LEMONS: A theoretical framework to improve physiotherapeutic motivational games

Manuel Jim Paolo Crepin

Supervised by Dr. Yun Fu

MSc Computer Science
September 9, 2017

Department of Computer Science
University College London

When life gives you lemons, don't make lemonade.

Make life take the lemons back!

Cave Johnson, Portal 2

Abstract

This report set out to dissect the current state of motivational serious games to better understand why there is a need for improvement. It was uncovered that the main distinction between specialised and commercial games in use in physiotherapy is that the former employ a form of Passive gameplay, while the latter display a more motivating Active type of gameplay. These findings were used, in conjunction with information gathered from patients with Cystic Fibrosis undergoing daily physiotherapy exercises, to develop a framework for the design of better gameplay for physiotherapeutic serious games. The frameworks, dubbed LEMONS, contains six main tenets for consideration during game design: Long-term focus, Experience customisation, Mobile design, Observe limitations, Neutrality, and Simplicity.

The framework was validated by designing a game which applied all its principles, and compared against an existing game to be used for the same purpose and end users. End users tested both games, simpler one first to avoid feature bias, and were asked to provide feedback on preferred game and gameplay features. In all cases, principles presented in the framework and applied in the new game were deemed appealing and were asked to be also implemented in the first game. The experiment was regarded as a success, notwithstanding the relatively small sample size. It can be concluded that the LEMONS framework is probably first of its kind in proactively encompassing gameplay aspects in its design considerations, and a valuable tool for the creation of better motivational serious games for physiotherapy. Future work should involve a study on the long-term effects of applying this framework, with particular emphasis put on overcoming the novelty bias and comparing rates of decreased motivation on games applying the framework and not.

Acknowledgements

I would like to thank the Institute of Child Health of University College London for launching the Fizzyo project, thus enabling the creation of the remote breath collection platform using motivational games for patients with Cystic Fibrosis.

Particular thanks go to Dr. Eleanor Main for allowing me to conduct my independent study on a topic which had not been considered, Sarah Rand for her insights, and Helen Douglas for consistently putting me in touch with the affected children at Great Ormond Street Hospital.

Special thanks to my supervisor Dr. Yun Fu for helping me shape my proposed topic from a vague collection of ideas into well-structured and refined arguments, and for providing useful suggestions and hints, while allowing me to develop and conduct my study independently.

Thank you to those who have inspired me along the way with ideas, feedback, and new points of view: the children with Cystic Fibrosis (such as Fredi) for their collaboration and enthusiasm, the researchers at Microsoft Cambridge for their insights, those Fizzyo team members for their collaboration, and specific classmates and special friends for their support and camaraderie.

Finally, my utmost gratitude goes to my family, which has encouraged and supported me in every way along my studies, and without whom I would not have made it this far. Particular thanks go to my parents, whom have always been there for me, and even proof read my entire dissertation in excruciating detail, a clear act of unconditional love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Cystic Fibrosis (CF) is the most common lethal genetic disease in white populations [1]. While there is no cure, life expectancy for afflicted patients has increased over the years thanks to early detection and more aggressive treatments, which are predicted to increase median survival age to over 50 in the UK [2]. Among other symptoms, overproduction of mucus in the lungs requires daily physiotherapeutic treatments to release and secrete the excess, which could otherwise catalyse the proliferation of pulmonary infections [1].

Physiotherapy is the provision of primary care using physical activity. Because of its mechanical nature aimed at improving specific bodily functions, motivation techniques are often used to try and improve the quality of the activity. In recent years, videogames have been introduced into physiotherapy routines to captivate patients and increase adherence, though they often lack in quality or were not designed for this purpose, making them less than ideal [3].

Project Fizzyo is an initiative from the Institute of Child Health at University College London in collaboration with Microsoft for the provision of a digital platform to remotely collect physiotherapeutic data from patients. The aim of this project is to motivate young patients (ages 6-16) to adhere to their prescription through the use of videogames. The award winning [4] project is currently under de-

velopment with multiple students generating several parts of the system in parallel. This report's primary concern is the motivational aspect of the project: improving physiotherapy adherence by designing better games.

1.2 Aims and Goals

The goal of this report is to dissect the current state of motivational serious games for physiotherapy to better understand why there is a need for improvement, to then propose a theoretical framework for the design of better gameplay for videogames to be integrated into the Fizzyo platform. A high quality specialised videogame will be delivered keeping these newly elucidated aspects of game design in mind, and compared to an existing game for evaluation, before being included in the platform.

The personal aim was to gain a deeper understanding of how videogame principles can be used in serious settings, and how to generate such a game using a freely accessible game engine such as Unity 5. Knowing this work would be applied in future, and would directly impact the lives of the young patients I had the pleasure to meet, supplied further motivation to deliver the highest quality work. The project also provided the valuable opportunity to learn C# and LaTeX 2_ε.

1.3 Project Structure

This project was taken out in two stages. Initially, a working application was needed as a base for the platform. The requirements and use cases were gathered and a working user interface was produced for the client application, which was then handed over to another team member for functionality implementation. The so-called Fizzyo Hub will be run on the user devices (tablets) and will communicate to a backend server and Microsoft HealthVault, as well as gather user health information and allow for games to be launched. While not directly connected to the scope of this report, it still provided a useful insight into client expectations and project dynamics.

The second, larger phase included the research into game design principles, learning a new programming language, getting familiar with the Unity engine, and development of the videogame. Many of these tasks were done in parallel, so user findings collection was taken out during game design research, so an appropriate design could be formulated and proposed to the clients before creation. The whole process was highly iterative, with regular checks with clients and even an excellent insightful opportunity to talk with researchers at the Microsoft Research HQ in Cambridge. After a game design was selected, it was developed in Unity over several weeks, while a framework for the development of better gameplay for physiotherapeutic games was researched and applied.

The final product was successfully presented to the clients and tested with end users, effectively validating the developed framework.

1.4 Report Structure

This report consists of a Literature Review which will provide a brief etymology and definition of serious games, along with an analysis of the state of the art games currently used for motivation in physiotherapy.

The Motivation chapter delineates the environment in which this project is set, thus providing a problem statement and aim for the report. The initial stakeholder interviews are described and the findings informing the framework are enumerated.

The LEMONS framework is then presented, and its six tenets elucidated. Its application is exemplified in the following chapter where a game designed using the framework is explained in detail.

In Discussion the game is tested against another for validation. The end user tests are analysed to evaluate the usefulness of the framework. Achievements, Limitations, and Further work conclude the report.

Chapter 2

Literature Review

In the context of this report it is useful to define and examine the origin of serious games, as well as provide examples of current games used in physiotherapy to determine the state of the art solutions in use, and identify their main characteristics and limitations.

2.1 A brief definition and etymology of serious games

To avoid participating in the philosophical arguments of the definition of games, from this point on the term *game* is used liberally to primarily refer to videogames as defined by Karhulahti [5]: electronic games which evaluate performance. As such, serious games can be considered as *any means of computerised game/game industry resources whose chief mission is not entertainment* [6].

The use of serious games is becoming ever more popular across a range of fields, as the increasing number of publications (see Figure 2.1) regarding them demonstrates, as Zagal's book [7] on ludoliteracy also suggests. The appeal of these so-called serious games is mostly motivational, leveraging our innate interest in gaming [8] to perform tasks that could otherwise be perceived as boring. While it is widely accepted that games are an important socio-cultural artifact [9], there are relatively few taxonomies or holistic studies regarding the usefulness of serious games in an applied field, potentially caused by ludology being a relatively young science [10].

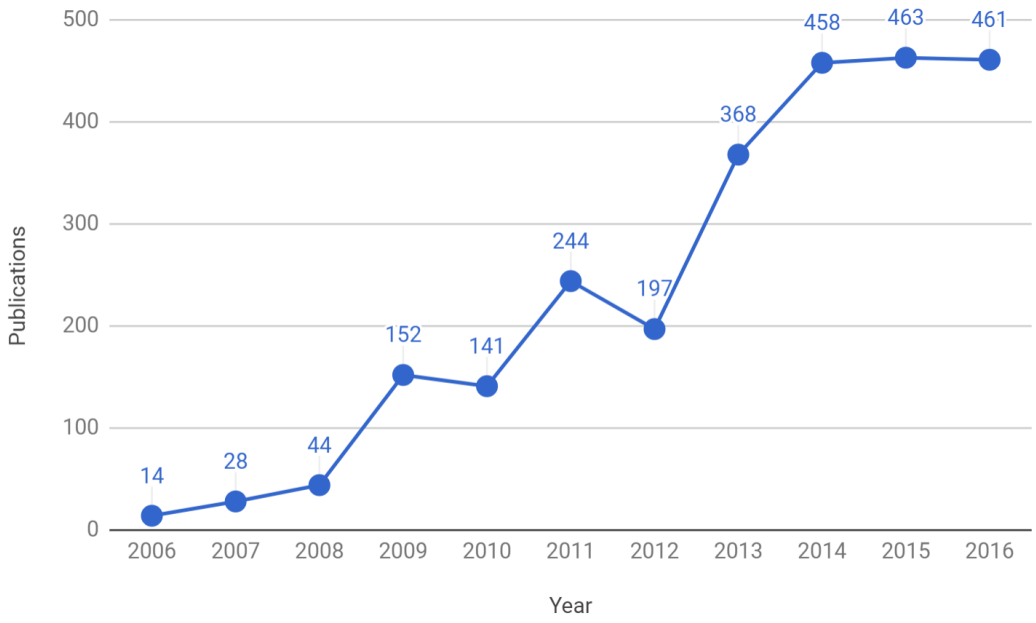

Figure 2.1: Number of publications returned for metadata search ”games in therapy” or ”serious games” or ”gamification” in IEEE Xplore Digital Library, per year.

The earliest occurrence of serious games is described by Manning [11] referring to the use of humour in serious topics during the Renaissance. The notion of *serio ludere* (serious gaming) was used as a stylistic expression, and was only revived in the early 20th century with novels handling the topic of adultery [12].

Use of the term closer to today’s notion of serious games can be found in Clark Abt’s 1970 book [13] titled *Serious Games* describing how games can be used as a method of learning abstraction through both a rational and emotional response, as well as explaining the motivational factors behind games. Clark based his book on his work on military simulations for the US Department of Defense, such as TEMPER (Technological Economic Military Political Evaluation Routine): the first global model of international conflict [14], eerily similar to the *Global Thermonuclear War* supercomputer game featured in the oxymoronically equivalent 1983 fictitious film *WarGames*. Abt did not, however, believe that ”the only winning move is not to play”, but rather encouraged the use of serious games.

It is important to note that, as Sawyer [6] proposes, a serious game project does not necessarily entail the creation of a videogame, but rather using the design meth-

ods or technologies used by game creators to produce a more compelling product. A compelling game does not limit itself to motivational benefits, as Westera's critical evaluation [15] of arguments for digital game-based learning unveils, but rather a range of advantages, from adaptable learning styles to problem ownership.

This process of gamification is evident primarily in the field of education or training, as a majority of publications about serious games seem to refer to (e.g. [16, 17, 18]). This phenomenon is however not unexpected, as using games in any form as an educational tool is almost a universal standard practice at this point in time. Hussain & Coleman [19] offer a valuable insight into this field with their collection of game design guideline essays.

This report restricts itself to the field of games for therapy, so while some arguments could be applied to educational serious games, they are not the target. Furthermore, there is an additional divide between games which aim to have a psychological or physical effect. The following literature review offers an insight into the landscape of serious games used for physical therapy, to better align the later proposed principles to current methodologies and common practices. It will also examine some appropriate generalised best practices for serious games.

2.2 A review of games for physiotherapy

The current state of serious games used in physical therapy can fall under several categorisations. The most common use of games is for rehabilitation purposes, though this kind of use does not exclude other forms of consumption: in all cases videogames are used as a motivational tool to carry out an active level of physiotherapeutic effort, the difference occurs in type of feedback and supervision. A more distinctive subdivision is that of the games themselves, and more qualitative in nature. Whether they have been specifically designed for use in physiotherapy or if they are commercial games repurposed to fill in this role typically characterises their playability, which in turn may affect adherence.

