Faculdade de Engenharia da Universidade do Porto

FEUP

ESTIMATING PLAYER PERFORMANCE AND ADAPTIVITY IN EXERGAMES AND LOCATION-BASED GAMES

João Tiago Pinheiro Neto Jacob

PHD THESIS

Programa Doutoral em Engenharia Informática

Supervisors: Prof. António Coelho

Prof. Rui Rodrigues

ABSTRACT

Exergames (games that promote some form of physical exercise in the gameplay) and location-based games (games that take into account the players' physical location) require obtaining or computing information regarding the players' physical activity and current location and context. Additionally, ensuring that the players are assigned challenges that are adequate to their physical ability, safe and adapted for the current context (both physical and spatial) is also important, as it can improve both the gaming experience and the outcomes of the exercise. So, the relevant information gathering of both player and context proves to be useful in determining the performance of the former in the latter. There are some models that conciliate the challenge's difficulty and the player's skill, for both regular games and exergames, such as the Game Flow Model or the Dual Flow Model. There is, however, no such model, broad enough so it can also be used in location-based games.

Such limitations called for a unified solution, and hence an innovative approach is proposed in this thesis, the SPaCiaL model. It takes into account the player's skill, physical condition, geo-temporal location and game challenge so as to optimize the gaming experience. Several components were developed (GeoStream, Grappher and GeoSensors) to provide information from different sources and a basis for developers to create the adaptive semantics for their games. Two case studies (GhostStand and GhostChase) were developed and integrated with the SPaCiaL framework tools to evaluate the potential of this solution. This allowed for the development and testing of different adaptivity profiles. Experiments show that the SPaCiaL framework is capable of adapting exergames and location-based games to the player's physical condition, geo-temporal location, difficulty and skill.

RESUMO

Exergames (jogos que incentivam e levam à prática de exercício físico de forma a serem jogados) e jogos baseados na localização (jogos que têm em consideração a posição física do jogador no mundo) requerem a obtenção ou computação de diferentes fontes de informação relativas aos contextos físicos do jogador e da sua localização. Adicionalmente, assegurar a segurança e adequabilidade dos desafios propostos aos jogadores e da sua adaptação aos atuais contextos (físicos e espaciais) e à habilidade do jogador é também de interesse, uma vez que pode melhorar a experiência de jogo e os resultados do exercício físico. Assim, a recolha de informação relevante relativa a esses contextos é necessária para determinar a performance do jogador no desafio proposto. Existem alguns modelos que procuram conciliar a dificuldade de um desafio com a habilidade do jogador ou até a sua capacidade física, como o modelo de "Game Flow" e o modelo de "Dual Flow". Não existe, no entanto, um modelo suficientemente abrangente que possa ser usado para jogos baseados na localização.

Estas limitações podem ser colmatadas por uma solução unificada. Como tal, uma nova abordagem é proposta nesta tese, sob a forma do modelo SpaCiaL. O modelo SPaCiaL, solução proposta para ultrapassar as limitações acima descritas, foi desenvolvido, implementado e testado. A framework desenvolvida com base neste modelo tem em consideração a capacidade física e habilidade de um jogador, bem como a dificuldade do jogo e a sua localização geo-temporal, com vista a otimizar a experiência de jogo. Várias componentes foram desenvolvidas (GeoStream, Grappher e GeoSensors) para fornecerem informação e bases para que programadores possam mapear os conceitos e as suas relações pertinentes para um jogo, criando perfis adaptativos. Dois casos de estudo (GhostChase e GhostStand) foram desenvolvidos e integrados com essas componentes da "framework" SpaCiaL para avaliar o potencial desta mesma solução. Resultados experimentais sugerem que a "framework" SPaCiaL é capaz de adaptar exergames e jogos baseados na localização à habilidade e condição física de um jogador, bem como à dificuldade de jogo e contexto espacial.

CONTENT

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

There are many games that require outdoor/indoor physical activity for the goals to be achieved, like indoor console exergames, mobile exergames and virtual reality games. Even though there are several solutions for capturing the current user-related bio-information, such as the Vital Jacket[20], and for obtaining the current context data (such as activity detection, ambient temperature, weather, world position, etc.) there is no solution that aggregates the data and information obtained from different sources with the goal of adapting the user to the game's challenges and vice-versa.

Assessment of the player's performance is, therefore, a problem. It can be analysed by fellow players, the game's system (to some extent), or it can be based upon the analysis of game reports or the completion of tasks. However, these methods often rely on incomplete information or a very subjective human analysis.

Considering today's widespread usage of mobile devices, such as tablets, laptops and smartphones, it is possible to design a truly mobile and generic approach to the problem of physical performance, player skill and local context analysis for adequate player-challenge matching. The framework proposed in this thesis - SPaCiaL - aims to solve a currently unsolved problem that exists in such games (location-based exergames): the need to know how a player is performing, both physically and in a game. The current game's real world location is also relevant to the difficulty of the current game's challenge, and can be determined through the usage of ubiquitous computing principles (an application's ability of discovering and capturing context information passively) and unobtrusively (by not being an impediment to the completion of context-specific tasks). This approach can be used to regulate the user's performance (demanding more or less in the game, whenever needed), and can also be used for parametrically generating challenges that are constantly adequate to the user's current physical performance, skill and location.

This chapter focuses on:

- Presenting the motivation behind this work,
- The thesis statement and hypothesis,

- Research questions relevant to this work,
- The goals that are to be attained by the end of this project,
- The publications associated to this work,
- The structure of the rest of this document.

5

1.1 MOTIVATION

Assessing physical performance and adaptivity of a player's game play session in a variable environment is needed and required in exergames or location-based games, so as to guarantee the safety and exequibility of the proposed game's challenges. Such an approach can help in guaranteeing the adequacy of the game's content (such as exercises, goals, challenges, difficulty levels, assets, etc.) that are both attainable and challenging for a user - a feature most useful for training and exergames scenarios.

The main motivation of this work is to give a contribution to help developers to more easily and precisely assess user and context related data and deciding how that information should be processed and integrated with a game's mechanics, with particular focus on the user's performance and adaptivity of the challenge in exergames, and how it can be used by said games. This is achieved by collecting sensor-captured information, for both the current context and the player, remotely-stored information, regarding player historical performance, both absolute and relative, and georeferenced information. This methodology is potentially useful in other, related scenarios, such as military training, physical ability and real-time cooperative environments as long as they share transversal traits with exergames or location-based games.

In terms of scientific contribution, this project also aims to approach some limitations of current adaptivity models, computation/ wearable computing issues as well as investigate the unexplored area of mobile adaptive content generation, applied to wearable computing systems. The focus of this thesis is the gathering of context and player-related relevant information through the mobile devices' sensors as ubiquitously and unobtrusively as possible, as well as generating adequate game content (such as challenges or game mechanics). The retrieved information is also to be used to assess performance of the player (so that an adaptive system may know what parts of the challenge the player is exceling/underperforming at). This knowledge can also be valuable for other scenarios where user and task adaptivity is important, for better fitting the capabilities and objectives of the user, having in mind the current surroundings and user conditions. However, the focus will be player adaptivity in mobile exergames.

1.2 THESIS STATEMENT

Based on the motivation behind this work, the thesis statement is the following:

"It is possible to establish a feedback/feedforward loop between a location-based exergame's challenge and the player's skill, physical ability, location and context, in order to enhance the game experience"

Current player context can be determined by combining the information gathered based on his/her current location (position, surrounding structures, weather, daytime), the current challenge and player skill or physical ability (if the challenge involves jumping, and the player is jumping, then he/she is probably aiming to achieve the given goal) and his/her biometric data (heart rate, effort, energy expenditure, pace, speed). Further information can be extrapolated by comparing the player's current context to previous attempts by the same player or even other players in similar contexts.

1.3 RESEARCH QUESTIONS

From the thesis statement, several research questions were identified:

- **RQ1: Is it possible to standardize a source of information to be used in a game?**

  The usage of several sources of information is the basis for a context-aware game. However, since there are a multitude of sources and types of data this often makes their usage in a context-aware game difficult. If it were possible to formalize what these sources could be, the addition of new ones to a game would greatly ease the process of testing and developing context-aware games.

- **RQ2: How can information relevant to a POI (point of interest) be inferred from different sources of data?**

  Several sources of geographical information exist (ArcGIS, Google Places, OpenStreetMap, etc) that provide information about POIs. However, some information is available in some providers but not in others, or it is available but contradictory. A possible means of gathering and condensing information from multiple sources and merging it would allow for location-based games to tap into several providers, potentially extending the detail and precision of the information gathered.

- **RQ3: How can a game flow theory model be applied to location-based exergames?**

  Flow theory relates a player's skill and the game's challenge with one another. If both are in a state of equilibrium (the player's skill is adequate for tackling the challenges at hand), the player is considered to be in a state of "flow". However, location-based exergames possess other variables than just those two, and, as such, the flow theory may prove inadequate in determining if the player is in a state of flow.

- **RQ4: How can geo-information be used as an input for a location-based game?**

  Location-based games are known for relying on sensor input (GPS) and geoinformation data (POIs, Maps, etc). However, there is a need for location-based games to be able to easily pair game mechanics with geo-information, as it can ease the process of developing and testing this type of games.

- **RQ5: Is it possible to associate information gathered from different sources to game mechanics?**

  When developing a game that is latter to be adapted, there are already well defined game mechanics the developer can use to adapt a game's challenge difficulty. If the developer were capable of defining the relationship between certain types of information and available game mechanics, it could be possible for the developer to describe an adaptivity semantics if not explicitly, implicitly.

- **RQ6: Is it possible to define a common game adaptivity semantics for both location-based games and exergames?**

  Although some location-based games can be argued to be exergames, both genres have different mechanics and sources of information at their disposal. However, if a single model or framework could satiate both game types' needs of creating adaptive experiences, it could further bridge the development of adaptive experiences in both genres.

- **RQ7: Do players perceive the differences in adaptive exergames when compared to non-adaptive ones?**

  Player adaptivity has the role of guaranteeing that the game experience is custom tailored to fit the player. If the player's experience isn't enhanced via adaptivity mechanics, then either the identified variables proved to be insufficient or the changes were subtle enough that the player couldn't identify them. Both direct and indirect measurements can help determine if a game's adaptivity mechanics had any impact towards the game experience.

- **RQ8: Can player safety in exergames be reinforced through adaptivity?**

  Another identified issue with location-based exergames is, since they are often played by interacting with and moving around the real world, additional threats to the players' wellbeing may exist. Since adaptivity aims to improve a player's game experience, it is possible that it could be used to improve the safety of the player in location-based games and exergames

1.4 OBJECTIVES

This PhD thesis's general objective is to research, develop and test a solution to the problem of "Estimating player performance and adaptivity in exergames and location-based games", meaning that the ultimate goal is to design a solution for easing the information gathering of both the players' physical efforts and the current context, to help determining the adaptivity and performance of the user, through the usage of mobile devices and relevant sensors. This information will be subsequently used in exergames for adaptative content generation, ensuring a custom, player-tailored experience.

More specifically, this thesis will focus on:

- Creating a transversal model for generic exergames and physical activity scenarios, through the identification of common elements and relevant information collected by users in different contexts.
- Identifying which information can be captured live and automatically from the players' system, what knowledge can be retrieved from it and cross it with the common elements and relevant information identified previously.
- Researching and creating performance evaluation heuristics.
- Designing a solution that would allow the users to link these sources of context-relevant data to context-adequate challenges or goals.
- Implementing a prototype framework based on the designed solution, that could be incorporated in exergames previously developed without that adaptive framework in mind.
- Evaluating the exergames, and determining if the games would need to adapt to better fit the players' current context (player's skill, physical ability, location context and game's challenge)
- Evaluating the framework integration with said games, by assessing the performance of the prototypes, and the designed model, through real-life tests, determining if this solution allows for clear and expressive creation of adaptation semantics, adapting the game to the current context.

1.5 PUBLICATIONS

Some parts of the work here presented have already been published in conferences, journals and in M. Sc. Thesis. Their contribution to this work is the following:

- Jacob, J., Coelho A. (2011). Issues in the Development of Location-Based Games. *International Journal of Computer Games Technology*, *2011*, 1-7. Doi: 10.1155/2011/495437. – This article identifies many of the issues that location-based games and exergames have, and suggests some solutions. As such, it was one of the main contribution for the work of this thesis. Some of these solutions will be further explored in this work.
- Jacob J., Coelho A. (2011). Geo Wars–The development of a location-based game. *Revista Prisma. Com*, (14), 1-13. – This work describes the development of a location-based mobile game that dynamically changes the level design to better suit the user. It depicts the importance of user-centred content generation and lays the foundation work for adaptivity in location-based exergames.
- Jacob J. (2011) A Mobile Location-Based Game Framework. *DSIE'11 Proceedings.*- This work details the location-based game framework that was developed for the creation of *Geo Wars*.
- Fernandes T., Jacob J. (2011). Virtual Location-Based Indoor Guide. *SGDA'11*.- In this work the authors have explored augmented virtuality in mobile devices, in the context of museum guides. It also shows the difficulty of accurately tracking a user's indoor position with current mobile devices.
- Jacob J. (2010) Location-Based Digital Game. *M.S. thesis, UP, FEUP, 2010* – This master thesis revolves around the game design specific requirements of creating a location-based game. It culminated with the creation of *Geo Wars* and the location-based game framework that was the basis for its creation.
- Jacob J., Da Silva H., Coelho A., Rodrigues R.,(2012) Towards Location-based Augmented Reality games. *Procedia Computer Science 15, 318-319* – The authors, in this work, focused on creating a multiplayer location-based augmented reality game, *wARms*. This game was created as a means to portray the benefits of allying augmented reality with location-based games.

- Da Silva H, Jacob J., Coelho A., Rodrigues R. (2012) 2C - Mobile Collaborative Fire Hazard Detection System . *CONVR 2012*- In this paper, the authors developed a mobile collaborative system for the detection and marking of fires. It shows that location-enabled application can benefit from the technologies and techniques used in location-based games, and vice-versa, as well as a mobile device's wide array of sensors.
- Meleiro P., Rodrigues R., Jacob, J., Marques, T. (2014) – Natural User Interfaces in the Motor Development of Disabled Children *Proceedings of Slactions 2014. Procedia Technology.* The development of a framework for the creation of exergames adequate for children with different disabilities showed that even within the same disability, children have different physical abilities. As such, even games designed with a specific ailment in mind will be player by children with different levels of physical capabilities and skill. Adaptivity in this games can help ensuring that the exercises have a therapeutic effect and are not harmful.
- Gonçalves, J., Rossetti, R., Jacob, J., Gonçalves, G., Olaverri-Monreal, C., Coelho, A. (2014) – Testing Advanced Driver Assistance Systems with a serious-game- based human factors analysis suite. *IEEE Intelligent Vehicles Symposium (IV).* The usage of GeoStream, a component of this thesis, for gathering location-relevant geographical information in driving assistance shows how this information and content generation can be used beyond location-based games and is precise enough for driving simulators.

1.6 STRUCTURE OF THE DOCUMENT

This thesis is composed by nine chapters, the first being this one, the introduction.

The second chapter presents a revision of the main literature on several areas deemed to be relevant for this work, with particular focus on human motion analysis/gait analysis, serious games, location-based services and games, player adaptivity, ubiquitous computing and exergames.

The third chapter proposes a new model (The SPaCiaL model) for accomplishing the thesis's goals. It also describes how this model differs from the current ones and what are its sources of data. Furthermore, it details the implementation of the model as a framework, the SPaCiaL framework, meant for designing adaptive location-based exergames, detailing the used technologies, components and deviations from the original design.

In the fourth, fifth, and sixth chapters are descriptions and implementation details of the GeoStream, GeoSensors and Grappher modules of the framework, respectively.

The seventh chapter presents the case studies: the two mobile exergames that were chosen to showcase the SPaCiaL framework

The eight chapter contains the experimental design of the tests as means of validating the SPaCiaL framework, through the previously presented case studies, and the results of those experiments.

The ninth chapter summarizes the findings of this thesis, its conclusions and the future work.

The annexes of this thesis contain detailed information of the SPaCiaL Game Data Sevice, Grappher nodes, raw experimental data and the informed consent form.

2 STATE OF THE ART

As this thesis's work focuses on delivering better tailored and more effective experiences in the context of physical training or exergaming, several areas of research were needed to be investigated. As such, this work explored the areas of serious games, location-based games, exergames and other close areas of knowledge, such as augmented reality, expended energy estimation, ubiquitous computing, among others.

This chapter is divided into the background section, where the most relevant areas, technologies and methodologies will be presented, and the sections of each area of interest for this thesis, containing relevant projects and detail of the respective, current state of the art.

2.1 BACKGROUND

Several areas have been identified as being useful in providing up-to-date application of current technology, practices, methodologies and techniques for this thesis. This chapter discusses the state-of-the-art for each of these areas - serious games, location-based services, collaborative systems, ubiquitous computing and wearable computing - while also providing insight as to how these areas may benefit the proposed work. At the end of each area's literature review it is explained the contribution the area gives to this work.

2.2 CONTEXT-AWARE GAMES

Context-aware games can be defined, by extension of the definition of context-aware applications made by Salber et al. [57], as games that make use of environmental information to improve user interaction or the game's mechanics. These games are usually available as location-based games, augmented-reality games, or a mix of both. This work will focus on location-based games.

2.2.1 LOCATION-BASED GAMES

Even though recent, location-based games have already become popular. Location-based games, games that use the player's physical location as input or for accessing or generating information are relatively recent. The first commercial game (2002) was Botfighters [64], a pay-per-locate game. Each player had to destroy enemy robots (other players) earning credits for each kill. These games have since then gained a considerable amount of popularity. For instance, Geocaching [50] is one such popular location-based game, with over one million of users. The idea behind that game is treasure hunting: a player physically stashes a cache somewhere in the world and creates a puzzle that other players must solve to get to the physical location of that cache. This location-based game does not require a mobile device to play. Some of these location-based games can also be considered exergames, games that require the user to perform physical activity as a means of interacting with the game. Games such as "Zombies, Run!" [23], a jogging- while-escaping-from-zombies exergame, or "Geo Wars" [34] a location-based strategy exergame, which promotes and rewards the player for performing physical activity in his/her surrounding environments.

Older, popular games such as Geocaching, from 2000, or Pac-Manhattan [42] also share some characteristics with exergames, as they require users to explore outdoor environments, by foot, in order to play.

Location-based games and exergames can benefit from analysing player performance and adaptivity of the proposed challenges. This way, levels and quests can be generated or adapted to better suit that particular player or the current context (both physical and spatial).

2.3 EXERGAMES

Exergames, video games that require physical exercise from the user as means of input, have been popularized by the Nintendo Wii and PlayStation Eye. However, exergaming can be traced back to the 1980's, through the Nintendo accessory, the PowerPad, and was also present in the 1990's via Konami's DDR (Dance Dance Revolution) [71]. Studies have shown how these games are more physically intensive than regular videogames and positively impact the user's health [63,74]. Although many of these games focus on being fun, mainly those sold by the entertainment industry, there are many with other goals, such as rehabilitation and health maintenance. These are called serious games, as they are games that do not focus on entertainment

2.3.1 SERIOUS EXERGAMES

Serious exergames are used, primarily as training, maintenance or rehabilitation tools [71].

As rehabilitation tools, thanks to the availability of the Nintendo Wii and Wii Fit, a large number of rehabilitation games appeared. These studies have shown, from an early start, that these games could be useful in therapeutic scenarios [65], and some even in how they could be useful in the areas of prognosis and treatment of physical disabilities [47] such as Cerebral Palsy [32] and Parkinson [7,62] . Even in scenarios where the goal is to treat psychological ailments, exergames show their potential [53].

Training exergame-based tools, such as the Online-Gym show the feasibility [16] of creating a virtual environment where physically remote people may interact and perform exercises virtually side by side. Balance and fall-prevention exergames have also shown positive results. A study of static balance assessment in patients suffering from chronic stroke showed that participants that used the Wii-Fit performed better than those training only with conventional methods, even after three months [33]. Another study with an elderly population showed how these games can potentially improve balance, particularly as they allow the patients to practice safely in their homes [2].

The potential benefits of serious exergames is widely recognized in many areas of application. These exergames could also benefit from adaptation and player specific profiles to potentially tailor and adequate their mechanics and goals to the player's physical condition and whereabouts.

2.3.2 HUMAN MOTION ANALYSIS, GAIT ANALYSIS AND EXPENDED ENERGY ESTIMATION

Human motion is the field that studies the identification, classification and measurement of the human bodies' movements. Although previously this field focused on computer vision techniques to extract human movements from video footage, thanks to the development of smaller and better sensors (heart rate, accelerometers, magnetometers, gyroscopes, etc.) it has recently began using them, so as to allow the study of the human motion with less restraints, and more reliable data.

Human gait analysis techniques can be used to extract helpful physical-related information, such as current body pose, expended energy estimation, and pace.

Bioelectrical Impedance Analysis (BIA) has also been used successfully to measure the percentage of fat, water, muscle and bone in a human body[40].

2.4 ADAPTIVITY

An adaptive game has been defined by [17] and [45], among other more recent authors, as games that recognize player input and are able to adequately change their behaviour (be it gameplay mechanics or game elements [10][44]) so as to ensure a better experience.

As such, this methodology can be applied to different games, such as exergames [59] or health games [62] [28], mobile games [14] and others, entertainment or otherwise [44].

The importance of game adaptivity in location-based games and exergames was also explained in [35], as a means to overcome the issues these types of games commonly present.

Adaptivity plays an important role in this work as it will cover the part of the solution where the game will dynamically change itself in order to better accommodate the current player's status.

2.4.1 USER PROFILING

The creation of player profiles to drive the adaptation of a game's behaviour is not recent. There is a patent dated from as early as 1997 that claims "A method of adaptive computer game play based on profiling" [8]. Furthermore, the relation between player personality types and game behaviour has also been explored with positive results, identifying personality types with actions in games [66] and vice versa, presenting a relationship between demographics, game preferences and in-game behaviour [41].

Player-centred game-design also benefits by the ability of correctly profiling users and their interests [17], be it to dynamically adapt the game to the player's preferences or to create a game that appeases to the tastes of a generic user profile. User profiling can be used as a basis for adaptivity, by matching a player's behaviour to the closest profile that fits, and treating the player as having that profile and game preferences.

User profiling can contribute towards player adaptivity, as it helps identifying the type of player and adequate challenges to them. Since location-based exergames have several dynamic game conditions, such as the player's physical state or the location the game is being played at, these variables, that may better define the context for adaptivity purposes, aren't considered when adapting a game solely through user profiling techniques.

2.5 CONTEXT-AWARE EXERGAMES

Context-aware exergames are defined by being exergames that have some awareness, via sensors or remote services, of data or information that is relevant to their goals and challenges [31], being similar to a context-aware application or service in that regard [57].

Unlike many exergames, these context-aware games monitor services and sensors to ensure the player is being challenged in a way that blends with the current context. Usually through GPS, heart rate and accelerometer sensors, the game can be aware of the player's physical condition. Mobile phone's camera and location services APIs can provide the application with some information regarding the surroundings of the player [14,27].

This context awareness can be used to not only improve the gaming experience itself [59], but the quality of measurements, such as more accurately calculating the expended energy in a gaming session [49].

The players of these games could potentially benefit from adaptivity. However, these games are currently very limited as to how aware they are of the player's surroundings and how they can take into account the geo-information and adapt the game's challenge and goals with it.

2.6 ENABLING TECHNOLOGIES

This section presents several areas of knowledge that contain technologies relevant to this thesis. These are used in varying degrees to support the work needed to answer the research question, although they do not provide direct answers to them.

2.6.1 LOCATION-BASED SERVICES

The expression "Location-based services" (LBS) refers to the use of the geographical position of a mobile device, such as a ***Smartphone*** or ***Personal Digital Assistant*** (PDA), to provide services based on that information. These services are of the utmost importance for location-based games as they are responsible for providing geo-information to the game. These services include but are not limited to:

- Navigation assistance,
- Localization in case of emergency or disaster relief,
- Social networking through finding friends,
- Presenting points of interest,
- Content-aware entertainment (such as location-based games)**,**
- Advertising discounts in nearby shops via ***SMS***.

Provision of location-based services requires a sufficiently accurate positioning technology, a geographical information system which maps the surrounding areas, and the service itself.

2.6.1.1 OUTDOOR LOCATION

Today, there are several methods for locating a device. These methods depend on the technology available on the country the device is at, the hardware of the device, and the locating service itself. These methods are suited also for distinct types of location, outdoor location being among them. [68] Outdoor location refers to localization techniques that work properly in outdoor environments. These methods are commonly used in location based games, making the game's challenges differ from place to place around the world [35].

GNSS LOCALIZATION

One such solution for outdoor localization is through a network of satellites that orbit the earth and communicate with a receiving device. The location is determined by comparing the different times that the signals from distinct satellites take to reach the ***GNSS (Global Navigation Satellite System)*** receiver (Figure 1). Although being a popular approach used in solving location issues, mainly due to its great accuracy (ranging between a couple of meters to 20 meters) it has several drawbacks:

- Requires the device to "warm-up", by locating the satellites available (this can range from a few seconds, to nearly half-an-hour). This issue has been minimized thanks to several techniques, like pre-downloading information about which satellites are available and when.
- Requires the device to have a rather direct line of sight with the sky. In older ***GNSS*** devices it was usual to lose the signal completely on rainy/cloudy days.
- Requires extra hardware, meaning extra cost and extra battery consumption.
- "Canyon effect", an effect where visibility of the satellites is intermittent, due to tall buildings or landscape narrowing the sky view [68].

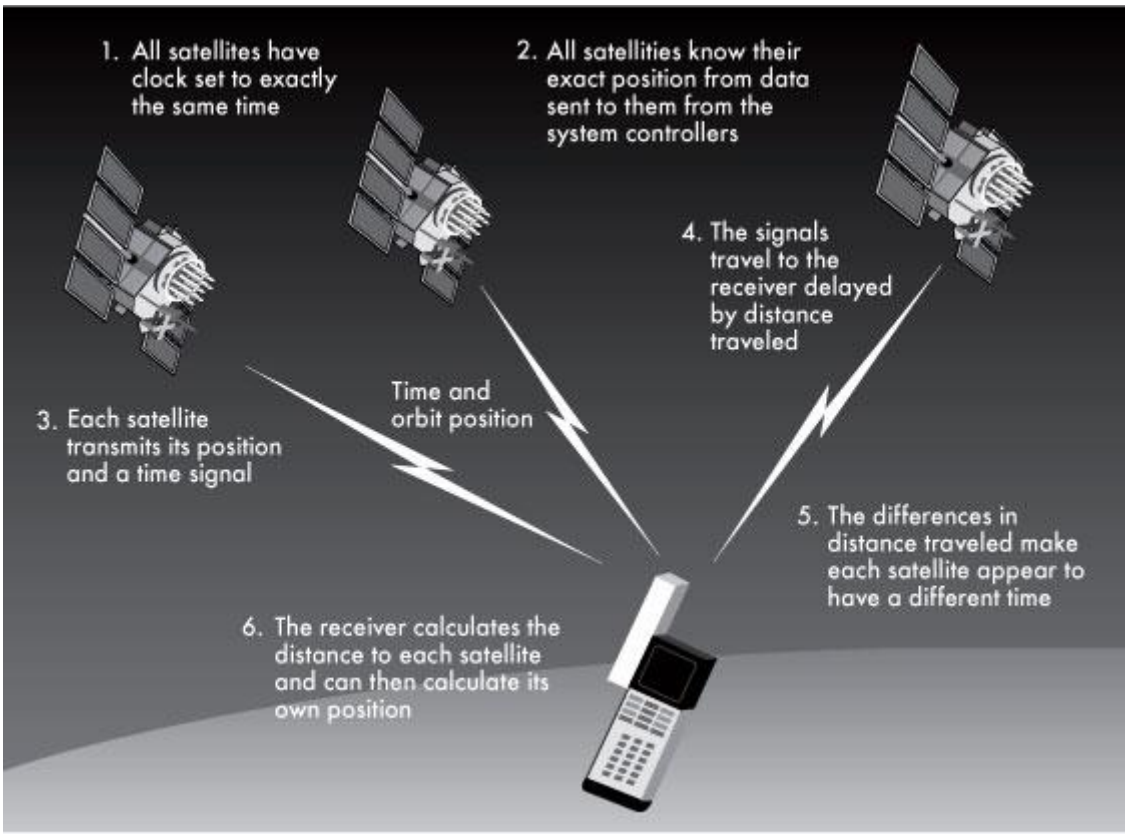


FIGURE 1 – IMAGE DEPICTING HOW LOCATION IS DETERMINED [19]

USING THE MOBILE PHONE NETWORK

The mobile phone can identify the ***Base Transceiver Station (or BTS)*** that the device is using, and knowing that station's position one can, albeit with some inaccuracy, determine the mobile phone's position. A ***GSM*** cell phone can be anywhere from 2 to 20 kilometres radius around the ***BTS***. Some techniques based on this method have a better accuracy, within 150 meters [68].

SHORT-RANGE POSITIONING BEACONS (LPS)

Using a triangulation technique like the one used in GPS but using multiple ***Wi-Fi*** spots or ***Bluetooth*** a device's location may be more accurately determined, (although not necessarily with a greater precision when comparing with the results GPS localization wields). Being cheaper and more accessible it is a popular choice. An example would be ***Google Wi-Fi***, a free wireless internet service offered to the city of ***Mountain View***. Using this service, people allow themselves to be located and to receive location-based advertisement. However, these solutions are more popular for indoor localization.

