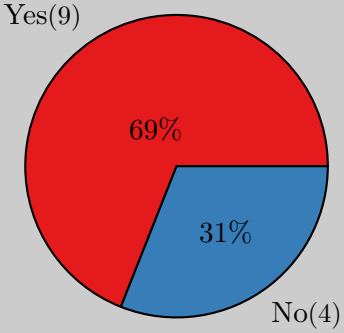


Table B.14: Pre-FC: C1.

C2: Did exercising affect your mood? *



C3: How? (C2: Yes)

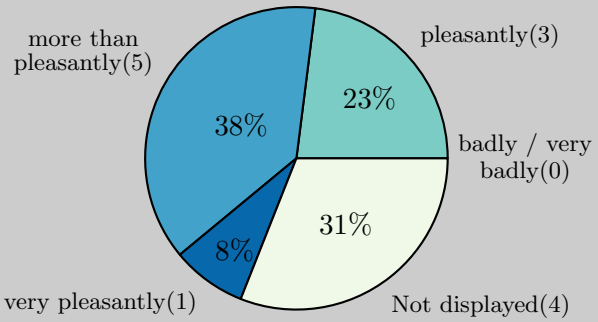
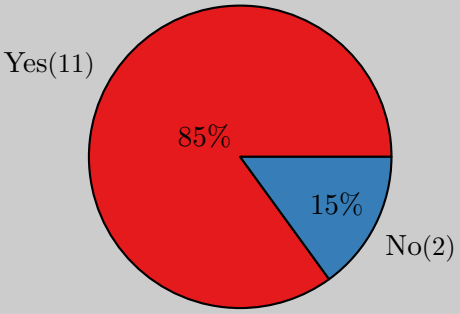


Table B.15: Pre-FC: C2–C3.

C4: Did the game's setting affect your mood? *



C5: How? (C4: Yes)

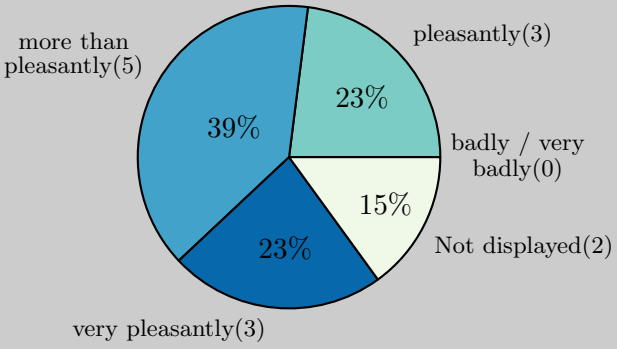


Table B.16: Pre-FC: C4–C5.

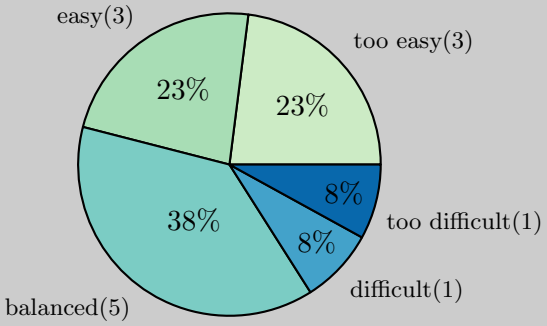
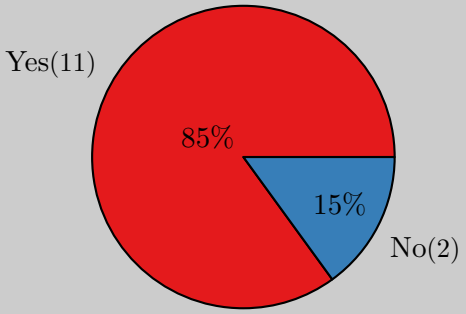
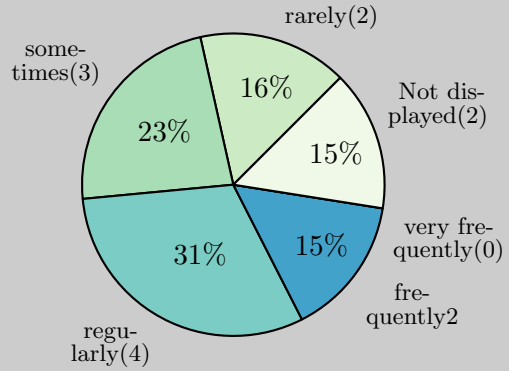
C6: How did you perceive the game's difficulty? *

C7: Why?
[C6: 1 too easy] Not enough game elements.
[C6: 2 easy] People often riding bikes have it easy.
[C6: 1 too easy] Difficulty should be adjustable.
[C6: 4 difficult] Biking is hard / Game is easy.
[C6: 5 too difficult] Biking is tough.
[C6: 2 easy] No challenge – at least for short periods too easy.
[C6: 1 too easy] Very constant experience – needs change.

TABLE B.17: Pre-FC: C6–C7.

C8: If possible, would you consider playing your favorite games with similar devices at home? *


C9: How often? * (C9: Yes)



C10: Would you thus be inclined to work out more often? * (C8: Yes)

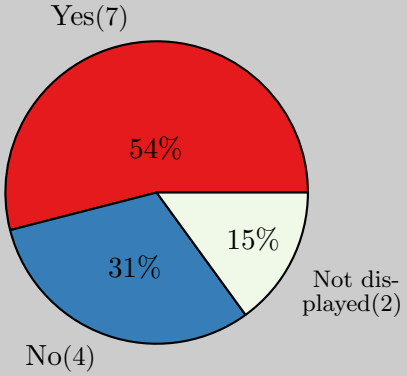


Table B.18: Pre-FC: C8–C10.

C11: How would you rate your overall experience during the experiment? *

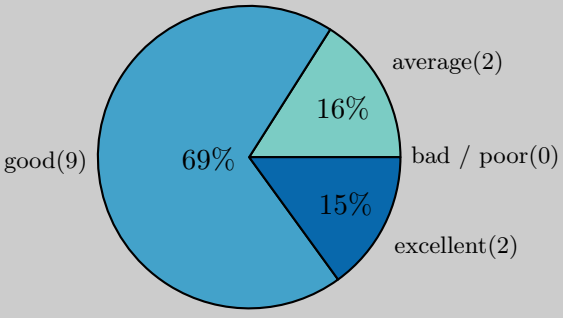


Table B.19: Pre-FC: C11.

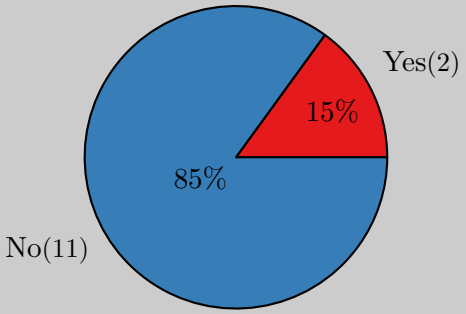
D1: Have you ever participated in an experiment similar to this one? *


TABLE B.20: Pre-FC: D1.

D2: Any other comments about what you liked or did not like, or things that should be changed during the course of this experiment?
Is fun, but lacks game goal.
No game challenge.
I would like to know the goal of this game.
What is the goal of the study? Flappy Cycling is a good idea.
Monotonous: no game goal, concentration on left sector of screen, where bird is.
Landscape is nice, but lacks realism. Would buy if refined.

TABLE B.21: Pre-FC: D2.

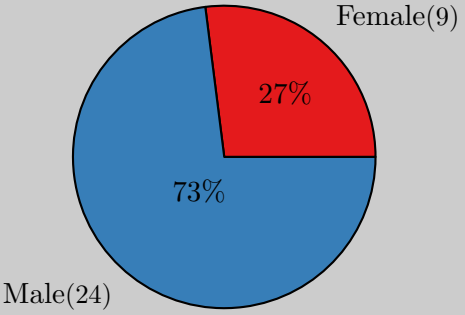
B.3.2 Main Flappy Cycling VS Quake 3 Arena Mod

General Statistics	
Flappy Cycling	**Quake 3 Arena Mod**
Total number of records: 33	Total number of records: 34

TABLE B.22: FC vs. Q3A: General.

A1: Please select your gender *

Flappy Cycling



Quake 3 Arena Mod

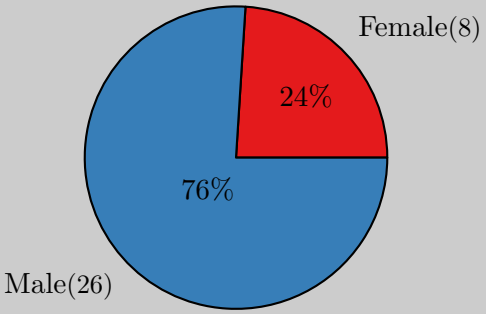
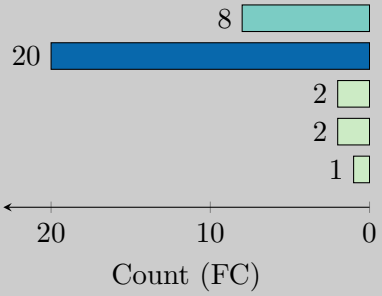


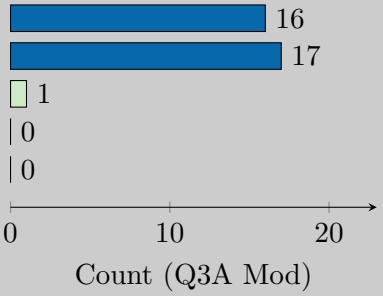
TABLE B.23: FC vs. Q3A: A1.

A2: Please enter your age *

Flappy Cycling



Quake 3 Arena Mod



Statistics

Calculation	Result
Count	33
Sum	960
Standard deviation	9.07
Average	29.09
Minimum	20
1st quartile (Q1)	24.5
2nd quartile (Q2)	26
3rd quartile (Q3)	31
Maximum	62.00

Statistics

Calculation	Result
Count	34
Sum	863
Standard deviation	4.02
Average	25.38
Minimum	20
1st quartile (Q1)	22
2nd quartile (Q2)	25
3rd quartile (Q3)	27
Maximum	36

Table B.24: FC vs. Q3A: A2.

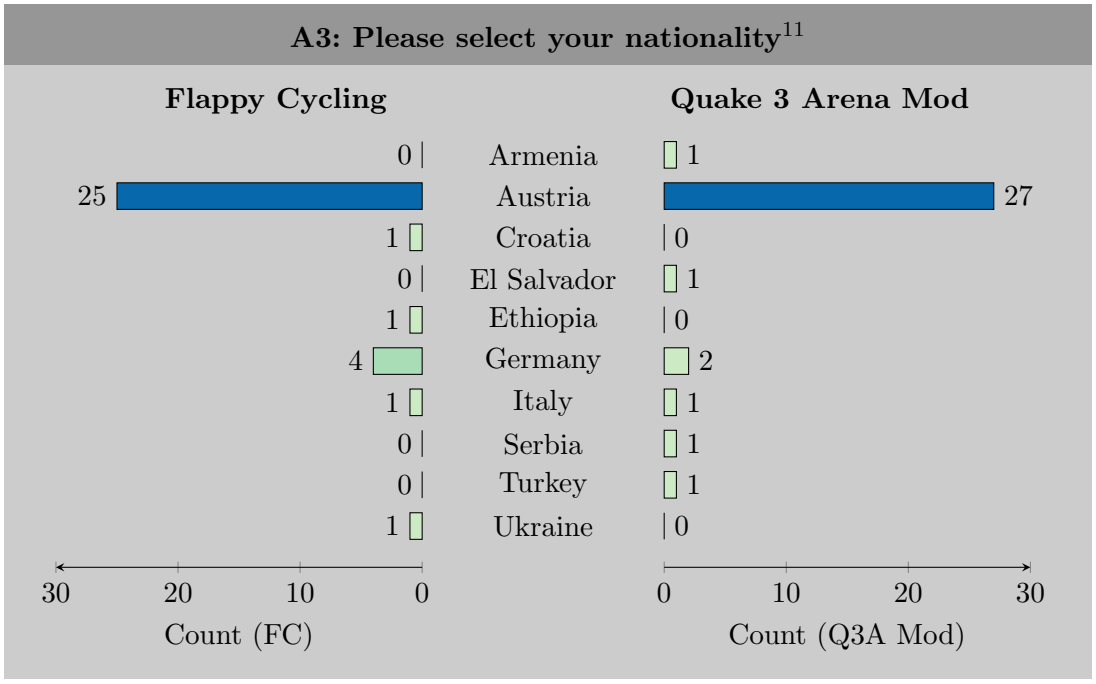


Table B.25: FC vs. Q3A: A3.