There is little agreement in practice on the precise definitions of gameplay and playability, which is also reflected in the literature. There are several ontologies approaching the subject from different perspectives attempting to define or model this term. In this report, *gameplay* is used liberally to refer to the broad set of rules, objectives, and interactions which are the basis of play activity, as suggested by Mello and Perani [20], who go in greater depth regarding specific definitions for gameplay and playability found in the literature. Gameplay in the physiotherapy context therefore refers to how these three aspects affect player participation in the game, and playability refers to the ability to interface with the game.

Games specifically developed for rehabilitation purposes (e.g. [21, 22, 23, 24, 25]) will tend to focus on the exercises that need to be completed by the patient and will often not develop their gameplay further. This often causes the games to offer more of a visual feedback mechanism of user effort rather than a truly interactive experience. They will also typically be aesthetically less pleasing and polished than commercial games, even those created by independent developers. A clear limitation in these projects, which affects quality, is the expense of game development. Limited budget and time constraints will have an obvious consequence on quality, but do not provide an explanation for bad game design and why they are just not very fun, which will be further discussed later on. Game hackathons frequently produce entertaining games, and are designed to challenge developers to make a game with limited time and often no budget. The difference must thus lie in the developer's inexperience in game design.

Commercial games on the other hand are designed to maximise entertainment, so are an ideal target for integration into physiotherapy sessions (e.g. [3, 26, 27, 28]). While these games actively engage with the user, they are not necessarily ideal for the physiotherapy session as they do not record specific exercise parameters the way specialised games do, or may not provide chances to execute the full range of needed exertions [29]. In this case, the game will typically be used for a limited amount of exercises. Another approach is to use mass mar-

keted game console movement tracking hardware such as the Xbox Kinect (image recognition), or Wii Fit (weight detection pad) or Wii Controllers (motion tracking joysticks). There have been attempts to convert specific detected movements as inputs to drive existing games (e.g. [28]), which solves the session recording issue (assuming gameplay is not affected by different inputs), but not necessarily the exercise type one. Alternatively, many specialised games were designed for use with this hardware, but again fall short in the areas previously described. Commercial gameplay therefore differs in a significant way from most specialised games: they are designed to be fun.

Almost all studies involving games in a serious context reference increased motivation (due to the presence of a game) as the driving factor for the study, as the assumption is that increased motivation will in turn bolster adherence to prescribed physiotherapy exercises. However, no referenced studies took into consideration the novelty of using a videogame when assessing its effectiveness in what is typically referred to as a repetitive and potentially boring treatment. These games may therefore be effective in the short-run to increase motivation by breaking routines, but are not proven to be effectively enjoyable to use. Some studies further reference Burke's Principle of Meaningful Play [30] (whereby a player's actions and choices have a clear in-game consequence and effect) as an additional explanation for their use of serious games, but provide little to no evidence of taking steps to making play meaningful.

Lazzaro's analysis of the player experience [31] explains why some games are more entertaining and meaningful to the gamer than others: there are four specific Types of Fun, which are briefly summarised as follows. *Hard fun* is found in the opportunities for challenge and problem solving, which incite feelings of frustration and fiero. *Easy fun* originates from a player's curiosity and encourages them to explore and unlock mystery. *Altered states* come from strong emotional reactions which can disconnect the player from real life. Finally, the *people factor* is triggered when a game offers social experiences of any nature.

An examination of several games used in physiotherapy, both specialised (e.g. [21, 22, 23, 24, 25]) and commercial (e.g. [3, 26, 27, 28]) in nature, shows that often only commercial games have an aspect of meaningful play, while also showing at least one of Lazzaro's types of fun [31]. A combination of these two concepts could thus be used as a benchmark to determine whether a game will likely be deemed enjoyable by a user or not, summarised as follows:

A game with Active Gameplay will seek to engage the user, by either providing some level of meaningful play in conjunction with at least one type of fun, or integrating more than one type of fun without it being necessarily meaningful.

A game featuring Passive Gameplay will instead seek to not entertain the user, but rather only fulfill its primary function of providing a form of visual feedback which still classifies it as a videogame, but removes any element of fun by not making an attempt to create emotions, instead just emulating a predetermined environment.

It is clear that a specialised game will better aid the patient in their recovery as the game can be designed to focus on a specific and wider range of exercises, which a commercial game may not cater for. However, the motivational aspect of commercial games is more likely to help the patient in the long run in their endevour to adhere to a physiotherapeutic program. It appears that there is somewhat of a trade-off between either quality of the gameplay or motivational factors, and the quality of the physiotherapy exercise. The logical solution is evident: specialised games need to be better designed to reap the benefits of both options while remaining within budget and time constraints.

The release of free to use (within certain profit limits) professional game engines such as Unity 5 and Unreal Engine 4 allow for new specialised games to be developed which have the potential to be semi-professional in quality. Appropriate accompanying game design principles for serious games in physiotherapy will be presented in Chapter 4.

Chapter 3

Motivation

Evidence found in the literature of poor gameplay design for physiotherapy games motivated the research for a better approach to specialised game creation. There is evidence in the literature that there are significant benefits to involving the users in the development process of games (e.g. [27, 29]), which is the approach taken in this report.

The purpose of this primary research is to build upon the work of Mader et al. [29] which suggests breaking down gameplay elements into specific user actions to match ability to appropriate physiotherapeutic goals. While an excellent step forward in the right direction, this model works retrospectively and solely focusses on the gameplay aspect of the game. The aim here is to develop a framework describing the main design elements to proactively consider when developing a specialised game for physiotherapy, in order to reduce creation time and maximise effectiveness.

Before approaching the users, an interview was conducted with the clients to gain as much background information on the typical users and their limitations as possible. Given that both the initial user interface design for the Fizzyo Hub and a complete game were part of this project, the interview focuses on both aspects, though most insights gained can be applied to either design.

One of the symptoms of Cystic Fibrosis is the overproduction of mucus, which

may lead to lung disease caused by the buildup of mucus which cannot be secreted. Lung problems are in fact the leading cause of death for people with Cystic Fibrosis, so respiratory exercises are crucial during treatment [1].

An important distinction in Cystic Fibrosis treatment to other forms of physiotherapy is thus the types of exercises prescribed. They will have to take out both physical exercise (PE) in the form of sports, and airway clearance (AC) exercises, which is done by actively blowing through a device. The client requirement for this project was a game that exercises the latter. Given the nature of the disease, patients will have to do AC for the rest of their lives, with frequency and intensity increasing over time. In the case of the test users (aged roughly 6-16) they typically would do 2 sessions per day, consisting on average of about 10 cycles of 10 breaths each, with a pause between cycles to ”huff” (cough). The medical devices typically used are either a PEP, Flutter, or Acapella: in all cases, their purpose is to exert a resistance against the patient's expiration to ensure an active breath is made.

Because of the intensity of the AC, it is crucial for the system to be straightforward to navigate as the users will typically have to integrate it into their routine, perhaps at times of little energy or motivation, and will not necessarily have access to support. The frequency of AC means variety in games would be welcome to break up the routine and make the exercises less tedious.

Having an initial appreciation of the challenges, a questionnaire was prepared to gain information from several patients about their usual game preferences and expectations. This included specific questions attempting to gauge exactly which game elements increased enjoyment. The findings are grouped and summarised below in an attempt to extrapolate the main desirable features from the wide range of preferences and opinions of the interviewed users.

In all cases users found their prescribed AC exercises tedious and would try to avoid doing them if possible. In some cases they would have to do them up to thrice a day, with the hardest session being the evening one because of fatigue. Most users

wished they could have more control over how they do their AC exercises.

The strongest desired feature by far was customisation, with many examples spontaneously provided. This mostly comes in the form of character customisation through a variety of cosmetic items. Often, these cosmetic items would have to be unlocked through a point system, which would drive the users to continue playing the game.

Other forms of unlockable content that fuelled the user to return to the game were level unlocking to progress, crafting recipes unlocked through discovery, unlocking new abilities to aid in progression, achievements, and so forth. A final type of gameplay that attracted users was procedurally generated levels or content, so that they would always have more game potential. Periodic challenges would also get the user to return, though it would depend on the type of game. Most games played were on a mobile platform, or occasionally console, as the patients can spend days if not weeks hospitalised at one time.

The findings are more qualitative than quantitative due to the relatively small sample size we had access to, and that due to their condition CF patients cannot meet to avoid cross contamination, for example for a focus group. However, the results aligned well with both clients (whom have been in contact with the patients for years) and personal expectations, so it can be assumed they provide a valid insight into the typical user.

Given these findings and the previous research examined in the literature, a framework for the creation of better games was developed, as seen in chapter 4, and applied to a videogame for physiotherapy to test whether it can be realistically and effectively applied.

Chapter 4

The LEMONS Framework

Taking into consideration the feedback gathered from CF patients and the limitations examined in the literature, the following tenets are proposed as a best practice guide for the development of physiotherapeutic motivational games with an active gameplay.

4.1 Long-term focus

Maximising the effective play time of the game is an important factor, given that the patient is likely to have to continue their treatment for an extended period of time. In the case of CF, the typical daily airway clearance exercises prescribed consist on average of two sessions per day, each lasting approximately half an hour. This would mean an average total play time of 365 hour per year. To put this in perspective, the majority of the current most popular games on Steam have an average total play time of about a third of that (see Figure 4.1), with the exceptions being Steam's all time best sellers. In the mobile game landscape, the situation is worse, with the average time for a game to reach maturity shortening to just 20 weeks from release, after which total number of active players declines [32]. While it could be expected that a patient is more motivated to play a game to do their prescribed exercises, it is evident that game length is still a critical issue.

A game designer would be forgiven for primarily thinking that the best way to tackle this issue is simply lengthening the story or increasing the amount of levels of

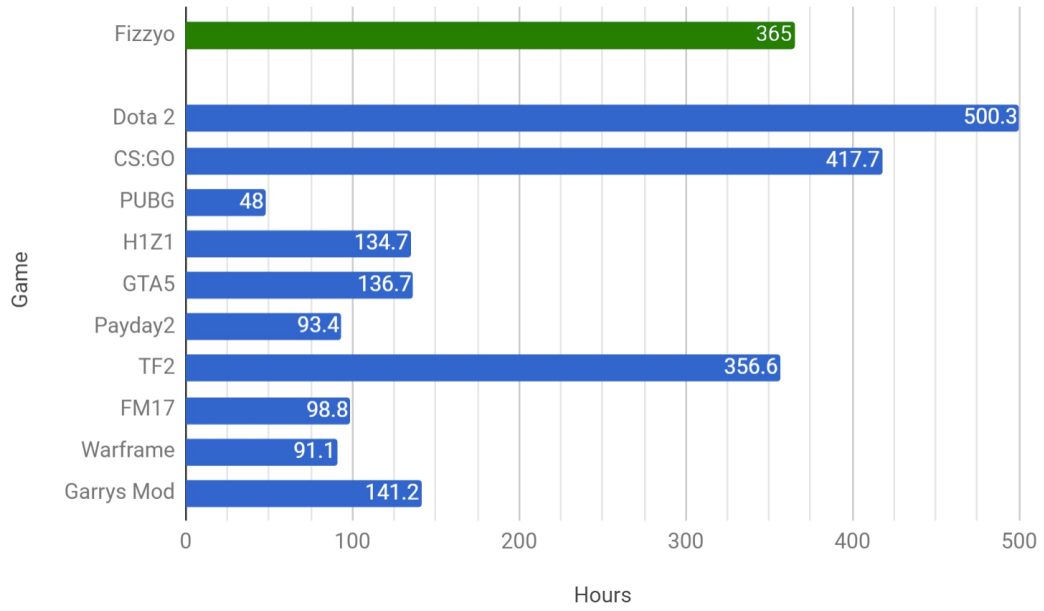

Figure 4.1: Average total play time per year for the top 10 most popular games released on Steam, data gathered July 2017.

a given game. While this would effectively increment the minimum expected play time, it would perhaps worsen the breakup effect of ending the game: most gamers will generally not go back to a game once they perceive it as finished. The reason is different across age ranges, with younger players having a shorter attention span and older ones having less time, but in both cases the presence of alternatives is the main factor preventing gamers from returning to finished games. Again, a patient may be more motivated to continue a game after it is finished, but the motivational factor will be diminished regardless.