2.6.1.2 INDOOR LOCATION

Some work has been developed for indoor-location, using several distinct technologies. Veljo Otsasson et al. [51] conceived a system that was able to provide user position in indoor environments using GSM triangulation. The idea behind this project is to use wide fingerprinting that uses GSM cells that are strong enough to be detected but too weak to be used in communication, in addition to the six cells defined in the GSM standards. This system has many advantages such as the range of signal coverage, the fact that any mobile phone could be used for positioning and that the system would be highly tolerant to power shortages. In order to be able to detect the user position accurately, this system requires some calibration that was performed by measuring both the 802.11 and the GSM signals in each division of the tested areas. By using the proposed algorithms that held the best results, this solution was able of reporting the user location with a median localization error between 2.5 and 5.4 meters.

On a different perspective, the Cricket project [54] uses both Radio-Frequency (RF) and ultrasound signals to identify a user's position. The utilization of both sensors is based on the fact that RF propagates in non-linear and possibly unpredictable ways inside buildings. Therefore, it was necessary to consider alternative ways of providing increased precision to the position calculation. So, to perform the calculations, the beacons send concomitantly RF and ultrasonic signals. As the speed of sound is smaller than the transmission speed of RF signals, the later will arrive sooner to the listeners. When a listener receives a RF, it uses the first bits as training information, enables the ultrasonic receiver and waits for the ultrasound emitted by the beacon. The calculation is then performed by using both the strength of the RF signal and the time difference between the arrivals of each signal. One of the great advantages of this project is the low cost that is required to buy all of the components. The error rate reported for mobile devices is however, somewhat big, being approximately within one meter.

HP also developed a solution that uses infrared beacons instead of ultrasound and typical RF emitters, called HP Cooltown [39]. To find its location, the user must point its infrared-enabled handheld device to the infrared beacons. This has the clear problem of requiring user interaction to work, but on the other hand, this method also protects the user's privacy, since he/she only interacts with the system when he/she really wants to.

Finally, F. J. González-Castaño and J. Garcia-Reinoso [3] [29] developed a system that attempts to provide user location in indoor environments using only Bluetooth devices. This proposal uses a network of Bluetooth devices, organized in hexagonal grids. Each node is either a slave or a master node. The user is equipped with a Bluetooth-enabled device (such as a mobile phone) or Bluetooth badges and broadcasts its address to the nodes. Every slave node receives the RSSI value from the user and sends it to the master node. The master node performs every calculation to triangulate the user position based on the RSSI values that were received as well as the slave nodes positions, and sends the computed data to some servers that will use that information for some service. This approach is very expansible since the system is able of auto-configuring itself automatically. Also, there are no collisions with other existing devices, because the work is centralized on the slave and master nodes which conform to a specific protocol. However, given that all the calculations are done by the master nodes, the system may quickly become overloaded, which brings performance problems in terms of response times, depending on the number of position calculations and Bluetooth devices in the network.

Location-based services are an important area of research for the proposed work, since the latter will require the usage of these services in order to further gather information for the user's current location. These services can vary, from maps, to weather, news and others deemed necessary during the design of the solution. Although this work does not aim to innovate in the area of location-based services itself, it is our belief that the usage of these location-based services in the context of this work is innovative, and is crucial for accessing remote information useful in certain collaborative contexts.

The distinct localization sensors coupled with the location-based services can provide information about the user's current context (such as current weather, traffic, etc) that can be useful in helping assess the challenge that the current context poses for the user.

2.6.2 COLLABORATIVE SYSTEMS

Collaborative systems are the systems that support CSCW (Computer-Supported Cooperative Work), a field that researches how collaborative activities can be enabled through computer systems [15].

This area of collaborative systems will provide knowledge of what elements of collaboration and cooperation are persistent throughout most cooperative/collaborative scenarios, which ones are more relevant for a goal to be achieved, and is thus useful for this work, specifically for the creation of collaborative gaming experiences and the creation of adaptive games collaboratively.

"Only in the second half of the 80s have collaborative systems with some kind of coordination mechanisms started to appear." [55]

Even so, only later have these solutions started to accommodate task interdependencies through Malone and Crowston [46]. They have defined coordination as being “the act of managing interdependencies between activities performed to achieve a goal”. The advantage of their presented model is that it is reusable in different coordination/cooperation scenarios since it takes into consideration the “fundamentally similar coordination phenomena”. They have identified several components of coordination such as goals, activities, actors and interdependencies. Thanks to this model, it was also possible to manage these interdependent activities through a workflow model [5].

COLLABORATIVE AUGMENTED REALITY

The usage of Augmented Reality as a means of easing the execution of collaborative tasks is not new [70] and has wielded interesting results. Nowadays, augmented reality has been gaining momentum in the real world thanks to the arrival of mobile Augmented Reality applications such as Wikitude or Layar [22], both mobile applications with great focus on user-generated content.

These solutions have proven that the ease of use of augmented reality is widely accepted by the general public, and as such means that the usage of augmented reality as a means of displaying both real and meta-information (e.g. through video Augmented reality as defined in [6]) can be of value to this thesis.

COLLABORATIVE SERIOUS GAMES

Since Serious Games´ main focus is education [4], and these games being an increasingly popular trend, growing at a steady pace, it is not surprising to see that the collaboration and cooperation nature of education has seen its way into the Serious Games trend [24], [75], [73]. These works have also taken into consideration usability, interface and user experience issues into their collaborative designs - models that will be useful in the future work of this thesis, as they are somewhat similar to Collaborative Virtual Environments, described in the next section.

COLLABORATIVE VIRTUAL ENVIRONMENTS

Collaborative Virtual Environments [9] are a mix of Virtual Reality and Computer-Supported Cooperative Work (CSCW). Some challenges were identified for this area[9]: scalability and interest management, distributed architectures, issues pertaining CSCW specifically and other 2D interfaces and new kinds of human factors.

However, these solutions, as their names imply, rely on virtual environments, meaning that the scenario, or context, is not real. This means that even though users are collaborating with one another, they are only doing so virtually. Consequently, for some dynamic contexts or situations that require real-life collaboration, these tools serve only for training, as they do not change the real world with their virtual actions nor are capable of changing along themselves with the real life context they represent. Even so, that same survey predicts that these systems will "move closer to mainstream commercialization" [9]. Additionally, it mentions the usage of ubiquitous, mobile and wearable computing as a future for these systems.

As predicted [9] [21], nowadays the gap between physical and virtual worlds has been considerably narrowed. The usage of augmented reality in collaborative systems allows for the virtual content to be more easily relatable to the real-world context [56]. If, on one hand, these augmented reality collaborative environments allow for a more realistic approach at cooperation, merging virtual and real contexts, these solutions prove to be cumbersome and expensive, as many rely on HMDs (Head Mounted Displays) and other very specific equipment [70]. However, there are recent developments in this area, such as the Oculus Rift and Google Glasses, with the promise of making this type of technology more affordable to the masses

Collaborative Environments and collaborative serious games can both serve as a context for the proposed methodology. Particularly collaborative contexts, such as firefighting, can benefit from a system that tracks personnel physical performance and affects personnel to missions / training exercises having in consideration their physical strengths and weaknesses. Game contexts that have cooperative aspects to them can also benefit from being considered a collaborative environment, where participants must match their expertise to their current context, contributing to a global goal. Additionally, the creation of adaptive profiles for a game can also be a collaborative task, with multiple game designers using a tool to concomitantly create and tweak adaptive profiles.

2.6.3 MOBILE COMPUTING

"Mobile computing consists of a paradigm in which applications can discover and take advantage of contextual information (such as user location, time of day, nearby people and devices and user activity)" [18].

Mobile computing and related sensors can be used to ubiquitously extract the current context data and user bio-information. This will be helpful in ascertaining both the user's current physical status and the current context-posed challenges.

There are already several sensors available, beyond the usual ones found in mobile phones (GPS, accelerometer, compass, magnetometer, gyro, etc), such as temperature, impedance, barometric pressure, humidity, and others that can be used to retrieve such information.

Examples of products that make use of these sensors are Sparkfun's 9DOF sensor stick [1], AmmSensors's 9DOF Bluetooth wrist-band[2], VitalJacket[3] and VariableInc's NODE sensor[4] platform, a solution that allows for modular built sensor devices.

The goals and challenges for ubiquitous computing can be summarized in these three topics [1]:

- Natural interfaces, that simplify the information exchange between humans and computers
- Context-awareness, the capability of applications to adapt their behaviour considering information retrieved from the physical surrounding environment
- Automated capture and access of live experiences

The added value of this area to collaborative virtual environments is significant, as it can serve as a way to determine the current context of the situation, using the mobile devices' array of sensors (as well as additional ones), without each user having to provide active input to the system.

1- https://www.sparkfun.com/products/10724
2- http://www.pcexpert.com.tw/ar/document/ammsensor/ammsensor.html
3- http://www.vitaljacket.com/?page_id=860
4- http://www.variableinc.com/node1

2.6.4 SERIOUS GAMES & SIMULATION

Serious Games is a growing trend in the research of applications with focus on many areas. Unlike their entertainment industry's counterparts, serious games focus on teaching skills or concepts while being played. Their growth was predicted by Michael & Chen [48] and was confirmed in this study [35]. Hence both the industry and the research communities have been focusing on developing more entertaining and serious games.

These games often rely on virtual environments and user interaction with them to get their message across. A good example of such games is VR Phobias [67], a virtual environment used with the goal of helping its users overcome anxiety disorders and phobias (most notably arachnophobia and the fear of driving), wielding interesting results (92% success rate with only 4.5% of participants dropping out). Another example that relies heavily on virtual environments would be Biohazard[11], a serious game with the goal of training firefighters in how to act during a terrorist attack. Considering that this game does rely on field-based exercises, the accuracy of, and interaction with, the virtual environment where the action takes place is very important. So, time of day, wind speed, temperature and number of victims are but a few of the variables this game has to offer regarding its environment. An issue that appeared during the development of this game was the importance that players give to small details. These details help build a true immersion and real simulation for the player.

Considering how much Serious Games have evolved in the last years, both in number and quality [4] it is reasonable to assert that realistic virtual environment design and interaction is an increasingly important aspect of serious games.

These types of games, however, have yet to broadly make use of location-based services and mobile computing (two aspects covered in this document that are known to work together [35]). However, since mobile location-based games are not only doable, but also an increasingly stronger trend [35] (thanks to the growth of smartphone usage), it is only natural for Serious Games to be able to follow.

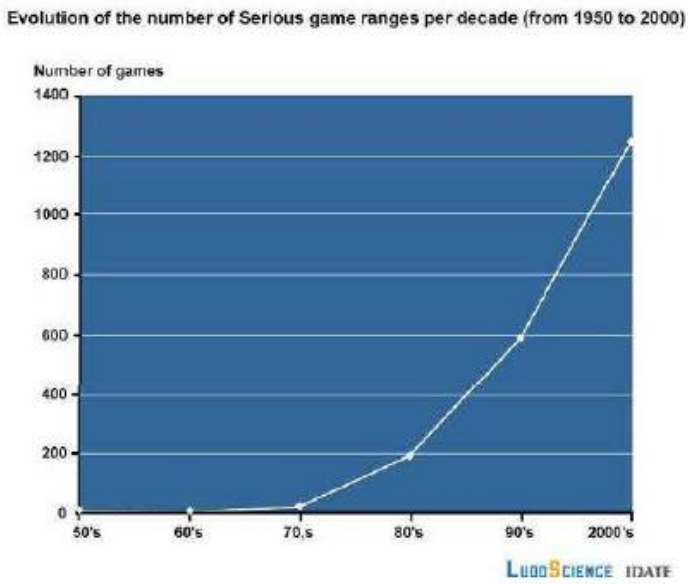


FIGURE 2 – SERIOUS GAMES GROWTH [4]

This area, considering that it has much common ground with the area of virtual environments and entertainment games, can provide useful design and interaction paradigms for the current work.

Serious games can be used in training and can benefit from procedurally generated content (such as challenges, game balancing, etc.). The rules for generating or adapting the content of these games can be created from the proposed solution. These types of games can become part of a test-bed, as they benefit from player evaluation and adaptivity, as the related work shows.

2.6.5 MOBILE AUGMENTED REALITY

Augmented Reality is a technology used by many context-aware applications and games. It superimposes virtual objects onto real world images or video. Often this requires the application to know the user's location and some characteristics of its surroundings (such as POIs, other users, etc).

Wikitude [Figure 3], a commercial mobile application, is capable of showing geo-referenced information in an augmented reality way. This content varies from YouTube videos, Tweets and Facebook publications, among others. The content showed can be selected through a list of sources, albeit almost all of them related to social networks.

FIGURE 3 - WIKITUDE APPLICATION IN USE

Similar to Wikitude, Layar [22][Figure 4] uses the concept of layers to show information related to certain contexts. However, unlike Wikitude, these contexts are not necessarily social networks. Layar allows the user to select from a list of layers the one that the user thinks is more relevant to the current situation. For example, the user can select a real estate's layer and "see" what houses are for sale in the surrounding area. Like Wikitude, the exact location of the user is not needed (although for some layers it is recommended), meaning the user can be located through GSM or WiFi triangulation. It is not possible to easily share or place content in a layer in Layar, as this requires the user to create the

content at the source where that content is being accessed. So, in the previous real estate layer, if a user would like to place a house for sale, it would be impossible to do so through the Layar application. However, the user could give that information to the respective real estate agency, with that information being then available in the Layar application (since the source for that information would be the real estate database). For more complex operations, the user would need to create the layer itself, something that requires the usage of the Layar publishing environment. This allows companies to place their own layers within the application, providing the user with distinct content, be it audio, video or 3D graphics.

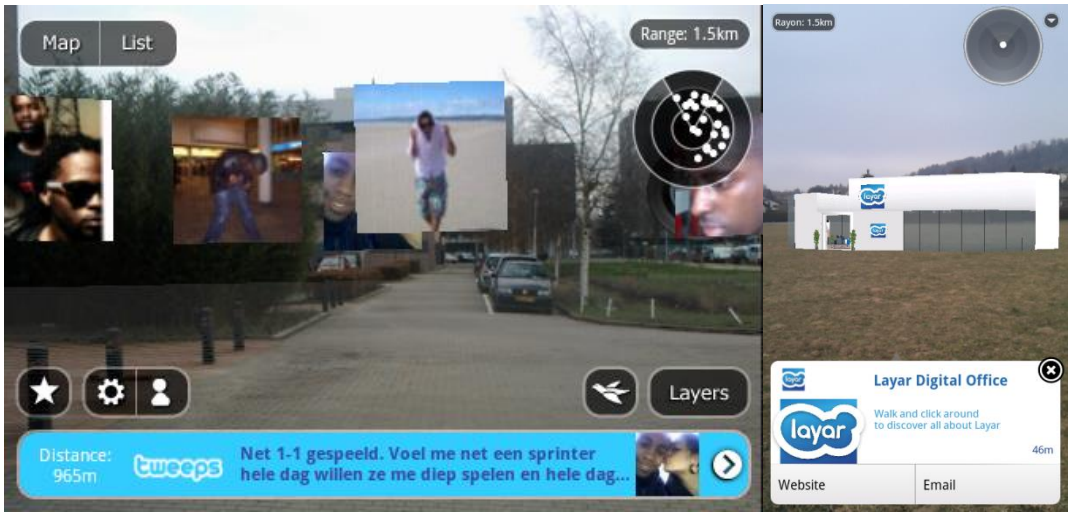


FIGURE 4 - LAYAR APPLICATION IN USE FOR 2D DATA (LEFT) AND 3D DATA (RIGHT) [22]

One limitation of both these applications (Layar and Wikitude) is that they are unable to infer the actual context of the user's situation. They rely only on localization technology in order to decide what content is shown, and the device's magnetometer so as to display the right content accordingly. Unlike them, Google Goggles [Figure 5] is capable of capturing some information on the current context aside from location and orientation. It relies on the usage of the mobile device's camera, apart from the other conventional localization technologies, in order to identify what the user is looking at. After a user snaps a photo, the application proceeds to analyse the image in search for a match in the servers' database. Although it works great for books, barcodes, text, logos and landmarks, it does not work well with furniture, people, animals or plants. The information returned from

the search depends on what was the identified content of the image. If it was a barcode, the returned information will be the product it refers to. If it is a book or painting, the information will be regarding the title and author (among other things). Additionally, Google Goggles also makes available an "Augmented Reality Mode" useful for searching for local places such as restaurants or stores. Unlike Wikitude or Layar, it is impossible for the user to add content, both through the application or through other sources.

FIGURE 5 - GOOGLE GOGGLES IN USE

Similar to Google Goggles, as it also uses the device's camera to capture information, is TAT Augmented ID [Figure 6]. This application is capable of identifying people, through facial recognition software from Polar Rose, and show information relative to that person in many social websites, such as Facebook or YouTube. It is still an unreleased application.

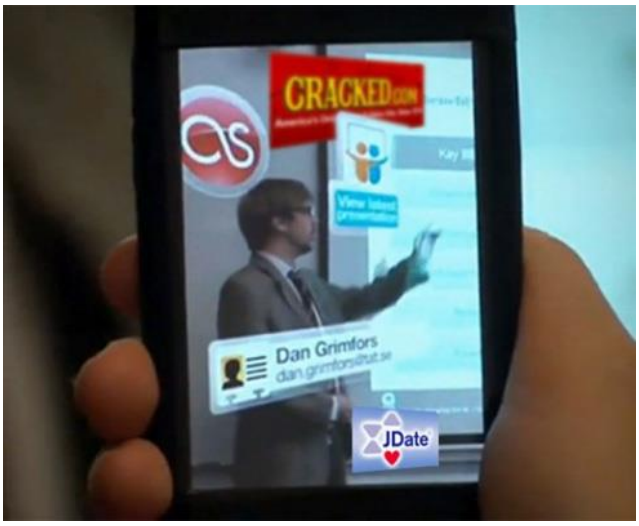


FIGURE 6 - TAT AUGMENTED ID IN USE

The issue with all the previously shown solutions is the lack of collaboration between users. In some of these applications users were able to share content with one another to some extent, but collaboration was not possible at all.

Mulloni and associates[52] developed a mobile, collaborative and location-aware game relying on augmented reality. This solution, based on Studiertube ES had the goal of letting players collaborate and compete within a virtual environment. Based on the Studiertube ES the game was able to run on mobile Windows Mobile CE devices. The collaboration in this game was made through computer-mediated interaction and through real world communication. The game consists of two teams, each in possession of a stable. In the game world there are cows that the teams must take to their respective stables. The team with the most cows rounded up in the stable wins. However, since the game requires the usage of fiducial markings, it can only be played locally, thus requiring all players to share the same space in order to play.

A handheld application, named Vidente, for civil engineering was developed by Shall and associates [58], relying heavily on augmented reality, location awareness, and infrastructural data.

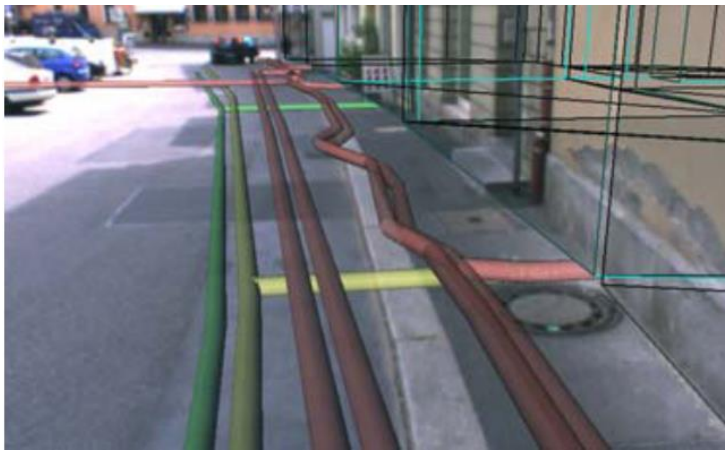


FIGURE 7 - VIDENTE SYSTEM WITH SUPERIMPOSED 3D INFRASTRUCTURE DATA [58]

As the above figure shows [Figure 7], it is capable of representing, through a 3D data overlay over video, infrastructural information useful for several tasks. The authors also mention that this solution not only allowed for an easier perception of the placement of certain infrastructures, but it also allowed for workflow optimisation. And since the information is portrayed more conveniently, it allowed for easier documentation, collaboration among workers and ultimately saving time and other resources.

There are several serious games that use augmented reality in order to achieve a greater immersion. One such example is the work of M. Juan et al [37] , that developed an application that uses augmented reality to treat phobias, most notably from spiders and cockroaches. The authors have used fiducial markers that when seen through the application spawn virtual insects on top of the camera view providing a realistic image. The results were positive with several test subjects considering that the virtual spiders were real enough and several others' anxiety levels decreased after the treatment. Another two games, ARPuzzle and ARBreakout also show the potential of augmented reality serious games in this work by Liarokapis and de Freitas [43] with very positive results, particularly ARPuzzle as it is a puzzle game that uses fiducial markers representing pieces of the puzzle (buildings of a school or campus) that must be arranged in a way similar to the way they appear in the world. ARBreakout, being an arcade classic game scored significantly less than ARPuzzle in terms of educational effectiveness.

A location-aware augmented reality game for emergency response education was developed by [72] and associates that is very relevant for the proposed thesis. The game, Rogue

Signals, relies on a human team that has to cooperate to find artefacts scattered in the real world, while avoiding virtual "predators" that roam the virtual world. The team consists of one coordinator and seekers. While seekers actively search for the artefacts, they have only a local and indirect view of the world, whereas the coordinator can observe the virtual world globally [Figure 8]. This means that while the seekers are physically able to capture the artefacts, they lack the information of the world to do so, whereas the coordinator knows that information but cannot directly act on the world, forcing them to cooperate. The authors claim that the scenario is very similar to fire emergency response teams, where each element holds a piece of the information of the whole picture.

FIGURE 8- COORDINATOR INTERFACE FOR ROGUE SIGNALS [72]

Popular, commercially available augmented reality games include Niantics' Ingress[1] and Pokémon Go[2]. These games require the player to explore their surroundings and points of interest to be played.

1- https://www.ingress.com/
2- http://www.pokemongo.com/

In ingress the player takes the role of an agent, part of a faction, that must find and identify portals, located at monuments and other POIs. These portals can be claimed by agents of a faction. If the portal belongs to a faction that is not the player's faction, it will be contested.

FIGURE 9- INGRESS PORTAL KEYS, AR ELEMENTS OF THE GAME (SOURCE: MIKEFRANCOIS.NET)

This level of immersion in Ingress made it the first widespread location-based AR game (Figure 9).

As for Pokémon Go, it takes the concept of Ingress and adapted it for GameFreak's Pokémon Universe. In it, players must physically search for Pokémon (fictional digital creatures) sightings around the world. The application shows a certain level of context awareness, as some of these creatures may only appear on grasslands, parks, water, or during the night or day. Weather also plays a role in changing the chance of encountering certain Pokémon.

FIGURE 10- POKEMON GO'S MAP VIEW AND AR COMPONENTS (SOURCE: TECHTUDO.COM)

In order to catch these creatures, the person must physically approach it and throw a Pokéball, an item to capture a Pokémon, as seen in Figure 10. Additionally, the player can also explore nearby points of interest, based on monuments, parks and other relevant points of interest in order to gain items to use in the game or challenge other player to an all-out Pokémon battle.

Augmented reality can help in displaying context sensitive information while overlaying that information over a real world feed. For the proposed work, it is a technique that should be considered as it displays context relevant information in an easy and ubiquitous way. Considering that the problem being tackled pertains to exergames/location-based games, an unobtrusive method of interacting with the game and displaying context-related information to the user will benefit from what AR/VR has to offer.

3 THE SPACIAL MODEL

The Dual Flow Model [61] is a revision of the GameFlow Model [69], particularly suited for exergames. It states that, regarding exergames, in addition to the need of being a balance between the players' skill and the game's difficulty, the player's physical condition must also be taken into account. The current issue is how can these three variables be considered when developing an exergame. The need to consider a player's location, when considering developing location-based exergames, can require the need for another variable to be taken into account: that of the adequacy of the location context, how it can be used to ensure that the player is being fairly challenged in it, or the importance of the location for both the user's performance and the game's mechanics to work. Both these models focus on the player's experience of the game, not necessarily the player's health or safety.

Since there is no current framework (at the time of the development of this work) that takes into account these variables when developing location-based exergames, the *SPaCiaL* Flow model was developed. This new model takes into account the player's skill, physical ability and condition, and the game's current challenge and location context in order to provide the player with an engaging experience.

3.1 WHAT IS THE SPACIAL MODEL?

The original Game Flow Theory [69] was revised, more recently, in the Dual Flow Model[60], dividing the player's flow state into both a game-induced state and a physical challenge conditioned state. This approach now enables games that use physical exercise as a game's input to evaluate the player's flow state. It, however, does not consider the context in which the game is being played.

This limitation can be noticed when applying this model in location-based games. As these games have another source of experience variability (the location where the game is being played at) that is not taken into account by any known Flow model, it can lead to inconsistent user experience [35]

The proposed *SPaCiaL* Flow model defines the flow of a game (F) at a particular moment of a gameplay session as being

$$F = f(S, P, C, L)$$

Where (S) is the skill of the player in playing that game, (P) is the physical exertion level and physical prowess of the player, (C) is the challenge or difficulty level that the game poses and (L) is the suitability or adequacy of where/when the game is taking place. It is also worth noting that some of these variables are co-dependent. For instance, an unsuitable location for a LBG to unfold may lead to higher levels of physical exertion than expected.

These are the definitions of the major *SPaCiaL* variables:

- **Skill** – Represents the current level of the technical ability the player has to play a given game. It often includes reflexes, adequate cognitive abilities, ability to plan ahead and fine motor skills (precision, quickness and often, hand-eye coordination, technique). It is somewhat variable from gameplay session to gameplay session, as the player's mental and emotional states change. However, a player's physical ability should not have any impact in this definition of "skill" in this context.

- **Physical Ability** – Denotes the current level of physical ability (the ability to carry out physical exercises or play sports), coordination and resistance. The player's own physical wellbeing and limitations are also part of this variable.

- **Challenge** – Represents the difficulty of overcoming the obstacles the game is currently posing to the player. The game can be challenging both in "Skill" and "Physical Ability".

- **Location** – Holds all the spatiotemporal context properties used by the game. Games that are location-based, time-based (where the date of when the game is being played is used in the game's mechanics) or both (ex: a game that uses real-time local weather as input) are considered to have a "Location" variable relevant to their players' flow state.

In games where both physical exertion and the suitability of the game's location(s) are negligible, this formula should be interpreted as that of the Game Flow theory (dependent only on (S, C)). If, however, physical exercise must be taken into account, it would assume the value of that of the Dual Flow theory (relying on (S, P, C, L)). Thus, SPaCiaL is a viable answer for **RQ3**

3.2 DETERMINING SOURCES OF (S, P, C, L)

One of the issues with the development of this framework was determining what values should be assigned to these variables. And if any of these variables is also dependent on others. The possibilities are immense and open to debate. They can also, as previously stated, carry very different weights depending on the games' design. The following table displays three distinct game scenarios where such issues are made more evident, and what game elements are representative of the values of (S, P, C, L).

TABLE 1- GAMES AND RESPECTIVE SOURCES OF (S, P, C, L)

Game Scenario	***2D Tetris Game***	***Zombies, Run!***	***GeoCaching***
(S)kill	The highscore in the game. Often dependent on the number of lines made, and time vs. number of lines. Some "luck" regarding the piece that falls may be involved.	Resource management and real-time decision making when running/jogging (to avoid the zombies) is dependent on the players' skill	Ability to solve the proposed puzzles. Number of caches discovered. Difficulty of the caches discovered.
(P)hysical Prowess	N./A.	Correlates with being able to avoid the zombies, completing the missions and gathering resources	Certain caches may be placed in location where a player has to go on foot (can be tiresome). Physical strain (running, jumping, etc) is uncommon.
(C)hallenge	The speed, variety, position of the falling pieces and starting state of the puzzle	Distance, Zombies' speed, Zombies' spawning position	Aspect of the cache. Placement of the cache. Hints given in the puzzle and the puzzle itself
(L)ocation	N./A.	Can physically strain the player (rain/wind and steep terrain). A crowded street or roadblocks can make zombie avoidance difficult or impossible	Can limit access to a cache (bad weather, landscape changes, limited road access).

The above table shows how differently some of the described variables affect the flow of the game. It also shows how these variables often encompass others, with different weights, and when correlations between variables can be expected. As such, standardization of a source of information can be difficult (as per).