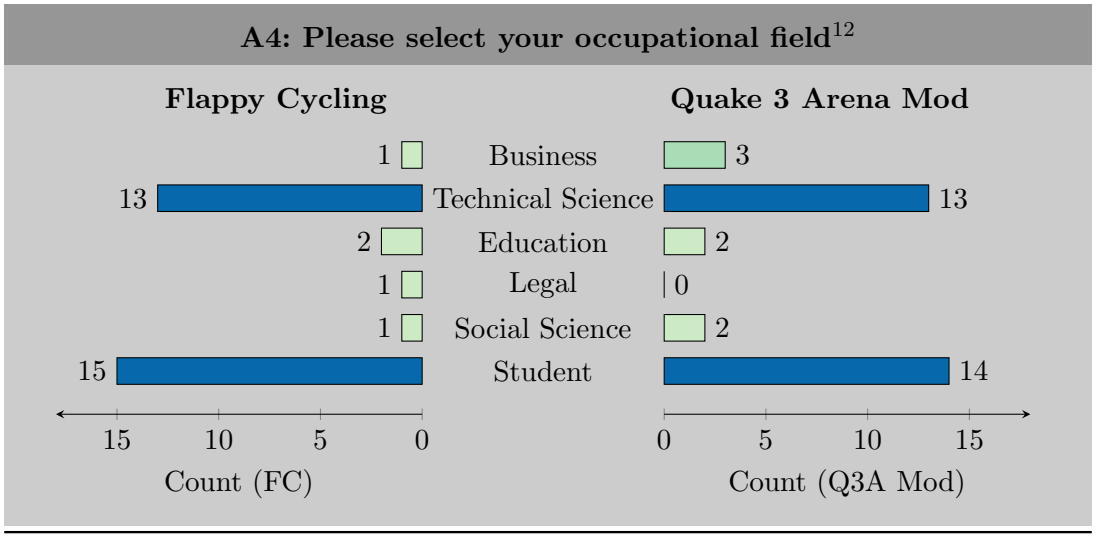
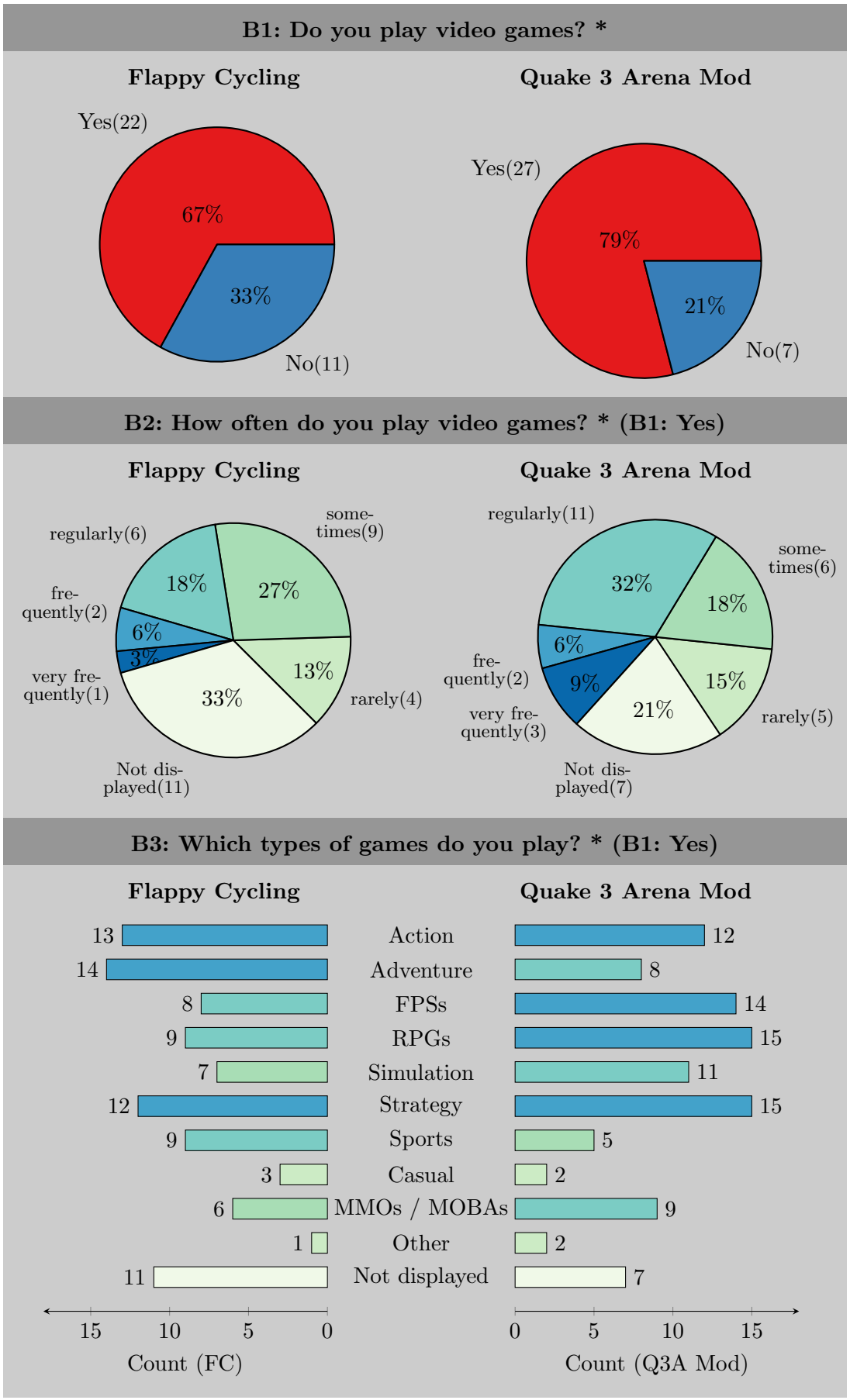


Table B.26: FC vs. Q3A: A4.

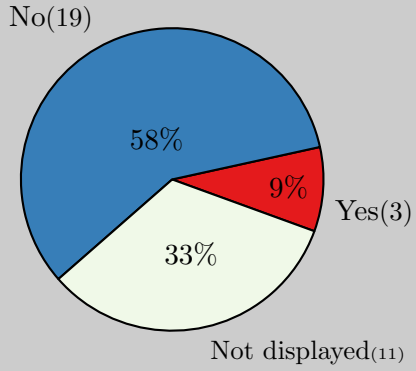
[11] Several not selected values omitted for abbreviation and increasing visibility.
[12] Labels abbreviated and several not selected values omitted (refer to Table B.1).


B1: Do you play video games? *
Flappy Cycling
Yes(22)
67%
33%
No(11)
Quake 3 Arena Mod
Yes(27)
79%
21%
No(7)
B2: How often do you play video games? * (B1: Yes)
Flappy Cycling
regularly(6)
some-
times(9)
fre-
quently(2)
18%
27%
6%
3%
very fre-
quently(1)
13%
rarely(4)
33%
Not dis-
played(11)
Quake 3 Arena Mod
regularly(11)
some-
times(6)
32%
18%
6%
fre-
quently(2)
9%
15%
very fre-
quently(3)
21%
rarely(5)
Not dis-
played(7)
B3: Which types of games do you play? * (B1: Yes)
Flappy Cycling
Quake 3 Arena Mod
13 Action 12
14 Adventure 8
8 FPSs 14
9 RPGs 15
7 Simulation 11
12 Strategy 15
9 Sports 5
3 Casual 2
6 MMOs / MOBAs 9
1 Other 2
11 Not displayed 7
15 10 5 0
Count (FC)
0 5 10 15
Count (Q3A Mod)

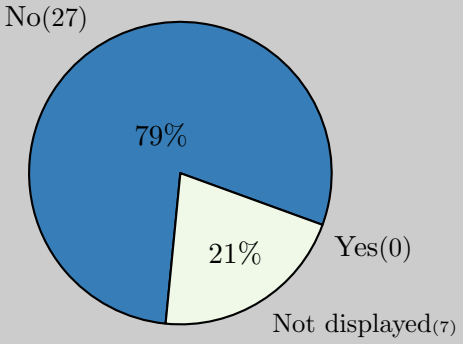
Other responses	Other responses
Jump & Run, Racing	emulator
	Manager

B4: Do you play computer games for fitness? * (B1: Yes)

Flappy Cycling

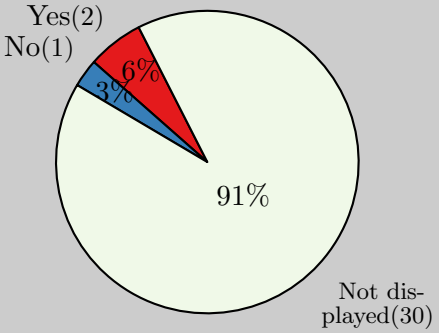


Quake 3 Arena Mod

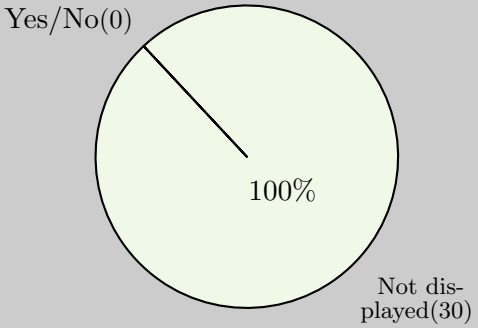


B5: Do you find those games satisfying? * (B4: Yes)

Flappy Cycling



Quake 3 Arena Mod



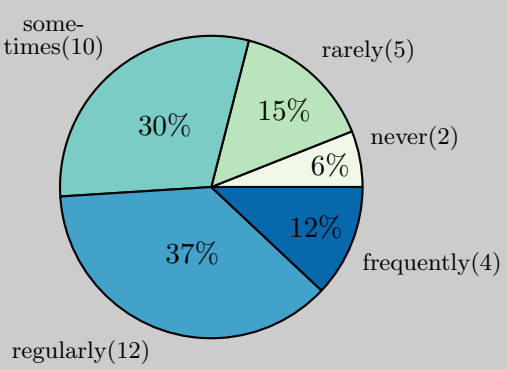
B7: Why? * (B5: Yes)

kinect is very precise, engaging experience → direct comparison with others, makes you dig deeper when exercising.
funny.