Another potential solution on the other side of the gameplay spectrum is disregarding story altogether and using a procedurally generated game to lengthen the replayability of the game. This is a common feature in most current mobile games, with randomly generated endless levels, typically with some mechanic of increasing difficulty over time, and points scored for distance travelled. In these games, motivation originates from non-story sources, which need to be designed carefully to maintain the sense of progression which would usually come with following a story, without becoming too boring or repetitive. This can involve unlockable con-

tent (such as levels, or items which can improve progression), or challenges to gain extra points (to use in a points shop, for example). The balance to strike in this case is to have enough content or challenges to progress to, and not increase the difficulty too quickly to a point that progression would frustrate the user.

The ideal way of preventing the breakup effect is making sure there is enough content for players to follow the designed progression of the game, but also enough freedom for players to have their own goals and objectives to follow in the game. These can differ depending on the game genre and player types, from exploration to player created mini-games; from unlocking every item or achievement, to competitively besting opponents [33]. It is however important to remember that potential game length alone is not sufficient to ensure a good motivational game for therapy, but is one of the important aspects to consider when designing such a game.

4.2 Experience Customisation

The aim of customising the user experience is to increase the cognitive consonance of using the medical device. In the case of Cystic Fibrosis, younger patients will already associate their medical device of choice for airway clearance with some negative connotations: boredom, fatigue, and a reminder of their condition. While playing a game is already an important distraction and improvement on the situation, furthering the customisation of the device and/or game can make the user feel more in control and overall positive.

Customisation can come at different stages of the game lifecycle: during the creation phase (design and testing), and once it has been developed. While both are important, customisation during the creation phase probably brings the most value to the overall product: developers tend to write software based on previous experience and assumptions, so a lack of understanding of the end user will inevitably cause developer bias.

Studies have shown [34] that game co-development (with user input, especially when testing) has the potential of better aligning user expectations and preferences with their needs, which can increase satisfaction with the product, as well as the perceived value. The interaction time with the game also tends to increase, which is likely to bolster the health transfer effect of using the device more effectively.

Customisation should also feature in the developed game: there are many ways to let the player customise their gameplay, perhaps the preferred being character customisation. The most effective way to implement customisation is probably by using a system of points which can be used to purchase items from a shop. Tiered items can give an additional sense of progression or an alternative goal for users. Items can be either purely cosmetic or give the player an advantage, though the latter should be well balanced (ideally temporary) as to not make the game too easy and shorten potential game time. Cosmetic items should be varied enough for there to be an item every user can identify with, in some games player hats or model skins play this role quite successfully.

A final way to adjust the experience of the game is the possibility to change the difficulty level, if at all. There is ongoing discussion as to which method (manual or dynamic) is best, and results in the best experience for the gamer [19]. A game might automatically change difficulty using dynamic difficulty adjustment to match player skill with the challenge at hand, and in the process manage the emotional intensity of the game. This could be an important factor in serious games as the aim is to have the user play the game as long as desired or possible. An alternative approach is to allow the player to manually change the difficulty. If well integrated in the game (not just in a menu) it could allow for a more varied experience for the gamer, but may result in a more difficult task recording scores in competitive social serious games. The final option is to not have a difficulty adjustment at all, which may better suit certain genres of game where challenge is not necessarily crucial for progression.

4.3 Mobile design

Technological mobility allows for medical devices based on a digital platform to be used not only in a clinical space, but also at home. This concept is at the core of the Fizzyo project, benefiting the user who can easily fulfill their prescribed exercises, and the clinicians who can better track and evaluate user performance. The users can thus have more flexibility in the way they take out their routines, which in turn may improve quality of life. In this light, minimising potential errors is crucial. While general Simplicity will be examined later, platform familiarity will be covered in this section.

Especially in the case of young patients, the prime source of entertainment is often a mobile device. This is particularly true when they are hospitalised, as they can spend days at a time in observation. When designing serious games, this knowledge can allow for a better understanding of user expectations in terms of usability. As Norman [35] suggests, using design conventions to convey affordance (the possible actions of something) can make the user better identify with the product. Especially important in the digital environment, which can be less intuitive than the physical environment, is to make sure user interaction is familiar and that the learning curve is minimised.

In practical terms, the ideal way to design the platform and serious game is to emulate current mobile apps. Touch input and small screens impose certain constraints, particularly the need for larger user interface (UI) elements. Clear icons and minimal text allow users to navigate apps quickly, and often platforms will have specific design patterns (such as Material design on Android or Fluent design on Windows) for developers to follow. These result in a higher quality experience as apps are better integrated in systems and users don't have to spend time understanding how the app works. Often mobile output is expected to be primarily visual, with audio and tactile feedback being less valued. This should not however inhibit accessibility measures, which will affect how a device will be interacted with, as

the next section discusses.

Mobile games have a particular set of characteristics (as opposed to computer or console games) which can lend themselves well to serious games. Often mobile games will have simpler game mechanics, partially caused by a lower possible amount of inputs, as well as some hardware limitations. Simplified mechanics make for simpler games, which can however become more addictive. Achievements and unlockables typically become more important than story (which often is absent), and objectives tend to be shorter in time, meaning gratification is constant and instant. However, these factors do mean that mobile games have a much shorter lifecycle, as discussed in the Long-term focus section.

4.4 Observe Limitations

When designing serious games for physiotherapy, it is easy to forget certain user limitations and make games with common game design principles. Simple game concepts such as obstacles or timers can in fact be infinitely more punitive for someone who has to make additional effort during input. In the case of CF, a user may need to quickly pause to cough, and may not have time to pause the game, so a timer limiting game time might not necessarily be fair in gameplay terms as it would keep on running. Similarly, medical devices used may also have their own limitations. In the case of Fizzyo, airway clearance devices require an active, steady flow of air to be effective, which will limit the ways the game will be able to interact with the device. It is therefore important to reduce the dependency on inexorable game limits that would effectively punish the players for having a condition. Any game mechanic that entails a limit should be examined to ensure it is not in fact punitive, but should not necessarily be simply removed.

Limitations should instead be used to empower gameplay and used as mechanics per se. As such, game developers should not replace a limited input with an artificial alternative, but rather embrace the limitation and repurpose it to be a use-

ful input. It is better for the game to be designed around the limitations than to use a potentially limited input for a secondary function, as when a user is immersed in a game they will focus less on the input methods than on the game itself. It is the responsibility of the developer to be informed of the correct exercise types and ensure the users will naturally perform these actions to drive gameplay, rather than being a result of the game. Hardware limitations must also be taken into account: a medical device may have only few input points, and controls may need to be coded creatively, for instance by using different timings or combinations. Given the amount of time the patient is likely to use the game for, a steeper learning curve is marginally less of an issue.

The effect of temporal challenge is also documented by Karhulahti [5], but in particular regarding the type of effort the gamer has to exert to play a videogame: whether a game is kinesthetic or non-kinesthetic affects gameplay, depending under which category the patients limitation falls under. In particular, the combination of different types of challenges can make a game more intense on physical effort (kinesthetic) timed correctly, or more cognitively difficult (non-kinesthetic) within a certain time frame. Ideally, the time challenge element should be paired with the type of effort which is not affected by a limitation to reduce frustrations. On the flip side, should the type of exercise to be undertaken require forced exertion, additional time challenge may improve effort by providing further motivation to the user.

4.5 Neutrality

In the videogame space, neutrality is often an overlooked feature as most games tend to take assumptions of a specific genre and apply those to the game. Neutrality does not refer just to the absence of bias perpetrated by assumptions, but an active effort in ensuring game content is adaptive or appropriate to as wide a range of audiences as possible. The simple reason for applying this sort of neutrality is that serious games, especially in a medical space, are relatively few due to their specialisation, and thus more likely to be applied to a wider audience, or not at all, in which case

they have failed their purpose.

Because of this reality, serious games need to actively try to be as inclusive and accessible as possible, as also suggested by Annema et al. [3]. This includes making considerations such as appropriate game genre, especially where skill could be compared between users, as some may already have pre-existing experience or preference. This can go as far as carefully selecting colour schemes, especially when designing for younger audiences, where the users are more likely to reject a game if they feel it doesn't represent them well enough.

Content does not however have to be bland, but rather strike a good balance between specific expectations of all types of users. For example, a younger gamer may prefer a simpler, yet faster paced game, with instant gratification satiated by a high amount of collectible items (such as coins) or rapid progression towards a goal. An older user instead may have a preference for a slower paced game, where there is a range of objectives to choose from, over which they have more control. A neutral approach would be, for example, to compromise by allowing lesser value items to be collected rapidly to satisfy the need of younger users, while having more valuable, yet challenging, objectives for a more advanced user.

A final point to be made for neutrality is in terms of emotions. While it is ideal for the user to have a high level of engagement with the game to maximise exercise effectiveness and cognitive consonance, it should not be maximised. Murphy [36] defines the $flow\ channel$ as the ideal engagement level between anxiety and boredom, and is the point where users will benefit most from what the game has to offer. Because of this, the game developer should avoid extreme areas of potential psychological burden, where an overly-challenging element, or too simplistic gameplay resulting in boredom, would disrupt $flow$. At the same time, a serious game does not necessarily benefit from being addictive, as this may potentially cause the users to lose sight of the overall benefit of playing the game, or overexert themselves in pursuing a goal that becomes more valuable than the intended outcome of playing the game, for instance an active but non-strenuous physiotherapy session.

4.6 Simplicity

Regardless of the environment in which serious games are to be used, it is important for all potential frustrations to be minimised, for all parties involved. Annema's study [3] makes a strong point for simplicity, as their research found that preparation and setting up of commercial games can take up to half the time of a therapy session. In particular, they found configuration, cutscenes, and numerous on-screen instructions and warnings to be the most frustrating an unnecessary consumptions of time during sessions. In their case, a simplified user interface and overall experience could save valuable time to the benefit of the patient.

In the case of games used at home, simplicity is also a necessity. The user must be able to operate, configure, complete, and potentially troubleshoot exercising with the device by themselves, so games should have as high a level of encapsulation as possible. Because time-consuming exercises for CF are integrated in daily routine, it is not uncommon for patients to have to complete their prescribed exercises at times of day when the user, especially if younger, may be more tired and less likely to make a willing effort to complete activities. In this case, minimisation of lengthy text, steps before initiating, and generally frustration, can lead to a better experience and upkeep of regular intended use.

Simplicity does not, however, necessarily apply to game content itself. Most patients expressed a preference for lengthier, feature-packed games rather than several simpler but shorter games. It is of critical importance for long-term use that the game be designed to require active participation, rather than being a simple result of the users exercise. An active game will seek to stimulate the player both physically and cognitively, spark interest, offer varied challenges, and allow for choices or control over the game. A passive game will be limited to responding to the performance of the input being exercised, and are more of a visual feedback mechanism for the physical input rather than a compelling videogame. As previously mentioned, there needs to be a balance of both kinesthetic and non-kinesthetic challenges.

Chapter 5

Game Design

To validate the effectiveness of the LEMONS framework an experiment had to be designed to measure the appeal of a game where application of the framework was maximised. To do this, a new game was developed from scratch using all the principles delineated in Chapter 4, and compared to an existing game. The Fizzyo project provided an ideal use case scenario, so the existing game used for comparison was one developed during a hackathon, and aimed at the same users. To better emulate a realistic developer use case, game creation time was limited to a month, and with no prior experience of game development.

5.1 Fizzyo Hub

Before game development could begin, a platform from which the games could be launched had to be built to simulate the final working environment. The initial FizzyoHub UI was proposed, adjusted, and developed to the specifications of the clients, though mostly contained placeholders for further development by other project members.