TABLE 2- CORRELATIONS BETWEEN (S, P, C, L) VARIABLES

	Physical Prowess	***Challenge***	***Location***
Skill	A highly skilled player can play "intelligently", thus reducing his/her physical strain (ex.: matching the pace of the zombies in "Zombies, Run!", when to move and where in "GeoWars"). Conversely, a physically fit player, can sometimes afford to make bad decisions (low skill) and compensate through physical strength/resistance/agility (ex.: wrong turn in "Zombies, Run!", forcing the player to quickly double back. Bad tower placement in "GeoWars", forcing him to run around and avoid incoming enemies)	As the player's skills grows, the challenge must also increase to keep the player from getting bored of the game (Game Flow Theory). Likewise, as the game's difficulty increases, so must the players' skill, in order remain competitive in the game (Game Flow Theory)	A skilled player can make the most of his/her surroundings (ex: Knowing shortcuts, paths that strain him less, best locations to player the game). If the game requires from the player knowledge of the current surroundings (ex: directing the player to POIs nearby without providing directions or cues), it can be considered that the player is unskilled if he/she has not enough knowledge to complete the challenge without guidance
Physical Prowess	N./A.	A fit player may or may not require a game that pushes his/her physical prowess to its limits. If the game aims to improve the fitness of the player, it must take the current state of it into consideration. Additionally, If the game's mechanics physically strain a player (as in an exergame), the challenge level can only increase with the player's physical fitness. At its limit, the player's own wellbeing may also be it risk (risk of injury or heart failure)	A fit player can overcome challenges the physical location poses more easily, such as bad weather, steep or long paths. An adequate location (and in this context, adequate time of day and weather) can lessen the physical strain the player is subjected to. Conversely, an inadequate location can both bore or place the player's health at risk
Challenge	N./A.	N./A.	A game's challenge level, if it is a location-based game, should also take into account current physical context (An easy game "go from point A to point B" can be easy in a leveled city or very difficult in a mountainous/forested region). The location context sets the base for what the challenge should be. An "easy" location, can be coupled with a "difficult" challenge to avoid boring the player. On the other hand, a "difficult" location should be met with an easier challenge [34]

As the above chart shows, dealing with the correlation between these variables is highly dependent on the game and availability (and quality) of the contextual information. Not only can the method of determining these variables' values, set by the game designer, be open to interpretation, but also it is possible for a game to have different, redundant methods to do so. Additionally, the relation of these variables between each other may not always be linear. A game designer may want the difficulty of a game to increase linearly/exponentially with the skill of the player, or he/she may want it to have some variation (a specific polynomial function or function composition relating these two variables). Finally, the SPaCiaL model function that is able to determine if a player is in a flow state, from these given variables is not set in stone, as different games will have different weights for these variables.

The following flux chart condenses the concept.

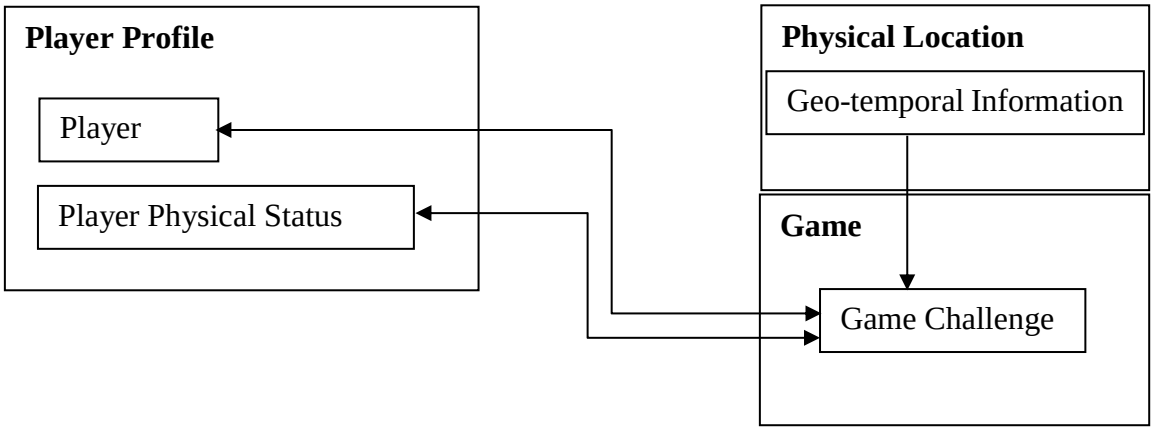


FIGURE 11- FLUXCHART DEPICTING DEPENDENCIES BETWEN SPACIAL MODEL COMPONENTS

As the flux chart shows, the four major variables, affect and are affected by each other. These can be computed from different sources of data and information:

- **Player Skill –** It is usually computed by comparing how well the player has fared at completing a certain challenge, with a baseline result. It can be determined **relatively** (what percentile is the player at, when comparing with the peers' results, or own results; the player gathered *x* more items than what was requested) or **absolutely** (player took *x* seconds to complete the task; player has 90% accuracy). Depending on the metrics used by the game, it can be computed as the challenge is being tackled or only when the challenge has ended.

- **Player Physical Status –** The depth, accuracy and information required by the game dictates what comprises the relevant player's physical status metrics. As it represents both the current physical condition of the player (tiredness, pace, intensity of exercise) and physical skill (maximum speed, strength, or other peak condition capabilities) it may require a multitude of sources. While the current condition of the player can be determined on the fly (by analyzing the player's physical activity intensity, heart rate, CO^2 concentration when expiring, etc.), physical skill can only be accurately determined over time, after the player's limits have been tested. In some scenarios, determining these limits can pose some degree of risk of self-inflicted injury (such as spraining a muscle or articulation).

- **Location Information –** This encompasses all the geo-temporal context required by the game. While some devices provide some degree of data through sensors (GPS position, relative humidity, barometric pressure, magnetic heading), some of the information has to be accessed from off-site sources (web-services, containing POIs information, maps, directions, weather, time-sensitive events descriptions) or computed (weather, inferred from the barometric pressure, or magnetic, calculated from the current time and position).

- **Game Challenge –** Considering adaptive games, the challenge itself or its difficulty can be pre-determined (by analyzing the player's skill level, for instance) prior to proposing the challenge to the player, or changed on-the-fly by analyzing the player's performance. In the case of location-based games or exergames, it should take into account the player's physical status and geo-temporal information. The challenge of a game can directly influence a player skill (as a challenging game will require the player to adapt and further hone his/her skill) and the player's physical status (as it can tire or physically strain the player).

3.3 PHILOSOPHY OF SPACIAL

Since the values of the SPACIAL model are both game-dependent and a valid function that is able to determine if the player is in a flow state, based on said variables, can also be difficult to model, a tool to allow for the testing of different functions that govern that flow state or measures the values of said variables is needed. A framework, based on the adaptive game-based learning framework [10] was therefore designed.

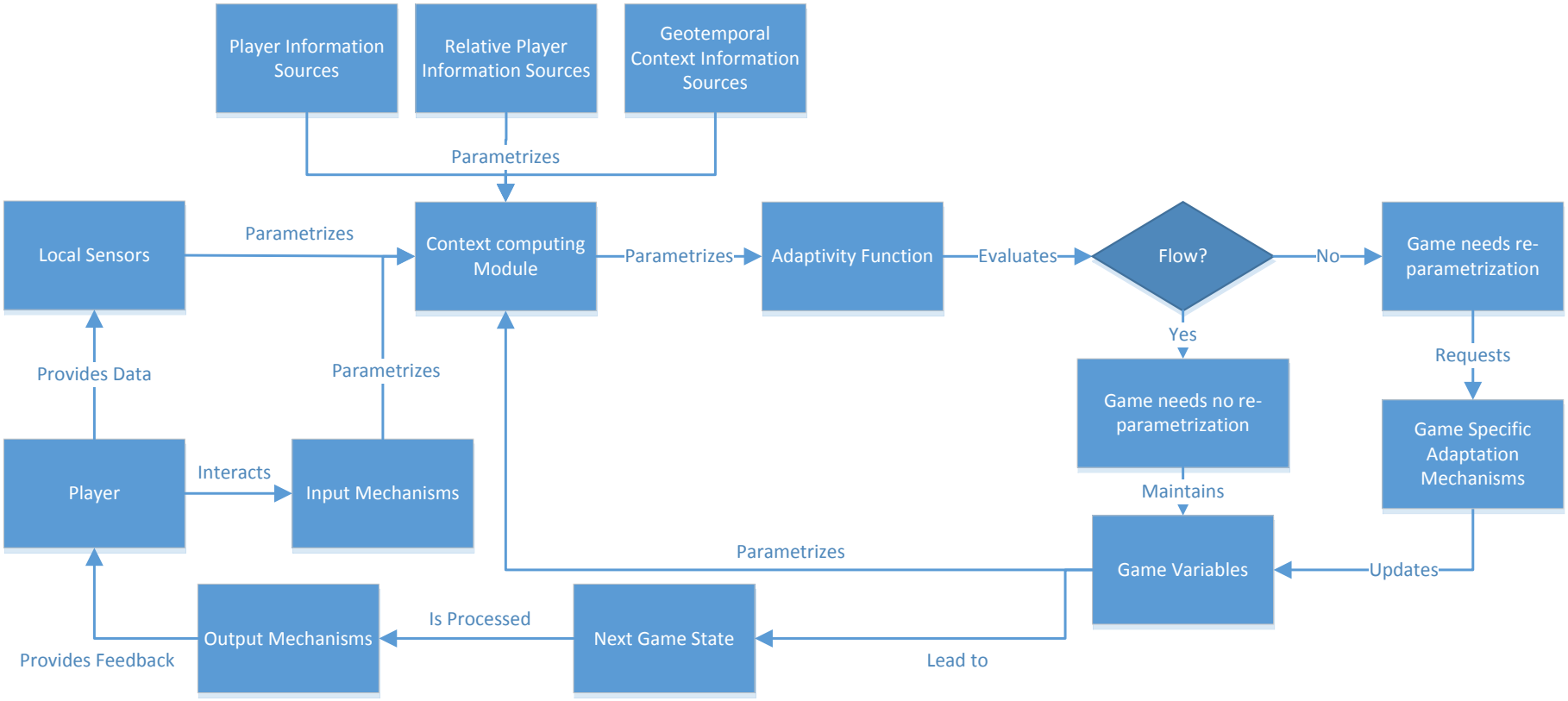


FIGURE 12- FLOWCHART OF THE SPACIAL FRAMEWORK

Figure 12 represents a derivative framework (from the one presented by Berger [10]) that introduces external sources of information (external to the game's mechanics) such as sensor data, player information (both absolute and relative) and geo-temporal context information.

The creation of this framework is based on the game loop and the interaction-feedback loop, but exposes key steps in those loops that can be explored when creating an adaptive, context-aware game. Particularly, the parametrization of the game through Game Specific Adaptation Mechanisms (explored in more detail in Chapters 7.1 and 7.2) and the computation of the current context, based on the local sensors/input mechanisms and remote sources of information. This context is then evaluated against an adaptivity function that is responsible for deciding if the game needs to be reparametrized (if the context is deemed to be inadequate) or not before computing the next game state. This new state will trigger changes in the player, that will be reevaluated in the next adaptivity loop.

- As it is possible to see, the difficulty of implementing such a framework lies on three parts: Context computing: calculating the current context based on several sources of data requires identifying the most important variables in determining the current context, discovering possible sources for that data as well as dealing with the need of potentially merging conflicting data or information from different sources. The context, for the purpose of this work, is the collection of information and data that are deemed relevant for a game.
- Adaptivity Function: the adaptivity function is responsible for mapping the contextual information to game-specific adaptation mechanisms, deciding when/if to trigger a certain mechanism. This function can effectively be a mathematical function (or multiple) that translate contextual information into parameters for adaptation mechanisms. As presented in Chapter 3.1, this function is meant to create a state of flow, taking multiple inputs, and triggering the needed changes in the game.
- Game-Specific Adaptation Mechanisms: these mechanisms are game-specific mechanics (for instance, fire weapon dealing Y damage) that are either parameterizable or only punctually triggered (spawn an enemy), or both. These mechanics are often implemented in games as isolated functions that specialize in doing that one task. As such, they can be already parametrized dynamically or called

only during context-specific events (player's heart rate is low, player is near a restaurant, etc).

This framework, the SPaCiaL framework, was developed based of the following summarized core principles:

- **Redundant and expandable sources of information should be exposed to developers**

Location-based games and exergames are often reliant on both web-services' information (POI information, maps, weather, sunrise/sunset, traffic, etc.) and sensors' data (accelerometer, pedometer, depth camera, etc.). In the particular case of mobile devices, the brand and model can influence the number of sensors available and the quality of their data. As such, it is often needed to have multiple ways to compute the same information. For instance, the pace the player is moving at can be computed via GPS position displacement or via a pedometer. The same can be applied to web-services, where the information available in some may not be available in others. Allowing a developer to select the information from multiple sources can help guarantee a more reliable and higher quality information flow, useful for both gameplay and adaptivity purposes (providing a definitive answer to **RQ1**).

- **Extendable centralized game analytic services are useful in helping developers evaluate UX and player progression**

Game Analytics provide developers information regarding how the players play the game and the issues they encounter. This information can be useful in creating game configurations that suit the game's players or in recommending the most adequate game configuration available for that game.

- **Tweaking of game/level adaptivity in real time with as little programing as possible**

More often than not, development for mobile platforms require the developer to deploy and test their projects often (as the platforms used for the development process usually lack the diversity of sensors of their mobile counterparts), test the application in emulators or manually program different scenarios for development and production environments. These solutions can make the process of tweaking the gameplay parameters or adaptivity

a slow and repetitive process. As such, a way of tweaking a game's behavior in real-time is desirable (see **RQ5**), as fine tuning the adaptivity function (or combination thereof) can be a repetitive and iterative task if a developer is required to tweak these parameters, deploy the new changes to the device, test and evaluate the results in different devices. Real-time reparametrization of a game can allow the developer to skip the deployment part of this process, substantially cutting down the time needed to obtain a desirable set of game parameters that govern the game.

- **Player profiling and unsupervised adaptivity**

Storing and identifying similar player profiles can help developers identify correlations between age, gender, physical fitness, BMI (Body Mass Index) and other parameters and game configurations. Additionally, this lays the foundation for the creation of a recommendation system that matches people with similar profile parameters with game configurations, leading to the existence of an unsupervised adaptivity.

These core principles are distinct enough that they can be separated into components of the framework, where they can operate independently, yet contribute to the overall usefulness of the framework in addressing the adaptivity problem of location-based exergames.

3.4 ESTIMATING PLAYER EFFORT

As previously presented in Chapter 2.3.2, there are several methods for calculating and estimating a person's gait, expended energy and effort. For the purpose of this work, only the following equations will be used for calculating the effort. The reasoning behind this is two-fold: not only are these equations well known, they also rely on the parameters of Age, Resting Heart Rate and Current Heart Rate, which are easily acquirable, as seen in Chapter 7.1.1, for the purpose of testing.

So, by using the Haskell equation for calculating the Maximum Heart Rate[26]

$$MaximumHR\ (bpm) = 220 - Age$$

and Karnoven's Heart Rate Reserve formula [38]

$$HRReserve\ (bpm) = MaximumHR - \text{RestingHR}$$

it is possible to determine the current intensity of the physical exercise using the Karvonen method [38].

$$Effort\ (\%) = \frac{CurrentHR - RestingHR}{HRReserve} \times 100$$

An effort value of 0%, when the current heart rate is the same as that of the resting heart rate, means that the person can be considered to be rested. Conversely, an effort of 100% is attained when the person's heart is performing at the theoretical maximum. As the above formula shows, effort values vary linearly with the CurrentHR (as the RestingHR and HRReserve are considered constants, even though they can vary from person to person).

3.5 ESTIMATING EXPENDED ENERGY

Expended energy can give us an approximation on how many calories are being burnt during an exercise. These can be indicative, when compared to the caloric intake of a person (i.e.: how much the person eats) if the exercise is potentially leading to weight loss.

Similar to what was explained in Chapter 3.4, the choices of the equations used in determining the Burn Rate rely on easy-to-obtain variables such as player Weight, Speed and the current Slope of the road or terrain the player is running on. These equations were studied and determined to over-estimate oxygen consumption [25]

By use of a player's weight, speed and slope to calculate the following variables:

$$Relative\ Gross\ VO2\left(\frac{\frac{mL}{Kg}}{\min}\right) = 0.2 * Speed + 0.9 * Speed * Slope + 3.5$$

$$Burn\ Rate\left(\frac{Cal}{\min}\right) = \frac{RGVO2}{1000} * Weight * 5$$

These formulas can be used to estimate the energy consumption of a running activity. The usage of this formula in the context of a location based game is shown in Chapter 7.2.1.

3.6 THE SPACIAL FRAMEWORK

The SPaCiaL framework, was broken down in several, specialized components. For testing purposes the framework was afterwards developed as a set of self-contained components. While the current chapter focuses on the design and concept of these components, and is therefore technologically agnostic, the chapters 4, 5 and 6 will detail the design and implementation of the three major components.

The SPaCiaL framework is heavily dependent on 3rd party remote web-services, device sensors and on the Game Analytics Web-Service. However, if the Game Analytics Service as well as the GeoStream Library are not needed, the SPaCiaL framework has no remote dependencies, beyond the device that is running the game.

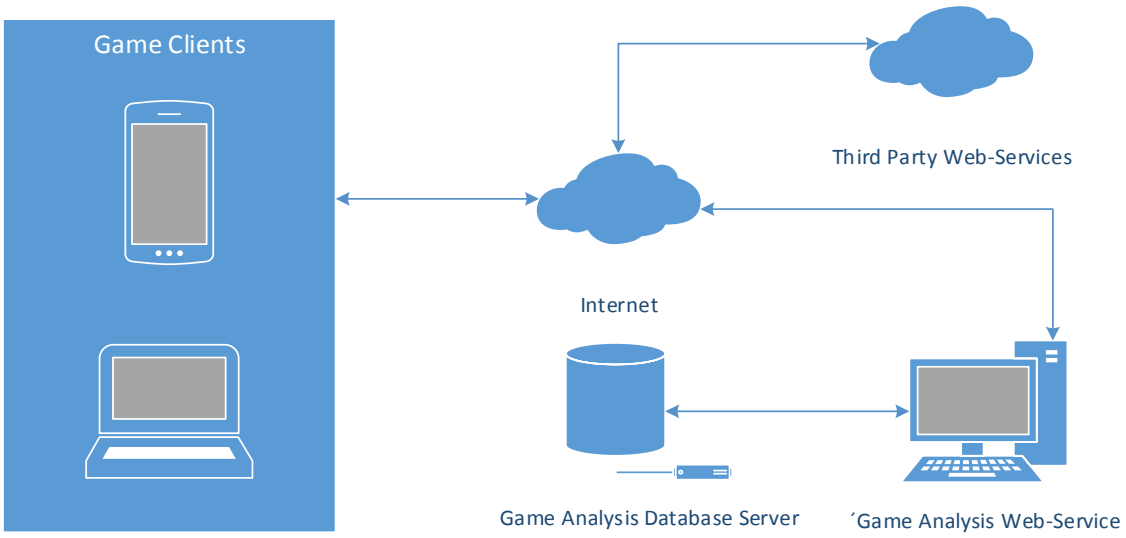


FIGURE 13- PHYSICAL ARCHITECTURE DIAGRAM FOR THE SPACIAL FRAMEWORK

Figure 13 illustrates the physical dependencies that the framework has. Has can be seen by that diagram, an internet connection is needed to make the most out of the capabilities of the framework. Without one, the framework will only be able to rely on local content, limiting the capabilities of GeoStream (as it can only access cached data) and rendering the Game Analytics Web-Service completely unavailable.

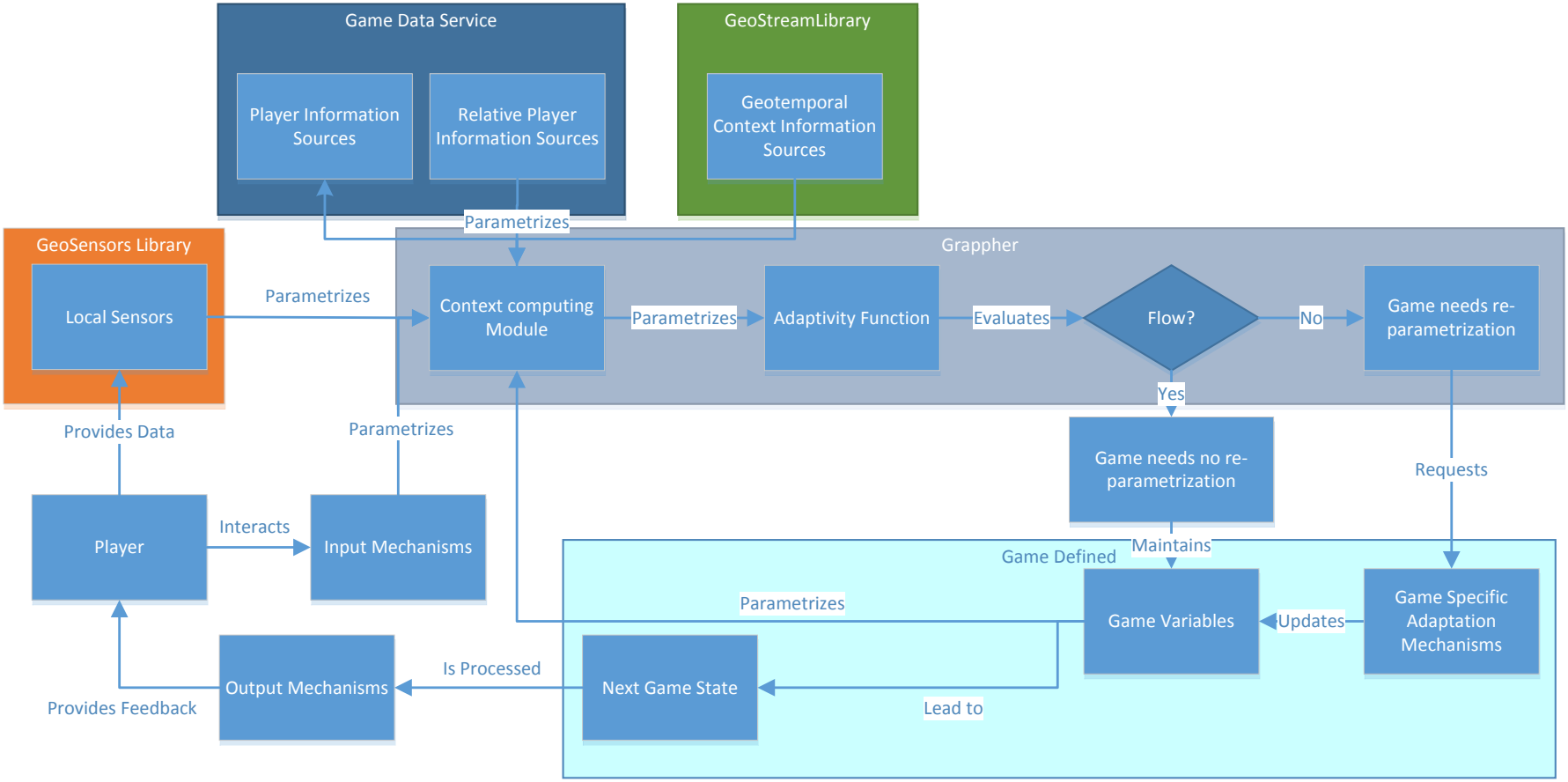


FIGURE 14- FLOWCHART OF THE SPACIAL FRAMEWORK (PER COMPONENT)

The SPaCiaL Framework prototype was designed with game engine integration in mind. It is comprised of the four components depicted in Figure 14 (highlighting how these components work in tandem:

- **GeoStream Library:** a library for accessing and computing geo-temporal data, useful in the development of location-based games and providing the developer possibilities on what to do with that information. Information computed includes POI meta-data, buildings footprints, weather information, satellite imagery, altimetry data and road networks information.

- **GeoSensors Library:** an API to abstract sensor data access and to allow for the creation of spoofing mechanisms in order to create false, but plausible, sensor data for when the real sensor is unavailable, as it is often when developing.

- **Game Data Service:** A remote service, meant for storing and processing player profiles, game sessions events, game sessions' results, progress tracking and game configurations for several games. This service lays the foundation for unsupervised adaptivity and player profiling as it could become a recommendation system, matching game session parameters across players with similar traits. As this is mostly a support service, not directly related to the research questions, its details were relegated to the Annex A.

- **Grappher:** a graph-based tool for the design and usage of game configurations and their mechanics. It is meant to aid a developer in creating custom high level nodes and graphs that are capable of generating, accessing and processing information relevant to a game or level, as well as triggering game-specific events, visually, with no programming needed. It provides a game designer with the ability to tweak a game's behavior without programming.

Each major SPaCiaL component (GeoStream, GeoSensors and Grappher) is detailed in its respective chapter, as follows

4 GEOSTREAM

The issue of gathering geo-referenced data and preparing it for usage in the context of location-based game is one of the main problems the creation of location-based games faces that is inexistent in regular, non-context aware, games.

As such, there is a need for accessing multiple sources of geo-referenced data and providing developers with means of using that data in location-based games.

GeoStream aims to become a solution for gathering remote geo-temporal data from 3rd party services, allowing an extensible means for developers to compute information from distinct sources.

The API allows for a developer to specify a scope: a geographical area for which data will be accessed.

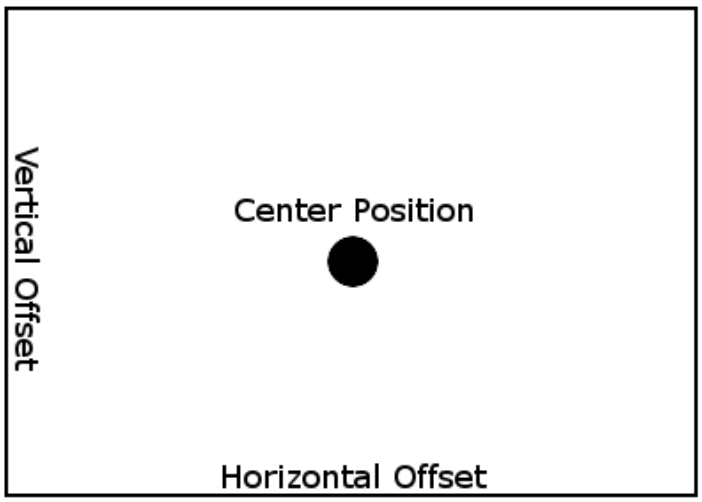


FIGURE 15- A MAP SECTION, DEFINED BY ITS BOUNDS

As Figure 15 shows, a map is comprised by the center position (in a Location-Based Game it will often be the Player's real physical position) and vertical and horizontal offset

values that, depending on the context, can be pixels, meters (for UTM coordinates) or degrees (for Latitude and Longitude coordinates), determining the bounding box where the game will take place. This approach, although usable for games where the player will not go out of the defined bounds, is unable to cope with the need to effective and continuously provide geo-temporal data as the player moves along the map. As such, a different approach was needed. The chosen approach was one that mimics most Map Viewing services and large-scale online games, such as Minecraft[1]: dividing the map in "chunks" and dynamically loading the close surroundings of the desired location. The next figure, Figure 16, further illustrates this approach.

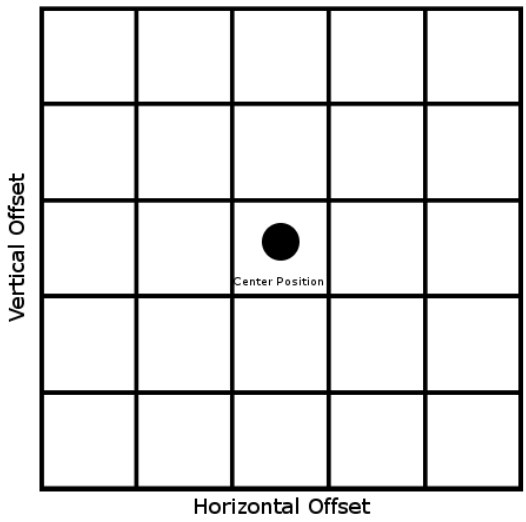


FIGURE 16- A MAP SECTION, DIVIDED IN SEVERAL CHUNKS

Such approach allows for the loading and unloading of chunks and their elements as they are needed.

1- https://minecraft.net/en/

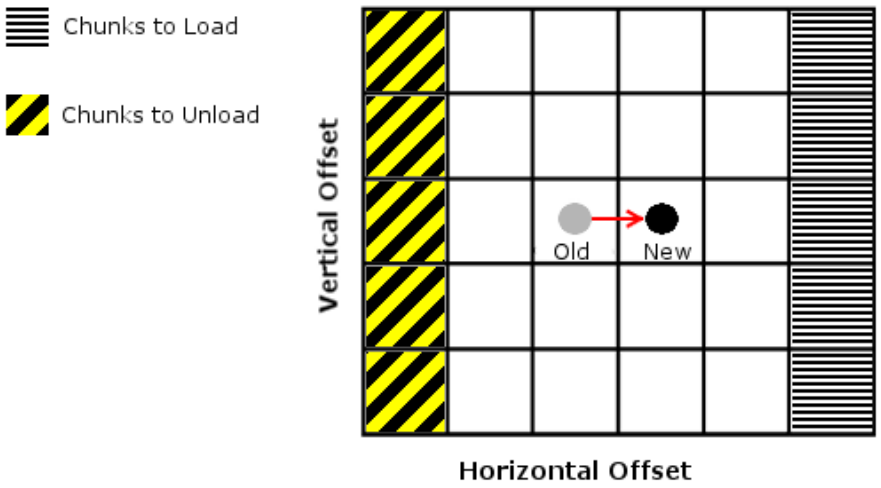


FIGURE 17- DEPICTION OF THE DYNAMIC LOADING AND UNLOADING OF MAP CHUNKS AS THE PLAYER MOVES EAST

As Figure 17 further illustrates, the map manager will choose what map chunks to unload depending on the position of the game entity it tracks. As the player moves toward a border of the map, whenever the limits of a chunk are traversed, new chunks are requested and older chunks may be discarded. Chunks that are marked to be discarded will be removed from the game, including all the georeferenced entities and information that were created with it. These entities, named features, comprise any type of geo-referenced structure, and are what populates these chunks. As such, features can be rivers, trees, bridges, buildings, bus stops, roads, sidewalks and the likes. However, since many of these structures are so big they can spread across several chunks (such as highways, large buildings, etc.), they cannot simply disappear if the chunk that created them is to be unloaded, as doing so would remove them from a chunk of the map that is not to be unloaded. Instead, each feature has a reference counter, a number that keeps track of how many Map Chunks it belongs to. Each time a Map Chunk unloads, all of the features it contains will have their reference counter decreased by one. Conversely, each time a Map Chunk loads, existing features it contains will have their reference counter increased by one, whereas new ones will have it set to one. When unloading a Map Chunk, if any feature has a reference counter of zero, meaning it is not present in any current Map Chunk, it will be removed.