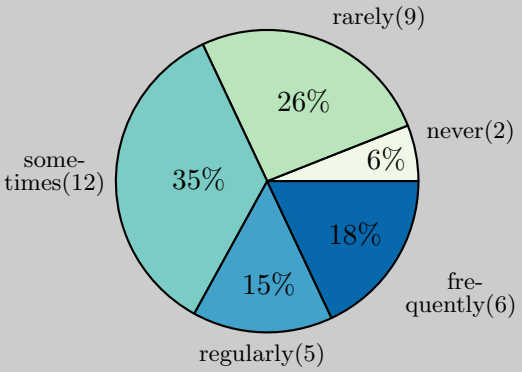
Table B.27: FC vs. Q3A: B1–B7.

B8: How often do you practice sports? *

Flappy Cycling

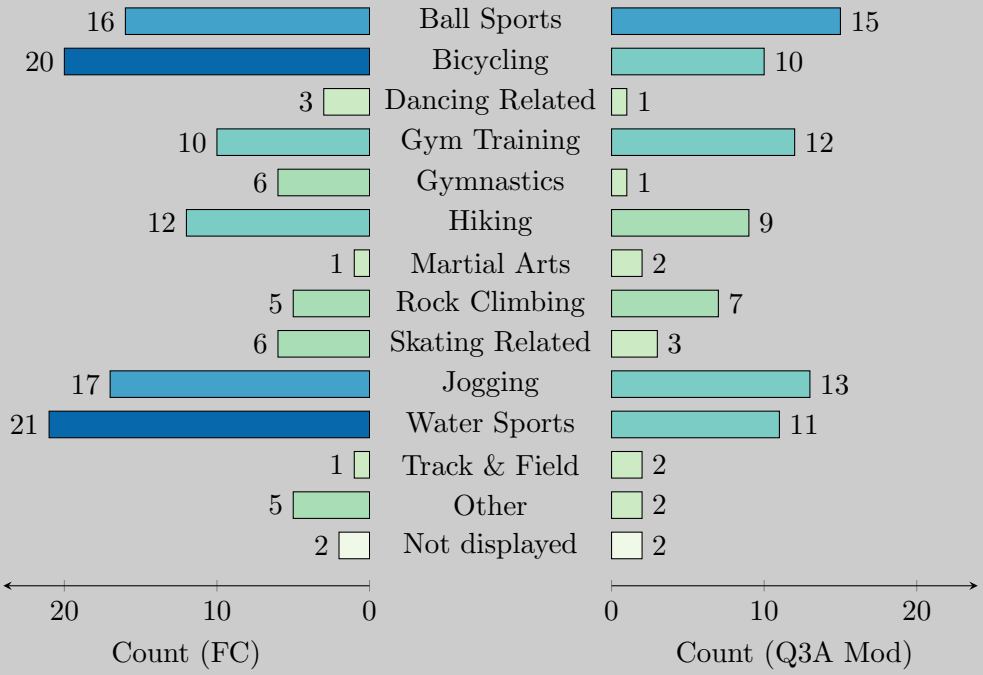


Quake 3 Arena Mod



B9: Which sportive activities do you like?[13] (B8 > 1 never)

Flappy Cycling

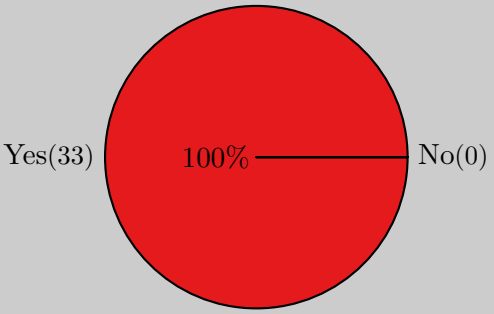
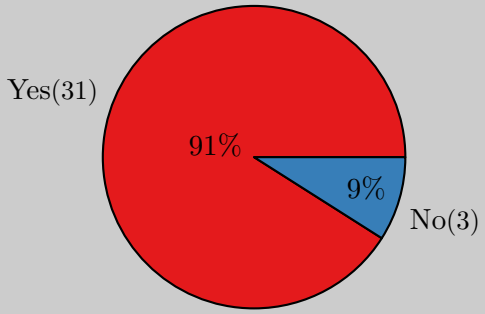


Quake 3 Arena Mod

Other responses	Other responses
Tennis	Kickboxing
think	Rugby
skiing	
cross-country skiing	
Kayak	

Table B.28: FC vs. Q3A: B8–B9.

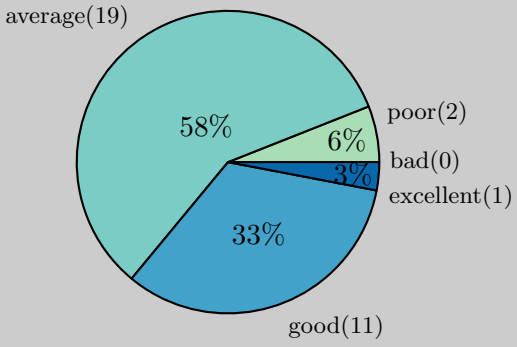
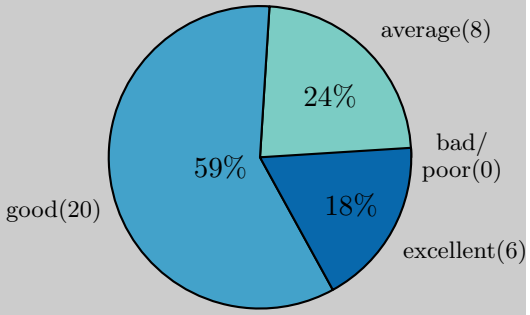
[13]Labels abbreviated (refer to Table B.2).

B10: Do you like the idea of combining video gaming and exercising? *	
Flappy Cycling	**Quake 3 Arena Mod**
	
B11: Why? (B10: Yes)	**B11: Why? (B10: Yes)**
more satisfying to also do something for your fitness while playing games.	awesome combination!
Fitness, better health, ...	Sports in combination with video games is the only sport that I would be interested in (probably healthy). If the implementation is done well, player immersion can give the feeling of being inside a scene.
because it sounds interesting.	combination of two things that are fun.
i like geocaching.	i like both.
Exercising becomes entertaining.	I play often but rarely do sports. Wii sports games to me are social games that I cannot and do not want to play alone.
Computer games get more meaning. More possibilites for gaming experiences.	Because it makes it easier to find motivation for doing sports. New gaming experience.
its funny.	Because computer games create fun, excitement and, as a side effect, it is possible to do something healthy for the body. Since one does usually not move when playing conventional games, this is a great opportunity to change.
more fun.	It's fun.

because playing video games isn't something you would call a healthy hobby, if you combine it with sports it's better. (Of course it's better to go outside to do sports, but it's better than nothing).	One has the opportunity to do sports and stay healthy.
it's a kind of motivation.	Good combination between virtual games and sports.
more competition with others, it's fun!	i always dreamed of that... cycling in the living room and having something to do meanwhile. or some kind of running on the treadmill and having some VR experience.
You lose weight while having fun!	keeps you healthy while having an additional motivation to do sports.
Mind occupied while stress on body.	Fun idea. I can imagine though that something like this is hard to market.
Thinking and doing sports is very effective to improve skills.	gain fitness out of my hobby :-).
The combination is fun.	motivating.
higher motivation if you're exercising on your own (there is competition even without opponents).	It's a perfect opportunity to interest lazy people in sports – many gamers are lazy and here they get to take action against that. Great idea.
Because it motivates to exercise along with an additional challenge.	It is fun and while playing one doesn't necessarily notice doing physical exercise.
fun. motivation. entertaining. time will pass faster.	Because it is possible to do so.
It is entertaining.	Something new.
This is a better motivation to start exercising.	It's always great to exercise, leads to a healthier life.
It would help people to work out more, though it is a danger that people become couch potatoes.	Because they combine the fun of playing computer games with physical activity.
So i don't get so fat while playing computer.	It is time saving doing both things in parallel.

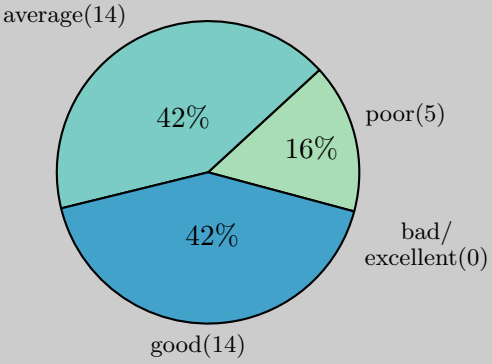
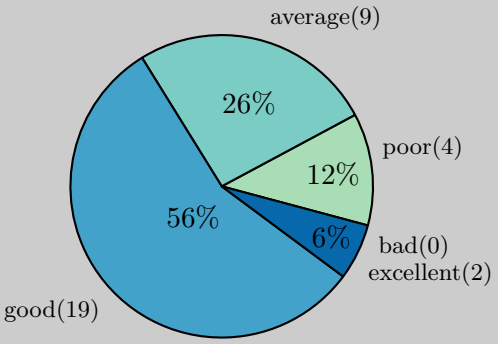
	more motivation to do physical exercise.
	It can make exercising more fun. It can help to improve coordination.
	B12: Why not? (B10: No)
	I hate Computer games.
	I associate gaming with relaxing and sports with exertion – for me they can hardly be combined.
	Videogames are rather for relaxation.

TABLE B.29: FC vs. Q3A: B10–B12.

C1: How was your experience regarding handling the ergometer? *	
Flappy Cycling	**Quake 3 Arena Mod**
	
C2: Any comments regarding your choice?	**C2: Any comments regarding your choice?**
[C1: 3 average] steering did not work for me.	[C1: 4 good] after some time it can get quite demanding - which is a very positive thing in my opinion.
[C1: 4 good] You just had to cycle.	[C1: 4 good] Playing on a crosstrainer would be very shakey, which can influence the gaming experience negatively.
[C1: 3 average] Changing the height was rather difficult.	[5 excellent] The ergometer level was set to high, other than that it was fine. In complicated situations with many enemies I unconsciously slowed down because I had to concentrate.

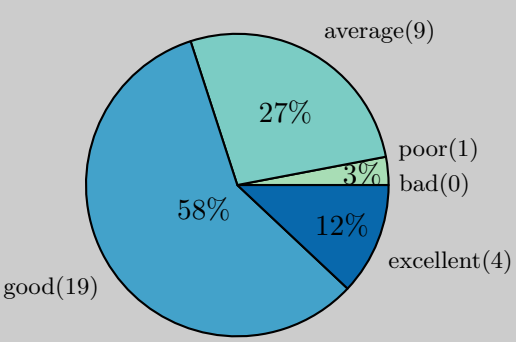
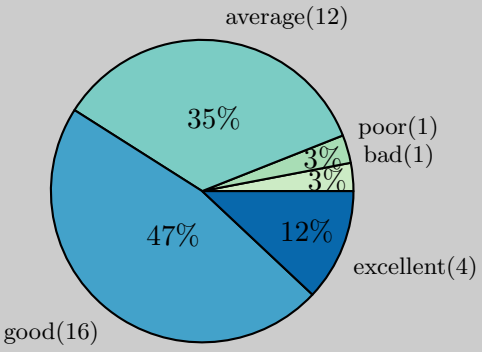
[C1: 5 excellent] good.	[C1: 4 good] Ergometer input is really good, it feels great how the character runs and that this can be influenced.
[C1: 3 average] it doesn't work at the beginning.	[C1: 4 good] Faster paddling positively influences gameplay only up to a certain point – after that precision is lost.
[C1: 3 average] Back button sometimes flies you forward. Paddling faster has delay for flying upwards (same as for downwards).	[C1: 4 good] Takes some getting used to, but functions well.
[C1: 4 good] was quite intuitive, good (fast enough) reaction.	[C1: 4 good] h.
[C1: 3 average] difficulties handling the height of the bird.	