The design was made to be accessible for users of all ages. A home page (see figure 5.1) gives an overview of the daily AC and PE exercise completion, shortcuts to games, and some data related to game statistics. The hamburger menu shows icons when closed and text when opened. There is a page for Games, Airway Clearance, Physical Exercise, and a User page which contains all the settings (such

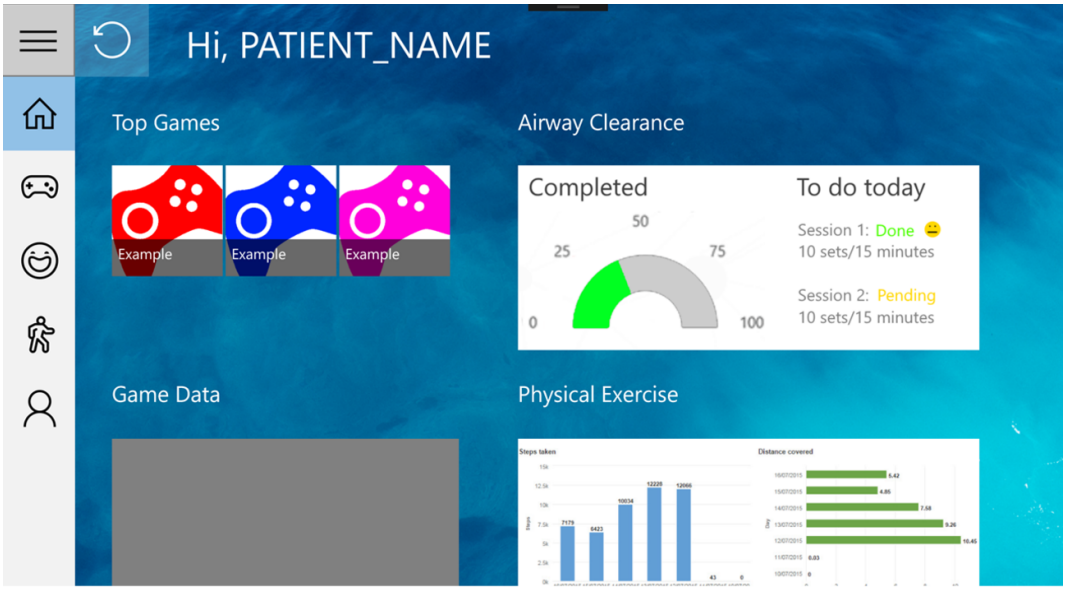

Figure 5.1: Fizzyo Hub Home Page

as background colour and bluetooth devices) and information about the project.

The Fizzyo Hub was also developed with LEMONS in mind. Long-term focus was ensured by making the system as modular as possible, so new menus or pages can easily be added in the future. The menu greeting the user (and their avatar) and allowing for backgrounds to be changed means the experience is more customisable, while the extensive use of buttons and avoidance of large blocks of text give it a more mobile feel. In general, the Fizzyo Hub was made as simple and neutral as possible to avoid any frustrations.

From a technical standpoint, the Fizzyo Hub was developed in Visual Studio 2017 as a UWP app using XAML, which should allow for easy conversion to mobile apps using Xamarin at a later stage. Styles were normalised across all pages and allow for standardised themes (such as Dark/Light) to be applied to the UI elements automatically. Navigation was implemented through a frame in *MainPage*. Most common pages could be accessed through the menu buttons on the left. *UserPage* had another frame for its child (destination) pages, which must therefore have back buttons to the parent (origin) *UserPage*.

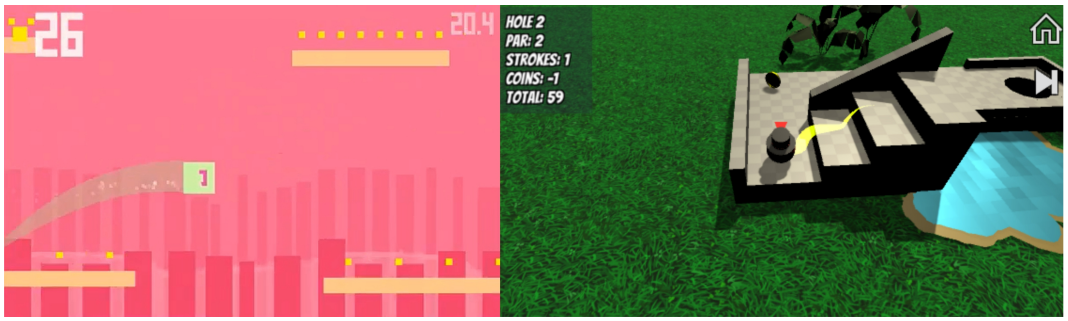

Figure 5.2: Game Screenshots. Left: Qubi. Right: Minigolf Madness.

The UI was passed on to another member of the project team for implementation of functionality. The Fizzyo Hub could successfully launch the game in the final demo and display game-related data.

5.2 Developed game

The Fizzyo project was an ideal candidate for testing the LEMONS framework as it requires specialised games for the desired type of exercise. Additionally, only one game (Qubi) previously developed for the project was mature enough to undergo testing, so there was a chance for comparison, and potential application to another game if deemed necessary. The games used can be seen in figure 5.2. The official clients for the game were Dr Eleanor Main, Sarah Rand, and Helen Douglas of the Institute of Child Health at UCL.

5.2.1 Requirements

The requirements were developed in collaboration with the clients to ensure full compatibility with the platform, hence ideal physiotherapy goals were known and aimed for. Below are the main requirements listed in order of importance as expressed by the clients.

Fizzyo device compatibility: The Fizzyo Hub and games are primarily going to be run on a Windows 10 tablet, so developed games cannot be too resource intensive as it is not a very powerful device. Furthermore, the Fizzyo device consists of a small

tube adapter with an air pressure sensor and a button on it, and these are to be the only inputs for the main gameplay, so the users can focus on breathing correctly.

Generate active breaths: Breathing correctly is a crucial component of AC exercises. Exhalations are the action being exercised, and need to be active, but not forced. Furthermore, breaths should not be prolonged as to put strain on the patient. This sort of breathing is ideal to displace mucus using the airway clearance device of their choice.

Allow for pauses to huff: The goal of airway clearance exercise is to displace and secrete excess mucus in the lungs, so after every set of a number of breaths, the patient is asked to ”huff” (cough out) secretions, which could be done with a game prompt. The user may also need to huff unexpectedly during a set, so pausing should be easy to do.

Avoid short breaths: For the airways clearance exercise to be effective, the patient has to exhale in an active manner. They should not therefore put less pressure in their breath or cut it short because of some gameplay mechanic. At the same time, they should not force themselves to breath out too hard.

Play time of about half hour: The typical session length (of 10 sets of 10 breaths, with huffs) is approximately half an hour. A gaming session should therefore be of at least this length, if it cannot be tailored to the prescription. The user should also be allowed to exercise longer than prescribed if they wish to do so.

Appeal to a wide range of patients: Because of the limited amount of games that will be included with the platform at launch, ideally games should aim to be as accessible and universally appealing as possible. This should also take into consideration that while the primary users are children, their age ranges from 6 to 16.

Adapt gameplay to prescription: Ideally, a game should be able to adapt its content to the patient’s prescription. The game could offer the same amount of levels as AC sets the patient has to do. This should not however prevent the player from

continuing if they so wish.

Support for achievements: A system of achievements was proposed to give a feeling of progression during exercises. Initially, given that the project is in collaboration with Microsoft, it was suggested to be linked to an Xbox Live account, but this did not happen. It was instead picked up by another project member who was preparing a game developer package, not finished at the time of writing.

Share to social media: To give a more social aspect to the games, the idea of creating leaderboards that could be shared on social media was proposed. Because of privacy concerns, social media was dropped, but global leaderboards for games should be included. This was also to be added in the game developer package, which is not finished at the time of writing.

5.2.2 Game proposals

After careful consideration of all the client requirements for the game, several options were developed and proposed to the client, based on user preferences collected during the initial interview phase, and combined with requirements following the LEMONS logic. Finally, the four finalists (sketches can be seen in figure 5.3) were judged by the clients, and are presented in order of increasing preference below. Each option explained basic gameplay, main features, and how the inputs (breath and button) would be used.

Vertical platformer: Inspired by Doodle Jump, this game's aim is to launch a character from one platform to the next using breath as the driving force. The player would gain points by distance, and would be able to move the character by holding down the button, whereas a click would change direction. Procedurally generated moving platforms and obstacles would make the game harder over time. Points earned could unlock more characters or powerups. The main flaw with this idea was that it was too timing based, and could punish pauses. Also the learning curve for the click/hold moving mechanic would make the game very skill-based, which

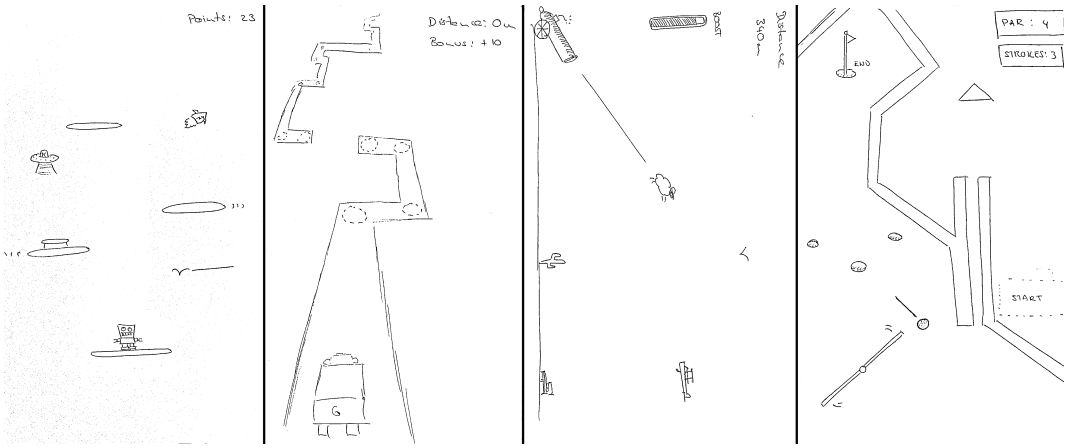

Figure 5.3: Proposed game sketches used for client's consideration. From left: Vertical platformer, Follow the path, Cannon launch, Golf game.

might not appeal to all players.

Follow the path: A simple game where a character has to move along a predefined, procedurally generated path, with obstacles in the form of corners and holes. In order to move forward, the player would have to blow, click at a corner to turn the right way, and hold the button to jump over holes. Points would be awarded for both distance and perhaps a multiplier for precision, which could then be spent on new characters or hats. Again, pauses could be a little punitive, especially if unexpected.

Cannon launch: Another classic-inspired game, cannon launch would launch a character in the air and the player would have to try and make them land as far as possible, using regular breaths to boost them upwards slightly. Points would be earned for distance covered in the procedurally generated world containing both obstacles and boosts. This kind of game would allow for many unlockable items of a large variety, and pauses would be non-punitive as it would be assigned to the button.

Golf game: Finally, the most popular proposal what that of a golf game, where the ball is moved by breath, and a held down button rotates the direction of the ball. There would be obstacles for added challenge and coins to collect to spend

in a shop. There could be several levels with different themes, unlocked once another level is finished. Most importantly, this game is rather neutral (not really skill based) and does not punish the player for pausing, as there is no time constraint.

5.2.3 Tools used

The game selected for development was the golf game, and dubbed Minigolf Madness as a placeholder, which stuck. The game was developed using Unity 5.6.1f1 Personal edition, along with several assets from the Unity Asset Store. Paid assets used were the ProBuilder Advanced v2.9.4f1 by ProCore3D, an invaluable tool for level creation and object modeling. Free assets include several texture and object model packs used in level creation, which do not affect the development of the game or require further documentation.

Objects created using ProBuilder had to be converted and exported to .obj files before being used in the final version of the game, especially if they are being cloned during runtime. Because of some limitations of the target platform (UWP/Windows Store), the ProBuilder package has to be deleted before a Build of the game is made. This does not affect static objects created, but it means assets such as shaders, materials, and textures used must be copied into the project before deleting the package.

All behavioural scripts were coded in C# using Visual Studio 2017. These include scripts for player movement, rotation, camera control, levels, and saving. The standard C# libraries were used together with the UnityEngine package. Other System packages such as IO and Xml.Serialization were used for saving the game state.

Because of delays in the production of the Fizzyo game developer pack, it was not included in any build of the game as of writing. However, an adapter class has been created which should plug into any future release of the game package, and requests functions such as breath detection, counts, and button presses, which at the

moment are either hard coded or simulated locally using mouse or keyboard button presses. Achievements and leaderboards have not been included as they are also part of this package, not finished at the time of writing.

The builds were made in UWP format and had to be launched using Visual Studio. To transfer the project to the tablet, the project had to be repackaged for the Windows Store, but not published to it. The resulting folder could then be distributed and installed using Windows PowerShell, which would take care of installing the correct certificates and the game.

5.2.4 Design considerations

The LEMONS framework was used to the fullest possible extent, guiding the design phase and integrating ideal factors into gameplay. To best fit both the client requirements and framework, several design considerations had to be made.