This architecture is further refined by the existence of a hierarchy between the Map, its Chunks and different services that require either a pair of coordinates to function (such as a Weather Service) or a bounding box to be defined (such as getting a Satellite Image of a portion of the earth). This hierarchy is show below.

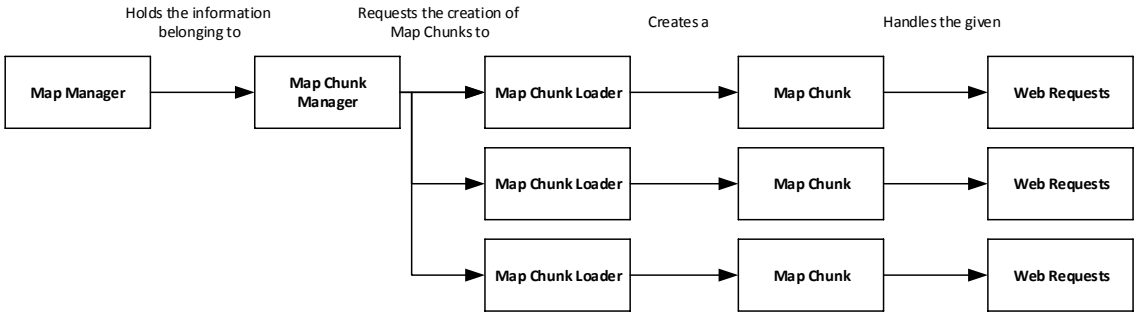


FIGURE 18- HIERARCHY AND INFORMATION FLOW OF A MAP MANAGER ENTITY AND ITS COMPONENTS

As the above figure (Figure 18) highlights, a Map will request the Map Chunk Manager to divide it into several Map Chunks, each of them being responsible for passing down their own coordinates (be it center or bounding ones) to the respective Remote Web Services that are registered in the Map Manager. This way, each Map Chunk is responsible for requesting, receiving and handling all the data referring to its own area (a portion of the Map). This approach not only allows for the progressive loading and unloading of needed and unneeded Map Chunks, respectively, it also allows for the parallelization of this process, as each Map Chunk can run on its own individual thread.

At its core the GeoStream API is comprised of the following components.

- **Map Chunk Loader**
    - The MapChunkLoader is capable of loading a single chunk with the specified bounds and RemoteServices in its own thread.
- **Map Chunk Manager**
    - This class is responsible for determining which chunks are necessary, based on the tracking of a game-object. If the game-object goes out of a

chunk, new ones are loaded, while unneeded ones are discarded. It serves as an interface between the MapManager and its MapChunks.

- **Map Manager**
    - The Map Manager holds the information of the Map: how many chunks will it be divided into, what RemoteServices should each Map-ChunkLoader use, what game-object should be tracked and the size of each Map-Chunk.
- **Geo UTM Converter**
    - Although many Geo-Referencing Web-Services use Latitude / Longitude coordinates, when representing 3D/2D Geo-referenced entities on a Screen, it is best to use a Map Projection technique to obtain Cartesian coordinates, that can be used to represent the world's features in a plane. This class is capable of converting WGS84 Latitude / Longitude coordinates to UTM coordinates and vice-versa.
- **MapFeature**
    - A MapFeature can be either a Node or a Way. The differences between them are described below.
- **Node**
    - Derived from the definition of a Node from OpenStreetMaps [30], a Node is a Dictionary of Key-Value pairs (*feature / value*) with a Pair of coordinates. It can be generically used to store information of a geo-referenced feature of any source, from buildings and roads to rivers and trees.
- **Way**
    - A Way, based on the definition from OpenStreetMaps [30] is an ordered set of Nodes. If the ending Node is the same as the starting Node, the Way represents a closed polygon. The way also has the same meta-information that a Node has (coordinates and features). It is often used, in OpenStreetMaps to hold information pertaining to roads, rivers, shorelines as

well as administrative areas, city limits, buildings, parks and other structures with large shapes.

- **MapElementManager**
    - A single MapElement Manager exists in each MapManager, and allows for a developer to query for specific Nodes or Ways (MapElements) that exist on said Map. This is useful for the creation of dynamic Location-Based Games, as the developer can query the MapElementManager about nearby features and transforming the returning Nodes as game elements.
- **RemoteService**
    - The definition of a Remote Web Service. Its functions can be implemented by other classes, allowing for the access of new Web Services to be made and the results of said call to be processed.

An issue that often affects location-based games is the need to download and compute information each time the game/level is played. In the case of the GeoStream framework, it allows developers to cache their results (either from downloads or computed game entities, such as Game Objects or Materials) so that if the player replays an already played level or location, it will not need to re-download / compute the whole level.

4.1 IMPLEMENTATION DETAILS

The GeoStream Library was developed as a Unity 3D component. This component has several public fields that are exposed in Unity's Inspector window when a GameObject that contains the GeoStream Map Chunk Manager.

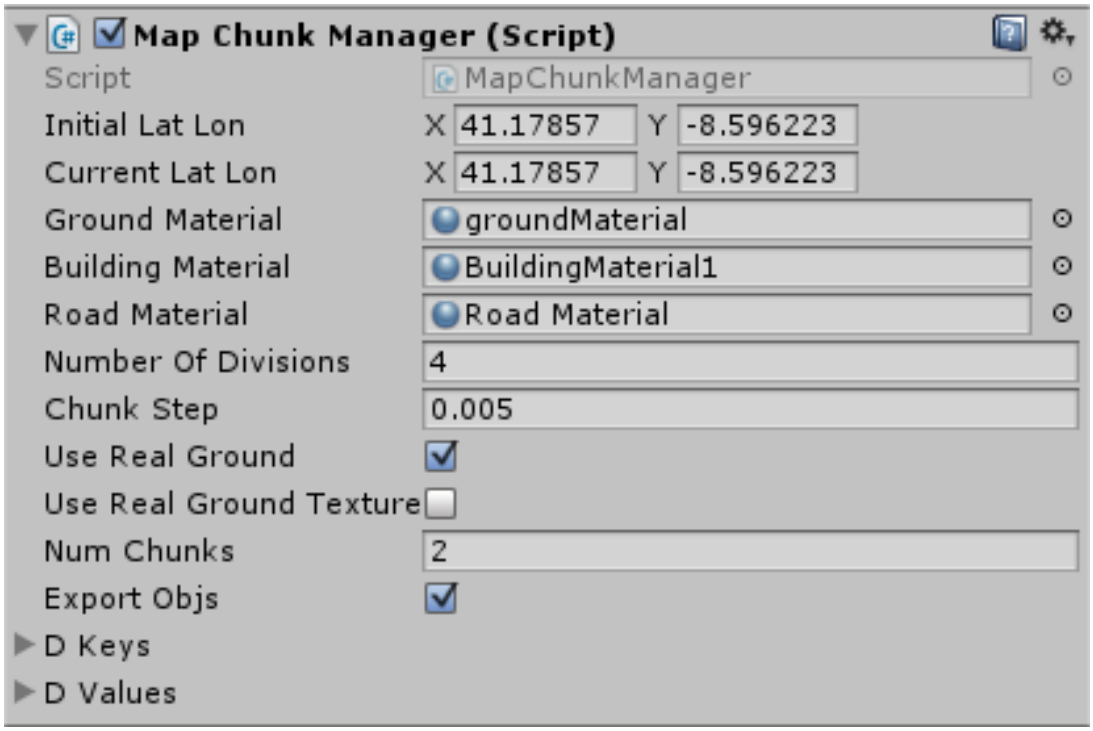


FIGURE 19- GEOSTREAM'S UNITY COMPONENT IN INSPECTOR WINDOW

The Unity component of the GeoStream library allows for some parameters to be changed on the fly (Figure 19). These parameters are as follows:

- **Initial Lat Lon:** specifies the initial pair of latitude longitude coordinates that represent the center of the Map (the map being comprised of one or more possible Chunks).
- **Current Lat Lon:** as the player or the object being tracked changes its position in the game, new map data may be needed to be downloaded, and some current map information may be discarded as well. This parameter specifies the current coordinate of the object being tracked.

- **Ground, Building and Road Materials:** these are the materials current being applied to Roads, Buildings or the Ground, if these structures are being created at all. Although other structures may have a 3D representation, as it will be explained ahead, as these are the most commonly needed, a faster way to changed them other than via code was also necessary. Thus, their inclusion as component parameters in the inspector window.
- **Number of Divisions:** each Chunk has its own Ground mesh. This mesh can be split into several lines and rows, depending on the values of this parameter. Particularly when Use Real Ground is enabled, a greater number of divisions can result in a more reliable ground mesh, detailing the elevation changes that are prevalent in that Chunk.
- **Chunk Step:** this determines the height and width of each individual Chunk, or the difference in maximum latitude and minimum latitude, as well as the difference in maximum longitude and minimum longitude that represent the area of each Chunk. Depending on the coordinates of the map, even though the ΔLatitude and ΔLongitude are the same, due to the distortion present in the WGS84 representation, some chunks will be taller or wider when converted to UTM.
- **Use Real Ground:** this parameter specifies if the programmer intends for the GeoStream framework to initially download real altimetry data for the specified Map and its Chunks. By default, this component makes use of the GoogleAltitudeRemoteService (based on the Google Altitude API)
- **Use Real Ground Texture:** specifies if the framework should download satellite imagery and apply it to the target Chunks. By default, it makes use of the GoogleMapsStaticRemoteService (based on the Google Maps Static API)
- **Number of Chunks:** as a Map is made up of several Chunks, this parameter dictates how many Chunks will be used in the initial Map. If the parameter is set to, for instance, 2, then four Chunks (2x2) will be instanced, with the specified ChunkStep, centered at the Initial Lat Lon parameter. As this value increases, initial set-up of the Scene can become computationally heavy, as each Chunk is processed in its own thread. As such, values greater than 4 (so, 4x4, 16 Chunks total) are not recommended. An alternative, to avoid using a great number of Chunks initially, is to decrease this parameter, and increase the Chunk Step pa-

rameter and the Number of Divisions parameter. This way, although having a reduced number of Chunks, their size and ground mesh resolution make up for it. However, as the satellite image will now have to cover a greater area with the same resolution, pixelation may be noticeable.

- **Export Objs:** in order to hasten the process of generating the Map and its Chunks in subsequent times, the parameter specifies weather caching of objects (be them 3D meshes, textures or remote web service responses) should occur. If so, the framework will check if any RemoteService request has a valid cache file (for the specified parameters) and if so, no web request made. In some cases, like the generation of buildings and roads, not only is there no web request made, there is also no need for triangularization, as 3D models are cached as *.obj files. This option can speed up the loading time noticeably. However, if the area being cached is big or heavily populated with MapElements that are of interest, caching can take a significant amount of space (in the order of dozens of MBs)
- **Dynamic Keys and Values:** As several MapElements are present in each Map and its Chunk, the programmer may want to specify how those MapElements should be treated. As such, in these fields, the programmer specifies what MapElements should be process by which Structure sub-class. Each Structure sub-class implements methods to build both a Node and Way. The specific structure type (road, bridge, tree, building, hospital, etc) of the Nodes or Ways that are passed to that Structure sub-class depends on what the programmer specified in the Dynamic Key of the corresponding Dynamic Value. As Figure 20 shows, in a possible scenario, “building” types of structures are handled by the BuildingScript. Note how both “highway” and “sidewalk” Nodes and Ways are also treated by the same RoadScript. These structure scripts are responsible for defining how the geographical information that the GeoStream framework aggregated should be present in the game.

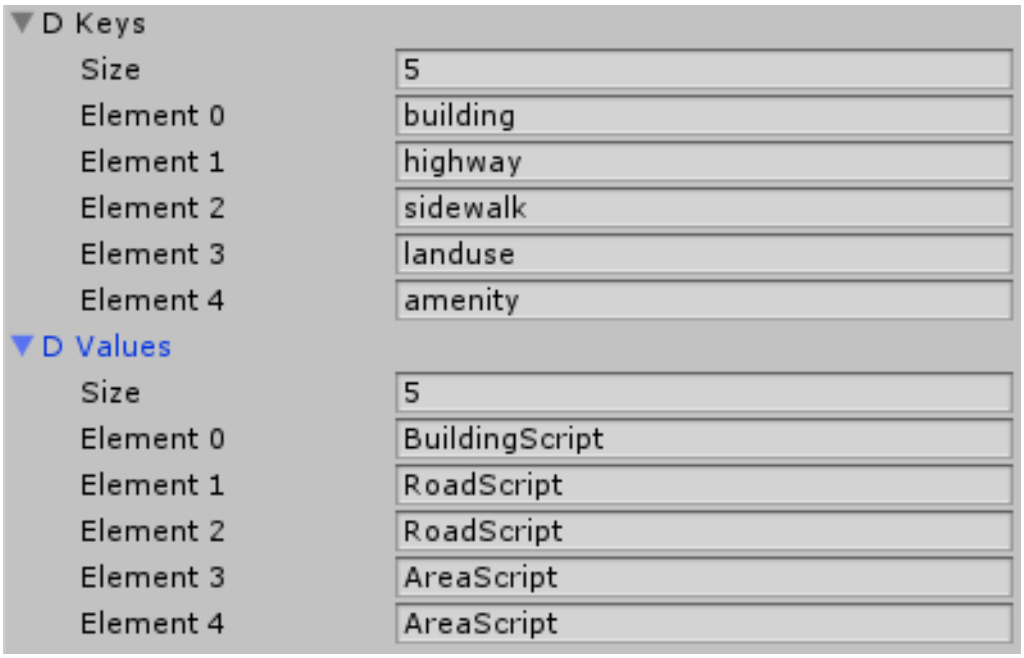


FIGURE 20- EXAMPLE OF DYNAMIC KEYS AND VALUES PAIRS IN THE MAP CHUNK LOADER COMPONENT

Several **RemoteService** classes where developed, making use of C# multiple inheritance. These can be a way of partially positively answering **RQ1**. These subclasses allow for remote information sources to be queried and their information to be processed. However, it is only applicable for remote sources. Examples of these implemented classes are as follows:

- **GoogleMapsStaticRemoteService**: Service responsible for downloading and caching a Google Maps Static API satellite image.
- **BingMapsRemoteService**: A similar service to the previous one, it downloads satellite images from the Bing Maps API.
- **MapQuestRemoteService**: Service responsible for downloading MapQuest map textures.
- **GoogleElevationRemoteService**: Service that downloads altimetry data and creates the ground geometry of a chunk based on said data
- **BingElevationRemoteService**: Similar to the previous service, but using Bing's altimetry data instead.
- **GoogleGeoReferenceRemoteService**: Service responsible for translating full addresses to their respective latitude/longitude coordinates, and vice-versa.
- **FlatElevationRemoteService**: Service that downloads no altimetry data, creating a flat ground mesh instead, with an elevation of 0 meters.

- **LocalWeatherRemoteService**: This service download weather information from the OpenWeatherMap API. This information is comprised of temperature, humidity, pressure, wind speed, wind direction, cloud intensity, precipitation and the type of weather (clear, cloudy, hazy, etc).
- **OpenStreetMapsRemoteService**: Service responsible for querying OpenStreetMaps XAPI for Nodes and Ways available in the requested area
- **GooglePlacesRemoteService**: Service that queries the Google Places Web Service API in a similar manner as that of the previous service. Places with a single "geometry" entry are considered Nodes, and Places with several geometry entries are considered Ways. The "types" described in each "place" by the Places API (such as "restaurant, establishment, building") are inserted as individual structure keys with the values of "yes".

The merging of information between the OpenStreetMaps and Google Places services is made through the name matching of the Nodes and Ways both have created. Other methods were developed and tested, such as assuring both structures were close to one another and shared the same category meta tags if available. However, since there are some discrepancies between them, merging both MapFeature representations into one is not always possible, often resulting in distinct MapFeatures representing the same structure coexisting and leading to some problems. As such, and due to the extensibility, simplicity, quality documentation and detailed coverage in big cities that OpenStreetMaps possesses, only the OpenStreetMapsRemoteService was used. However, this work provides an answer to **RQ2**, as to "how location's features of interest information can be aggregated from different sources." although, as mentioned, this is a problem with no easy or universal solution. The OpenStreetMaps ontology provides an expansible and flexible schema for defining and describing a map's feature. However, it can be difficult to understand the type of values defined in this ontology as well as measurement units.

The following images attest to the quality and detail of the generated scenarios that the GeoStream framework is capable of computing. These serve as partial visual representations of some of the framework structures' information, as there are several (such as sidewalks, traffic lights, fire hydrants) that are not represented visually, due to no Structure script capable of handling them being implemented at the current time.

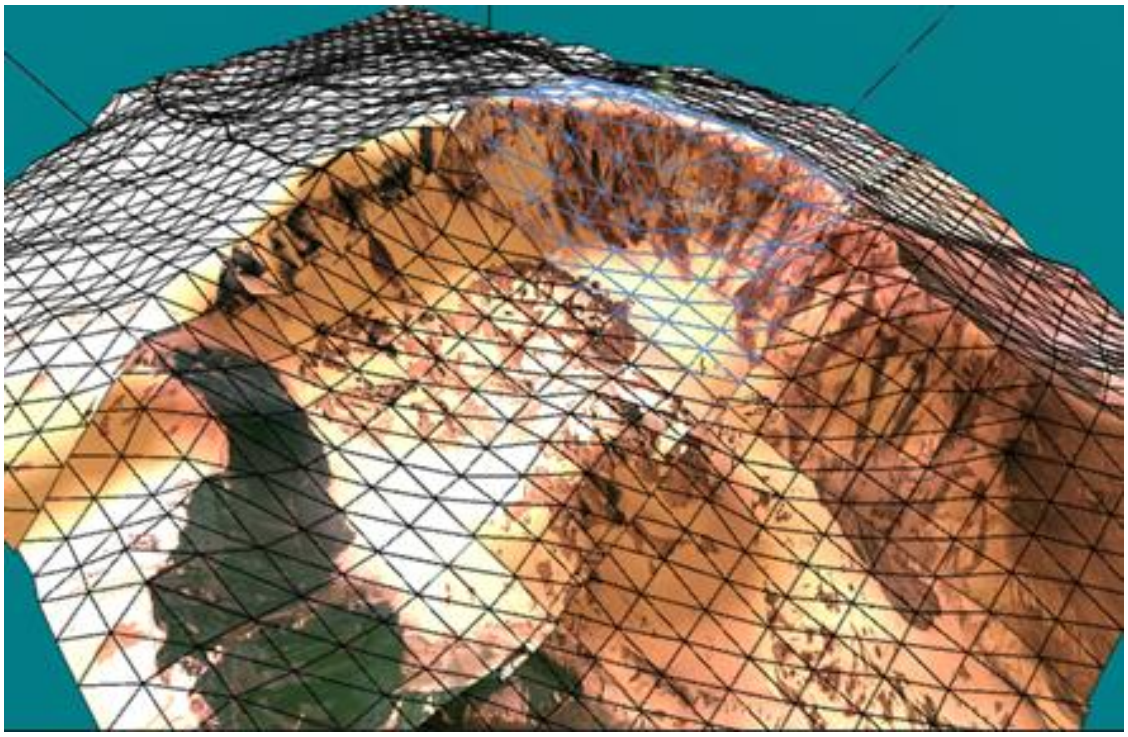


FIGURE 21- CRATER OF ST. HELEN'S MT. (GENERATED IN GEOSTREAM)

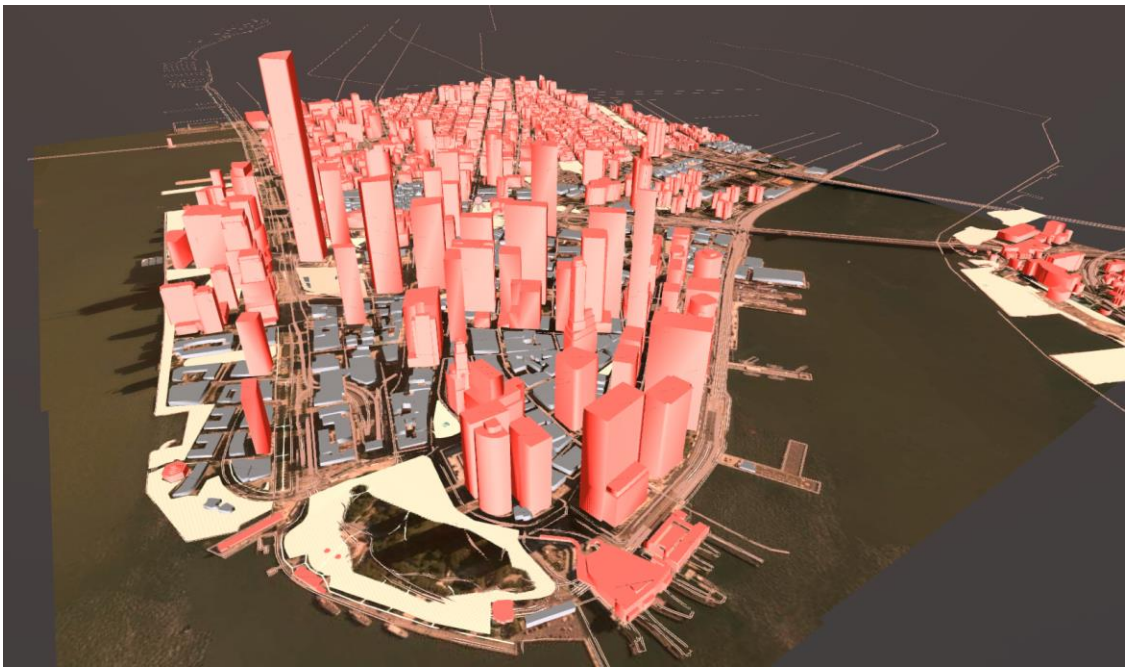


FIGURE 22- ISLAND OF MANHATTAN (GENERATED IN GEOSTREAM)

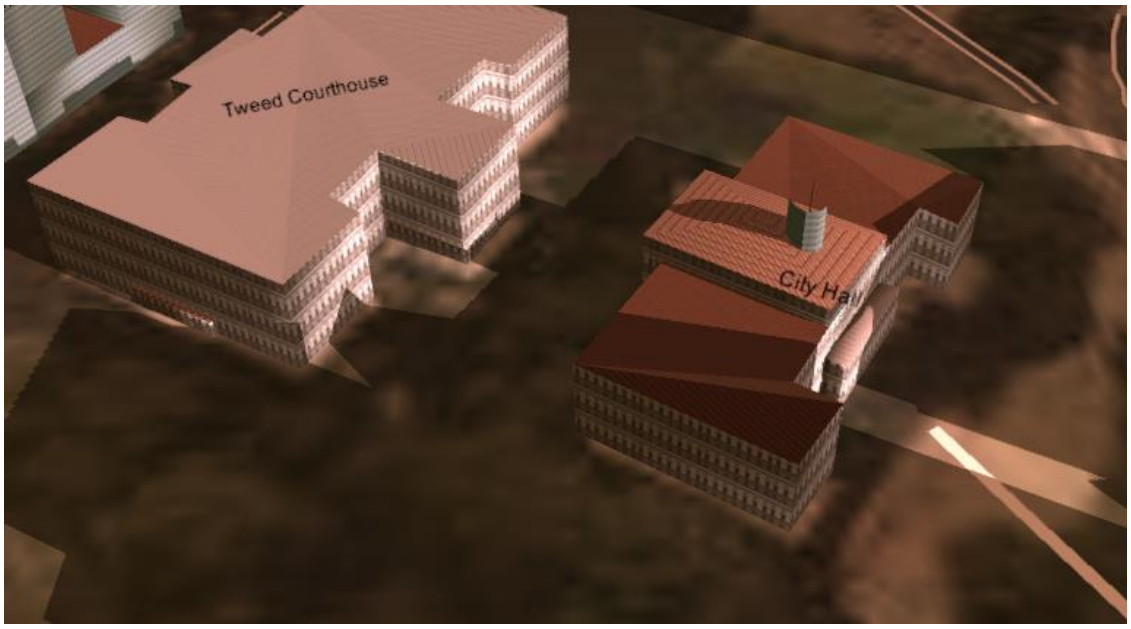


FIGURE 23- BUILDING GEOMETRY IN DETAIL (GENERATED IN GEOSTREAM)

FIGURE 24- PORTO DOWNTOWN (GENERATED IN GEOSTREAM)

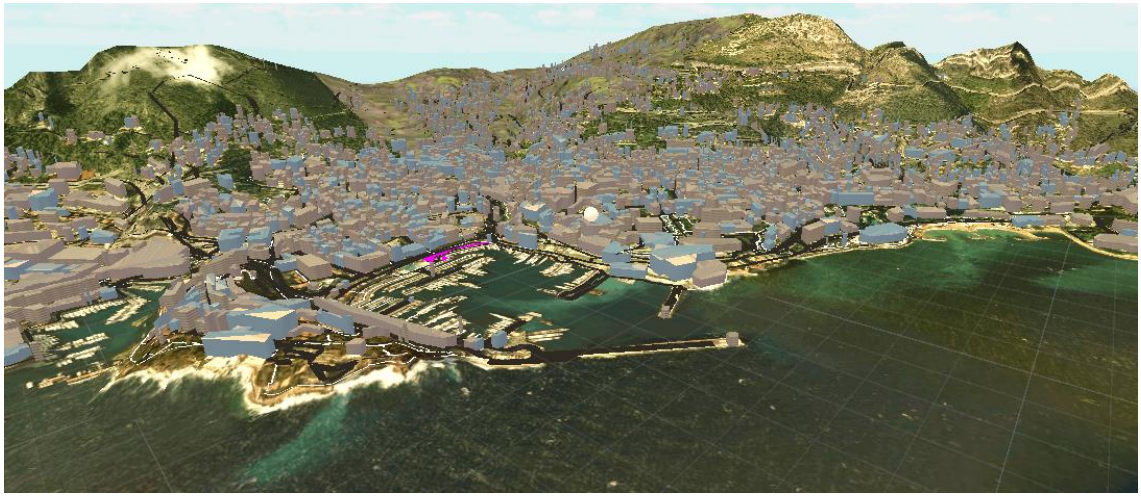


FIGURE 25- MONACO (GENERATED IN GEOSTREAM)

These images showcase how it is possible to represent the computed geographical information in different ways with GeoStream. Some cities display only single coloured buildings, others, fully textured and intricate. This is done by changing the script that is responsible for handling the structures that contain the type "building". Also noticeable in Figure 21 is the Chunks' delimitations and the density of the ground mesh. As all structures and Chunks have a relative positioning (the centre of a Map begins always at the coordinates of {0,0,0}, and every coordinate is offset from that initial UTM coordinate), every structure has a 1:1 scale, in meters. This makes modelling and simulation based on the GeoStream framework both easy and precise, out of the box [36].

5 GEOSENSORS

As both exergames and location-based games require the developer to access several sensors and specify how that data is transformed into game relevant information, an approach to allow consistent, configurable and extensible access to such sensors becomes needed.

All current, major, mobile platforms (Android, iOS, Windows Phone) give developers a basic abstraction layer to access devices' sensor data. However, it often is inconsistent in providing access to that information, while attempting to keep the API readable. For instance, in Android, accessing Compass Orientation can be done with the following line of code:

public static float[] SensorManager.getOrientation(float[] R, float[] values)

or the developer can register a callback to the *OnSensorChanged* event for the specific sensor.

However, for instance, accessing GPS data is done by creating a *LocationListener* and registering it to a *LocationManager*. While the first approach allows for the developer to just read the sensor value or register an event whenever the sensor is polled, the second approach is different, allowing only for the registering of a different event.

Additionally, a developer cannot easily extend these APIs, adding access to either new sensors the device or another remote device has, or new ways to read the data from already existing sensors.

Some Game Engines such as Unity or Unreal already provide some abstraction layer to access a device's sensor data. However, these usually do not provide the same level or depth of information the Native APIs provided by the mobile platforms do, nor the same variety of sensors. In the particular case of Unity it is not possible to register callbacks to Sensor Events nor is it possible to access some sensors such as barometers or heart rate sensors.

So, since the development of location-based games and exergames is so reliant on the access to sensors' data and its computation, a library for easing this task was designed, named GeoSensors.