Table B.30: FC vs. Q3A: C1–C2.

C3: How was your experience regarding playing the game? *	
Flappy Cycling	**Quake 3 Arena Mod**
	
C2: Any comments regarding your choice?	**C2: Any comments regarding your choice?**
[C3: 4 good] pulse sensors weren't always detected at the beginning because of the screws on the back of the handles.	[C3: 4 good] My physical efforts payed off – at least virtually (Character moved faster plus I stayed in an appropriate pulse area, which yielded more power ups).

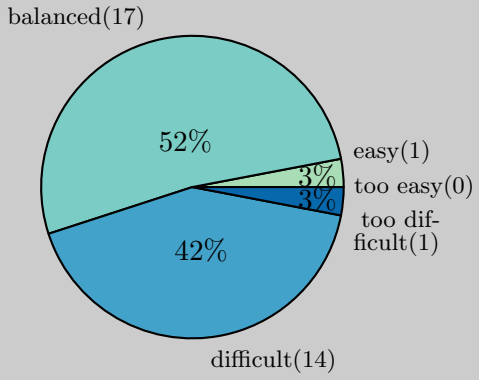
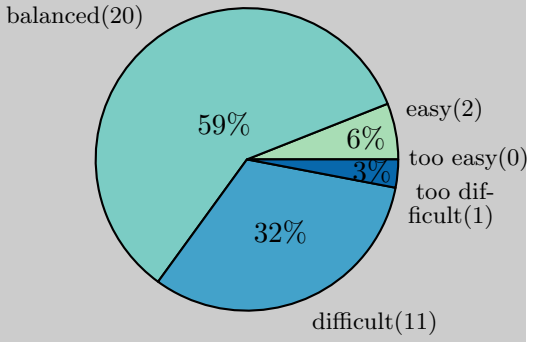
[C3: 3 average] could not dodge so many birds.	[3 average] would need more time to get used to the controls, especially in combination with the ergometer.
[C3: 2 poor] Got the feeling of not being in control of the bird.	[C3: 5 excellent] It was fun!
[C3: 4 good] interesting. http://kal.at	[C3: 3 average] funny.
[C3: 3 average] huge hitbox; front-back controls a little wonky.	[C3: 2 poor] I'm really bad at playing video games.
[C3: 4 good] good.	[C3: 2 poor] i am a bad ego shooter.
[C3: 3 average] I don't like games where input latency affects your gameplay.	[C3: 4 good] Mouse would help.
[C3: 2 poor] I find it difficult to dodge the birds.	[C3: 3 average] I'm not good at FPS, had problems with aiming.
[C3: 3 average] birds very often come on the same height as the player. You'd need to avoid them by going up and down which is quite difficult.	[C3: 2 poor] I am not an expert at videogames.
[C3: 3 average] It is difficult to increase the birds height fast.	
[C3: 4 good] it was fun but a little frustrating that I could not navigate in a perfect way.	
[C3: 2 poor] to much crashes, difficulties handling the height of the bird.	
[C3: 4 good] It was fun.	
[C3: 4 good] nice graphics, input could be improved.	

Table B.31: FC vs. Q3A: C3–C4.

C5: How would you rate the game? *	
Flappy Cycling	**Quake 3 Arena Mod**
	
C6: Why?	**C6: Why?**
[C5: 3 average] the steering controls could be improved.	[C5: 4 good] Quake 3 is a classic game :-D. The integration was good, the rpm and bpm calculations seemed to have recognizable influence on the gameplay .-).
[C5: 4 good] nice idea.	[C5: 3 average] again, i would need more time to make a more detailed and informed statement about the quality of the game, but the idea of combining quake with the ergometer is very engaging.
[C5: 3 average] training.	[C5: 3 average] I hate shooting games.
[C5: 3 average] graphics could be a bit better; Game seems a bit random (birds).	[C5: 2 poor] I don't like shooting games.
[C5: 3 average] it is a fun game, but i think it will get boring after a while.	[C5: 5 excellent] i am a bad gamer but i always like q3a.
[C5: 4 good] nice idea.	[C5: 3 average] I usually don't like shooters, but combined with exercise it is fun.
[C5: 4 good] Nice graphics, concept.	
[C5: 4 good] No leaderboard at the end of the game. no option to enter the name of the player.	
[C5: 2 poor] i find it very difficult to dodge the birds.	

[C5: 3 average] birds very often come on the same height as the player. You'd need to avoid them by going up and down which is quite difficult.	
[C5: 4 good] I had fun and was entertained. It kept my attention.	
[C5: 4 good] a new game experience for me. nice idea to combine workout with playing computer games.	
[C5: 4 good] Time goes by pretty fast.	

TABLE B.32: FC vs. Q3A: C5–C6.

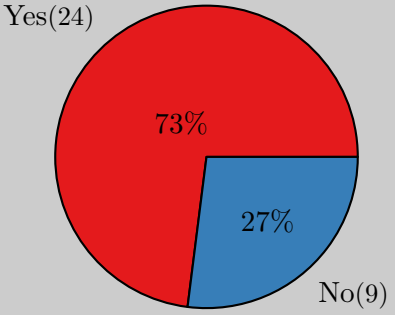
C7: How did you perceive the game's difficulty? *	
Flappy Cycling	**Quake 3 Arena Mod**
	
C8: Why?	**C8: Why?**
[C7: 4 difficult] difficult to dodge the evil birds.	[C7: 4 difficult] mainly difficult because i wasn't used to the controls.
[C7: 5 too difficult] trying to avoid the birds is too hard.	[C7: 3 balanced] Difficulty is adjustable.
[C7: 4 difficult] could not dodge birds. had feeling of not being in control of my bird.	[C7: 4 difficult] Controls are hard.
[C7: 3 balanced] was ok.	[C7: 4 balanced] With more practice would make it a lot easier.
[C7: 4 difficult] Birds fly towards you more aggressively than you can react to.	

[C7: 4 difficult] I find it difficult to dodge the birds.	
[C7: 3 balanced] sometimes I could not navigate properly.	
[C7: 4 difficult] difficulties to control the bird with the ergometer. more fine tuning of the ergometer is necessary.	
[C7: 4 difficult] The reaction are quite slow.	

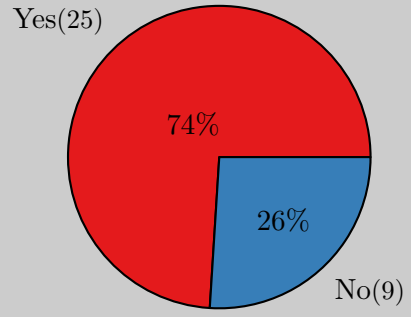
TABLE B.33: FC vs. Q3A: C7–C8.

C9: If possible, would you consider playing (FC) such games / (Q3A) your favorite games with similar devices at home? *

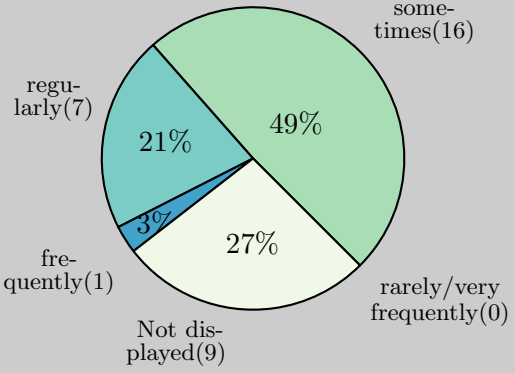
Flappy Cycling



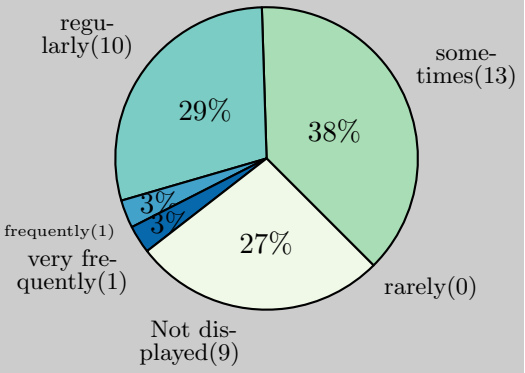
Quake 3 Arena Mod



C10: How often? * (C9: Yes)



C10: How often? * (C9: Yes)



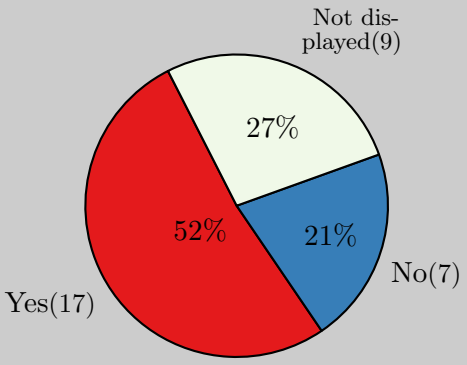
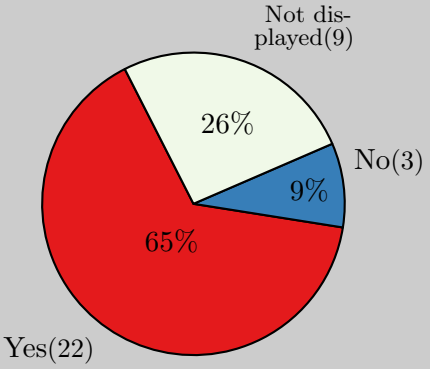
C11: Would you thus be inclined to work out more often? * (C9: Yes)	C11: Would you thus be inclined to work out more often? * (C9: Yes)
	
C12: Why not? (C11: No)	**C12: Why not? (C11: No)**
Maybe i would, if it was more challenging.	I would be too lazy.
Don't have enough space.	
I don't usually play video games.	
takes a lot of space. too hard. feeling of immersion is suffering if the actions performed are not the same as in reality and virtual reality. cycling on a virtual bicycle might improve that.	

Table B.34: FC vs. Q3A: C9–C12.

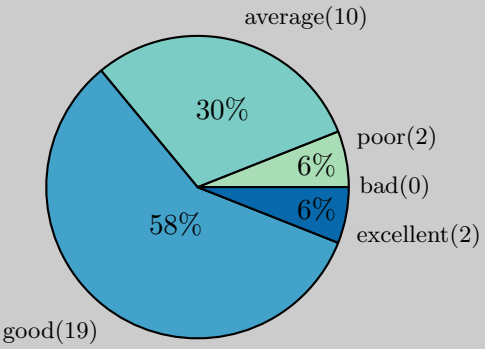
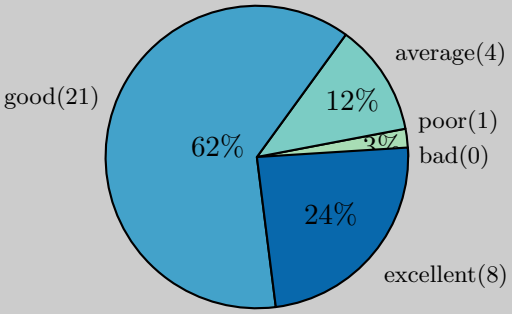
C13: How would you rate your overall experience during the experiment? *	
Flappy Cycling	**Quake 3 Arena Mod**
	

Table B.35: FC vs. Q3A: C13.