It was decided that moving the ball directly with breath instead of charging a power bar would make the game more immersive and positive towards the notion of exercising, effectively Observing limitations. It also allows for more continuous application of breath, so that there are no pauses when waiting for the ball to stop. This notion was further reinforced by making the ball heavy, so that it would require more breath to get it rolling in the first place. To avoid the user interrupting a breath to change direction, the *PlayerContoller* script only allows the player to change direction when movement speed is below a certain threshold, so a player has to commit to a certain direction. Rotation was sped up for a minimal pause between breaths, and could realistically be done while inhaling.

The levels were designed in a way to be as replayable as possible. It was difficult to try and generate terrain procedurally, making sure there would always be a path to the hole. This would have made holes too simplistic, and would complicate predicting the amount of strokes (breaths) it would take to reach the end. Therefore, level design is fixed, with each level having its own theme. Coins are spread throughout each minigolf-style course which players can collect and later use at the

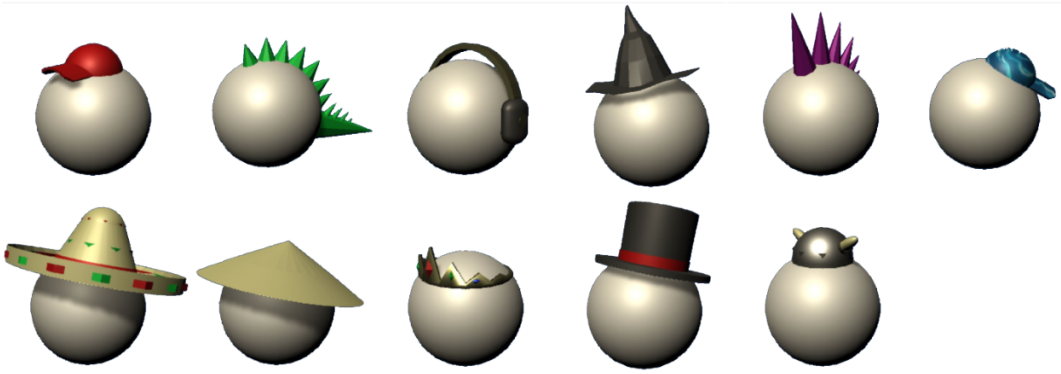

Figure 5.4: Minigolf Madness Hats. Special care was taken to provide a diverse set.

shop to buy items. Each course (level) has a variable amount of holes, but the total par is of at least 100, to ensure the average cycle of 10 breaths for 10 sets was covered. Players can also choose to continue if they haven't reached the end of the level. If they return to a level which they have not completed, they can skip holes, but at the price in coins equal to the hole number they want to skip. This means initial holes can be skipped cheaply, but they cannot simply unlock more levels by skipping.

The Shop is a relatively large feature of the game and took much design and technical work. The main focus was to ensure that the users could customise their ball to make it feel their own. They can change the colour, add a trail, or outfit it with a hat. While a range of trails and colours are provided, the hats were a little trickier. To maintain a good level of neutrality, a dozen models were produced with a wide range of cultural and social indicators (see figure 5.4), so every person could find a hat that would appeal in some way to them. Some of the hats are the result of suggestions from users at the initial interviews.

5.2.5 Interfaces

Each level is driven by two main scripts. A generic *LevelContent* script which contains most reusable code shared across all levels, and a child class holding the same name as the level. Children classes contain the spawn points and par for each new hole. The parent *LevelContent* class handles all the counters (coins, holes, etc)

and events (coin colleted, out of bounds, etc), and updates the UI accordingly. As such, adding new levels is relatively easy: all that needs to be done is link up the *LvlHUD* (level Heads-Up-Display, the UI) template prefab items in a new scene to an empty GameObject containing the *LevelContent* childs script, which inherits all its parents features.

The *UserInput* adapter class (see appendix E.2) is the main point of access to the game from an external point of view. It is designed to accept inputs from the Fizzyo game developer package yet to be produced, and will link Fizzyo device controls such as breath detection and button press detection (booleans) and breath counters (integer) to existing functions. Other inputs dependent on this package such as prescription information have yet to be integrated into the game, but a function to provide popups already exists, which may be used to prompt the user to huff at the appropriate time.

5.2.6 Implementation of Key Features

Correctly moving the player using breath and a button press as input was challenging and required several custom classes to work together: the *PointerController* (PC, used to determine direction), the *CameraController* (CC, used to maintain camera position), the player *Rigidbody* (RB, used in physics events), and the level information parent *LevelContent* (LVL, containing common level functions). The code snippet regarding this section can be found in appendix E.1, pseudocode below:

```
1 if (PlayerIsMoving) disallowRotation(pc, cc); else {
2     if (UserIsPressingButton) rotateAndGetDirection(pc);
3     else stopRotatingAndUpdateDirection(pc, cc); }
4 if (ValidBreathDetected) rb.AddForce(direction * speed);
```

It was determined that the player should not be able to rotate while in motion, so above a certain *thresholdSpeed* the PC would *showAsInactive* (by turning yellow instead of red) and *stopRotating*, while CC would *updateDirection* (rotate the

camera to always look at player, and position it at a constant offset distance). When slow enough, during a button press the PC can *Rotate* and *getDirection*.

In case of a *ValidBreathDetected*, the PC *direction* would be converted to a *forceDirection* vector, which would then be multiplied by a predetermined ideal *speed* and applied to the player RB. In case of collision, the appropriate action would be taken depending on the tag of the object touched, which could result in a respawn, coin collection, boost, teleport to next hole, or just a physics event.

The *LevelContent* class was created to avoid code repetition and simplifying individual level classes. It handles all HUD updates, coin operations, popup functions, and return to menu. Examples seen below are shown in appendix E.4.

One gameplay mechanic is that users can skip a hole by using the coins collected (where price = hole number). The game must determine if the player has gathered enough coins in the past by calling the *SaveState* through *SaveManager.Instance.state.coins* and comparing with the value of *currentHole*. If possible, the player flag *isSkippingHole* is set to true, which the level-specific child *Update* function will handle by asking the parent to teleport the player to a new coordinate it specifies, and canceling momentum to avoid a Portal effect (given that momentum is conserved). The parent class instead handles the HUD updates (counters) and appropriate informative popup.

The popups featured in the game use a lightweight Coroutine to generate an additional thread to display a message overlay and avoid pausing the game. The method simply takes a *message* string and *time* in seconds (float) for popup duration (though there is also an override without time). The time is used in a simple while loop which yields for the *WaitForSeconds* function and decreases the *time* until it reaches zero.

Finally, saving the game is done after each shop purchase or level exit through the *SaveManager* class, which holds the references (declared in the Unity editor) to the shop items that can be purchased (and whether they have been), and saves this information to the *SaveState* file, as seen in appendix E.3.

```
Example: gameObject array[4] (containing 4 shop items)

shopIndex (item):   3 2 1 0   (nothing unlocked)
Bitwise:            0 0 0 0   -> int 0

shopIndex (item):   3 2 1 0   unlockItem(shopIndex(2))
Bitwise :           0 1 0 0   -> int 4

shopIndex (item):   3 2 1 0   unlockItem(shopIndex(0))
Bitwise:            0 1 0 1   -> int 5
```

Table 5.1: Example of bitwise operations for shop item unlocks

The *Awake* method is used to load the *SaveState* on launch and generates a cloned static object for use in runtime using *DontDestryOnLoad* on a gameObject containing the script in the main menu. This way this gameObject will be kept when a new level (scene) is loaded, and equipped items will be displayed on the player.

When an item is purchased in the shop, the *SaveState* is first probed to ensure there are enough coins. If there are, the object is unlocked. The Unlock functions use bitwise operations to toggle bits as flags for each consecutive item, as exemplified in table 5.1. This allows for a standardised approach to unlocking items as their automatically generated *shopIndex* is used as the item identifier, thus removing the need to declare new variables, and saves both memory and time (at least for the first 32,767 items in a given shop category, which is unlikely to be exceeded).

5.2.7 Client Feedback

The clients were very impressed with the initial demo of the game, and agreed to more levels being developed after the end of the project, to allow more time for testing. Therefore, feedback is based on the Tutorial and Woods levels. The shop is fully functional and does not need further updates, but additional content is always desirable. One client request was to ensure the feature to provide a popup when a huff by the user was needed, easily implementable once the developer package is available. All other feedback came from the user tests, as shown in section 6.1.3.

Chapter 6

Discussion

In this chapter, the appeal of Minigolf Madness as a motivational game was analysed by comparing it to the other game prepared for the Fizzyo platform: Qubi. As a result, the effectiveness of the LEMONS framework was tested and evaluated.

6.1 Testing

Given that the game produced will be included in the initial release of the Fizzyo system, appropriate testing had to be conducted to ensure industry-grade release quality. A number of tests were run, ranging from qualitative game tests to assessing minimum hardware requirements.

6.1.1 Hardware Tests

Several systems were tested to try and identify the minimum requirements of the game. Unity games reportedly require a minimum of a DX9 GPU and a CPU with SSE2 instruction set support, though this was not the case using certain test devices which meet these requirements (see table 6.1). The game did however run on the target device (Linx 1020), so hardware testing can be considered a success. For desktop versions of the game, a minimum of a dual-core Intel CPU with integrated Intel HD graphics is recommended to run the game smoothly. No other type of CPU or device (e.g. mobile) were tested because of time constraints.

Device	*OS*	*CPU*	*GPU*	*Run?*
Linx 1020 (Target device)	Windows 10	Intel Atom x5-Z8300	Intel HD Integrated	Yes
Razer Blade Stealth	Windows 10	Intel i7-6500U	Intel HD 520	Yes
Razer Blade Stealth	Windows 10	Intel i7-6500U	Nvidia GTX1060	Yes
Microsoft Surface Pro 4	Windows 10	Intel i5-6300U	Intel HD 520	Yes
Terra Pad 1050	Windows 8.1	Intel Atom N455	Intel GMA 3150	No
Terra Pad 1050	Windows 10	Intel Atom N455	Intel GMA 3150	No

Table 6.1: Test devices used to check hardware compatibility

6.1.2 Game Tests

Game testing was taken out on several types of users to gain varied feedback. End users tested the games from which experiential feedback was gathered. The clients tried the game to ensure suitability for the end users. Other project members tested the game to seek out bugs and ensure proper integration to the platform. Some external testers were also used to ensure the game was free of bugs and fit for independent use. Tests were largely successful, with only minor bugs surfacing and being fixed shortly after. A list of test queries used to identify bugs is listed below.

- Game can run smoothly as an UWP app (if at all)
- All UI elements functional (if interactive)
- Coin balance correct after shop transactions
- All shop objects spawn on player upon purchase
- Game state saved on purchase (balance and unlocked items)
- Popups correctly displayed from first hole to end
- HUD counters change correctly (hole/coins/par/strokes)
- Player collisions (coin collection/respawn if out of bounds/obstacles)
- Player spawns at correct next hole on skip/completion

6.1.3 End User Tests

In order to determine the effectiveness of the LEMONS framework in making a game for physiotherapy more appealing, Minigolf Madness was compared to another game which already exists for the Fizzyo platform: Qubi, a platformer where the user exhales to charge a power bar, and clicks to jump to higher platforms. The higher up the player goes, the more coins they get. There are no obstacles other than gravity, and the only way to end the game is by completing the amount of breaths and sets provided at the start.

Both games were tested with several end users to gather feedback on both game specific preferences (what they would improve) as well as overall experience evaluation (what features they enjoyed the most). Qubi was tested first to avoid feature bias. The results will be analysed in chapter 6.

The overall feedback for Qubi was that, while exciting, it lacked progression, meaning they had no long-term goal to aspire to. Nevertheless, the chance to collect more points higher up was an appealing challenge, especially for younger users. Older users would have also preferred more obstacles, ideally increasing over time. Finally, a strong desire for customisation and use of collected points was expressed, with several users suggesting unlockable content such as power-ups, cosmetic items, or new levels.

After Qubi, Minigolf Madness was tested. The most common feedback was that it felt a little slow in pace. However, older users said they did t mind a more relaxed game. Some users requested further coins to be included as an optional objective, even if it meant higher shop prices. The shop was well appreciated, and a further suggestion of adding unlockable music was brought up. Finally, some users asked for levels to have longer, perhaps more challenging holes.

Figure 6.1: Final user testing. Image included with carer's consent, see appendix D.