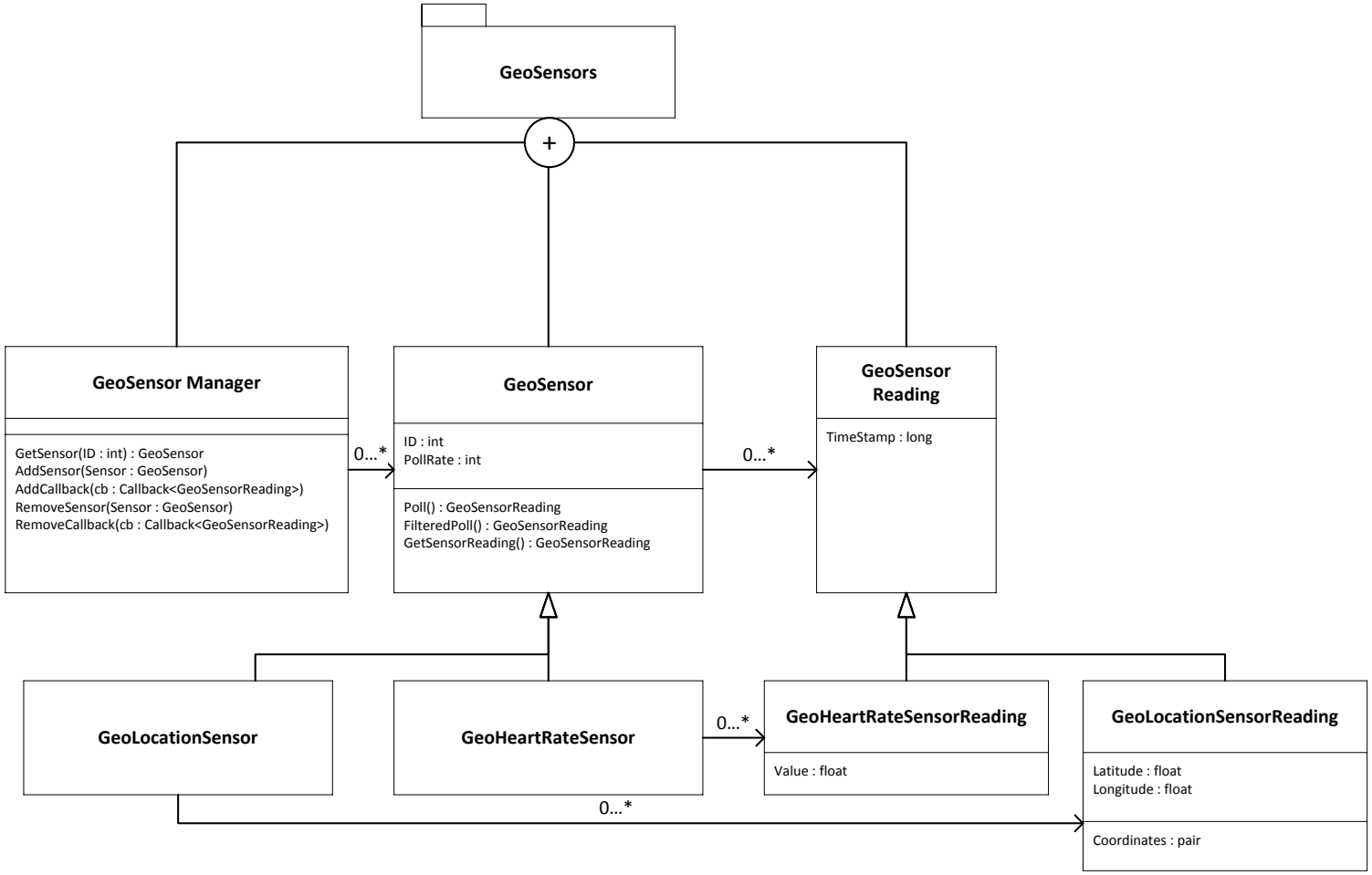


FIGURE 26- CLASS UML DIAGRAM OF THE GEOSENSORS LIBRARY

The core of the GeoSensors library is comprised of three classes:

- **GeoSensorManager**
    - This class is responsible for governing all GeoSensors and their callbacks, following the observer design patern. It allows for developers to add new GeoSensors and to associate callbacks to be run whenever a specific sensor is polled. The polling of the sensors is done by this class, taking into account the poll rate specified in each GeoSensor.
- **GeoSensor**
    - An abstract class, meant to be extended into concrete representations of a Sensor. It is comprised of an ID, a polling rate, a list of GeoSensorReadings and methods to get new GeoSensorReadings. GeoSensorReadings are considered as facts and are not deletable by the developer. These GeoSensorReadings are kept so as to allow for filtering of their values.
- **GeoSensorReading**
    - It represents what the data read by a sensor is. Since sensors and their data vary greatly, it is only an abstract class, with a timestamp, meant to hold the information of when said GeoSensorReading occurred.

The UML diagram shown in Figure 26 details the GeoStream library and its core methods. It also presents four other classes that are not part of the core package. These classes represent a location sensor and a heartrate sensor, and their respective readings, showcasing how this architecture can be extended to include other sensors. While the sensors themselves do not appear different in the diagram, they would mostly differ in their polling methods, as they access different sensors and work with different data. Their respective readings show the type of information a developer can expect when dealing with those sensors.

5.1 IMPLEMENTATION DETAILS

The GeoSensors Library is comprised of a series of Unity 3D scripts, an Android library used as a Unity plugin, an Android Monitor Application and an Android Wear Application. This was developed in order to overcome the limited access to a mobile devices' sensors that Unity 3D currently offers. The Android library was developed in Android Studio and exported as a *.jar library. Changes to the Unity's Android Manifest file were made so that it was possible to invoke the library methods from Unity via JNI, through a Broadcast Receiver service.

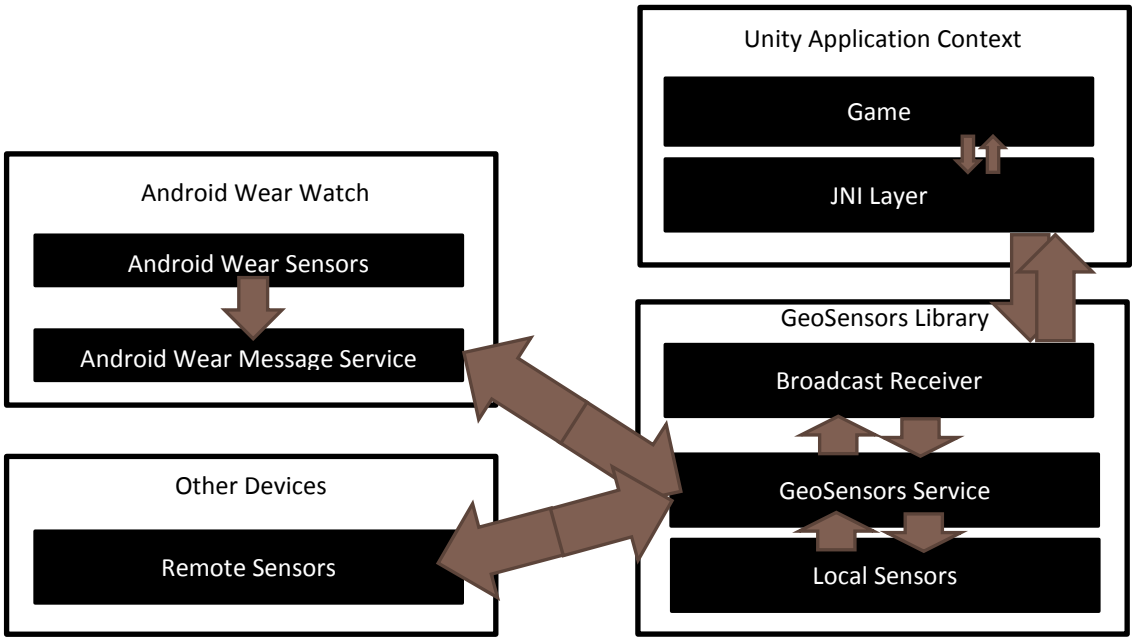


FIGURE 27- GEOSENSORS' COMPONENT DIAGRAM

Figure 27 further describes how the GeoSensors Library serves as a middleware for allowing Unity to access both local and remote sensor data. The GeoSensors Service, running in the background, collects the data from the sensors that were registered in the game. The programmer begins this process by soliciting, via a Unity script, what sensor from which type of device (optional) he/she wants data from. Currently, only two (local and android wear) sources of sensors are available. For the sensors themselves, they vary for each android wear and android device.

If the solicited device is the local device, then this request is then passed on through the JNI Layer onto the Broadcast Receiver, that further relays this request to the GeoSensors Service. If the device is registered in this service and that type of sensor is available, a **void onSensorChanged (int sensor, float[] values)** event is registered to that local sensor and, as the sensor is polled, new data from will then be sent to the Broadcast Receiver

where it will be stored in an array of sensor readings, for each sensor requested. The game developer, in Unity, needs only to request the latest readings for an available and registered sensor.

In the particular case of accessing other devices' sensors, although for the game developer in Unity the process of accessing and registering that device is the same as that for a local device. However, other devices (such as Android Wear devices) require changes in the GeoSensors Service to accommodate their existence. In the particular case of the supported Android Wear devices, it requires the smartwatch to run a specific application that sends to and receives data from the GeoSensors Service, via the Android Wear Message API. This required the GeoSensors Service to be manually changed, as it now has to make use of the Message API when dealing with data of sensors belonging to Android Wear devices. As such, the GeoSensors Service, when solicited to provide an Android Wear sensors' data, will, if that sensor has not yet been registered, send a message to the Android Wear app to register a listener for that sensor. The Android Wear app will then, whenever that sensor has a new reading available, forward that readings' values to the GeoSensors Service (and it to the Broadcast Receiver, via Intent), making it available to Unity thanks to JNI.

This approach can be generalized for other devices, such as Arduinos and IOT devices. However, it would most likely still require changes in the GeoSensors Service, at the very least, for the integration to occur.

The GeoSensors Library provide access to these sources of sensors, in Unity3D, via a series of classes, deriving from the generic GeoSensor and GeoSensorReading classes. These classes provide a series of abstractions for generic handling of the requested data from the BroadcastReceiver. Each GeoSensor, in Unity, has a list of GeoSensorReadings (a moving window of a variable number of readings), filtering (such as Kalman or Bayes, commonly used in sensor data to eliminate noise), prediction virtual functions and callback C# Actions to be run when a new reading is received. There is also a manager for all the currently registered GeoSensors. This manager guarantees that all the sensor's data is in sync with the latest available from the BroadcastReceiver via JNI by updating once every frame. If no new sensor data is available in a certain frame, the previous valid reading is maintained as the latest available one. The following types of sensors and readings are supported, currently, in Unity via the GeoSensors library:

TABLE 3- DEVICES AND SENSORS AVAILABLE IN UNITY VIA GEOSENSORS

	Device	
Sensor	**Phone**	**Android Wear**
Accelerometer	yes	yes
Compass	yes	no
Orientation	yes	yes
Location	yes	no
Heartrate	no	yes
Magnetometer	yes	no
Altitude	no	yes
Touch	no	yes

Other unsupported sensors may still be accessed through it, but additional parsing of the raw string data provided by the JNI call to the Broadcast Receiver will be needed. However, it shows how device independent sensors can be accessed in a game engine that does not support it natively.

(Page intentionally blank)

6 GRAPPHER – GAME PROFILE EDITING TOOL

Using a graph-based editor provides a very visual image as of how the information flow occurs in a graph. Adapting this approach for the design of adaptive profiles can ultimately make their design easier to do, understand and change, as the adaptive semantics for a game are implicitly specified. The process of fine tuning a game's difficulty can be often repetitive, as it requires developers to constantly tweak or reprogram parameters, redeploy, test and evaluate the game. A tool for allowing the design of adaptive game configuration to be made in real-time, as a game is being tested, would allow developers to focus on the task of tweaking game difficulties to specific player profiles.

Grappher was designed as a means to create adaptive gameplay configurations. It follows a visual programming paradigm, as this allows for the design of implicit semantics without the need to change the code of the game. Additionally, this paradigm also fits the Rapid Application Development paradigm, as it allows for continuous changes to the game's profiles with no programming or redeployment of the game needed. The configuration files specify what and how game variables or events change or are triggered, based on contextual information. It should also allow for the game designer to freely decide how said contextual information is computed from whichever data sources are deemed needed.

6.1 IMPLEMENTATION DETAILS

The Grappher Editor Plugin is a Unity 3D Editor Plugin. This gives the game designer the possibility to test and create graphs in design time. These graphs are designed to easily bridge the gap between information sources (both real-time and offline data, be it from a local or remote source) and game actions and its behaviours.

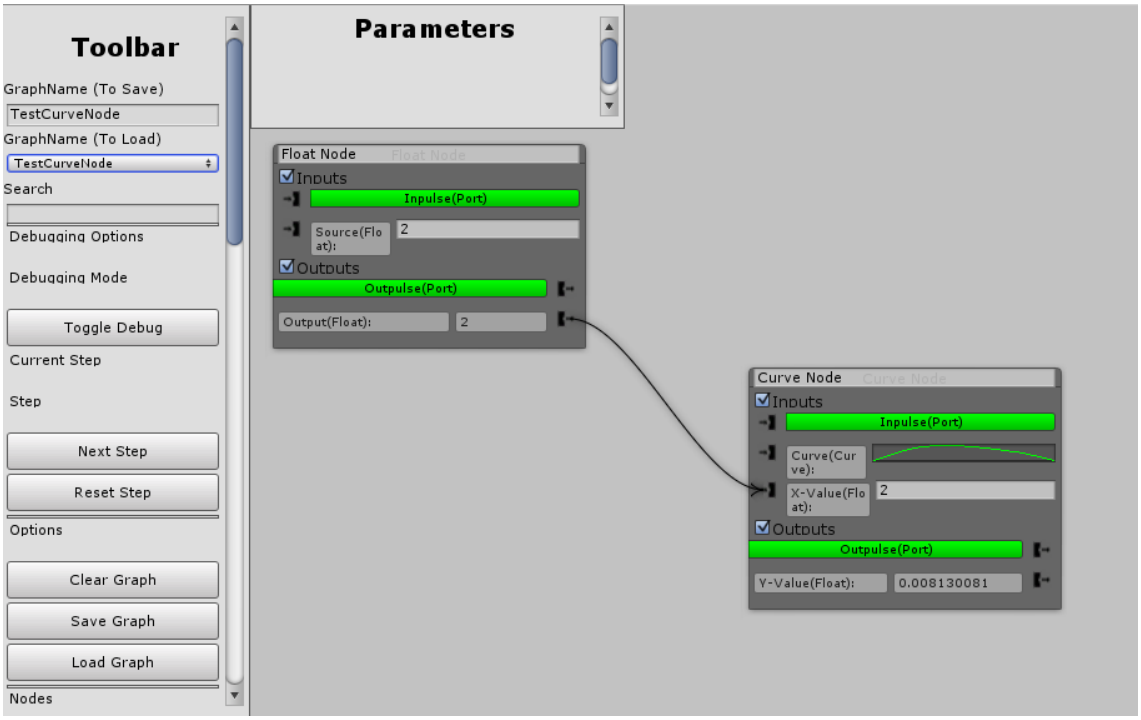


FIGURE 28- IMAGE OF A SIMPLE GRAPH IN GRAPPHER

Figure 28 displays the layout of the Grappher Editor window in Unity. This tools allows for the creation of graphs by connecting nodes. Each node is comprised of a series of inputs and outputs. A node's inputs and outputs are always visible in the editor, starting with the inputs (each of them preceded by an input icon) and with the outputs following close by. Both inputs and outputs display their name, type and current value. Inputs can be filled via the outputs of other nodes or by manually inserting a static value in them. At least one input, called Inpulse, and an output, named Outpulse are present in any given node. The Inpulse port defines if the node should be executed. The Outpulse port will be "true" only if the node was executed correctly. Each input will reference the value of a single output. However, one output may be referenced by several inputs. The execution order of any given graph follows this algorithm:

1. Find a node without any connection to its inputs (there has to be at least one node that follows this rule)
2. Store the node in a list, remove it from current graph (as well as any connections from this node to any other node) and repeat step 1 with the current graph. Go to step 3 when no more nodes are left in the graph.

3. The list now contains one possible execution order for a given graph. Some extraordinary circumstances may create other possible intended execution orders that might require explicit disambiguation.

This ordering algorithm is applied whenever a new node, or connection between input and output is designed. As said, some particular circumstances may occur where the intended execution order is not explicit.

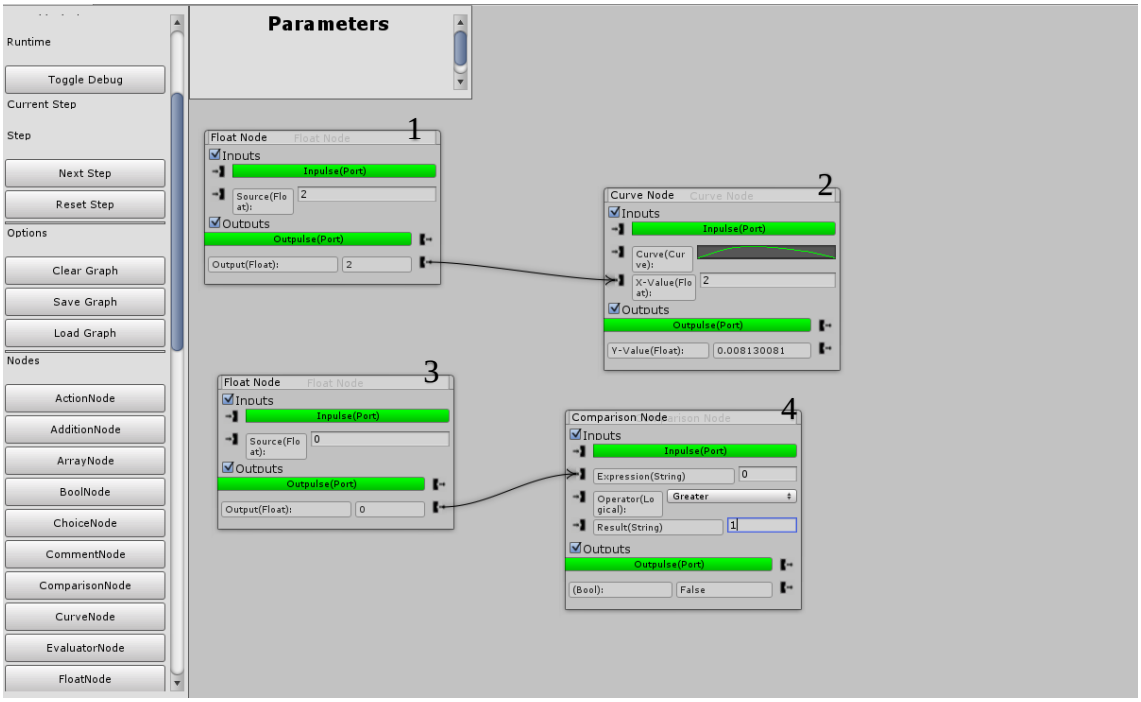


FIGURE 29- EXECUTION ORDER AMBIGUITY IN GRAPHS

Situations where one graph ends up having two inner graphs that are not interconnected (thus effectively having two distinct graphs) has no clear execution order (Figure 29). As such, for the above example, possible execution orders, following the aforementioned sorting algorithm are:

- 1→ 2 → 3 → 4
- 1→ 3 → 2 → 4
- 1→ 3 → 4 → 2
- 3→ 1 → 2 → 4
- 3→ 4 → 1 → 2
- 3→ 2 → 4 → 2

In this particular example, choosing one execution order over any other will have no influence, as no racing condition exists. However, there are situations where it might occur (Node 2 changes a game variable, and Node 3 makes use of that variable to pass to Node 4, for instance) and, as such, an execution order needs to be made explicit.

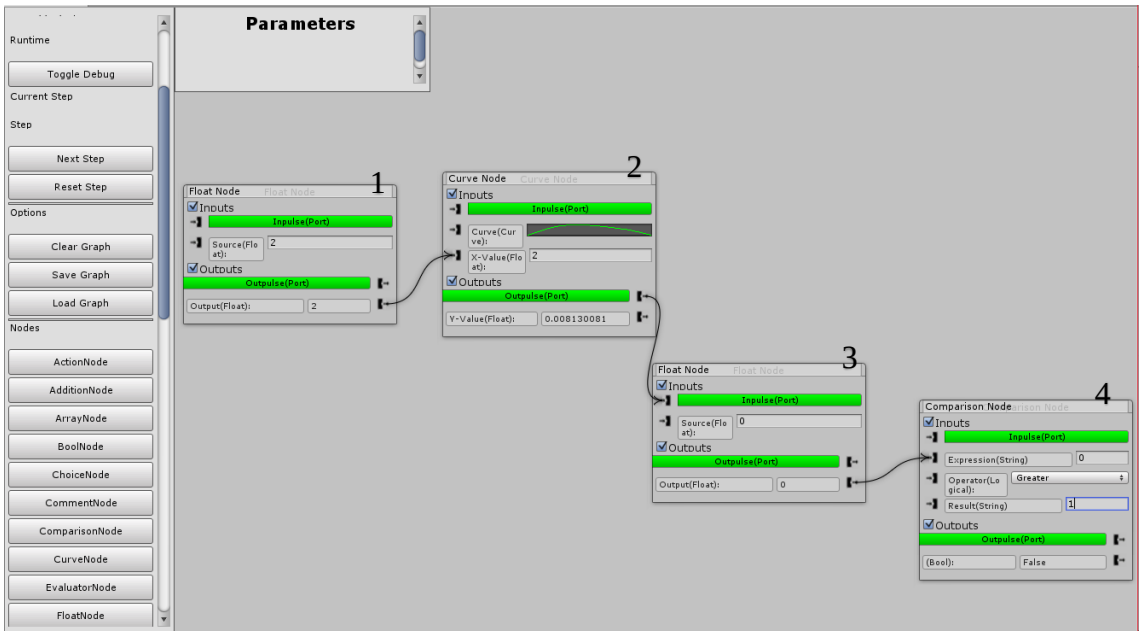


FIGURE 30- EXECUTION ORDER DISAMBIGUATION IN GRAPHS

Since Outpulses are only made "true" once a node has finished executing, and Inpulses require a value of "true" in order for their respective nodes to be evaluated, connecting the Outpulse of a node to the Inpulse of another node means that the former node must precede the latter. In Figure 30, node #2's Outpulse is connect to node #3's Inpulse, explicating the intended precedence.

On the left side of editor window, there is a toolbar (Figure 28) . This toolbar is responsible for the placement of different nodes, saving, erasing and loading a graph. It possesses a quick search bar, allowing for the filtering of toolbar options or nodes. The toolbar also allows for the debugging of a graph to be enabled. This debugging allows for a graph to be executed only up to a specific node. There are several possible states a graph may have.

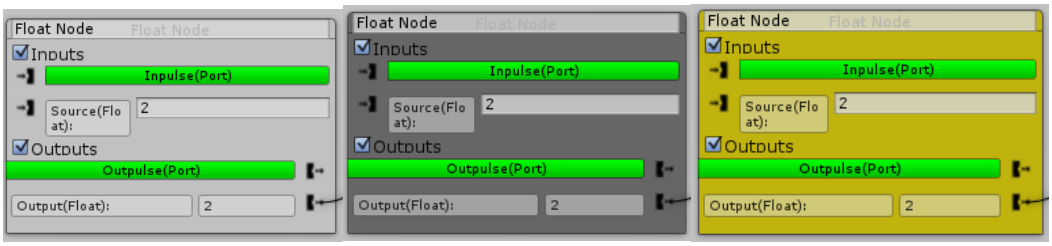


FIGURE 31- 3 POSSIBLE VALID NODE STATES, FROM LEFT TO RIGHT: "EXECUTED", "DID NOT EXECUTE" AND "WAS STEPPED OVER"

A node may have three valid execution states:

- **Executed:** meaning that the node was executed with no errors in the current iteration
- **Did not Execute:** all nodes start of in this state. If they end up being executed (if their Inpulse port is set to true), it will change to the Executed state. This allows the designer to have a "code coverage" view of the graph, being able to see which parts of the graph are being currently executed and which ones are not.
- **Stepped Over:** in debug mode, this highlights which nodes have been "stepped over". As the user presses the "Next Step" button, the next node in execution order will be evaluated, while the previously evaluated ones remain in this state.

A fourth, invalid state also exists, a node error state. The node is also drawn in yellow, and a log in the console is shown. The graph will attempt to execute the next node in the graph, meaning that there is a certain level of robustness. For instance, arithmetic operations that might result in error (for example, dividing by zero) will lead to this state, even though the resulting error is not critical, and the execution may continue. This, however, if left unattended can result in bugs or other errors along the execution of said graph.

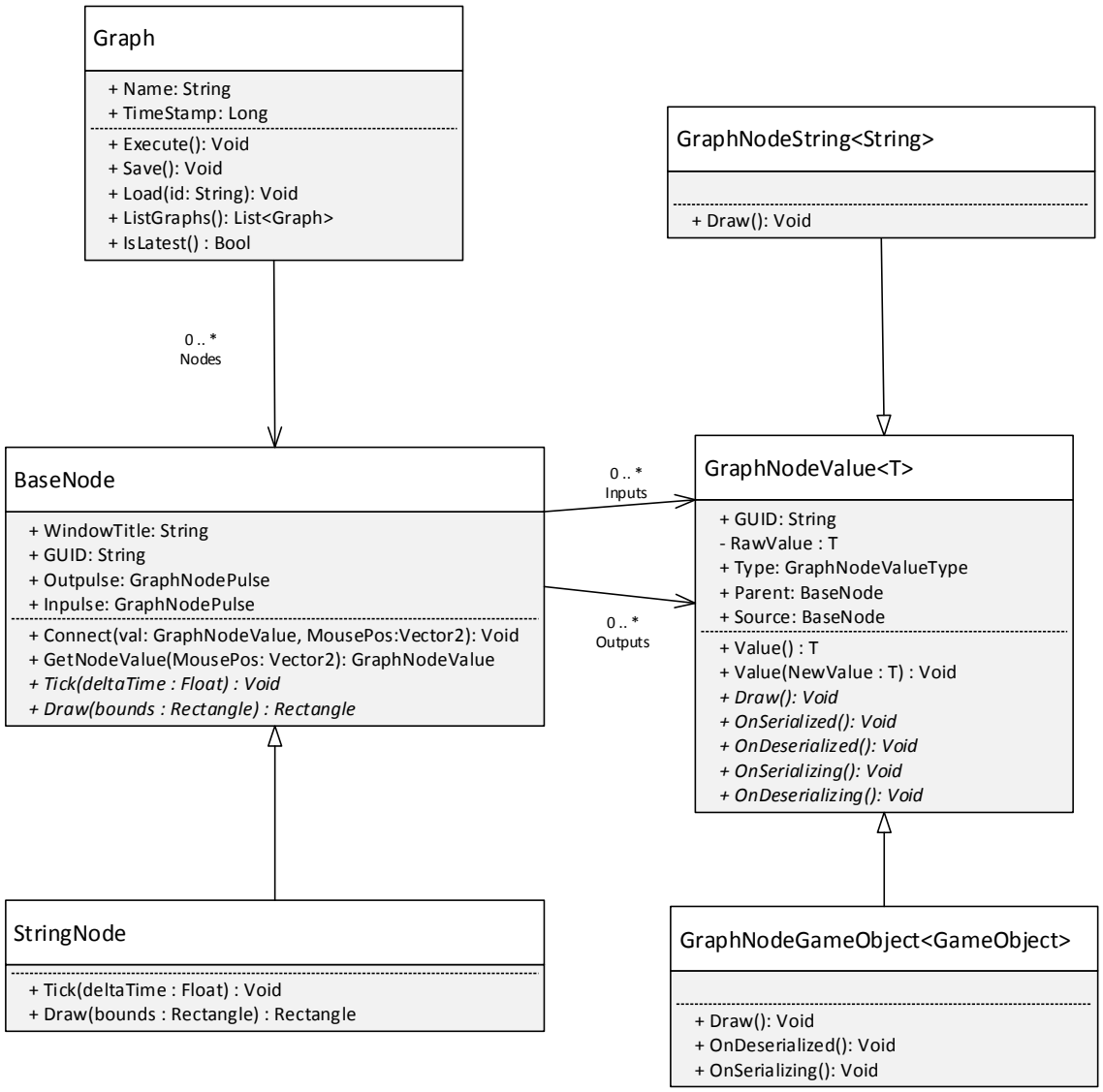


FIGURE 32- UML CLASS DIAGRAM DEFINING THE MAJOR GRAPPHER CLASSES

Each Node in Grappher is usually comprised of several GraphNodeValues, as either inputs or outputs. The Inpulse and Outpulse of a Node, previously mentioned, are also a specific subclass of GraphNodeValue, GraphNodePulse.

The possibility for developers to create possibly needed Nodes and NodeValues was contemplated in scenarios where the implemented Nodes and NodeValues are not sufficient. As such, these were designed with expansibility and coherence in mind, as C# generic design patterns are enforced, particularly in the GraphNodeValues subclasses.