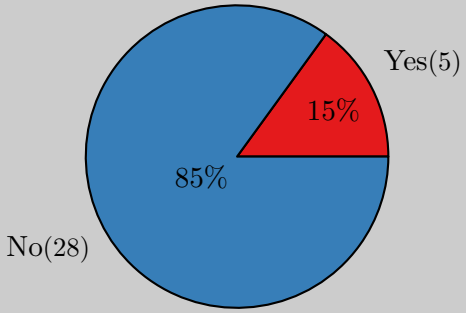
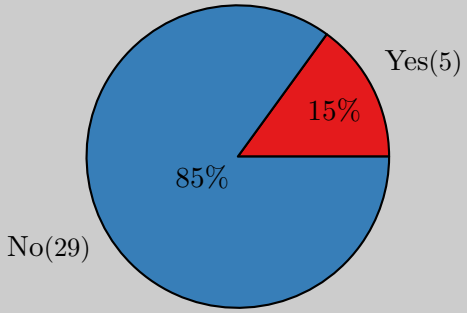
D1: Have you ever participated in an experiment similar to this one? *	
Flappy Cycling	**Quake 3 Arena Mod**
	

Table B.36: FC vs. Q3A: D1.

D2: Any other comments about what you liked or did not like, or things that should be changed during the course of this experiment?	
Flappy Cycling	**Quake 3 Arena Mod**
dodging the birds is way too difficult, otherwise very cool.	the experiment itself was conducted very well.

It would be more fun to try to hit the enemy birds.	no nothing, it was a great experience doing it on that way combine sport and game.
Improve the controls! Nevertheless, great idea, motivates for exercise.	I really like the topic you chose for your thesis. Good luck for your experiment :-).
VR headset.	The idea was very original, but it is difficult to implement with multiple genres (MOBA, strategy) and therefore restricted to only a few. For competitive online play it would be counterproductive, since I would be much faster with the keyboard I think. This "tool" is well suited for combining physical exercise with gaming, but I would not classify it as market-ready.
Nice Game.	I would like it, if the transition between slow and fast would be more fluent.
The experiment was conducted in a great way. The examiner was very helpful, motivated and nice.	Go on with your experiment – great project.
need a shower now.	Generally, one could use an ergometer playing any game (without games specifically created for it), but nobody or very few would do that because ingame rewards provide a much better motivation.

Table B.37: FC vs. Q3A: D2.

B.4 Mann-Whitney-Wilcoxon Tables

TABLE B.38: MWW U Tables: $n_B = 3$ or 4. Probabilities of obtaining an U not larger than u when comparing n_A and n_B samples: $P(U \leq u)$. Source: based on [142].

	$n_B = 3$			$n_B = 4$			
	n_A			n_A			
u	1	2	3	1	2	3	4
0	0.250	0.100	0.050	0.200	0.067	0.029	0.014
1	0.500	0.200	0.100	0.400	0.133	0.057	0.029
2	0.750	0.400	0.200	0.600	0.267	0.114	0.057
3		0.600	0.350		0.400	0.200	0.100
4			0.500		0.600	0.314	0.171
5			0.650			0.429	0.243
6						0.571	0.343
7							0.443
8							0.557

TABLE B.39: MWW U Tables: $n_B = 5$ or 6. Probabilities of obtaining an U not larger than u when comparing n_A and n_B samples: $P(U \leq u)$. Source: based on [142].

	$n_B = 5$					$n_B = 6$					
	n_A					n_A					
u	1	2	3	4	5	1	2	3	4	5	6
0	0.167	0.048	0.018	0.008	0.004	0.143	0.036	0.012	0.005	0.002	0.001
1	0.333	0.095	0.036	0.016	0.008	0.286	0.071	0.024	0.010	0.004	0.002
2	0.500	0.190	0.071	0.032	0.016	0.429	0.143	0.048	0.019	0.009	0.004
3	0.667	0.286	0.125	0.056	0.028	0.571	0.214	0.083	0.033	0.015	0.008
4		0.429	0.196	0.095	0.048		0.321	0.131	0.057	0.026	0.013
5		0.571	0.286	0.143	0.075		0.429	0.190	0.086	0.041	0.021
6			0.393	0.206	0.111		0.571	0.274	0.129	0.063	0.032
7			0.500	0.278	0.155			0.357	0.176	0.089	0.047
8			0.607	0.365	0.210			0.452	0.238	0.123	0.066
9				0.452	0.274			0.548	0.305	0.165	0.090

10		0.548	0.345		0.381	0.214	0.120
11			0.421		0.457	0.268	0.155
12			0.500		0.543	0.331	0.197
13			0.579			0.396	0.242
14						0.465	0.294
15						0.535	0.350
16							0.409
17							0.469
18							0.531

TABLE B.40: MWW U Table: $n_B = 7$. Probabilities of obtaining an U not larger than u when comparing n_A and n_B samples: $P(U \leq u)$. Source: based on [142].

	$n_B = 7$						
	n_A						
u	1	2	3	4	5	6	7
0	0.125	0.028	0.008	0.003	0.001	0.001	0.000
1	0.250	0.056	0.017	0.006	0.003	0.001	0.001
2	0.375	0.111	0.033	0.012	0.005	0.002	0.001
3	0.500	0.167	0.058	0.021	0.009	0.004	0.002
4	0.625	0.250	0.092	0.036	0.015	0.007	0.003
5		0.333	0.133	0.055	0.024	0.011	0.006
6		0.444	0.192	0.082	0.037	0.017	0.009
7		0.556	0.258	0.115	0.053	0.026	0.013
8			0.333	0.158	0.074	0.037	0.019
9			0.417	0.206	0.101	0.051	0.027
10			0.500	0.264	0.134	0.069	0.036
11			0.583	0.324	0.172	0.090	0.049
12				0.394	0.216	0.117	0.064
13				0.464	0.265	0.147	0.082
14				0.536	0.319	0.183	0.104
15					0.378	0.223	0.130
16					0.438	0.267	0.159
17					0.500	0.314	0.191

18			0.562	0.365	0.228
19				0.418	0.267
20				0.473	0.310
21				0.527	0.355
22					0.402
23					0.451
24					0.500
25					0.549

TABLE B.41: MWW U Table: $n_B = 8$. Probabilities of obtaining an U not larger than u when comparing n_A and n_B samples: $P(U \leq u)$. Source: based on [142].

	$n_B = 8$							
	n_A							
u	1	2	3	4	5	6	7	8
0	0.111	0.022	0.006	0.002	0.001	0.000	0.000	0.000
1	0.222	0.044	0.012	0.004	0.002	0.001	0.000	0.000
2	0.333	0.089	0.024	0.008	0.003	0.001	0.001	0.000
3	0.444	0.133	0.042	0.014	0.005	0.002	0.001	0.001
4	0.556	0.200	0.067	0.024	0.009	0.004	0.002	0.001
5		0.267	0.097	0.036	0.015	0.006	0.003	0.001
6		0.356	0.139	0.055	0.023	0.010	0.005	0.002
7		0.444	0.188	0.077	0.033	0.015	0.007	0.003
8		0.556	0.248	0.107	0.047	0.021	0.010	0.005
9			0.315	0.141	0.064	0.030	0.014	0.007
10			0.388	0.184	0.085	0.041	0.020	0.010
11			0.461	0.230	0.111	0.054	0.027	0.014
12			0.539	0.285	0.142	0.071	0.036	0.019
13				0.341	0.177	0.091	0.047	0.025
14				0.404	0.218	0.114	0.060	0.032
15				0.467	0.262	0.141	0.076	0.041
16				0.533	0.311	0.172	0.095	0.052
17					0.362	0.207	0.116	0.065
18					0.416	0.245	0.140	0.080

19	0.472	0.286	0.168	0.097
20	0.528	0.331	0.198	0.117
21		0.377	0.232	0.139
22		0.426	0.268	0.164
23		0.475	0.306	0.191
24		0.525	0.347	0.221
25			0.389	0.253
26			0.433	0.287
27			0.478	0.323
28			0.522	0.360
29				0.399
30				0.439
31				0.480
32				0.520

TABLE B.42: MWW critical U value Table for $\alpha = 0.10$. H_0 can be rejected when observing a u less or equal to the listed value (one-tailed). For two-tailed tests α is doubled. Source: [151].

n_A n_B = 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1																			
2		0	0	1	1	1	2	2	3	3	4	4	5	5	5	6	6	7	7
3	0	1	1	2	3	4	5	5	6	7	8	9	10	10	11	12	13	14	15
4	0	1	3	4	5	6	7	9	10	11	12	13	15	16	17	18	20	21	22
5	1	2	4	5	7	8	10	12	13	15	17	18	20	22	23	25	27	28	30
6	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	34	36	38
7	1	4	6	8	11	13	16	18	21	23	26	28	31	33	36	38	41	43	46
8	2	5	7	10	13	16	19	22	24	27	30	33	36	39	42	45	48	51	54
9	2	5	9	12	15	18	22	25	28	31	35	38	41	45	48	52	55	58	62
10	3	6	10	13	17	21	24	28	32	36	39	43	47	51	54	58	62	66	70
11	3	7	11	15	19	23	27	31	36	40	44	48	52	57	61	65	69	73	78
12	4	8	12	17	21	26	30	35	39	44	49	53	58	63	67	72	77	81	86
13	4	9	13	18	23	28	33	38	43	48	53	58	63	68	74	79	84	89	94
14	5	10	15	20	25	31	36	41	47	52	58	63	69	74	80	85	91	97	102
15	5	10	16	22	27	33	39	45	51	57	63	68	74	80	86	92	98	104	110

16	5	11	17	23	29	36	42	48	54	61	67	74	80	86	93	99	106	112	119
17	6	12	18	25	31	38	45	52	58	65	72	79	85	92	99	106	113	120	127
18	6	13	20	27	34	41	48	55	62	69	77	84	91	98	106	113	120	128	135
19	7	14	21	28	36	43	51	58	66	73	81	89	97	104	112	120	128	135	143
20	7	15	22	30	38	46	54	62	70	78	86	94	102	110	119	127	135	143	151

TABLE B.43: MWW critical U value Table for $\alpha = 0.05$. H_0 can be rejected when observing a u less or equal to the listed value (one-tailed). For two-tailed tests α is doubled. Source: [151].

n_A $n_B =$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1																				
2					0	0	0	1	1	1	1	2	2	3	3	3	3	4	4	4
3			0	0	1	2	2	3	4	4	5	5	6	7	7	8	9	9	10	11
4			0	1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18
5		0	1	2	4	5	6	8	9	11	12	13	15	16	18	19	20	22	23	25
6		0	2	3	5	7	8	10	12	14	16	17	19	21	23	25	26	28	30	32
7		0	2	4	6	8	11	13	15	17	19	21	24	26	28	30	33	35	37	39
8		1	3	5	8	10	13	15	18	20	23	26	28	31	33	36	39	41	44	47
9		1	4	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54
10		1	4	7	11	14	17	20	24	27	31	34	37	41	44	48	51	55	58	62
11		1	5	8	12	16	19	23	27	31	34	38	42	46	50	54	57	61	65	69
12		2	5	9	13	17	21	26	30	34	38	42	47	51	55	60	64	68	72	77
13		2	6	10	15	19	24	28	33	37	42	47	51	56	61	65	70	75	80	84
14		3	7	11	16	21	26	31	36	41	46	51	56	61	66	71	77	82	87	92
15		3	7	12	18	23	28	33	39	44	50	55	61	66	72	77	83	88	94	100
16		3	8	14	19	25	30	36	42	48	54	60	65	71	77	83	89	95	101	107
17		3	9	15	20	26	33	39	45	51	57	64	70	77	83	89	96	102	109	115
18		4	9	16	22	28	35	41	48	55	61	68	75	82	88	95	102	109	116	123
19		4	10	17	23	30	37	44	51	58	65	72	80	87	94	101	109	116	123	130
20		4	11	18	25	32	39	47	54	62	69	77	84	92	100	107	115	123	130	138