6.2 Analysis

There are several differences between the games in terms of how the LEMONS framework can be applied, summarised in table 6.2. The game Qubi shows relatively low levels of compatibility with the framework, while Minigolf Madness was designed to maximise adoption of suggested practices.

From a feature standpoint, the mechanisms put in place to ensure Long-term focus (such as the shop in Minigolf Madness) were deemed very desirable features. It would seem that progression over time, or a feeling of accomplishment, is very much valued by the users. This is in line with Burke's Principle of Meaningful Play [30], which states that a game will be more effective if the user feels their effort is rewarded. Furthermore, the sense of control over the character they customise is likely to make the user feel more emotionally connected to the game. Given that the shop was an appreciated feature, it could be concluded that Experience customisation is a valuable tenet to follow.

LEMONS	*Minigolf Madness*	*Qubi*
Long-term focus	Several levels, unlockable content, periodic challenges.	Single level, no progression, no unlockable content.
Experience customisation	Custom characters, inclusive design, variable difficulty.	Exclusive design, static difficulty, no customisation.
Mobile design	Simple mechanics, clear interface, mobile compatible.	Simple mechanics, clear interface, mobile compatible.
Observe limitations	Non-punitive, limitation drives main game mechanic.	Punitive, non-driving limitation, therapy adjusted.
Neutrality	Range of varied difficulty objectives. Neutral genre.	Skill based, competitive genre.
Simplicity	Ease of access, stimulating game.	Ease of access, simplistic game.

Table 6.2: Comparing Qubi and Minigolf Madness using LEMONS

The preference for Mobile design was most evident when testing the game Qubi, where the game genre was one that is commonly found in mobile platforms. Given that the target device is a tablet, user expectations are closer to that of a mobile device such as a phone rather than a computer game. Interestingly though, the game genre was not a point of preference for any of the test users, but rather the game's simple interface. They appreciated the shallow learning curve and use of common mobile design principles. It is in fact well known that a user will prefer a system which is highly usable and that will not make them feel technologically impaired.

The biggest advantage for the test users is the ability to use videogames to take out their prescribed daily exercises, making a previously tedious task into an enjoyable one. The most important factor in this category is that the game will be driven by the exercise they are doing, rather than being a pleasant distraction during exercise. In the games, Qubi relies on button clicks to jump, while breath makes the character go faster. To resolve the issue of users coasting through the level without breathing, the developer turned breath input into a charger for a power bar, effectively turning the exercise to be done into an additional feature, rather than a driving game factor. This decreases the direct impact of correct exercise on

gameplay, which risks returning over time to the previous state of tedium where it is an expected task rather than significantly affecting the game.

When Observing limitations, conversely, Minigolf Madness ensures that user breath is not only a gameplay driver, but requires an active and conscious effort to control character movement. While both games employ the limitation as a game driver, Qubi restrains its use to kinesthetic response, where Minigolf Madness puts it at the centre of gameplay by requiring both physical and cognitive effort. This maintains better $flow$ over time, which was evidenced in the game tests, as the only way to progress was using active breaths.

Simplicity ties into this argument through the gameplay itself. Having direct control over the character and the ability to make decisions, Minigolf Madness stimulates the user in a number of ways, through both implicit and explicit objectives. User feedback, as previously mentioned, showed a desire for Meaningful Play, which could be obtained by a more complex variety of objectives. As suggested in the framework, simplicity should be in terms of usability rather than content.

Overall, every test user seemed to have a preference for mechanisms and gameplay design considerations which would maximise the application of LEMONS to either game. Often, users expressed interest in adapting features found in the second game to the first. While it is normal for some users to have a preference in genre for either game, the features they found most appealing could be applied in both. It could therefore be deemed that the LEMONS framework is an effective guide towards best practices and tool in the design of better gameplay and development of ideal motivational serious games for physiotherapy.

6.3 Limitations

One limitation affecting the evaluation of these videogames for use in physiotherapy, specifically in the case of Cystic Fibrosis, is the reduced number of patients available to take part in the study. In this report, results were collected from seven

distinct individuals, with the permission of their carers and the staff at the Institute of Child Health: three during the initial user experience information gathering stage, and five (one participating in both) during the final game testing and feedback stage. However, the test users are in fact the end users, so it could be said that development was tailored to their best interest, to be as effective as possible. Furthermore, only two videogame genres were used, whereby the framework should ideally be tested for effectiveness in all genres, even though the tenets are designed as genre-independent, if not game-agnostic.

The issue of novelty is a major general limitation during the test of serious games as a motivational tool to increase the effectiveness of physiotherapy. Physiotherapeutic exercises are by definition repetitive and strenuous for the user, as they aim to train a certain bodily function in order to regain better control. In many cases, this sort of treatment can last for long periods of time, or even the lifetime of the patient. It would be only natural for new methods and interactive tools to spark the interest of patients who often perceive physiotherapy as a chore, especially in the case of younger users who may not understand the importance of continuous and repetitive exercise.

Similarly to all other papers examined, in this case the videogames would be regarded as an absolute success in terms of improving the adherence of patients to their physiotherapeutic regime, given that changes to the routine are always welcome in an attempt to make sessions more interesting. However, further work is necessary in order to examine the long-term effectiveness of videogames as motivational tools in physiotherapy. In particular, it would be interesting to see whether games developed using LEMONS tend to capture user interest longer than other specialised games, or if both types of games decrease in effectiveness over time at the same rate. This would also be an ideal indicator to quantify how many games the user would expect to have access to, or at least the average lifespan of such a game.

Chapter 7

Conclusions

This report set out to dissect the current state of motivational serious games in physiotherapy in order to better understand why there is a need for improvement and how to solve this problem. The achievements are listed below, and future work is suggested based on the limitations and natural progression of the arguments.

7.1 Achievements

The literature review exposed a clear distinction in the games currently used in physiotherapy: specialised games, created for the purpose of generating a certain type of exercise, and commercial games, used as a motivational tool and adapting their gameplay to fit exercise where possible. These types of games have a high correlation with the type of gameplay they feature: specialised games are Passive in engaging the user, while commercial games are Active in giving meaning to gameplay and enticing fun. However, commercial games lack in quality of exercise and reporting abilities, while specialised games tend to be not be as appealing. Given that the purpose of using serious games is mainly to motivate the user, it was determined that a framework to create specialised games designed proactively with Active gameplay is needed.

This inspired the creation of a framework dubbed LEMONS. It features 6 tenets delineating principles and design considerations which should be examined in all phases of game development. The framework is summarised as follows.

Long-term focus: reducing the breakup effect of ending a game by introducing alternative objectives in the form of varied unlockable content, periodic and/or alternative challenges, or allowing for user-defined goals. Types of Fun should also be taken into consideration.

Experience customisation: increasing cognitive consonance by providing control to the user. This can be done during creation, through co-development, or at runtime, allowing for customisation (in various forms, be it cosmetic or gameplay-enhancing) or difficulty level adjustments.

Mobile design: the use of a familiar platform (or emulation of their design principles) can also reduce cognitive dissonance, and lower learning curves. Monitoring user progress and adherence remotely adds to treatment flexibility, also enabling a previously unseen social aspect.

Observe limitations: ensuring the limitation being treated is the force driving gameplay is of crucial importance to adherence. Inevitably, creativity is required on the developer's part to ensure gameplay is non-punitive, yet stimulating, both physically and cognitively. The exercise should not be second to the game.

Neutrality: serious games need to be as accessible and open to users as possible, avoiding developer bias. Pragmatically, developed specialised games are relatively few, and must appeal to a large audience. The game should not cognitively burden the user by being too simple or challenging.

Simplicity: frustrations and unnecessary steps are to be eliminated. Games should be both time efficient and, where applicable, allow for independent operation by the patient. Simplicity does not apply to game content: users prefer a more complex, feature-filled game than several simplistic ones.

The LEMONS framework was tested by comparing an existing game for the Fizzyo platform and a newly purpose-created one, and it was found that users appreciated all features generated from applying the framework, and requested for them

to be added to the other game. It can therefore be concluded that the LEMONS framework is first of its kind, to our knowledge, in encompassing gameplay aspects in its design considerations, and a valuable tool for the creation of better motivational serious games for physiotherapy. It is important to note that, as highlighted by user testing, while game genre does not affect the utility of the framework, it does not guarantee a game to be appealing, this being a highly subjective area. As mentioned in the framework, an appropriate game genre should be used to address the target patients.

Personal achievements include gaining an in-depth understanding of the state-of-the-art games for physiotherapy, which can be extended for use in serious games in all areas. Together with a study of game design principles and C# scripting for Unity, a high quality game was created: Minigolf Madness, which was enjoyed by the children who tested it. The game will be launched with the Fizzyo platform in the pilot stage, and included in the final release, expected to be distributed at national level. It includes the code necessary to function with the Fizzyo breathing device, a Tutorial level, a Woods themed level, functioning shop and automatic progress saving. The shop allows for the colour of the ball and the trail to be changed, and a number of hats to be purchased to the user's desire. The Deployment Manual can be found in appendix A, and the User Manual explaining game mechanics is in appendix B.

7.2 Future Work

The limitations expressed in section 6.3 suggest future work on two fronts: the technical aspect of generating extra content for the Minigolf Madness game, and further theoretical studies on the effectiveness of the LEMONS framework.

The game could benefit from additional content in terms of themed levels. The process of level creation is rather long as generating enough holes to reach total par of a hundred with a variety of length and difficulties is time consuming. Additional

effort was put into making the courses overlap and intersect to add further interest. The game has menu placeholders for a Snow, Desert, and Space levels for further development. The System Manual in appendix C explains how the game can be extended.

Further studies could be done to test the LEMONS framework. Because of time and medical constraints (non-related CF patients should not meet to avoid cross contamination) the game tests could be done with only a limited amount of patients; a wider test with more CF patients would provide a better validation of the framework and provide valuable feedback for game design. Furthermore, it would be interesting to examine whether the LEMONS framework is effective in the long run by examining motivation over time in a game using the framework, thus with Active gameplay, and one with Passive gameplay. This would also help determine the effect of the novelty bias, by which a patient will, in the short run, be more motivated due to a new way of performing physiotherapy.

The framework could also benefit from application to other game genres and forms of physiotherapy to further test its validity. In particular, it would be interesting to examine game improvements in the field of rehabilitation, which is the focus of most physiotherapeutic motivational games. Other studies could be made to determine if the LEMON tenets are also applicable to serious games in other disciplines, such as education, psychotherapy, and gamification in a labour context.

Finally, application of the framework to games used as motivational tools (such as in the Fizzyo platform) should improve physiotherapy adherence for unsupervised exercises, thus enabling further indirect studies on the effectiveness of domestic treatments using them. In the case of Cystic Fibrosis, this would allow for more consistent exercises and conclusive results on unsupervised physiotherapy, which have failed to show consistent benefits in the past [37], potentially due to low adherence.

Bibliography

[1] B. P. O'Sullivan and S. D. Freedman, "Cystic fibrosis," *The Lancet*, vol. 373, no. 9678, pp. 1891–1904, 2009.

[2] J. Dodge, P. Lewis, M. Stanton, and J. Wilsher, "Cystic fibrosis mortality and survival in the uk: 1947-2003," *European Respiratory Journal*, vol. 29, no. 3, pp. 522–526, 2007.

[3] J.-H. Annema, M. Verstraete, V. V. Abeele, S. Desmet, and D. Geerts, "Video games in therapy: a therapist's perspective," *International Journal of Arts and Technology*, vol. 6, no. 1, pp. 106–122, 2012.

[4] AbilityNet, "Tech4good awards: Fizzyo, finalist category: Digital health award," 2017, available at https://www.tech4goodawards.com/finalist/fizzyo/.

[5] V.-M. Karhulahti, "A kinesthetic theory of videogames: Time-critical challenge and aporetic rhematic," *Game Studies*, vol. 13, no. 1, 2013.

[6] B. Sawyer and D. Rejeski, "Serious games: Improving public policy through game-based learning and simulation," 2002.

[7] J. P. Zagal, *Ludoliteracy: defining understanding and supporting games education*. ETC Press, 2010.

[8] J. McGonigal, *Reality is broken: Why games make us better and how they can change the world*. Penguin, 2011.

[9] J. Huizinga and R. F. C. Hull, *Homo Ludens. A Study of the Play-element in Culture. [Translated by RFC Hull].* Routledge & Kegan Paul, 1949.