The overall structure of a graph in Grappher is described in Figure 32 . This covers the most relevant methods of each class. Graphs can be simply described as aggregations of BaseNodes, which, in turn, are usually comprised of several GraphNodeValues. The figure also presents some examples of GraphNodeValues subclasses, the GraphNodeString

and GraphNodeGameObject. While the GraphNodeString is simple a class that actually consists of the GraphNodeValue generic with a String type, and works with no other change, other than having to implement the virtual Draw function, the GraphNodeGameObject class dos implement the virtual OnDeserialized and OnSerializing methods. Since the GameObject class belongs to Unity.GameEngine namespace it is not serializable. As such, when serializing a Graph, each GraphNodeGameObject will store, not the GameObject reference itself, but a unique id that referes to that particular GameObject. When being deserialized, the GraphNodeGameObject will search the scene for that particular GameObject, and will set its Value to the GameObject it finds. Other GraphNodeValue subclasses that make use of unserializable classes (mostly from the Unity.GameEngine namespace) use similar approaches, serializing only the needed values to recreate or lookup the Value they hold (such as the GraphNodeAudioClip or the GraphNodeCurve).

Grappher does not currently enforce strong typing when connecting outputs to inputs of different types. In order to do so, it follows this algorithm for converting a Value of a current Type to its intended Type:

- If the Value is already of the needed Type, simply return said Value.
- Execute a known, programmer defined, converter function between the two Types (current and intended), if it exists, returning the converted Value.
- If it is a conversion between an Enum and a String, it will parse the String as the given Enum type and returns that Enum value (or String representation of it).
- If the intended type is String it will execute the ToString() function of the current Value.
- The last attempt is to make an ChangeType call to System.Convert for the current Value to the intended Type.
- If no conversion was possible, an exception occurs.

Several GraphNodeValue derived classes are available, with further details present in Annex B (Grappher Node):

- GraphNodeAction
- GraphNodeArray
- GraphNodeBool
- GraphNodeCurve

- GraphNodeDropDown
- GraphNodeFloat
- GraphNodeImpulse
- GraphNodeInt
- GraphNodeBooleanOperator
- GraphNodeMaterial
- GraphNodeObject
- GraphNodeString
- GraphNodeVector2
- GraphNodeVector3

Generic Nodes, derived from the BaseNode class were also implemented, based on the needs of the developed prototypes. Details about their functions are present in Annex C (Grappher Core Nodes)

- ActionNode
- AdditionNode
- ArrayNode
- BoolNode
- ChoiceNode
- CommentNode
- ComparisonNode
- CurveNode
- EvaluatorNode
- FloatNode
- FloatToStringNode
- ForNode
- GameObjectSelectNode
- GrappherNode
- IntNode
- MultipleChoicesNode
- ParameterGetterNode
- ParameterSetterNode
- PlayerDataNode
- RandomNode
- StringNode
- StringToFloatNode
- TimerNode
- Vector2Node
- Vector3Node
- WebRequestNode

Specialist Nodes are also currently made available in the Grappher Editor Window. These Nodes exist in different packages:

- **GeoSensors Nodes,** based on the GeoSensors API definition of Sensors and their Readings. The execution of these nodes is dependent on the definition of their sensors in the GeoSensors API, their availability and provided readings. They are described in detail in the Annex D (Grappher GeoSensors Nodes).
- **GeoStream Nodes**, based on the GeoStream API, so as to provide easy access to geographical data. This data can be later used to parametrize subsequent nodes or graphs. They are described in detail in the Annex E (Grappher GeoStream Nodes).

In addition to implement a new class that derives from the BaseNode class, Nodes can also be created from encapsulating other nodes. This can be used to simplify sections of a graph that have multiple connected nodes responsible for doing a single task.

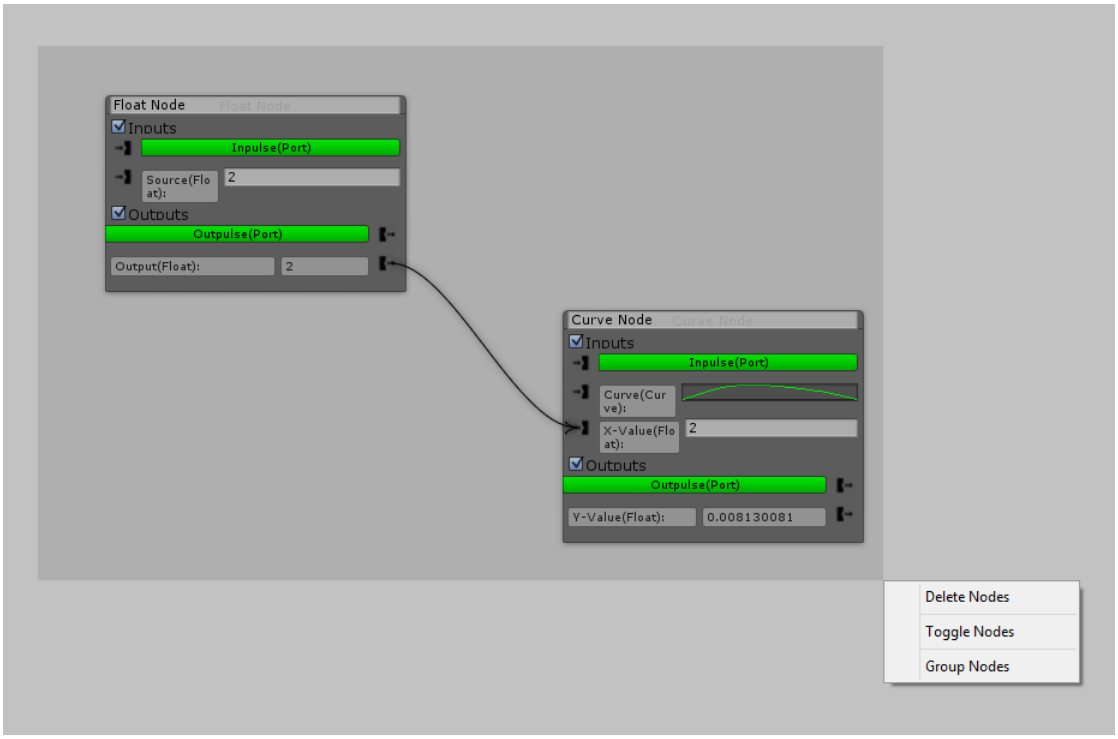


FIGURE 33- CONTEXTUAL MENU FOR MULTIPLE SELECTED NODES

As Figure 33 shows, it is possible to drag and select multiple nodes, allowing for the options of deleting, disabling or grouping said nodes. In the specific example above, the user is grouping two nodes (a Float node and a Curve node) that are connected.

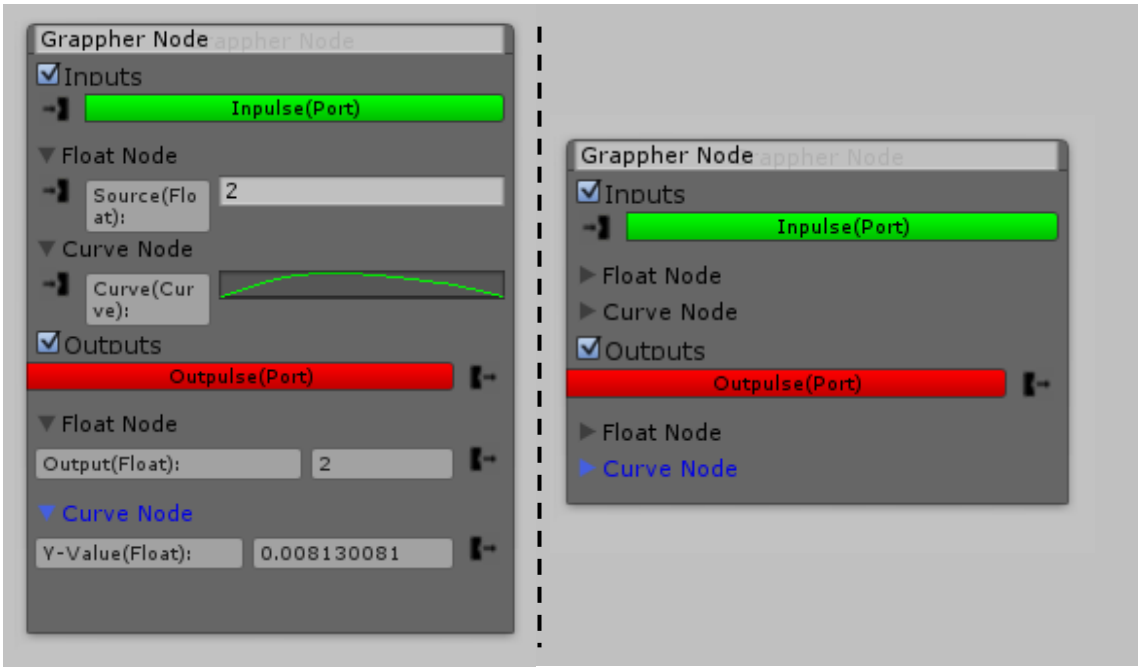


FIGURE 34- EXAMPLE OF AN ENCAPSULATED SET OF NODES (EXPANDED AND COLLAPSED)

The result (see Figure 34) is a new node that contains all the unconnected inputs and outputs of the nodes that comprise it, grouped under the node that originally contained it and that is functionally equivalent. This approach, although potentially visually simplifying the graph (and its readability by humans) hides the connections between the nodes that comprise the encapsulating node and their occupied inputs (in this example, the input "X-Value" from the curve node is missing, as it is being used). This also means that the node must be broken into its encapsulated graph if the user intends to either see how it works (what connections comprise it) or if one wants to edit it.

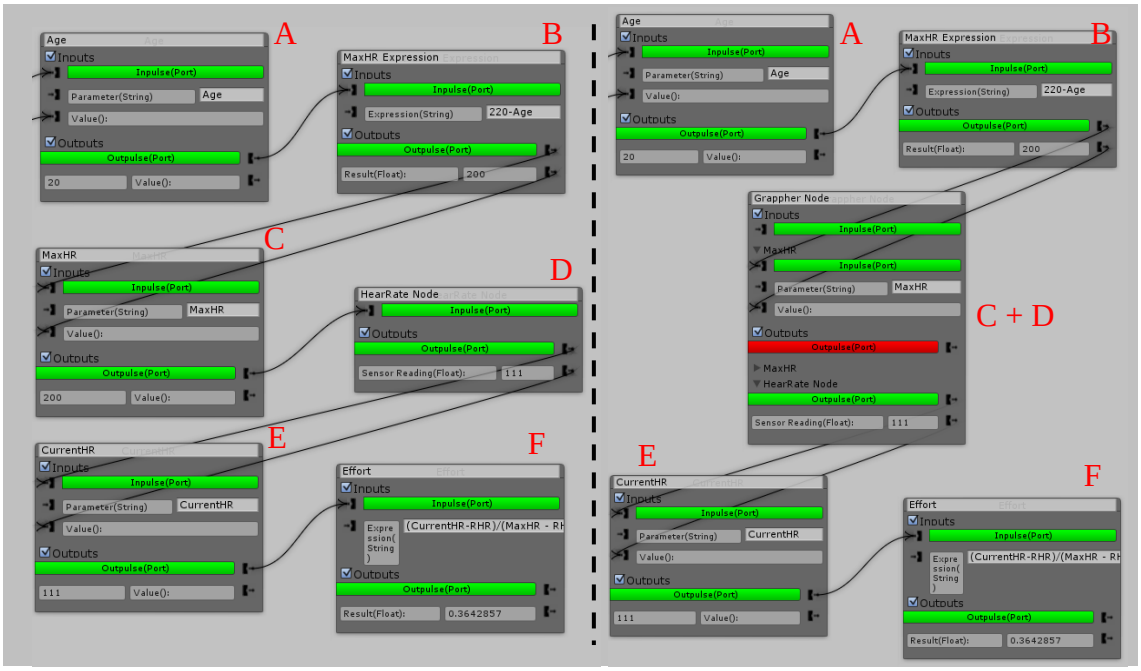


FIGURE 35- PARTIAL GRAPH ENCAPSULATION (BEFORE AND AFTER)

Figure 35 shows a more realistic scenario where only part of a big graph was encapsulated. The nodes MaxHR and HeartRate were merged into a single one. Notice how the Outpulse port of the HeartRate node, now encapsulated, is exposed, as it is used as an input by the CurrentHR node. All inputs and outputs required by the rest of the graph from the newly encapsulated node are exposed, allowing for the graph to be functionally the same. The GrappherNode, explained in Annex C (Grappher Core Nodes), takes this encapsulation a step further, allowing for graphs to invoke other graphs within a single node, with a behaviour similar to this method of grouping nodes. This allows for graphs to be reusable, and for a user to abstract from knowing how a graph works in order to use it.

Whenever a graph is saved, the editor will serialize it (via the JSON.NET library) and upload it to a MySQL database for later use. Only then can a designed graph be executed outside the Grappher Editor Window. In order to use any created graph in a Unity application, the GrappherLoader component must be added to a GameObject that will execute the needed graph for as long as needed. Other GrappherLoader components may also be added (to the same object or others) so as to execute additional graphs.

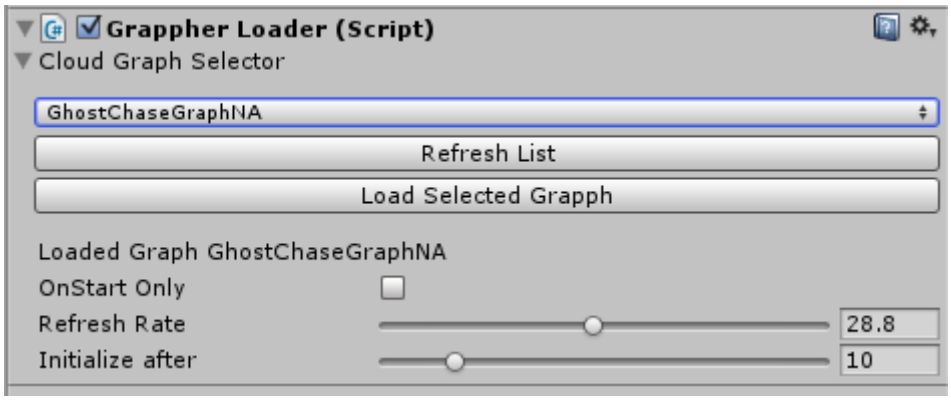


FIGURE 36- GRAPPHER LOADER COMPONENT IN UNITY'S INSPECTOR VIEW

The Grappher Loader allows for a user to select any saved graph from a dropdown list and mark it for execution. The execution itself can either be only once (on start) or for as long as the GameObject this component is attached to exists, executing the graph once per frame. Some other parameters allow for the user to specify when a graph should be executed, in seconds, after the GameObject was instanced (useful for when loading a scene guaranteeing that all the needed resources have loaded) and to choose weather and when should the Grappher Loader query the server for new versions of the graph it is executing (it asks the server if the current graph's timestamp is the latest. If not, the server responds with the new graph's content). This allows for changes that are made in the Grappher Editor Window that are saved to a graph that is already in use in a deployed application to be "updated" with no redeployment needed. During the development of the prototypes of this thesis, as all of them are mobile applications, the possibility of tweaking a graph's parameters and adding new nodes without having to redeploy the whole application proved to be useful. However, in circumstances where some code was changed, it would still need to be redeployed. A simple example is a C# function that was written to be invoked via an ActionNode. In the editor, it works correctly as the function is available. In the remote application, since it would still not have that new function, the graph execution would not work as intended.

Grappher shows that it is possible to standardize sources of information for games via graph nodes, allowing for information processing and game mechanics to be bridged with little to no programming needed (as per **RQ5**).

7 CASE STUDIES

To assess the applicability, usability and effectiveness of the framework, two case studies were designed, consisting of mobile games. These were developed, in a first stage, without any reference or usage of the SPACIAL framework. The framework was only included later, to show how it can be easily adopted by existing games without specific requirements. This allowed for one to informally test how the integration of such games with the framework would be. GhostStand, one of the implemented games, is a mobile Virtual Reality exergame (using a mobile Head Mounted Display, smartphone and smartwatch), that places the player in an immersive environment, where the player must fend off restless spirits with a virtual sword.

In order for the player to wield the sword, a smartwatch, responsible for tracking the player's movements, must be worn. This game was developed as it would be independent from the location, time of day or weather conditions. The other game, TrackDay, has the player jogging, attempting to escape pursuing enemies, similar to the Zombies, Run! game.

After developing these games, and deeming them playable, the SPaCiaL framework and its tools were then used to develop and test several adaptive game-configurations. The games and some of those configurations are presented and explained in detail below as a showcase of what the capabilities of the SPaCiaL framework's tools are.

7.1 GHOSTSTAND

The GhostStand game is a VR (Virtual Reality) First Person Exergame where the player must fend off enemy ghosts from his/her position, using a sword.

FIGURE 37- USER PLAYING GHOSTSTAND WITH A MOBILE HMD, HEADPHONES, SMARTWATCH AND SWORD

In reality, the user is equipped with a mobile head mounted display, headphones, a smart-watch and an improvised sword as per Figure 37. The game requires the player to be alert as the ghosts can come from different directions. Interactions are limited to looking around (3DOF) and swinging the sword (3DOF) at the incoming enemies.

FIGURE 38- CONCEPT ART FOR THE GHOSTSTAND GAME

As Figure 38 shows, a dark and gloomy atmosphere characterizes the game from an early start. This was to help minimize possible problems with immersion or discomfort, as with current mobile VR technology it is not uncommon for the space between pixels to be visible to the naked eye. So, a darker image helps in masking this effect.

The game has the following characters and items:

- **Ghosts**: these characters are the enemies in the game. They are attracted to the player and will attempt to kill him.
- **Pillars of Light**: four of these structures are present in the scene. They serve two purposes. To serve as beacons, making the orientation of the player relative to them an easier task, and as a spawn location for the Ghosts.
- **Restless Samurai Spirit**: the player embodies this character, a spirit continuously tormented by the Ghosts
- **Spirit Sword**: the player's character wields a sword capable of defeating the incoming Ghosts. It can be used defensively, by blocking the Ghosts' attempts to harm the player or by shoving them away, or offensively, by violently hitting the Ghosts, banishing them definitely.

- **Gloomy Moon**: A big, dark moon rests low and centred between the four Pillars of Light. It serves no purpose, other than providing ambiance and as a very specific point of reference to the player, as it is used for the calibration process of the sword.

These game elements behave in accordance to the following mechanics:

- **Pillars of Light:**
  - Spawn Ghosts at certain intervals
  - Spawned Ghosts share the same characteristics, within parametrized limits (speed, hit speed resistance and time to live)
- **Ghosts:**
  - Have varied maximum speeds, resistance (the minimum sword swinging speed they must be hit at in order to die) and a TTL (time to live) of 40 seconds.
  - Will attempt to converge at the Player's position. They will gravitate towards the player, from their own spawn points. However, they are susceptible to bumping against the ground, each other, the Player, and the sword, meaning that their initial straight trajectory can become curve or elliptical.
  - When bumping against the player, will remove one hit point from the player's life points pool.
  - When hit, will either flash in a red colour and move away from the player, if the speed it was hit was less than their resistance, or, if said speed is greater than their threshold, they will instantly die.
  - Cast a red shadow on the platform the Player's avatar is at, making it easier to guess their distance and position.
- **Player:**
  - Can freely look around
  - Can move the sword around by moving his/her right hand around the three axis (the player cannot, however, thrust the sword forward)
  - Can block Ghosts from attacking, by placing the sword between the ghosts and himself/herself
  - Can push away Ghosts by swinging or flicking the sword gently
  - Can kill Ghosts by hitting them with great arm speed

- Is awarded points based on the speed at which each Ghost was killed
- Can die when player life points pool reaches zero points, from an initial 100 points
- Can re-align his/her field of view with the sword by looking at the Gloomy Moon, placing his/her right hand on the chest and, with the left hand, touching the Smartwatches display. A small vibration is felt on the wrist, indicating that the calibration process was concluded

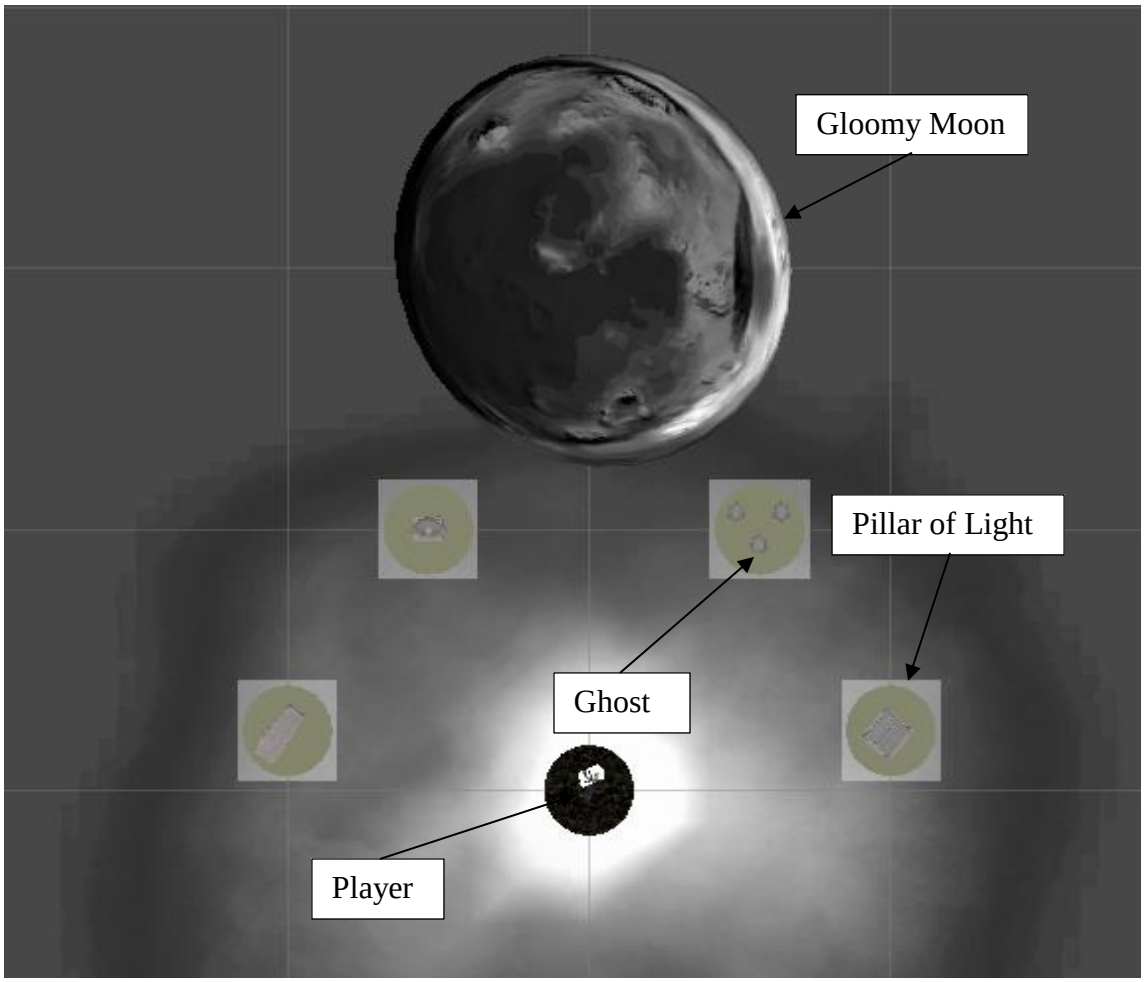


FIGURE 39-LAYOUT OF A LEVEL IN GHOSTSTAND

As Figure 39 shows, the Player is standing between the four pillars of light. Since the player does not know from where the next enemy spawned will be coming from, he/she must periodically look around at the pillars of light.

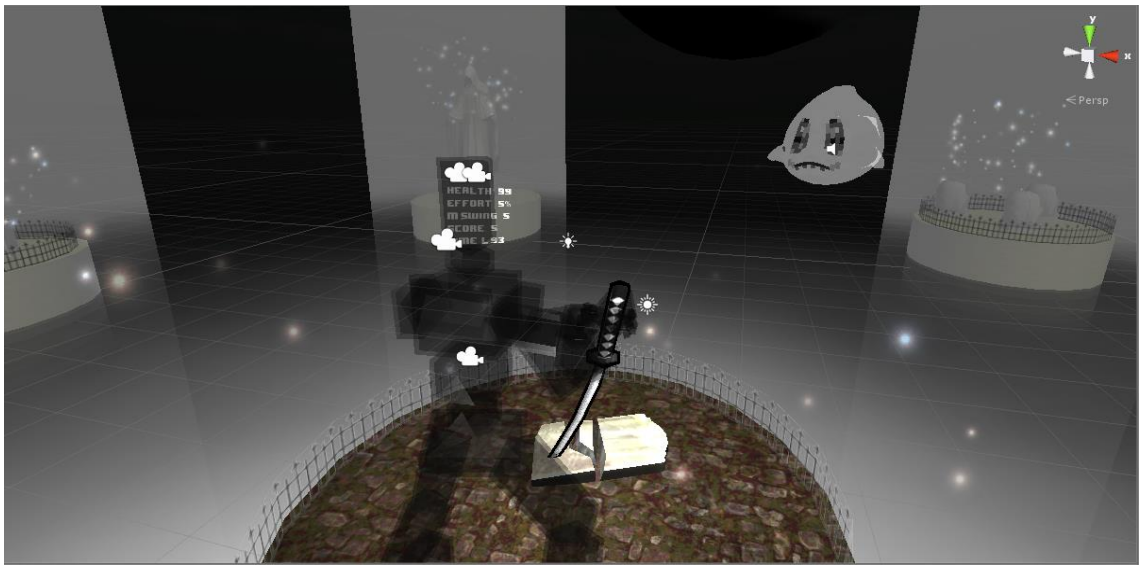


FIGURE 40- THIRD PERSON VIEW OF THE GAME IN DESIGN TIME

Figure 40 shows a perspective view of the game, in design time. The player's avatar body is translucent, so as to giving the player the ability to see his/her avatar's body, in first person, but without intruding too much on the game. Also noticeable is that the avatar's arms and wrists follow the swords orientation. This is thanks to the avatar being fully rigged as a humanoid, allowing for the game engine's (Unity3D) inverse kinematic features to be used, fully. As such, as the player moves around, so do the avatars hands, arms and shoulders, further enhancing the player's experience. The size of the platform the player is standing at is roughly the same length of that of the sword. This means that whenever the ghosts cast their red shadow on the platform, they will most likely be in range of the player's sword.

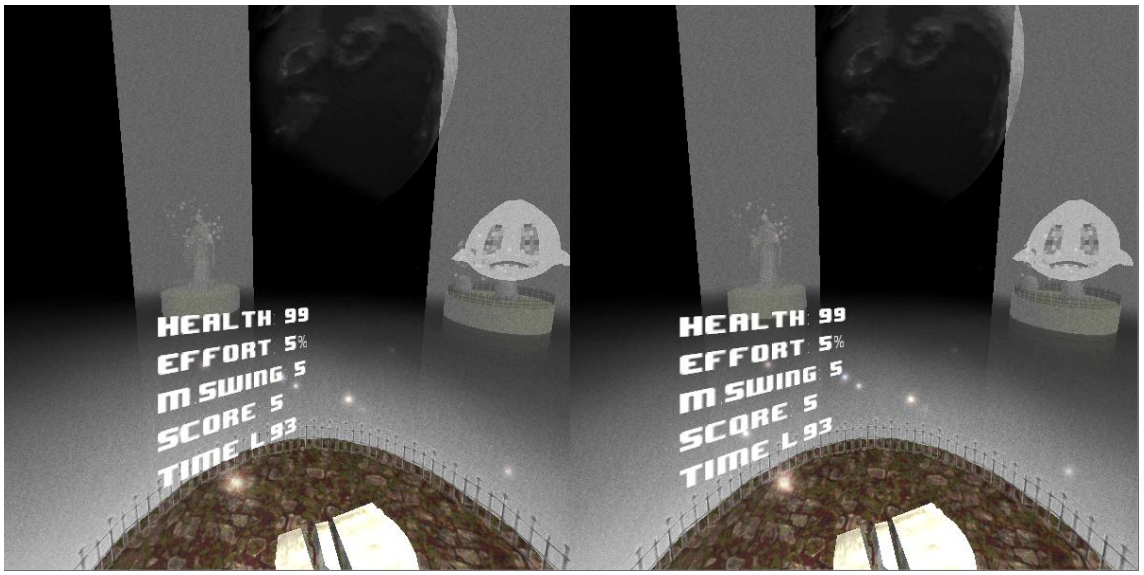


FIGURE 41- IN GAME SCREENSHOT OF GHOSTSTAND

Figure 41 is representative of what the players see when experiencing the GhostStand game. In this particular screenshot, the player did not have the sword within the view. The game's UI present in this screenshot is meant for debugging only and was changed for the experiments, by removing Effort, Maximum Swing Speed and Time Left from it, and leaving only Health and Score for the player to see.

A slight vignette effect and monochromatic noise were added to each individual camera image, so as to better hide the current limitations of the smartphone-based head-mounted displays used, as they usually have very big, visible pixels and the dark space between them is also noticeable. Sound effects for ghost spawning, dying, and hitting the player were added, as well as for the sword swinging and hitting the ghosts. An eerie music with random sound effects (such as heavy breathing, maniacal laughter and storm brewing) was added in order to muffle external sounds and enhance the experience.