TABLE B.44: MWW critical U value Table for $\alpha = 0.01$. H_0 can be rejected when observing a u less or equal to the listed value (one-tailed). For two-tailed tests α is doubled. Source: [151].

n_A $n_B =$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	2												0	0	0	0	0	0	1	1
	3						0	0	1	1	1	2	2	2	3	3	4	4	4	5
	4				0	1	1	2	3	3	4	5	5	6	7	7	8	9	9	10
	5			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	6			1	2	3	4	6	7	8	9	11	12	13	15	16	18	19	20	22
	7		0	1	3	4	6	7	9	11	12	14	16	17	19	21	23	24	26	28
	8		0	2	4	6	7	9	11	13	15	17	20	22	24	26	28	30	32	34
	9		1	3	5	7	9	11	14	16	18	21	23	26	28	31	33	36	38	40
	10		1	3	6	8	11	13	16	19	22	24	27	30	33	36	38	41	44	47
	11		1	4	7	9	12	15	18	22	25	28	31	34	37	41	44	47	50	53
	12		2	5	8	11	14	17	21	24	28	31	35	38	42	46	49	53	56	60
	13	0	2	5	9	12	16	20	23	27	31	35	39	43	47	51	55	59	63	67
	14	0	2	6	10	13	17	22	26	30	34	38	43	47	51	56	60	65	69	73
	15	0	3	7	11	15	19	24	28	33	37	42	47	51	56	61	66	70	75	80
	16	0	3	7	12	16	21	26	31	36	41	46	51	56	61	66	71	76	82	87
	17	0	4	8	13	18	23	28	33	38	44	49	55	60	66	71	77	82	88	93
	18	0	4	9	14	19	24	30	36	41	47	53	59	65	70	76	82	88	94	100
	19	1	4	9	15	20	26	32	38	44	50	56	63	69	75	82	88	94	101	107
	20	1	5	10	16	22	28	34	40	47	53	60	67	73	80	87	93	100	107	114

Appendix C: Assets

All assets used throughout implementing the prototypes and writing this thesis are listed in Table C.1.

Table C.1: Asset references (cc: Creative Commons).

asset	usage	license	origin (URL)
`thesis`			
Treadmill	clipart (Figure 5.1)	`cc-by`	Noun Project user Sathish Selladurai (`tinyurl.com/glrsmgb`)
Joystick	clipart (Figure 5.1)	`unlimited commercial use`	Openclipart user sagism (`tinyurl.com/hxdnebv`)
Game Pad	clipart (Figure 5.18)	`unlimited commercial use`	Openclipart user qubodup (`tinyurl.com/hurh6ue`)
Desktop PC	clipart (Figures 5.1, 5.5, 5.10, 5.18)	`unlimited commercial use`	Openclipart user averpix (`tinyurl.com/hhdtf2g`)
Ultrasport Ergometer	image (Figures 5.2a, 5.5, 5.10, 5.18)	`Ultrasport`	Amazon EU (`tinyurl.com/jlqnbna`)
Arduino Leonardo	image (Figure 5.2b)	`cc-by-sa`	Arduino USA (`tinyurl.com/qc2xzda`)
Pulse Sensor	image (Figures 5.2c, 5.19)	`open hardware`	World Famous Electronics (`tinyurl.com/jlqnbna`)
Heart Beat	clipart (Figure 5.18)	`cc-by`	Noun Project user Plainicon (`tinyurl.com/jcxgqkg`)

Fritzing Cliparts	clipart (Figures 5.5, 5.10, 5.14, 5.15, 5.18, 5.19)	open-source initiative	Fritzing (fritzing.org)
flappy cycling			
Bird	sprite	cc0	OpenGameArt user MoikMellah (tinyurl.com/zwndqsy)
Crow	sprite	cc-by	OpenGameArt user Redshrike (tinyurl.com/jkuqf5q)
Background	sprite	cc0	Tiny Speck submitted by OpenGameArt user jakegamer (tinyurl.com/jjdukog)
Ambience	sound	cc-by	Freesound user Rick Hoppmann (tinyurl.com/gmdk3s7)
Ding	sfx	cc-by	Freesound user Gabriel Killhour (tinyurl.com/j9yu4rj)
Screech	sfx	cc-by	Freesound user Robinhood76 (tinyurl.com/lcbpkro)
Crow	sfx	cc-by	Freesound user davidworksonline (tinyurl.com/jsty27n)
Tick	sfx	cc-by	Freesound user jorickhoofd (tinyurl.com/jb5ssms)
Wings	sfx	cc-by	Freesound user AgentDD (tinyurl.com/jjtbdvj)
Fall	sfx	cc-by	Freesound user J.Zazvurek (tinyurl.com/jfqgoqf)
Lose	sfx	cc-by	Freesound user (tinyurl.com/jew4jlw)

Bibliography

[1] Julian Hoppit. Understanding the industrial revolution. *The Historical Journal*, 30(01):pp 211224, mar 1987. URL DOI:10.1017/S0018246X00021993.

[2] Joachim Merz and Tim Rathjen. Time and income poverty: An interdependent multidimensional poverty approach with german time use diary data. Working Papers 126, ECINEQ, Society for the Study of Economic Inequality, 2009. URL http://EconPapers.repec.org/RePEc:inq:inqwps:ecineq2009-126.

[3] Hal Pashler. Dual-task interference in simple tasks: Data and theory. *Psychol. Bull.*, 116(2):220–244, sep 1994.

[4] I. Barshi, L.D. Loukopoulos, R.K. Dismukes, P.S. Dekker, and C.D.E. Maurino. *The Multitasking Myth: Handling Complexity in Real-World Operations*. Ashgate Studies in Human Factors for Flight Operations. Ashgate Publishing Limited, 2012. ISBN 9781409485919. URL https://books.google.at/books?id=DunCUUfQehEC.

[5] F. A. Drews D. L. Strayer, J. M. Watson. Cognitive distraction while multitasking in the automobile. *The Psychology of Learning and Motivation*, 54(2):29–58, jan 2011.

[6] Gabe Zichermann and Christopher Cunningham. *Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps*. O'Reilly Media, Inc., 1st edition, 2011. ISBN 1449397670, 9781449397678.

[7] Marco Aurélio Monteiro Peluso and Laura Helena Silveira Guerra de Andrade. Physical activity and mental health: the association between exercise and mood. *Clinics*, 60:61 – 70, 02 2005. ISSN 1807-5932. URL http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1807-59322005000100012&nrm=iso.

[8] Darren ER Warburton, Crystal Whitney Nicol, and Shannon SD Bredin. Health benefits of physical activity: the evidence. *Canadian medical association journal*, 174(6):801–809, 2006.

[9] Jan Prinzhausen. *Warum nehme ich nicht ab? So wehrt sich der Körper gegen das Schlanksein. [Why don't I lose weight? This is how the body combats slenderness.]*, volume 1. Ketoline, Hinter Malzens Garten 5, 07751 Milda, first edition, may 2015. ISBN 978-3-9817247-0-7.

[10] Nick Montfort and Ian Bogost. *Racing the beam: The Atari video computer system.* Mit Press, 2009.

[11] Mark Overmars. A brief history of computer games. Department of Computer Science, Utrecht University, jan 2012. URL `http://www.cs.uu.nl/docs/vakken/b2go/literature/history_of_games.pdf`.

[12] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: Defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, MindTrek '11, pages 9–15, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0816-8. doi: 10.1145/2181037.2181040. URL `http://doi.acm.org/10.1145/2181037.2181040`.

[13] Marc McDonald, Robert Musson, and Ross Smith. Using productivity games to prevent defects. *The Practical Guide to Defect Prevention, Microsoft Press, Redmond*, pages 79–95, 2008.

[14] John Ferrara. *Playful Design: Creating Game Experiences in Everyday Interfaces.* Rosenfeld Media, first edition, 2012.

[15] Chris O'Brien. Get ready for the decade of gamification. *San Jose Mercury News*, 24, 2010.

[16] Kevin Werbach and Dan Hunter. *For the win: How game thinking can revolutionize your business.* Wharton Digital Press, 2012.

[17] Thomas W. Malone. What makes things fun to learn? heuristics for designing instructional computer games. In *Proceedings of the 3rd ACM SIGSMALL Symposium and the First SIGPC Symposium on Small Systems*, SIGSMALL '80, pages 162–169, New York, NY, USA, 1980. ACM. ISBN 0-89791-024-9. doi: 10.1145/800088.802839. URL `http://doi.acm.org/10.1145/800088.802839`.

[18] Thomas W. Malone. Heuristics for designing enjoyable user interfaces: Lessons from computer games. In *Proceedings of the 1982 Conference on Human Factors in Computing Systems*, CHI '82, pages 63–68, New York, NY, USA, 1982. ACM. doi: 10.1145/800049.801756. URL `http://doi.acm.org/10.1145/800049.801756`.

[19] E K Morris, J T Todd, B D Midgley, S M Schneider, and L M Johnson. The history of behavior analysis: Some historiography and a bibliography. *Behav Anal*, 13(2):131–58, 1990. ISSN 0738-6729. URL `http://www.biomedsearch.com/nih/history-behavior-analysis-Some-historiography/22478061.html`.

[20] Svend Waltman. Behaviorism. *EDF 607*, 2004. URL `http://pangea.selu.edu/~swaltman/behaviorism.pdf`. This is an electronic document. Date of publication: May 1, 2004. Date retrieved: October 28, 2012.

[21] Geir Overskeid. Looking for skinner and finding freud. *The American psychologist*, 62(6):590–5, 2007. ISSN 0003-066X. URL `http://inside.bard.edu/~luka/documents/SkinnerandFreud.pdf`.

[22] Charles Stangor (made available by Andy Schmitz). *Beginning Psychology*, volume v.1.0. Creative Commons License 3.0 by-nc-sa (http://tinyurl.com/2qn665), dec 2012. URL `http://2012books.lardbucket.org/books/beginning-psychology/index.html`.

[23] Richard M Ryan and Edward L Deci. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American psychologist*, 55(1):68, 2000.

[24] Susan Harter. Effectance motivation reconsidered. toward a developmental model. *Human development*, 21(1):34–64, 1978.

[25] Robert W White. Ego and reality in psychoanalytic theory. *Psychological issues*, 1963.

[26] Roy F Baumeister and Mark R Leary. The need to belong: desire for interpersonal attachments as a fundamental human motivation. *Psychological bulletin*, 117(3): 497, 1995.

[27] Harry T Reis. Domains of experience: Investigating relationship processes from three perspectives. *Theoretical frameworks for personal relationships*, pages 87–110, 1994.

[28] R de Charms. Personal causation. *New York*, 1968.

[29] Edward L Deci. Intrinsic motivation. new york and london, 1975.

[30] Mihaly Csikszentmihalyi and Mihaly Csikzentmihaly. *Flow: The psychology of optimal experience*, volume 41. HarperPerennial New York, 1991.

[31] Richard A. Bartle. Hearts, clubs, diamonds, spades: Players who suit muds, apr 1996. URL `http://mud.co.uk/richard/hcds.htm`.

[32] Stuart E Dreyfus and Hubert L Dreyfus. A five-stage model of the mental activities involved in directed skill acquisition. Technical report, DTIC Document, 1980.

[33] Hubert Dreyfus, Stuart E Dreyfus, and Tom Athanasiou. *Mind over machine*. Simon and Schuster, 2000.

[34] Patricia Benner. From novice to expert. *The American Journal of Nursing*, 82(3): 402–407, 1982. ISSN 0002936X. URL `http://www.jstor.org/stable/3462928`.