[10] D. Williams, “Bridging the methodological divide in game research,” *Simulation & Gaming*, vol. 36, no. 4, pp. 447–463, 2005.

[11] J. Manning, *The emblem*. Reaktion Books, 2004.

[12] S. G. Society, “Origins of the serious game name,” 2016, available at https://seriousgamessociety.org/2016/09/21/origins-of-the-serious-game-name/.

[13] C. C. Abt, *Serious games: The art and science of games that simulate life in industry, government and education*. Viking Press: New York, 1970.

[14] Raytheon, *T.E.M.P.E.R. Volume 1, Orientation Manual*. US Dept. of Defense, 1965, available at http://www.dtic.mil/cgi-bin/GetTRDoc?AD=AD0470375.

[15] W. Westera, “Games are motivating, aren´t they? disputing the arguments for digital game-based learning,” *International Journal of Serious Games*, vol. 2, no. 2, 2015.

[16] C. Murphy, D. Chertoff, M. Guerrero, and K. Moffitt, “Design better games: Flow, motivation, and fun,” *Design and Development of Training Games: Practical Guidelines from a Multidisciplinary Perspective*, pp. 146–175, 2014.

[17] A. De Gloria, F. Bellotti, and R. Berta, “Serious games for education and training,” *International Journal of Serious Games*, vol. 1, no. 1, 2014.

[18] B. F. Marques and M. M. De Paula, “An exploratory analysis about requirements of serious games for learning support,” in *12th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, 2017, pp. 1–6.

[19] T. S. Hussain and S. L. Coleman, *Design and Development of Training Games*. Cambridge University Press, 2014.

[20] V. Mello and L. Perani, “Gameplay x playability: defining concepts, tracing differences,” *Simpósio Brasiliero de Jogos e Entertenimento Digital*, vol. 11, pp. 157–164, 2012.

[21] P. M. Bingham, J. H. Bates, J. Thompson-Figueroa, and T. Lahiri, "A breath biofeedback computer game for children with cystic fibrosis," *Clinical pediatrics*, vol. 49, no. 4, pp. 337–342, 2010.

[22] J.-D. Huang, "Kinerehab: A kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities," in *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility*. ACM, 2011, pp. 319–320.

[23] B. Lange, S. Koenig, E. McConnell, C.-Y. Chang, R. Juang, E. Suma, M. Bolas, and A. Rizzo, "Interactive game-based rehabilitation using the microsoft kinect," in *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*. IEEE, 2012, pp. 171–172.

[24] I. Pastor, H. A. Hayes, and S. J. Bamberg, "A feasibility study of an upper limb rehabilitation system using kinect and computer games," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*. IEEE, 2012, pp. 1286–1289.

[25] A. Santos, V. Guimarães, N. Matos, J. Cevada, C. Ferreira, and I. Sousa, "Multi-sensor exercise-based interactive games for fall prevention and rehabilitation," in *Proceedings of the 9th International Conference on Pervasive Computing Technologies for Healthcare*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2015, pp. 65–71.

[26] C. ODonovan and J. Hussey, "Active video games as a form of exercise and the effect of gaming experience: a preliminary study in healthy young adults," *Physiotherapy*, vol. 98, no. 3, pp. 205–210, 2012.

[27] A. Y. Wang, "Games for physical therapy," in *Proceedings of CHI (ACM Conference on Human Factors in Computing Systems)*, 2012.

[28] R. Unnikrishnan, K. Moawad, and R. R. Bhavani, "A physiotherapy toolkit using video games and motion tracking technologies," in *Global Humanitar-*

ian Technology Conference: South Asia Satellite (GHTC-SAS), 2013 IEEE. IEEE, 2013, pp. 90–95.

[29] S. Mader, G. Levieux, and S. Natkin, "A game design method for therapeutic games," in *8th International Conference on Games and Virtual Worlds for Serious Applications (VS-Games)*. IEEE, 2016, pp. 1–8.

[30] J. W. Burke, M. McNeill, D. K. Charles, P. J. Morrow, J. H. Crosbie, and S. M. McDonough, "Optimising engagement for stroke rehabilitation using serious games," *The Visual Computer*, vol. 25, no. 12, p. 1085, 2009.

[31] N. Lazzaro, "Why we play games: Four keys to more emotion without story," 2004.

[32] B. Sinclair, "Mobile games lifecycle shortening - app annie," 2016, available at http://www.gamesindustry.biz/articles/2016-01-20-mobile-games-lifecycle-shortening-app-annie.

[33] R. A. Bartle, *Designing virtual worlds*. New Riders, 2004.

[34] M. M. van Dooren, V. T. Visch, R. Spijkerman, R. H. Goossens, and V. M. Hendriks, "Personalization in game design for healthcare: a literature review on its definitions and effects," *International Journal of Serious Games*, vol. 3, no. 4, pp. 3–28, 2016.

[35] D. A. Norman, "Affordance, conventions, and design," *Interactions*, vol. 6, no. 3, pp. 38–43, 1999.

[36] C. Murphy, "Why games work and the science of learning," 2012.

[37] S. Ledger, H. Douglas, L. Sarria-Jaramillo, P. Rayner, A. Goldman, A. Giardini, S. Prasad, A. Wade, P. Aurora, and E. Main, "Inspire-cf: A randomised trial evaluating the longitudinal effects of a weekly supervised exercise programme on children with cystic fibrosis," in *WCPT Congress 2017*. World Confederation for Physical Therapy, 2017.

Appendices

A Deployment Manual

The source code for the game can be found in the following repositories:
`https://uclix.visualstudio.com/Fizzyo/_git/MinigolfMadness`
`https://github.com/MJPCrepin/Fizzyo-MinigolfMadness`
Both repositories are maintained and the same updates are pushed to both; the former is owned by the project client while the latter is publicly accessible. Pull requests are more likely to be considered on the latter.

To deploy the game it will first need to be built in Unity (5.6.1 or higher) and then packaged using Visual Studio (2017 or higher).

Once opened in Unity, ensure *ProBuilder* has been deleted (found in *_Imports*) and all objects in any Scene are not marked as *Static* before attempting a build. The Build Settings need to include all finalised Scenes in the correct order (*HomePage* as Scene 0), see appendix C for further details. The Windows Store module needs to be downloaded and installed (UWP SDK 10.0.15063.0 or higher). The SDK used was *Universal*10 with Target Device being *PC*. UWP Build type used was *D*3*D* to run on the Local machine.

When the build is completed, the game should be opened using the generated .sln file in Visual Studio. From here, the game can be directly run on the Local Machine (x86 or x64) which will install the game. To export to another Windows machine without Visual Studio, it will need to be packaged as a Windows Store application first.

For integration with the Fizzyo Hub, *App.cs* and *Package.appmanifest* need to be edited as described below.

The *Declarations* tab in *Package.appmanifest* needs to have a *Protocol* added which defines a name for the UWP application, in this case the Fizzyo Hub calls for *minigolfmadness*. Other settings can be left as default.

Then, the following *App.cs* method needs to be edited as follows in order to allow the defined protocol name to be called and launched by another UWP application successfully.

```
1 private void ApplicationView_Activated(
     CoreApplicationView sender, IActivatedEventArgs args)
2 {      CoreWindow.GetForCurrentThread().Activate();
3        if (args.Kind == ActivationKind.Protocol)
4            CoreWindow.GetForCurrentThread().Activate(); }
```

To package the opened project as a Windows Store application, navigate to and select the following menu item:

```
Project > Store > Create App Packages...
```

After refusing to upload to the Windows Store, the app can then be packaged locally, and the resulting build can be freely distributed to other machines. To install, the generated .appxbundle file can be run and the game will be installed automatically with the correct certificates on the machine. Alternatively, the powershell (.ps1) file can be run as an Administrator at the same effect but with debug information.

B User Manual

This manual refers to the final build before submission of the code (Minigolf Madness v1.0.0.0). When the game is launched, the user is greeted with the home page, where there are three buttons: *Play*, *Shop*, and *Quit*.

Play brings the user to the level selection page, where on a button press the game will start at the selected level.

In the Tutorial level, some functions are disabled and popups prompt the user to do certain actions to learn how the game works. By the last hole the user has full control, as defined in table 1. Once a level is finished, the user will be automatically brought back to the main menu.

In the Woods level, the user has full control. This includes the Skip hole button (underneath the Home button), however this will cost the same amount in coins as the current hole number. Coins are gained by running into them, and more are available at later holes. The coins are only respawned every time the level is loaded, and can also be spent in the Shop. Once the ball is put, the player is shown their golf score in relation to the Par amount. Each level has a total Par of about 100. The difficulty is progressively increased, with each following hole becoming either more challenging or making coin collection harder. If a player falls off the course onto the ground, they will be respawned at the beginning of the hole, but the *Stroke* counter will not reset. Similarly, collected coins won't respawn.

Back at the main menu, the Shop button brings the user to the coin shop, where the collected coins can be used to unlock three types of items: Colours, Hats, and Trails. Purchased items can be Previewed directly in the shop, underneath the current coin amount displayed.

Colour refers to the colour of the ball itself, there is a range of 10 different colours to choose from.

Trail is a string of colour that comes out of the ball as it moves, temporarily tracing its path. It also has 10 different colours.

Hats are models that can be spawned on the player and like the other items are

Input Device	*Control: move forwards*	*Control: turn direction*
Mouse	Left click	Right click
Keyboard	Up arrow	Right arrow
Fizzyo device	Breath	Button press

Table 1: User Manual: Game Controls per Input type for Minigolf Madness

purely cosmetic. There's a variety of different hats to choose from, which have been designed so that there is a hat that could represent each individual in some way.

Items are currently priced at a constant price of 10 coins, but this is likely to change in future updates, giving Hats a higher price, and to better reflect the value of coins. The initial amount of 100 coins is also for testing purposes and will be removed in the final release.

To purchase an item, the player has to select the item and click the appropriate *Buy* button which displays the price of the selected item. If the item is already owned, it displays no *Locked* overlay and the *Buy* button becomes an $Equip$ button.

Throughout the game progress is saved automatically each time an item is purchased or once a level is exited. The currently selected items and coin balance are loaded when the game is relaunched.

The user can leave the game using the *Exit* button on the home page.

C System Manual

While the game is intended for use with the Fizzyo platform, full integration using the developer package has not been completed, as the latter was not finished at the time of writing. An adapter class *UserInput.cs* (found in *Assets*) was however made to easily change user inputs (such as new device commands in case of firmare updates). The game should however function with the device at this point in time with the following included lines:

```
1 FizzyoDevice.Instance().Pressure() || Input.GetAxisRaw(
    "Horizontal"); // Breath value (float)
2 FizzyoDevice.Instance().ButtonDown() || Input.GetKeyDown(
    "joystick button 0"); // Button press (bool)
```

As mentioned in the conclusion, the game could be further worked upon by adding more content in terms of levels, and perhaps at a later stage shop contents.

Adding a level

All level scripts are a child of the *LevelContent.cs* class (see appendix E.4), so that common functions (regarding coin collection, popups, and HUD counters) can be inherited and used easily. The child class only needs to call the initialisers for the HUD counters prefab and specify the respawn points for each hole in the course, see appendix E.5 for the *Woods* example. The child *LevelContent* script needs to be assigned to an empty gameObject in Unity, and all the public counters (inherited from the parent) linked to the appropriate HUD gameObjects. To make a new level script therefore the developer just needs to create a child of *LevelContent*, or simply use the *Woods* script as template.

Each course should follow a predetermined theme (such as Desert/Snow/S-pace) and have a total Par of 100, as this is the minimum length of the average airway clearance exercise session. Holes should be varied in length and difficulty, and give no more coins than half the value of the hole number as a rule of thumb. Each level could also feature a certain related challenge, for example Desert could

be windy (constant slight sideways force), Snow could be slippery (momentum only decreases to a certain threshold, thus ”slipping”), and Space could have low gravity. It is important for each course to be interesting, and follow the LEMONS framework. The ground should be set with the *Deathzone* tag for respawn if out of bounds. As it stands, respawn locations are dependent on the hole number, so holes shouldn't be too long. Nevertheless, remember that the patient would rather continue playing the game than be cut short. Most importantly: DO NOT PUNISH THE PLAYER FOR INACTIVITY.