7.1.1 GHOSTSTAND SPACIAL INTEGRATION

In the previous Figure 14, the overall design and interaction between SPaCiaL components has been specified. Considering that the GhostStand game is not a location-based game, it will not make use of the Location component and thus, it will not make use of the GeoStream component. As such, one can only make use of the Skill, Physical Prowess and Challenge variables present in the proposed framework for this game. The game required access to both the mobile phone's orientation sensor (for determining the direction the player is looking at) and the smartwatch's orientation sensor (for determining the orientation of the player's right hand so as to infer the orientation of the sword). Additionally, access to the smartwatch's touchscreen sensor was also needed for the purpose of calibration. The first SPaCiaL profile created was the GhostStandB profile (Non-Adaptive) to be used later in the GhostStandT and GhostStandB testing versions of the game.

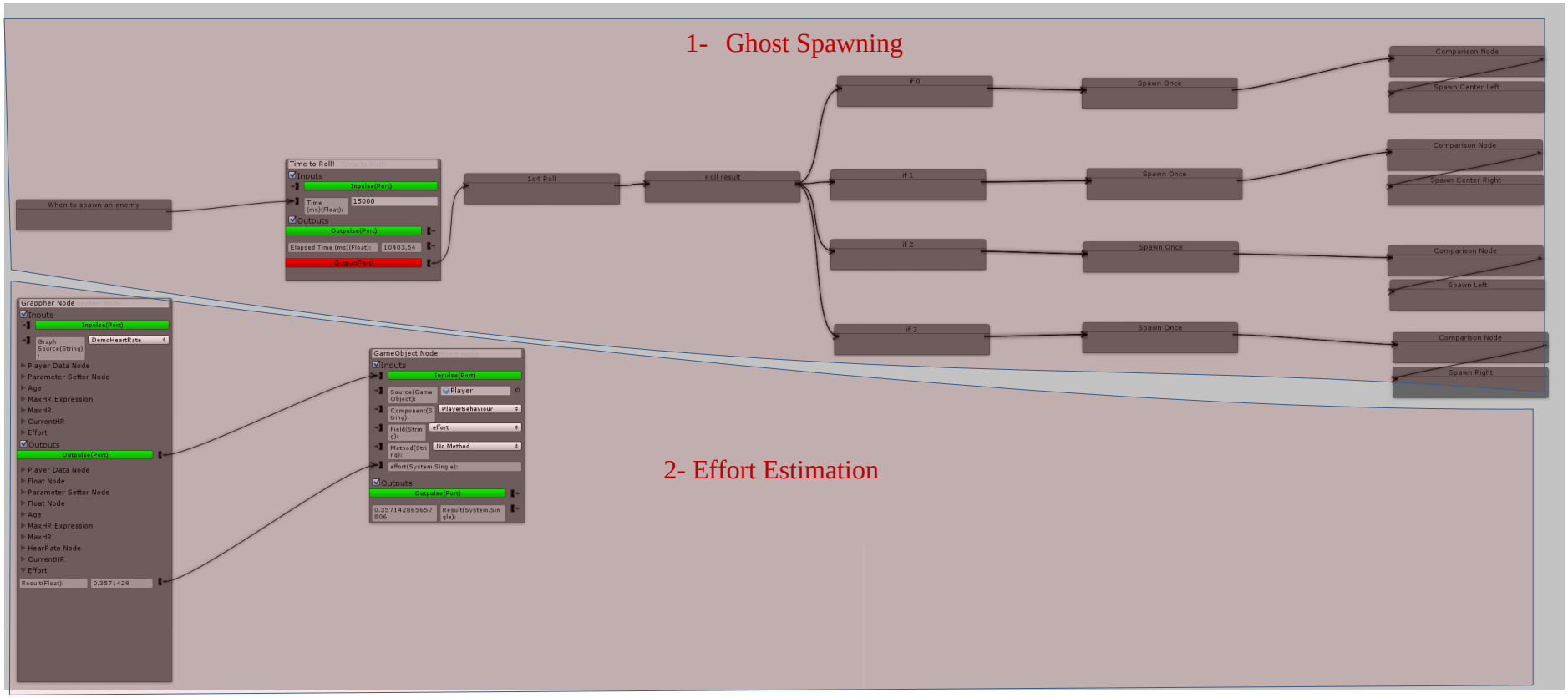


FIGURE 42- GHOSTSTAND NON-ADAPTIVE PROFILE CREATED IN GRAPPHER

The non-adaptive profile contains two sub-graphs. One, responsible for controlling the game's spawning behaviour and another for calculating the player's current effort, as seen in the above Figure 42.

The upper sub-graph (1- Ghost Spawning) is tasked with the spawning of ghosts. Note that in order to improve readability most nodes are collapsed. That graph, readable from left to right, does the following operations:

- Define a constant spawn interval (15000 milliseconds)
- Define a recurring timer that triggers an Outpulse port every 15 seconds
- When triggered, a random number from 0 to 3 is generated
- If the number is 0, 1, 2 or 3 it will spawn a Ghost, only once, in the Center Left, Center Right, Left or Right spawn areas, through an ActionNode calling the Spawn function of the GhostSpawner Component of the respective GameObject. Ghosts are spawned with the default spawning parameters.

With that simple graph, the spawning behaviour of ghosts is assured, every 15 seconds, from a random spawn point to keep the game somewhat unpredictable.

The lower sub-graph (2) simply sets the effort field of the PlayerBehaviour component. However, in order to do so, it makes use of another previously defined graph (Player Effort Graph), responsible for calculating the current effort level of the player.

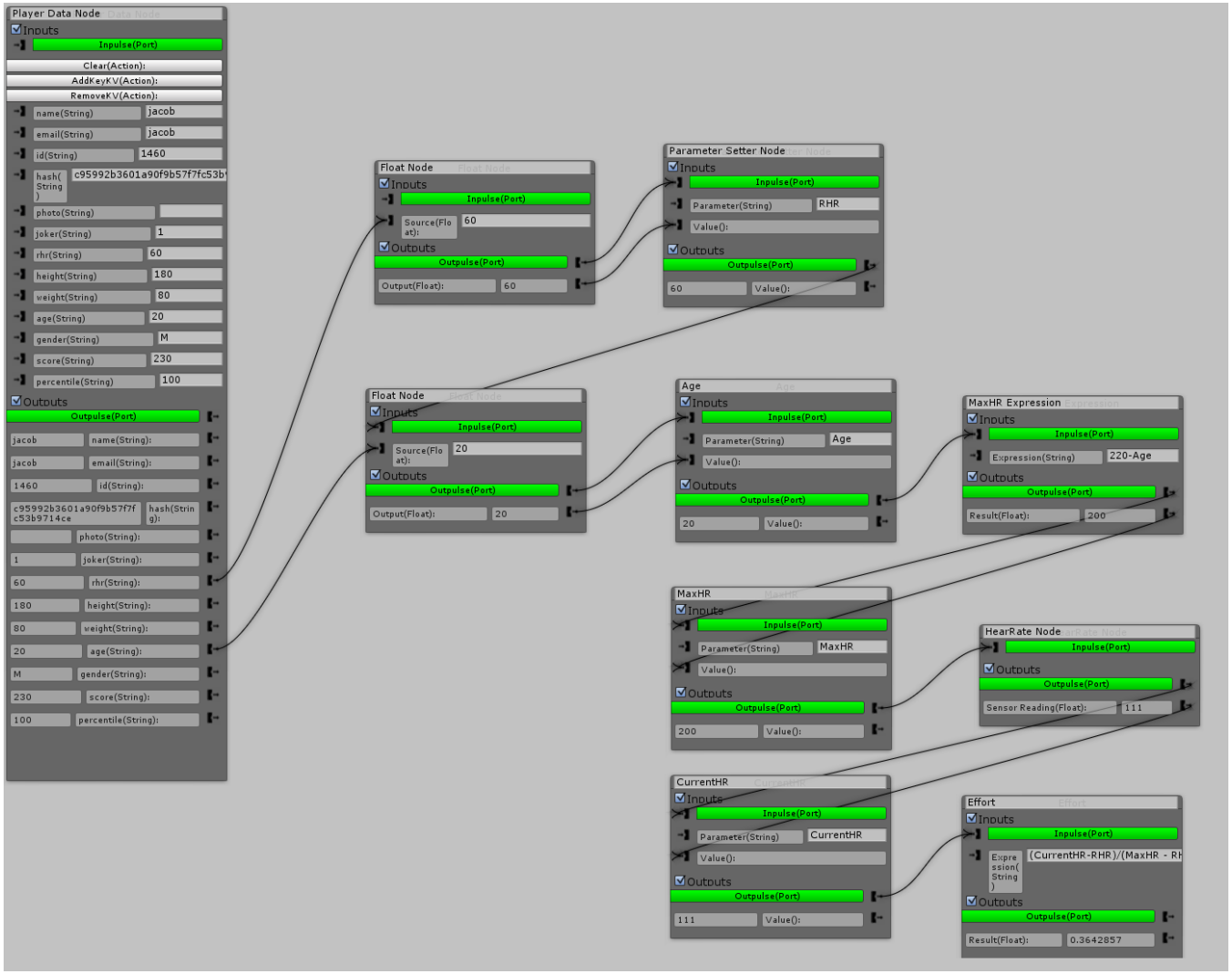


FIGURE 43- PLAYER EFFORT GRAPH CREATED IN GRAPPHER

The graph responsible for calculating the player effort is the foundation for subsequent adaptivity profiles that make use of the player's current effort to vary the game's challenge. Based on some of the player's profile parameters, such as age and resting heart rate, and using current heart rate, the graph is capable of estimating the player's effort, by implementing the equation shown in chapter 3.4.

The graph, most notably, makes use of the PlayerDataNode for getting the current player's data and the HeartSensorNode for getting the latest heart rate reading from GeoSensor. As the effort is used in both games and their profiles as means to estimate the current relative physical exertion, it merited its own graph, making it easy to change its equations or effort calculation whenever needed, without needing to reparametrize other graphs based on it.

So, the GhostStandB graph is able to access the current player state and generate a challenge. However, since they have no relation altogether (as seen in Figure 42), this graph makes no use of the SPaCiaL model for adaptive location-based exergames, even though it uses some of the framework's components separately (Physical Prowess and Challenge). No function relates these two variables.

Another profile, this time an adaptive one, dubbed GhostStandA was created. It was based on the GhostStandB profile. However, instead of the Ghost Spawning time being constant, it varies based on a custom curve and the player's estimated effort.

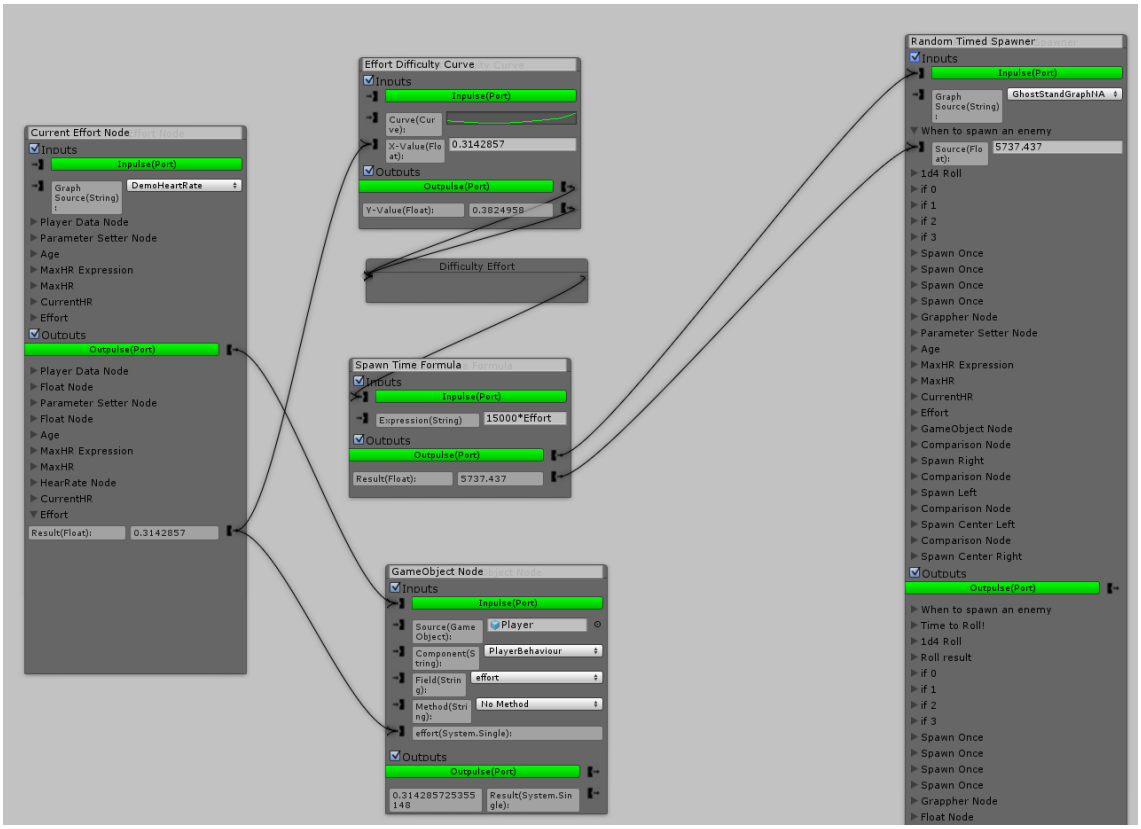


FIGURE 44- GHOSTSTAND ADAPTIVE PROFILE CREATED IN GRAPPHER

Although, by glancing the above Figure 44, this new profile may appear to be simpler, it relies on both the previously specified graphs: the GhostStandB and Player Effort graphs. The Player Effort graph calculates the player's current effort, as explained in Chapter 3.4 .

The difference between the GhostStandB and GhostStandA profiles lies in the spawning time. Whereas in the GhostStandB graph it was defined as having a 15 seconds cadence, in the GhostStandA profile, it is variable, based on the player's current effort and an effort-difficulty curve that defines the function that relates Challenge with Physical Prowess, and thus, making partial use of the SPaCiaL framework.

FIGURE 45- EFFORT (X-AXIS) VERSUS GHOST- SPAWNING-TIME(Y-AXIS) FUNCTION

The curve was defined and iterated over multiple times in order to achieve the desired effect. It defines the target effort of the exercise to be that of approximately 30%, as it can be seen in Figure 45. When the effort is 0% (at the start of the game, presumably), the spawn interval is already roughly 70% of the default values of 15 seconds. As the player plays the game, getting more and more exerted, the game's spawning interval decreases to a minimum of about 40% of the initial value when the player's effort is at 30%. If the player continues to push him/herself, the game will gradually slow down the spawn interval, up to almost double the amount of the default value. However, as the player's heart rate begins to descend, the spawning rate will increase again since the player is now physically available to be challenged more aggressively once more.

Player Skill could also be used parametrize the game, concomitantly with the Effort/Ghost-Spawning function. Skill could be absolutely determined via the current number of ghost kills, current health or maximum striking speed, or relatively via the Game

Data Tool, comparing the current score with other player's or himself. However, for experience purposes it was deemed easier to define only one relationship between two SPaCiaL variables instead of several. One would be enough to show that it is possible to use Grappher, GeoSensors and the Game Data Tool to create an adaptive game. This shows that a graph-based approach can be used to parametrize game mechanics with processed information from multiple sources, answering **RQ5**.

7.2 GHOSTCHASE

The second game, GhostChase, is a mobile location-based exergame. Contrary to GhostStand, where the player is asked to remain in the same place fighting off ghosts, in GhostChase the player must physically flee from pursuing ghosts, reaching a safe haven before they catch him/her.

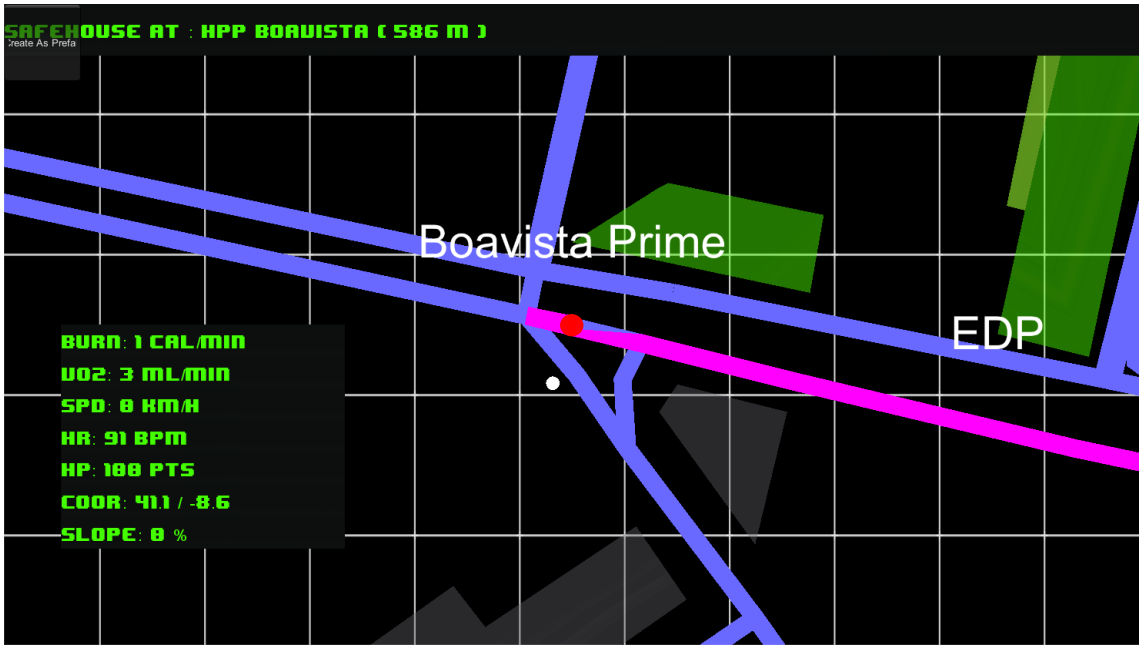


FIGURE 46- GHOSTCHASE LOOK AND FEEL

Once the game starts, the player can see and hear the game (Figure 46). The player is the red dot at the centre of the screen. White dots represent the position of ghosts and the purple line is the recommended path to the safehouse. Additional information (such as burn rate, volume of oxygen consumed and other) is conveyed in the left corner of the screen. Since the game requires the player to run around, it encourages only glancing the phone when absolutely needed, as it guides the player with turn-by-turn navigation to the safehouse. Whenever ghosts are getting closer to the player, a continuous beeping sound is played. The pitch, tempo and volume of that sound increase whenever the ghost gets closer to the player. Conversely, they fade into silence when the player evades the ghost(s). If a ghost touches the player, it will make the player lose health at the rate of one health point per second. If the player's health reaches zero, it is game over. If the player safely reaches the safehouse, the game is won.

GhostChase makes use of GeoStream to generate the 3D buildings, calculate the safehouse position (a hospital in the area, if none exists, the game will only spawn ghosts

and effectively start when the player's surroundings contain one) and determine the shortest known path to it. It also makes use of the GeoSensors library to determine the player's real world position, speed and heartrate.

7.2.1 GHOSTCHASE SPACIAL INTEGRATION

Being GhostChase a Location-Based Exergame, it differs from GhostStand as it makes full use of the SPaCiaL variables (Skill, Physical Ability, Challenge and Location). The game requires access to the smartphone's GPS and the smartwatch's Heart Rate sensor, to determine the player's physical position in the real world and physical condition. The first profile created was the GhostChaseB (Non-Adapting) to be used later in the GhostChaseT and GhostChaseB versions of the game.

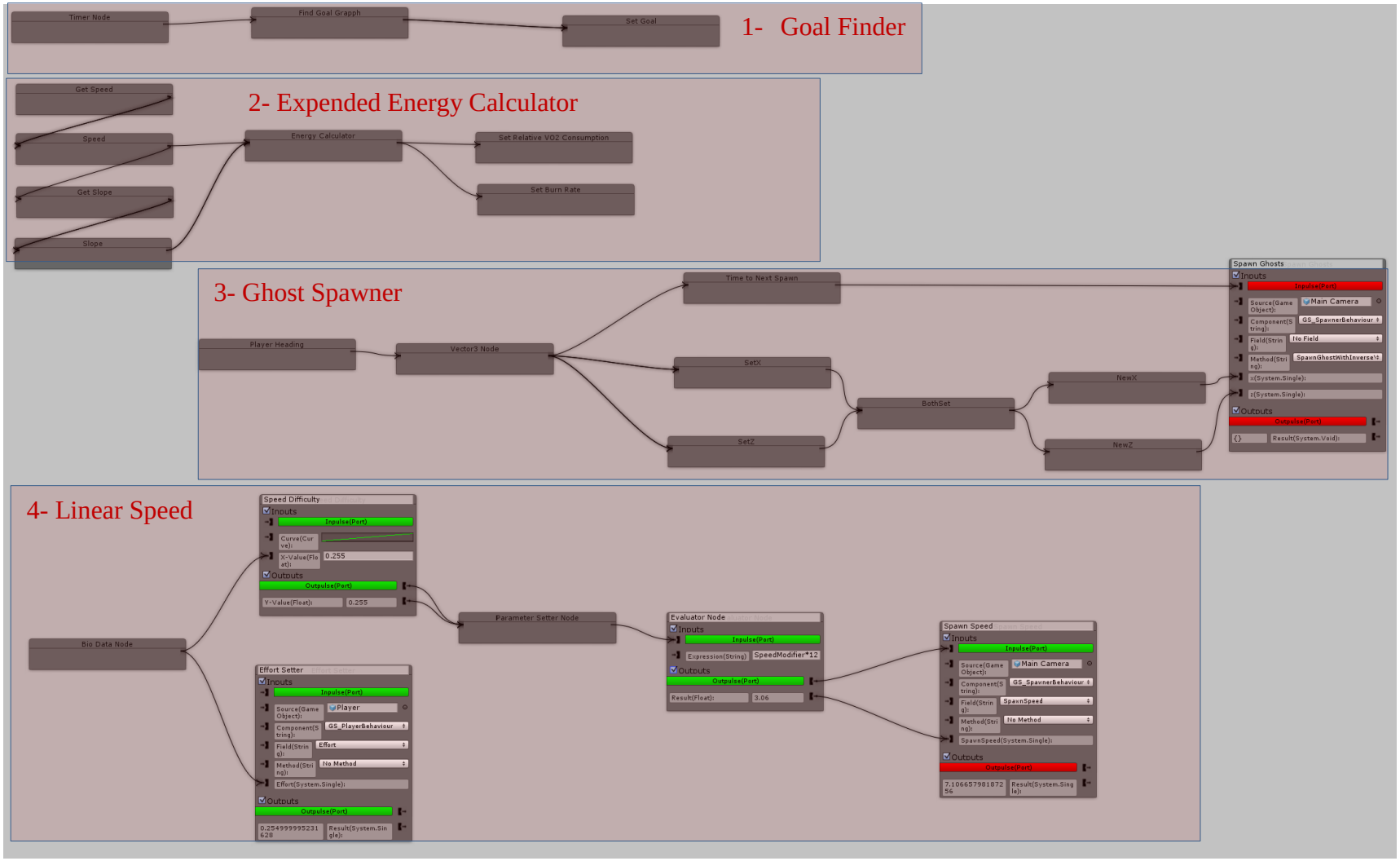


FIGURE 47- GHOSTCHASE NON-ADAPTIVE PROFILE CREATED IN GRAPPHER

Figure 47 depicts a more complex graph than that of the GhostStand Non-Adaptive profile (Figure 42). The GhostChaseB actually contains a series of sub-graphs (numbered 1 through 4 in the above image), that require other additional graphs. In the above image, the upper sub-graph (1- Goal Finder), comprised of 3 nodes, is responsible for periodically looking for new safe houses and setting said safehouse as a goal. It requires the usage of another graph for locating the goal.

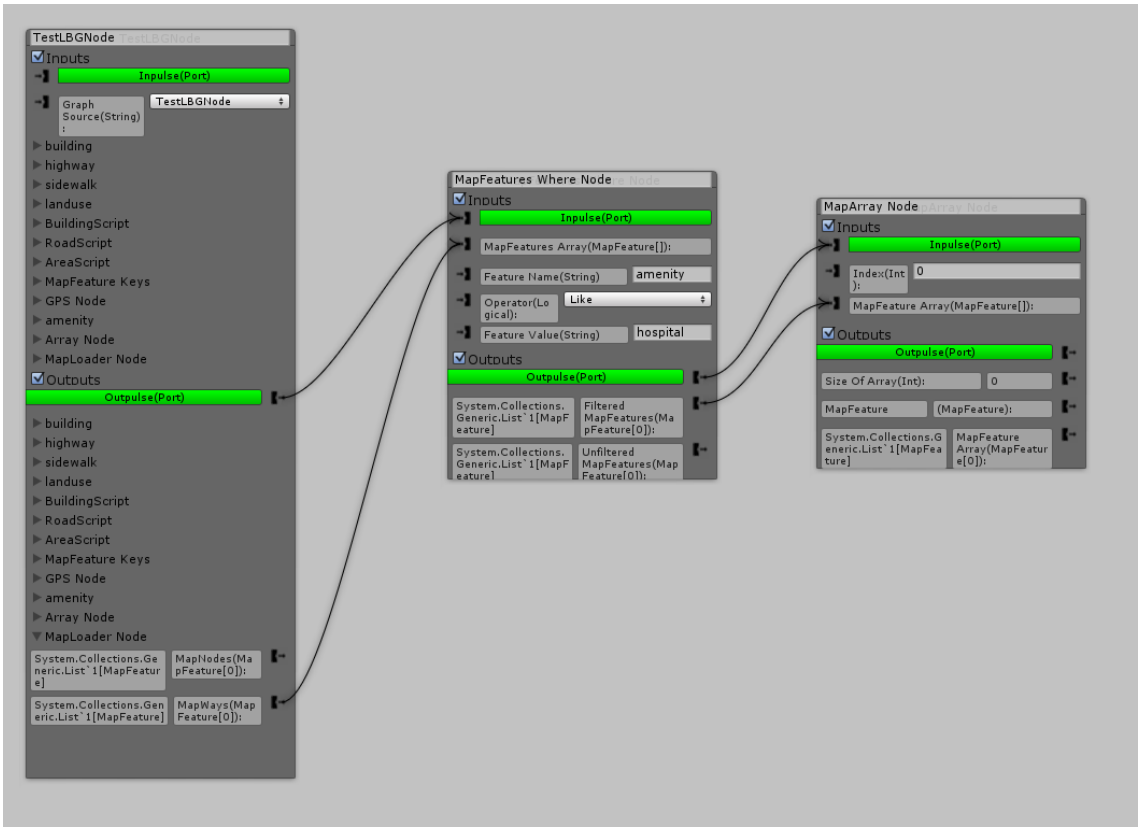


FIGURE 48- GHOSTCHASEFINDGOAL GRAPH CREATED IN GRAPPHER

As specified in the description of the game, the goal of GhostChase is to reach the safehouse. These safehouses are located in the real-life location of hospitals. The above graph (Figure 48) uses the results of an encapsulated graph that contains a MapLoader-Node, filters the MapFeatures that have an "amenity" feature with the value "hospital" and selects the first one of the results. This value is then used in the GhostChaseB graph to parametrize a call to the function "SetGoal". Notice the usage of the graph TestLBGNode graph in the above image.