[35] Robin Hunicke, Marc LeBlanc, and Robert Zubek. Mda: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, volume 4, 2004.

[36] S Nicholson. Strategies for meaningful gamification: Concepts behind transformative play and participatory museums. meaningful play 2012. lansing, michigan. *Meaningful Play*, 2012.

[37] Pankaj Jalote. *An integrated approach to software engineering*. Springer Science & Business Media, 2012.

[38] Mark R Lepper, David Greene, and Richard E Nisbett. Undermining children's intrinsic interest with extrinsic reward: A test of the" overjustification" hypothesis. *Journal of Personality and social Psychology*, 28(1):129, 1973.

[39] Jon Radoff. *Game on: energize your business with social media games*. John Wiley & Sons, 2011.

[40] Scott Nicholson. A recipe for meaningful gamification. In Torsten Reiners and Lincoln C. Wood, editors, *Gamification in Education and Business*, pages 1–20. Springer International Publishing, 2015. ISBN 978-3-319-10207-8. doi: 10.1007/978-3-319-10208-5_1. URL `http://dx.doi.org/10.1007/978-3-319-10208-5_1`.

[41] Malcolm Gladwell. *Blink: The power of thinking without thinking*. Back Bay Books, 2007.

[42] Fiat Eco Drive. Eco-driving uncovered: The benefits and challenges of eco-driving, based on the first study using real journey data, 2010.

[43] Nicole Lazzaro. Why we play games: Four keys to more emotion without story. *Available online: http://www.xeodesign.com/xeodesign_whyweplaygames.pdf*, 2004.

[44] A. Marczewski. *Even Ninja Monkeys Like to Play*. BLURB Incorporated, 2015. ISBN 9781364955526. URL `https://books.google.at/books?id=vGxOjgEACAAJ`.

[45] Benjamin Heilbrunn, Philipp Herzig, and Alexander Schill. Tools for gamification analytics: A survey. In *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*, pages 603–608. IEEE, 2014.

[46] Philipp Herzig, Kay Jugel, Christof Momm, Michael Ameling, and Alexander Schill. Gaml - a modeling language for gamification. In *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, UCC '13, pages 494–499, Washington, DC, USA, 2013. IEEE Computer Society. ISBN 978-0-7695-5152-4. doi: 10.1109/UCC.2013.96. URL `http://dx.doi.org/10.1109/UCC.2013.96`.

[47] Amir Matallaoui, Philipp Herzig, and Rudiger Zarnekow. Model-driven serious game development integration of the gamification modeling language gaml with unity. In *System Sciences (HICSS), 2015 48th Hawaii International Conference on*, pages 643–651. IEEE, 2015.

[48] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: defining gamification. In *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*, pages 9–15. ACM, 2011.

[49] David Codish and Gilad Ravid. Adaptive approach for gamification optimization. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 609–610. IEEE Computer Society, 2014.

[50] David Codish and Gilad Ravid. Personality based gamification-educational gamification for extroverts and introverts. In *Proc. 9 th CHAIS Conference for the Study of Innovation and Learning Technologies: Learning in the Technological Era*, 2014.

[51] D Codish and G Ravid. Detecting playfulness in educational gamification through behavior patterns. *IBM Journal of Research and Development*, 59(6):6–1, 2015.

[52] Baptiste Monterrat, Élise Lavoué, and Sébastien George. Toward an adaptive gamification system for learning environments. In *Computer Supported Education*, pages 115–129. Springer, 2014.

[53] Geiser C Challco, Dilvan A Moreira, Ig I Bittencourt, Riichiro Mizoguchi, and Seiji Isotani. Personalization of gamification in collaborative learning contexts using ontologies. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, 13(6):1995–2002, 2015.

[54] Dominik Hurtienne, Ulrik Schroeder, and Christian Spannagel. It engages!– adaptierbare gamification in einer anfänger-programmiervorlesung. [it engages! – adaptive gamification in a programming course for beginners.]. *HDI 2014–Gestalten von Übergängen: 6. Fachtagung Hochschuldidaktik der Informatik; 15.-16. September 2014, Universität Freiburg*, 9:27, 2015.

[55] Juho Hamari, Jonna Koivisto, and Harri Sarsa. Does gamification work?–a literature review of empirical studies on gamification. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 3025–3034. IEEE, 2014.

[56] Katie Seaborn and Deborah I Fels. Gamification in theory and action: A survey. *International Journal of Human-Computer Studies*, 74:14–31, 2015.

[57] Juan Vargas-Enríquez, Lilia García-Mundo, Marcela Genero, and Mario Piattini. A systematic mapping study on gamified software quality. In *Games and Virtual Worlds for Serious Applications (VS-Games), 2015 7th International Conference on*, pages 1–8. IEEE, 2015.

[58] Philipp Herzig, Michael Ameling, and Alexander Schill. A generic platform for enterprise gamification. In *Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on*, pages 219–223. IEEE, 2012.

[59] Yefeng Liu, Todorka Alexandrova, and Tatsuo Nakajima. Gamifying intelligent environments. In *Proceedings of the 2011 international ACM workshop on Ubiquitous meta user interfaces*, pages 7–12. ACM, 2011.

[60] Alexander Müller, Mathias Lux, and Laszlo Böszörmenyi. The video summary gwap: summarization of videos based on a social game. In *Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies*, page 15. ACM, 2012.

[61] Aaron D Mason, Georgios Michalakidis, and Paul J Krause. Tiger nation: Empowering citizen scientists. In *Digital Ecosystems Technologies (DEST), 2012 6th IEEE International Conference on*, pages 1–5. IEEE, 2012.

[62] Mathias Lux, Alexander Müller, and Mario Guggenberger. Finding image regions with human computation and games with a purpose. *AIIDE*, 12:41–43, 2012.

[63] Michael Riegler, Ragnhild Eg, Mathias Lux, and Markus Schicho. Mobile picture guess: A crowdsourced serious game for simulating human perception. In *Social Informatics*, pages 461–468. Springer, 2014.

[64] Mathias Lux, Mario Guggenberger, and Michael Riegler. Picturesort: gamification of image ranking. In *Proceedings of the First International Workshop on Gamification for Information Retrieval*, pages 57–60. ACM, 2014.

[65] Michael Riegler, Ragnhild Eg, Lilian Calvet, Mathias Lux, Pål Halvorsen, and Carsten Griwodz. Playing around the eye tracker-a serious game based dataset. In *GamifIR@ ECIR*, pages 34–40, 2015.

[66] Benjamin Rainer, Stefan Petscharnig, Christian Timmerer, and Hermann Hellwagner. Is one second enough? evaluating qoe for inter-destination multimedia synchronization using human computation and crowdsourcing. In *Quality of Multimedia Experience (QoMEX), 2015 Seventh International Workshop on*, pages 1–6. IEEE, 2015.

[67] Carsten Eickhoff, Christopher G Harris, Arjen P de Vries, and Padmini Srinivasan. Quality through flow and immersion: gamifying crowdsourced relevance assessments. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 871–880. ACM, 2012.

[68] Elaine Massung, David Coyle, Kirsten F Cater, Marc Jay, and Chris Preist. Using crowdsourcing to support pro-environmental community activism. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 371–380. ACM, 2013.

[69] Michael Koch and Alexander Richter. *Enterprise 2.0: Planung, Einführung und erfolgreicher Einsatz von Social Software in Unternehmen. [Enterprise 2.0: planning, introduction and successful application of social software in enterprises.]*. Oldenbourg Verlag, 2009.

[70] Alexander Richter. Gamification im firmennetzwerk. [gamification in company networks.]. *Wirtschaftsinformatik & Management*, 2013(1):80–85, 2013.

[71] Rosta Farzan, Joan M DiMicco, David R Millen, Casey Dugan, Werner Geyer, and Elizabeth A Brownholtz. Results from deploying a participation incentive mechanism within the enterprise. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 563–572. ACM, 2008.

[72] Rosta Farzan, Joan M DiMicco, David R Millen, Beth Brownholtz, Werner Geyer, and Casey Dugan. When the experiment is over: Deploying an incentive system to all the users. In *Symposium on Persuasive Technology*, 2008.

[73] Jennifer Thom, David Millen, and Joan DiMicco. Removing gamification from an enterprise sns. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1067–1070. ACM, 2012.

[74] João Fernandes, Diogo Duarte, Claudia Ribeiro, Carla Farinha, João Madeiras Pereira, and Miguel Mira da Silva. ithink: A game-based approach towards improving collaboration and participation in requirement elicitation. *Procedia Computer Science*, 15:66–77, 2012.

[75] Petra Schubert, Johannes Hager, and Ludwig Paulsen. Auswirkungen von gamification in enterprise collaboration systems.[impacts of gamification on enterprise collaboration systems.]. In *Mensch & Computer*, pages 3–14, 2014.

[76] Krishna Depura and Mayank Garg. Application of online gamification to new hire onboarding. In *Services in Emerging Markets (ICSEM), 2012 Third International Conference on*, pages 153–156. IEEE, 2012.

[77] Ferdinand Cornelissen, Mark A Neerincx, NJJM Smets, Leo Breebaart, Paul Dujardin, and Mikael Wolff. Gamification for astronaut training. *SpaceOps, Stockholm*, 2012.

[78] Dominic Gorecky, Katharina Mura, Iulia Von Falkenhausen, Judith Apold, and Frank Arlt. Spielebasiertes training gestalten und integrieren. [how to design and integrate game-based training.]. *atp edition-Automatisierungstechnische Praxis*, 55(05):40–47, 2013.

[79] David Crookall. Serious games, debriefing, and simulation/gaming as a discipline. *Simulation & gaming*, 41(6):898–920, 2010.

[80] Natalie J Sintek and Nicolaas P Pronk. Optimizing work with play: A gamification primer. *ACSM's Health & Fitness Journal*, 17(3):35–39, 2013.

[81] Juho Hamari, Kai Huotari, and Juha Tolvanen. Gamification and economics 5. *The gameful world: Approaches, issues, applications*, page 139, 2015.

[82] Sanat Kumar Bista, Surya Nepal, Nathalie Colineau, and Cecile Paris. Using gamification in an online community. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on*, pages 611–618. IEEE, 2012.

[83] Sanat Kumar Bista, Surya Nepal, and Cecile Paris. Engagement and cooperation in social networks: Do benefits and rewards help? In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pages 1405–1410. IEEE, 2012.

[84] Luis De-Marcos, Adrián Domínguez, Joseba Saenz-de Navarrete, and Carmen Pagés. An empirical study comparing gamification and social networking on e-learning. *Computers & Education*, 75:82–91, 2014.

[85] Jordan Harris Frith. *Constructing location, one check-in at a time: Examining the practices of foursquare users*. North Carolina State University, 2012.

[86] Juho Hamari. Transforming homo economicus into homo ludens: A field experiment on gamification in a utilitarian peer-to-peer trading service. *Electronic commerce research and applications*, 12(4):236–245, 2013.

[87] Jonna Koivisto and Juho Hamari. Demographic differences in perceived benefits from gamification. *Computers in Human Behavior*, 35:179–188, 2014.

[88] E. Killian. *Gamification 2.0 - A Concept*. A Coffee Shop Read. Eamonn Killian, 2013. URL `https://books.google.at/books?id=wJJbAgAAQBAJ`.

[89] Thom Hartmann. *Thom Hartmann's Complete Guide to ADHD: Help for Your Family at Home, School, and Work*. Recording for the Blind & Dyslexic, 2003.

[90] Louis George Alexander and John Holder. *K's first case*. Longman, 1975.