Because of how Unity handles lighting, it is important for there to be no static objects before a build, as the lightmaps will take an excruciatingly long time to generate. Level content has been generated using ProBuilder Advanced which is extremely useful for map creation. However, before building as a UWP app this import needs to be deleted as the UWP platform will not package the executables linked to the tools. This means PB gameObject may need to be converted to regular .obj files and re-imported before they can be used in a build (currently, this is a PB Advanced feature only). Vanilla Unity build tools can be used otherwise. Ensure only copyright free (or purchased) assets are used in the game.

It is important for HomePage to be first (Scene 0) as this will be the first scene rendered on launch. Subsequent levels need to also be in order as button listeners are assigned to existing buttons on launch. For example, in the *HomePage* script, the *OnLevelSelect* method's case 0 refers to *Tutorial*, and case 1 refers to *Woods*.

```
1 private void OnLevelSelect(int currentIndex) {
2     switch (currentIndex) {
3         case 0: SceneManager.LoadScene(1); break;
4         case 1: SceneManager.LoadScene(2); break;
5         default: SceneManager.LoadScene(0); break; }
6      }  // Default loads Main Menu (+ debug.log)
```

Special consideration must therefore be taken when assigning text to the buttons, and their order in the Unity editor.

Adding shop items

Test users always provide great ideas of content to add in shops. Both specific hats and whole other categories have been suggested (e.g. background music or accessories).

To add an extra category, the relevant PlayerController initialisers, HomePage script shop methods and Unity gameObjects can be used as templates and duplicated. In particular, the HomePage script features most variables or functions in sets of three, clearly labelled with the appropriate shop (Colour/Trail/Hat). It should therefore be relatively simple to copy this code to add a new category.

Additional similar changes need to be done in the *SaveState.cs* file (used to save the game on item buy or level end) and *SaveManager.cs* which saves the game state and doubles as a holder for the array of gameObjects to be sold, and additional purchasing functions which can be copied. The purchased and active items flags are saved using bitwise operations on integers in the *SaveState.cs* file to save space.

New items can then be assigned to the array of gameObjects in *SaveManager* through the Unity editor in the *SaveManager* gameObject in the *HomePage* scene. Images can be assigned to the individual scrollPanel buttons (add more buttons and lengthen appropriate arrays for more objects), and individual prices can be set in the *HomePage* shopPrices arrays.

D Consent Form

UCL
GREAT ORMOND STREET
INSTITUTE OF CHILD HEALTH

Great Ormond Street NHS
Hospital for Children
NHS Foundation Trust

PROJECT FIZZYO Consent Form
for photography and filming

The PROJECT FIZZYO team would like to photograph children when they are attending the hospital for an admission or at outpatient clinics and helping us with the project.

This form allows us to make sure we have accurate information about your child and records the fact that you and your child have agreed to your child being photographed and/or filmed by a member of the PROJECT FIZZYO research team.

We may like to use the photographs for training or educational purposes for healthcare professionals, on the hospitals website (www.gosh.nhs.uk), the charity website (www.gosh.org) and posters or oral presentations relevant academic conferences. Additionally, the photographs may be used as graphic material that forms part of the write-up of Computer Science student projects. The material will not be used in academic journals or text books without further permission being sought from you.

You will be given a copy of this consent form; additionally a copy will be kept in the PROJECT FIZZYO research files. The digital images will be stored on a secure UCL computer database and server.

Your child's details:

Child's name: Ronni Hilton
Date of birth: 29.07.08
Address: 15 Ridgewell Ave
Orsett
Essex
RM16 3HP
Hospital No: 882512

Name of parent/guardian: Kristie Aldwinckle
Daytime telephone no/s: 07506751648
Email: Fredronny@outlook.com
Parent/guardian signature: Kaldwinckle
Date: 25/8/17
Child's signature: RH

How you are happy for us to use the interview, photographs or film of your child

I am happy for the photographs and/or film of my child to be used for the following purposes:

- ☑ Training and education of healthcare professionals
- ☑ Academic conferences
- ☑ Great Ormond Street Hospital website
- ☑ Great Ormond Street Hospital Charity website

Photographs and filming are only undertaken with your and your child's permission.
If you have any questions or concerns about what the graphic materials will be used for, please contact Helen Douglas on h.douglas@ucl.ac.uk for more information.

E Code Listing

Code may be edited from original format to better fit paper copies (e.g.: parenthesis relocated and lines/spaces deleted). Only relevant snippets are included.

E.1 PlayerController.cs

The *FixedUpdate* method is used for physics events, such as for player behaviour.

```
1  void FixedUpdate() {
2      var PlayerIsMoving = rb.velocity.magnitude >
           thresholdSpeed;
3      var ValidBreathDetected = UserInput.isExhaling();
4      var UserIsPressingButton = UserInput.
           isHoldingButtonDown();
5
6      if (PlayerIsMoving) {
7          pc.showAsInactive(); pc.stopRotating();
8          cc.UpdateDirection(); }
9      else {
10         pc.showAsActive();
11         if (UserIsPressingButton) { pc.Rotate();
               direction = pc.getDirection(); }
12         else { pc.stopRotating(); cc.UpdateDirection();}
13         }
14     if (ValidBreathDetected) {
15         var convertedDirection = direction * (float)Math.
               PI / 180;
16         var forceDirection = new Vector3(speed * (float)
               Math.Sin(convertedDirection), 0.0f, speed * (
               float)Math.Cos(convertedDirection));
17         rb.AddForce(forceDirection * speed); }
18      }
```

E.2 UserInput.cs

Adapter class designed for an easier integration to the Fizzyo developer package (for communication with Fizzyo Hub and Device) yet to be delivered.

```
1  using UnityEngine;
2
3  public static class UserInput {
4
5  public static bool isExhaling() {
6      var validInput = (
7          Input.GetMouseButton(0) || // mouse control
8              isValidBreath() || // fizzyo device control
9              Input.GetKey(KeyCode.UpArrow)); // keyboard
10     return validInput; }
11
12 public static bool isHoldingButtonDown() {
13     var validInput = (
14         Input.GetMouseButton(1) || // mouse control
15         Input.GetKeyDown("joystick button 0") || //fizzyo
16         Input.GetKey(KeyCode.RightArrow) ); // keyboard
17     return validInput; }
18
19 public static bool isValidBreath() { // Would need to
      know valid breath lower and upper thresholds
20     return Input.GetAxisRaw("Horizontal") > 0; }
21
22 public static bool aBreathIsDetected() {
23     // Used to increment the LevelContent stroke counter
24     return Input.GetMouseButtonDown(0) || Input.
          GetKeyDown(KeyCode.UpArrow) || isValidBreath(); }
25 }
```

E.3 SaveManager.cs

The *SaveManager* class holds the references to the shop items that can be purchased, whether they have been purchased, and saves the game state. It is woken on launch and is cloned and static in each scene during runtime.

```
1  using UnityEngine;
2  public class SaveManager : MonoBehaviour {
3
4      // Controls a SaveState
5      public SaveState state;
6      public static SaveManager Instance { get; set; }
7
8      // Arrays holding purchasable objects
9      public Material playerMaterial;
10     public Color[] playerColours = new Color[13];
11     public GameObject[] playerHats = new GameObject[13];
12     public GameObject[] playerTrails=new GameObject[13];
13
14     private void Awake() {
15         //ResetSave(); // Used to reset save file
16         DontDestroyOnLoad(gameObject);
17         Instance = this; Load(); }
18
19     public void Save() {
20         PlayerPrefs.SetString("Save", Tools.Serialise(
               state)); }
21
22     public void Load() {
23         if (PlayerPrefs.HasKey("Save")) {
24             state = Tools.Deserialise<SaveState>(
                   PlayerPrefs.GetString("Save")); }
25         else { state = new SaveState(); Save(); }
26     }
```

Check if a shop item is owned (similar for Trail and Colour).

```
1    public bool IsHatOwned(int index) {
2    // If int bit is set, hat owned
3        return (state.hatOwned & (1 << index)) != 0; }
```

Purchase a shop item (similar for Trail and Hat).

```
1    public bool BuyColour(int index, int price) {
2        var HasEnoughCoins = state.coins >= price;
3
4        if (HasEnoughCoins) { state.coins -= price;
5            UnlockColour(index); Save();
6            return true; }
7        else return false; }
```

Bitwise operations used to unlock items (no function to lock).

```
1    public void UnlockTrail(int index)
2    { // | = toggle on, ^ = toggle off
3        state.trailOwned |= 1 << index; }
```

Handy operation to reset save state, only for debug.

```
1    public void ResetSave() {
2        PlayerPrefs.DeleteKey("Save"); }
```

The SaveState.cs file is simply a collection of declared integers manipulated by the *SaveManager*, condensed below.

```
1  public class SaveState {
2      public int coins = 100; public int completedLevel =
           0; // Will be used for level progression
3      // Owned flags
4      public int colourOwned = 0; public int hatOwned = 0;
           public int trailOwned = 0;
5      // Equipped flags
6      public int activeColour = 0; public int activeHat =
           0; public int activeTrail = 0; }
```

E.4 LevelContent.cs

This class acts as a parent to all specific level classes, and contains all common functions. Following are some more interesting code snippets.

An example of a counter update.

```
public void UpdateCoinCounter(int x)
{ // x = counter change (can be negative)
    pickupCount+=x;
    coinstxt.text = "Coins: " + pickupCount.ToString();
    UpdateTotalCoins(); }
```

Player can skip holes. Demonstration of *SaveState* coin handling.

```
public void SkipHole() {
    if (SaveManager.Instance.state.coins >= currentHole)
       { // Check balance first (price = hole number)
        player.isSkippingHole = true;
        StartCoroutine(ShowPopup("Skipped hole (-"+
           currentHole+" coins)", 1.5f));
        UpdateCoinCounter(-currentHole); strokeCount=-1;
        UpdateCurrHole(); UpdateStrokeCount(); }
    else { player.isSkippingHole = false;
        StartCoroutine(ShowPopup("Not enough coins to
           skip!", 1.5f)); }   }
```

The concurrent popup function used for all notifications.

```
public IEnumerator ShowPopup(string msg, float time)
{ // Example: StartCoroutine(ShowPopup("Win", 3f));
    ShowPopup(msg);
    float currCountdownValue = time;
    while (currCountdownValue > 0) {
        yield return new WaitForSeconds(1.0f);
        currCountdownValue--; }
    HidePopup(); }   }
```

Demonstration of how the player is moved to another hole.

```
1 public void SetNewPosition(float x, float y, float z)
2 { // Used for teleports (eg next hole)
3     player.CancelMomentum(); // Avoid Portal effect
4     player.transform.position = new Vector3(x, y, z); }
```

E.5 Woods.cs

The *Woods* class is an example of a level-specific *LevelContent* child which inherits parent methods and builds upon it with specific content (such as spawn coordinates), frame update instructions, etc. Entire class as follows.

Level initialisers and common update functions.

```
1  using System.Collections;
2  using UnityEngine;
3
4  public class Woods : LevelContent {
5
6      private void Start() {
7          initCounters(); HidePopup();
8          StartCoroutine(SpawnAtNewHole(1)); }
9
10     private void Update() {
11         DetectBreathTrigger();
12         if (player.isAtEndpoint) GoToNextHole();
13         if (player.isSkippingHole) { player.
               isSkippingHole = false; StartCoroutine(
               SpawnAtNewHole(currentHole)); }
14         if (player.isInDeathzone) StartCoroutine(
               SpawnAtNewHole(currentHole)); }
15 [...]
```

Demonstration of how the player is moved to another hole.

```
1  public void GoToNextHole() {
2      EndpointReached(); // Pause before respawning
3      StartCoroutine(SpawnAtNewHole(currentHole)); }
4
5  private IEnumerator SpawnAtNewHole(int holeNumber) {
6      player.isInDeathzone = false; // Used when player was
           out of bounds and needed reset
7
8      float currCountdownValue = 1f;
9      while (currCountdownValue > 0) { yield return new
          WaitForSeconds(1.0f); currCountdownValue--; }
10
11     // Only section needing editing for other levels
12     switch (holeNumber) {
13         case 1:
14             SetNewPosition(-0.7f, 1f, -7f);
15             SetPar(2); break;
16 [...]
17         case 16:
18             SetNewPosition(79f, 9f, -125f);
19             SetPar(7); break;
20         default: Debug.Log("This shouldn't happen");
              StartCoroutine(BackToMainMenu()); break; }
21     }
22 }
```