[91] Ben Leong and Yanjie Luo. Application of game mechanics to improve student engagement. *Proceedings of International Conference on Teaching and Learning in Higher Education.*, 10(1.368):1256, 2011.

[92] Wei Li, Tovi Grossman, and George Fitzmaurice. Gamicad: a gamified tutorial system for first time autocad users. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 103–112. ACM, 2012.

[93] Paul Denny. The effect of virtual achievements on student engagement. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 763–772. ACM, 2013.

[94] Lasse Hakulinen, Tapio Auvinen, and Ari Korhonen. Empirical study on the effect of achievement badges in trakla2 online learning environment. In *Learning and Teaching in Computing and Engineering (LaTiCE), 2013*, pages 47–54. IEEE, 2013.

[95] Adrián Domínguez, Joseba Saenz-de Navarrete, Luis De-Marcos, Luis Fernández-Sanz, Carmen Pagés, and José-Javier Martínez-Herráiz. Gamifying learning experiences: Practical implications and outcomes. *Computers & Education*, 63:380–392, 2013.

[96] Rudy McDaniel, Robb Lindgren, and Jon Friskics. Using badges for shaping interactions in online learning environments. In *Professional Communication Conference (IPCC), 2012 IEEE International*, pages 1–4. IEEE, 2012.

[97] Geoff Goehle. Gamification and web-based homework. *Primus*, 23(3):234–246, 2013.

[98] Ohad Inbar, Noam Tractinsky, Omer Tsimhoni, and Thomas Seder. Driving the scoreboard: Motivating eco-driving through in-car gaming. In *Proceedings of the CHI 2011 Workshop Gamification: Using Game Design Elements in Non-Game Contexts*, 2011.

[99] Anton Gustafsson, Cecilia Katzeff, and Magnus Bang. Evaluation of a pervasive game for domestic energy engagement among teenagers. *Computers in Entertainment (CIE)*, 7(4):54, 2009.

[100] Jorge F Silva, João Emílio Almeida, Rosaldo JF Rossetti, and Antonio Leca Coelho. Gamifying evacuation drills. In *Information Systems and Technologies (CISTI), 2013 8th Iberian Conference on*, pages 1–6. IEEE, 2013.

[101] Nick Cavill, Sonja Kahlmeier, and Francesca Racioppi. *Physical activity and health in Europe: evidence for action*. World Health Organization, 2006.

[102] American College of Sports Medicine et al. *ACSM's guidelines for exercise testing and prescription*. Lippincott Williams & Wilkins, 2013.

[103] François Modave, Jiang Bian, Trevor Leavitt, Jennifer Bromwell, Charles Harris III, and Heather Vincent. Low quality of free coaching apps with respect to the american college of sports medicine guidelines: A review of current mobile apps. *JMIR mHealth and uHealth*, 3(3), 2015.

[104] Joshua H West, P Cougar Hall, Carl L Hanson, Michael D Barnes, Christophe Giraud-Carrier, and James Barrett. Theres an app for that: content analysis of paid health and fitness apps. *Journal of medical Internet research*, 14(3):e72, 2012.

[105] Cameron Lister, Joshua H West, Ben Cannon, Tyler Sax, and David Brodegard. Just a fad? gamification in health and fitness apps. *JMIR serious games*, 2(2), 2014.

[106] Yu Chen and Pearl Pu. Healthytogether: exploring social incentives for mobile fitness applications. In *Proceedings of the Second International Symposium of Chinese CHI*, pages 25–34. ACM, 2014.

[107] Y Chen, J Zhang, and P Pu. Exploring social accountability in pervasive fitness apps. *Proceeding of the UBICOMM2014, Rome, Italy, August 24*, 28, 2014.

[108] Ahmed Allam, Zlatina Kostova, Kent Nakamoto, and Peter Johannes Schulz. The effect of social support features and gamification on a web-based intervention for

rheumatoid arthritis patients: randomized controlled trial. *Journal of medical Internet research*, 17(1), 2015.

[109] Juho Hamari and Jonna Koivisto. working out for likes: An empirical study on social influence in exercise gamification. *Computers in Human Behavior*, 50: 333–347, 2015.

[110] Jennifer Sween, Sherrie Flynnt Wallington, Vanessa Sheppard, Teletia Taylor, Adana A Llanos, and Lucile Lauren Adams-Campbell. The role of exergaming in improving physical activity: A review. *Journal of physical activity & health*, 11(4): 864, 2014.

[111] Lisbeth H Larsen, Lone Schou, Henrik Hautop Lund, and Henning Langberg. The physical effect of exergames in healthy elderlya systematic review. *GAMES FOR HEALTH: Research, Development, and Clinical Applications*, 2(4):205–212, 2013.

[112] Sari Mokka, Antti Väätänen, Juhani Heinilä, and Pasi Välkkynen. Fitness computer game with a bodily user interface. In *Proceedings of the second international conference on Entertainment computing*, pages 1–3. Carnegie Mellon University, 2003.

[113] James J Lin, Lena Mamykina, Silvia Lindtner, Gregory Delajoux, and Henry B Strub. Fishnsteps: Encouraging physical activity with an interactive computer game. In *UbiComp 2006: Ubiquitous Computing*, pages 261–278. Springer, 2006.

[114] Marco Pasch, Nadia Bianchi-Berthouze, Betsy van Dijk, and Anton Nijholt. Movement-based sports video games: Investigating motivation and gaming experience. *Entertainment Computing*, 1(2):49–61, 2009.

[115] Fabio Buttussi and Luca Chittaro. Smarter phones for healthier lifestyles: An adaptive fitness game. *Pervasive Computing, IEEE*, 9(4):51–57, 2010.

[116] Philipp Brauner, André Calero Valdez, Ulrik Schroeder, and Martina Ziefle. Increase physical fitness and create health awareness through exergames and gamification. In *Human Factors in Computing and Informatics*, pages 349–362. Springer, 2013.

[117] Stephen Yang, Brian Smith, and George Graham. Healthy video gaming: Oxymoron or possibility? *Innovate: Journal of Online Education*, 4(4):5, 2008.

[118] Elaine Biddiss and Jennifer Irwin. Active video games to promote physical activity in children and youth: a systematic review. *Archives of pediatrics & adolescent medicine*, 164(7):664–672, 2010.

[119] Tom Baranowski, Dina Abdelsamad, Janice Baranowski, Teresia Margareta OConnor, Debbe Thompson, Anthony Barnett, Ester Cerin, and Tzu-An Chen. Impact of an active video game on healthy childrens physical activity. *Pediatrics*, 129(3): e636–e642, 2012.

[120] Robin Mellecker, Elizabeth J Lyons, and Tom Baranowski. Disentangling fun and enjoyment in exergames using an expanded design, play, experience framework: A narrative review. *GAMES FOR HEALTH: Research, Development, and Clinical Applications*, 2(3):142–149, 2013.

[121] Soh Masuko and Junichi Hoshino. A fitness game reflecting heart rate. In *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, page 53. ACM, 2006.

[122] Ekaterina Kurdyukova and Matthias Rehm. Interactive fitness game for public places. In *Proceedings of the International Conference on Advances in Computer Enterntainment Technology*, pages 437–438. ACM, 2009.

[123] Feng Lu and Kimberly Turner. Improving adolescent fitness attitudes with a mobile fitness game to combat obesity in youth. In *Games Innovation Conference (IGIC), 2013 IEEE International*, pages 148–151. IEEE, 2013.

[124] Darren ER Warburton, Shannon SD Bredin, Leslie TL Horita, Dominik Zbogar, Jessica M Scott, Ben TA Esch, and Ryan E Rhodes. The health benefits of interactive video game exercise. *Applied Physiology, Nutrition, and Metabolism*, 32(4):655–663, 2007.

[125] Miru Ahn, Sungjun Kwon, Byunglim Park, Kyungmin Cho, Sungwon Peter Choe, Inseok Hwang, Hyukjae Jang, Jaesang Park, Yunseok Rhee, and Junehwa Song. Running or gaming. In *Proceedings of the International Conference on Advances in Computer Enterntainment Technology*, pages 345–348. ACM, 2009.

[126] Taiwoo Park, Chungkuk Yoo, Sungwon Peter Choe, Byunglim Park, and Junehwa Song. Transforming solitary exercises into social exergames. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 863–866. ACM, 2012.

[127] Yue Gao and Regan Mandryk. The acute cognitive benefits of casual exergame play. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1863–1872. ACM, 2012.

[128] Stefan Göbel, Sandro Hardy, Viktor Wendel, Florian Mehm, and Ralf Steinmetz. Serious games for health: personalized exergames. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1663–1666. ACM, 2010.

[129] Mohamad Hoda, Reem Alattas, and Abdulmotaleb El Saddik. Evaluating player experience in cycling exergames. In *Multimedia (ISM), 2013 IEEE International Symposium on*, pages 415–420. IEEE, 2013.

[130] Nasser H Dardas, Juan M Silva, and Abdulmotaleb El Saddik. Target-shooting exergame with a hand gesture control. *Multimedia tools and applications*, 70(3): 2211–2233, 2014.

[131] A Donner, D Goldstein, and J Loughran. Health e-games market report: Status and opportunities. *San Francisco, CA: Physic Ventures*, 2008.

[132] David Kushner. *Masters of Doom: How two guys created an empire and transformed pop culture*. Random House Incorporated, 2004.

[133] Shawn Holmes and Andy Smith. *Focus on MOD programming in Quake 3 Arena*. Premier Press, 2002.

[134] Andrew S Jackson. Estimating maximum heart rate from age: is it a linear relationship? *Medicine & Science in Sports & Exercise*, 39(5):821, 2007.

[135] Gerald F Fletcher, Gary J Balady, Ezra A Amsterdam, Bernard Chaitman, Robert Eckel, Jerome Fleg, Victor F Froelicher, Arthur S Leon, Ileana L Piña, Roxanne Rodney, et al. Exercise standards for testing and training a statement for healthcare professionals from the american heart association. *Circulation*, 104(14):1694–1740, 2001.

[136] Jodi L Whitaker and Brad J Bushman. remain calm. be kind. effects of relaxing video games on aggressive and prosocial behavior. *Social Psychological and Personality Science*, 3(1):88–92, 2012.

[137] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.

[138] Martin B Wilk and Ram Gnanadesikan. Probability plotting methods for the analysis for the analysis of data. *Biometrika*, 55(1):1–17, 1968.

[139] SS Shaphiro and MB Wilk. An analysis of variance test for normality. *Biometrika*, 52(3):591–611, 1965.

[140] Likert-Type Scales. Analyzing and interpreting data from. *Journal of graduate medical education*, page 541, 2013.

[141] Frank Wilcoxon. Probability tables for individual comparisons by ranking methods. *Biometrics*, 3(3):119–122, 1947.

[142] Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.

[143] Jürgen Bortz and Christof Schuster. *Statistik für Human- und Sozialwissenschaftler. [Statistics for human and social scientists.]*. Springer-Verlag, 2011.

[144] Harald Cramér. *Mathematical methods of statistics*, volume 9. Princeton university press, 1945.

[145] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2): 81–93, 1938.

[146] Maurice George Kendall. Rank correlation methods. 1948.

[147] Edward E Cureton. Rank-biserial correlation. *Psychometrika*, 21(3):287–290, 1956.

[148] Karl Pearson. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58:240–242, 1895.

[149] Bernard Rosner. *Fundamentals of biostatistics*. Nelson Education, 2015.

[150] Jacob Cohen. Statistical power analysis for the behavioral sciences. 2nd edn. hillsdale, new jersey: L, 1988.

[151] Daniel Zwillinger and Stephen Kokoska. *CRC standard probability and statistics tables and formulae*. Crc Press, 1999.