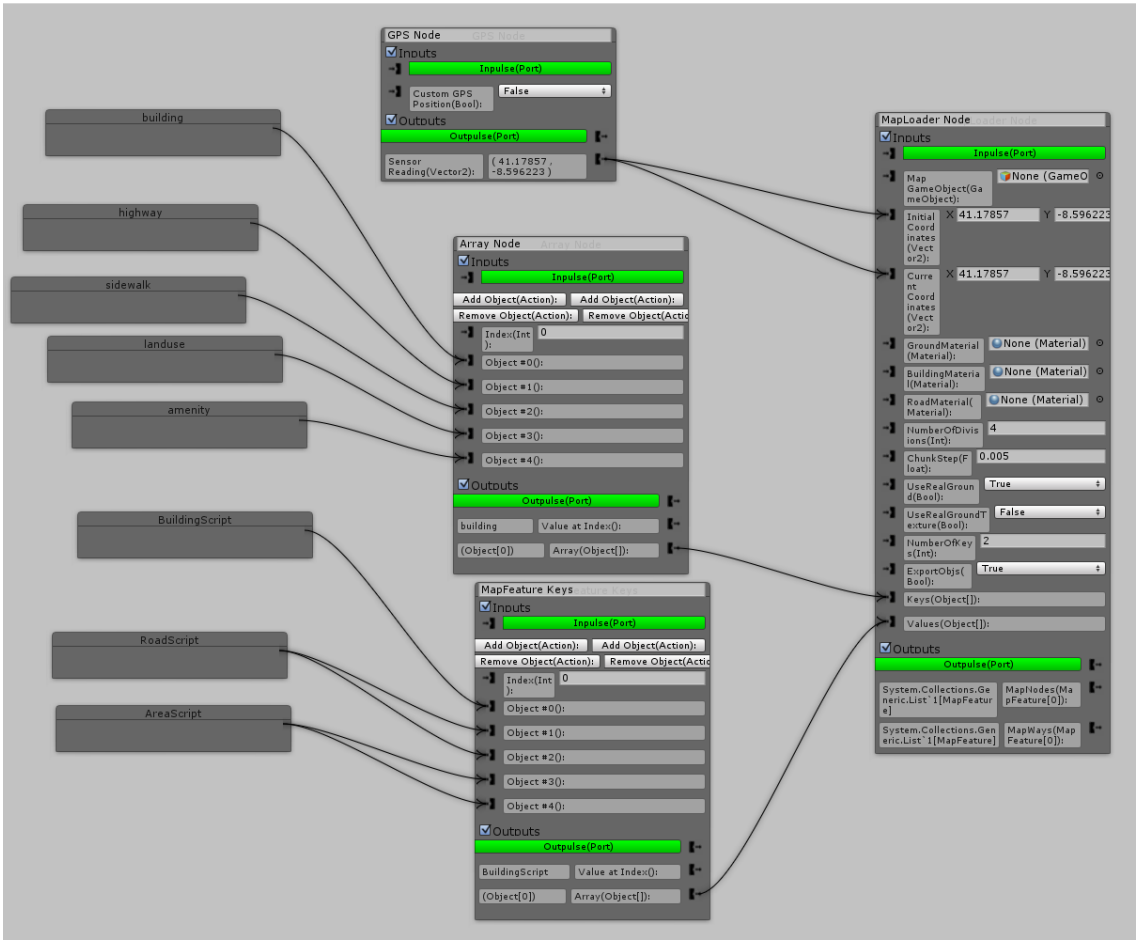


FIGURE 49- TESTLBGNODE CREATED IN GRAPPHER

The TestLBGNode graph was created to encapsulate the creation of a GeoStream's MapLoader, responsible for the creation of the 3D representation of the game's real world scenario and tasked with handling the geographical information of the nearby location.

Figure 49 shows how the MapLoaderNode can be parametrized in Grappher. Note how it is fed the coordinates of the GPSNode, keeping track of the player's location. Two arrays are also created, describing the scripts that will be responsible for recreating certain Map-Features. In GhostChase, buildings, highways, sidewalks and "landuses" (parks and other areas) are represented in the map, as described by the TestLBGNode.

Now, for the remaining sub-graphs of Figure 47. The next subgraph (2) is responsible of displaying the player's energy consumption, calculating the consumption using some established formulas. For the Energy Consumption graph to be correctly used, it requires the parameters of speed (default value of 100 m/ min), slope (default value of 0) and player weight (default value of 100 Kg) to be known.

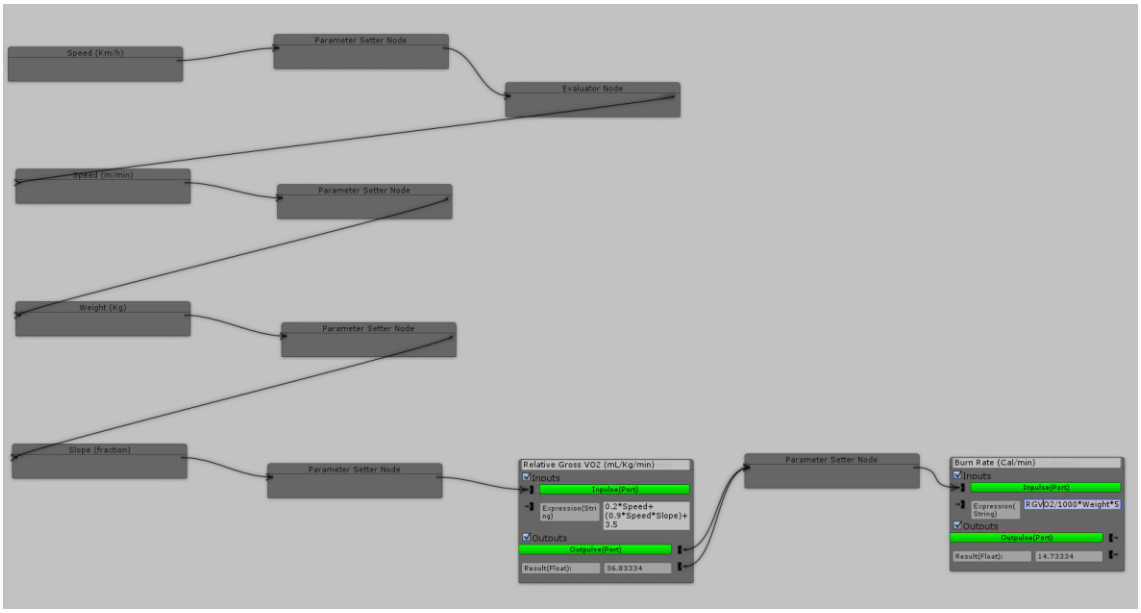


FIGURE 50- ENERGY CONSUMPTION GRAPH CREATED IN GRAPPHER

The Energy Consumption Graph makes use of the formulas presented in Chapter 3.5.

Other formulas can be used, by simply altering the Energy Consumption Graph, adding new variables or simply rewriting the EvaluatorNodes' expressions. Note that in the GhostChaseB graph, the Energy Consumption Graph is used only to change some values shown in the game's UI, providing visual feedback to the user regarding the immediate intensity of the exercise. The Burn Rate could be used to calculate the total number of burnt calories.

In the GhostChaseB graph, the sub-graph immediately below to the "Energy Calculator" graph (3) dictates when and where should a new ghost spawn. The Time to Next Spawn Node is a TimerNode that activates an output every 30 seconds. The Player Heading Node is a ActionNode that gets the current GPS heading of the player, as a Vector3. The X and Z values of the player's normalized orientation vector are used to determine where the Ghost should spawn, relative to the player's position. The NewX and NewZ are EvaluatorNodes that specify that specify new values to be used as parameters in a function:

$$NewX = -50 * PlayerDirectionX$$

$$NewZ = -50 * PlayerDirectionZ$$

The Action "SpawnGhostWithInverseVector" creates a new Ghost with the following position

$$\text{SpawnGhostWithInverseVector}(NewX, NewZ) = GhostPosition(PlayerPositionX + NewX, PlayerPositionY, PlayerPositionZ + NewZ)$$

As can be seen by the above equations, the new Ghost's position is dependent on the current players' position and orientation. The Ghosts spawn 50 meters away from the player and will appear in the opposite direction the player is running towards. This guarantees that the Ghosts are used simply to motivate the player to run faster, never appearing by surprise in front of him.

Finally, the final sub-graph (4) is used to determine the movement speed of the spawned ghosts. By using the Player Effort Graph (see Figure 43), this sub-graph uses the effort value to calculate the current speed of new spawns:

$$SpawnSpeed = 12 * PlayerEffort$$

So, as the player's heartrate increases, so does the speed of the ghosts (up to 12 Km/h). This linear, yet adaptive measure was not initially intended to be included in the GhostChaseGraph's Non-Adaptive profile.

Finally, another profile, GhostChaseA (Adaptive) was created, serving as a basis for the GhostChase version of the game, used in tests.

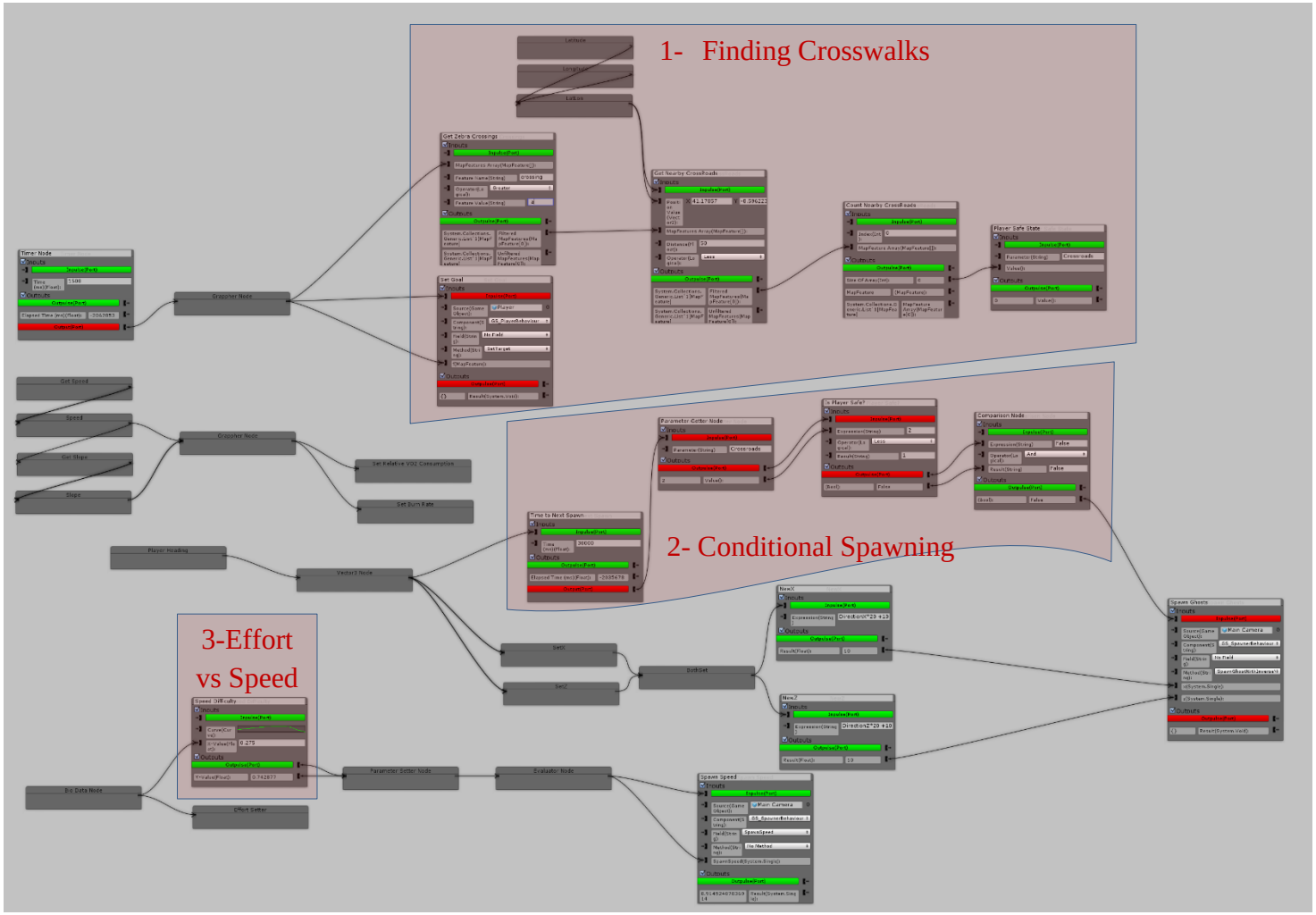


FIGURE 51- GHOSTCHASEA GRAPH CREATED IN GRAPPHER

This graph is considerably more complex than that of the GhostChaseB one. Contrary to what was done with GhostStandA and GhostStandB, GhostChaseA did not make use of an encapsulated GhostChaseB graph to change some of its parameters, as many changes to the graph were needed. The 3 red transparent squares mark the changes that GhostChaseA has when compared to the GhostChaseB

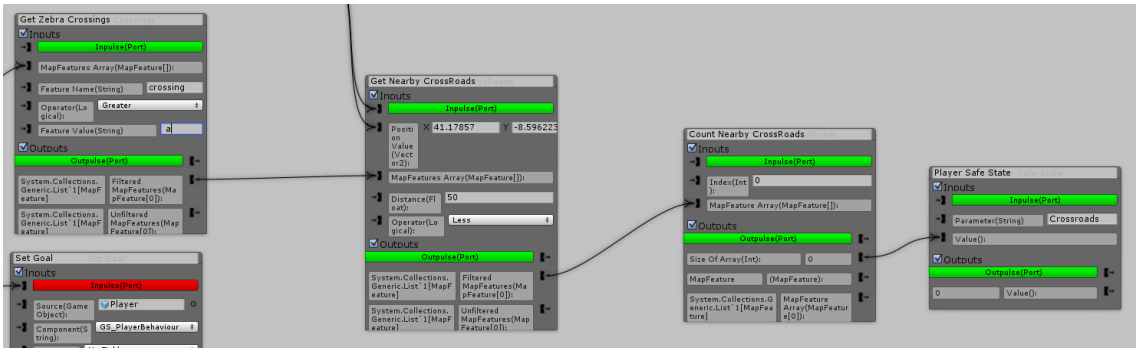


FIGURE 52- FINDING CROSSWALKS (CHANGES IN GHOSTCHASEA VS GHOSTCHASEB)

In GhostStandA, it was decided that ghosts should not spawn when the player is close to a crosswalk. This is to avoid possible health hazards that some of these location games often pose. If no ghosts spawn when the player is near a crosswalk, the player is not instigated into crossing the road as fast as possible, possibly disregarding the incoming traffic.

Figure 52 highlights the differences present in the first red rectangle. This section is responsible for searching for nearby MapFeatures that have a feature of "crossing". If a nearby crossing exists (within 50 meters of the player's position) then a variable "CrossRoads" is set to the number of nearby crossings. This variable is then later used in another section of the graph and shows how geo-information can be freely used for changing the behaviour of a location-based game (see **RQ4**).

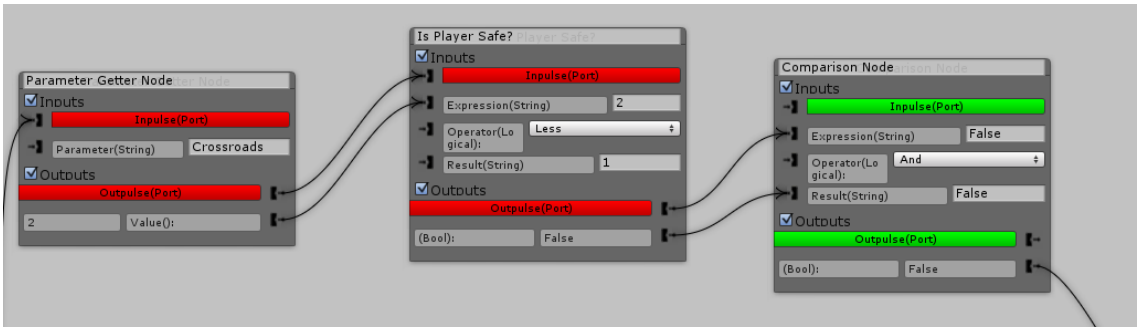


FIGURE 53- CONDITIONAL SPAWNING (CHANGES IN GHOSTCHASEA VS GHOSTCHASEB)

The second change in the graph (highlighted in the second red rectangle) is responsible for deciding if a spawn is possible or not. Every 30 seconds the GhostChaseA graph will decide if it is safe to spawn a new ghost. Figure 53 shows that the previously defined parameter "Crossroads" is decisive in determining if the ghost can be spawned or not. If the number of "Crossroads" is less than 1, then it is safe to spawn the new ghost. Otherwise, this spawn will not be made, and only after 30 seconds will the graph test if it is safe to spawn a new one. Additionally, the ghost's relative spawning position was also made closer to the player, instead of the previous 50 meter fixed distance. Ghosts now spawn 20 meters away from the player.

The final changes made to the GhostChaseB to make it into the GhostChaseA were straightforward changes to the speed of the ghosts, based on the player's current effort.

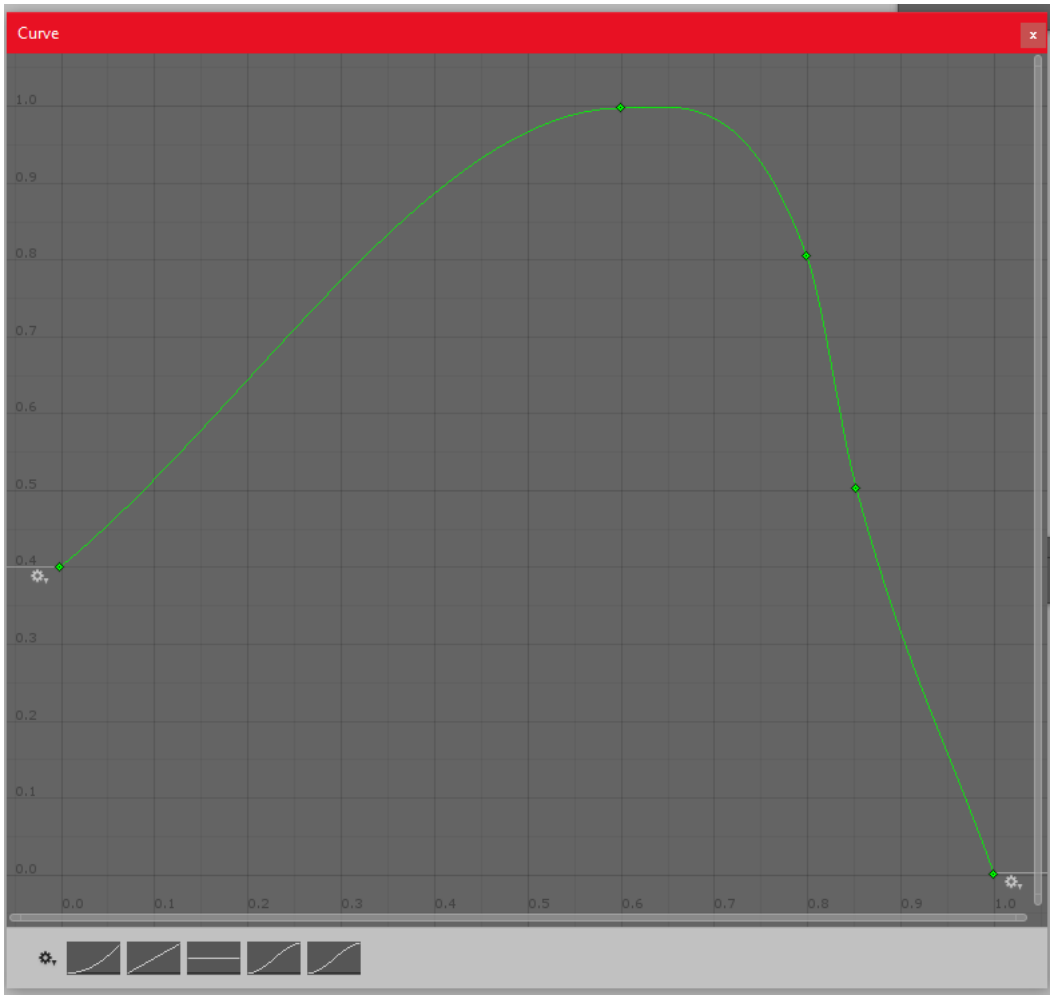


FIGURE 54- EFFORT VS SPEED (CHANGES IN GHOSTCHASEA VS GHOSTCHASEB)

This was made by simply changing the function of effort to speed modifier. As the above image suggests (Figure 54), a non-linear curve was drawn, opposed to the linear GhostChaseB one. This curve ensures that if the player is relaxed, the ghosts will still move towards him. Additionally, as the player's effort increases, so does the ghosts' speed. However, once the player's effort reaches a value of 50%, the ghosts' speed decreases, to a minimum of zero.

With all these changes, the GhostChaseA graph provides a greater health-aware adaptivity than that of the GhostChaseB one. The GhostChaseA also shows how there are some semantics that can be used transversally between exergames and location-based games regarding the player's physical context (answering positively **RQ6**). Also, it takes into account the player's Location, Physical prowess to regulate the Challenge, as per the SPaCiaL framework. The concern for the player's current physical effort coupled with the awareness of the player's real world dangerous surroundings is used to show that, using Grappher, GeoStream, GeoSensors and the Game Data Tool, it is possible to design an almost completely different game with no programming necessary.

(Page intentionally blank)

8 EXPERIMENTAL DESIGNS AND RESULTS

In order to ascertain if the implemented SPaCiaL framework was working as expected, two distinct experiences were designed. Additionally, both games were condensed into a single Unity Android application (called "GhostDilemma: Fight or Flight"), with several scenes added to it. These scenes aimed to make registering the players, and collecting their information possible, just before playing the actual games. The following figures further illustrate the chosen approach.

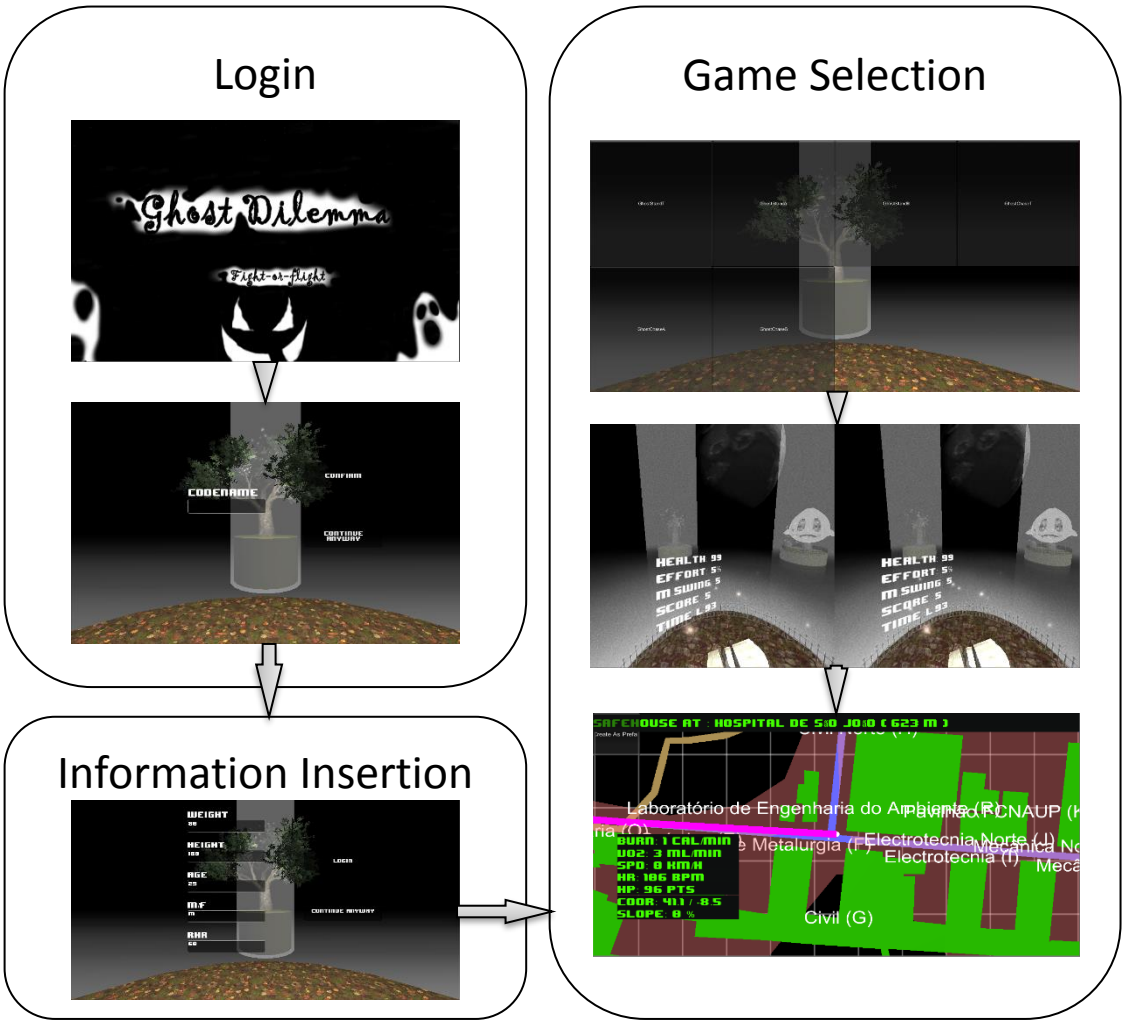


FIGURE 55- GAME PHASES DURING EXPERIENCE

As seen in the above flowchart (Figure 55), both the experiences required the tester or the researcher to insert the tester's codename, some information about him (only needed

on the first login), regarding weight, height, gender, resting heart rate and age, information needed to feed the game adaptation profiles. After selecting the experience, and version of it, to partake in, the respective game would start.

The experiments were divided into two main experiments: The GhostStand game experiment and the GhostChase experiment. Both of these experiments would present the player with three possible variations: T (tutorial), A (adaptive) and B (non-adaptive). Each of these is described in more detail next:

- **Version T (GhostStandT or GhostChaseT):** A version of the game meant for serving as a tutorial and for debugging purposes only. It uses the exact same adaptivity configuration as that of the next version (B). Even though all data is logged, since it is logged in the game data service as corresponding to a T variation of the game, it is ignored in the subsequent statistical analysis.
- **Version A (GhostStandA or GhostChaseA):** The A version of both games relies on the implementation of the version B. The difference is that game relevant variables, present in the B version, are now manipulated with different sources of information, such as real time and offline player data, his/her performance and location-sensitive information. This version relies on the previous versions of the games (version B) so as to better showcase both the possibility of encapsulating adaptivity configurations into each other and guarantee that the only differences between versions is the sources of information and how it is manipulated to affect game mechanics.
- **Version B (GhostStandB or GhostChaseB):** This version of the games provides no adaptivity. This means that the game will be played in a predictable and unvarying pace, as the game will not take into account any context-relevant variables whatsoever. Even so, this profile was designed via the Grappher tool, so that it would serve as the basis for the Version A of the game.

The first experience was designed so as to minimize the number of variables present in the game and the adaptivity configurations used. As such, the GhostStand game was used, since it does not bear the location-based component that its counterpart (GhostChase) does.

8.1 GHOSTSTAND EXPERIMENTAL DESIGN

The GhostStand experience was designed for indoors, limiting the amount of possible external factors. An indoor location with controlled lighting and sound is desirable. As such, the chosen location was that of the Sound Computing Laboratory (room I -104) at the Faculty of Engineering of the University of Porto.

8.1.1 MATERIAL

The following material was needed for the experiment to take place:

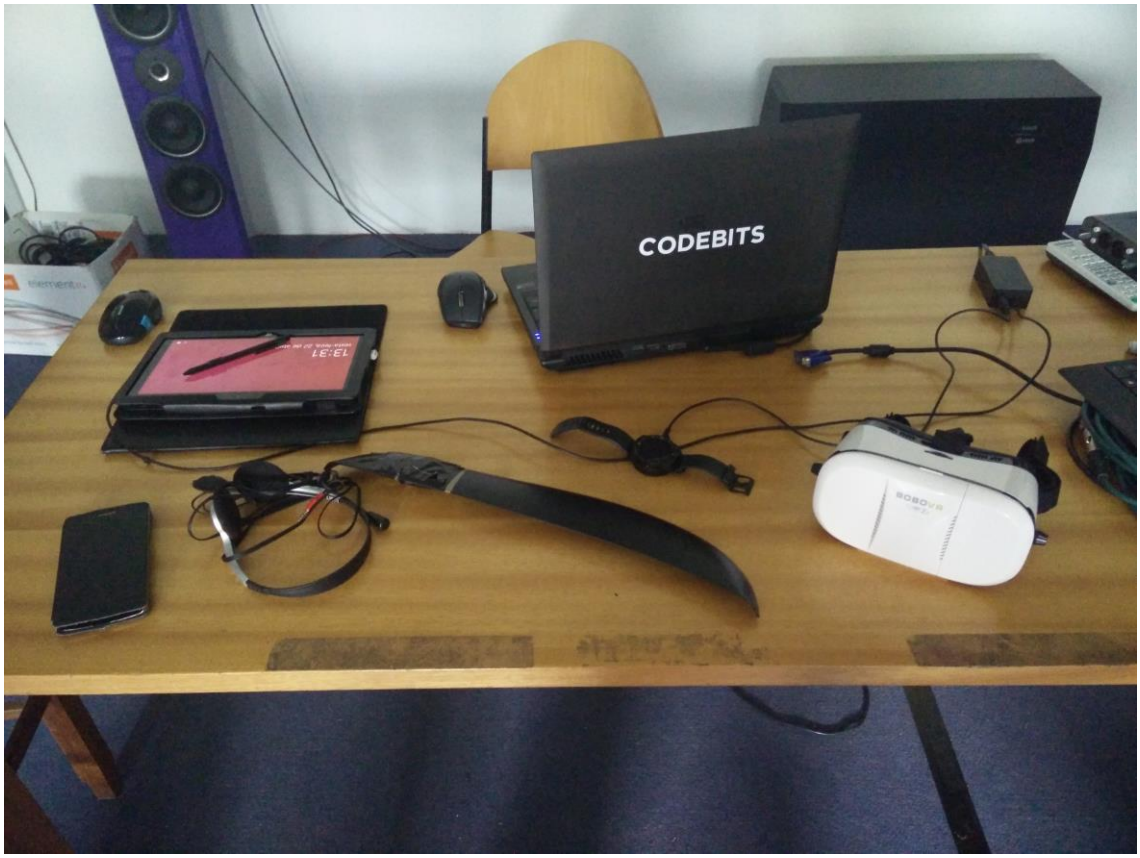


FIGURE 56- GHOSTSTAND EXPERIMENT EQUIPMENT

The equipment required for the user to play the game, visible in Figure 56, is listed below

- **Equipment**
  - A BoboVR Z3 mobile HMD case for the mobile phone to be placed in it
  - A OnePlus One smartphone, running Android 6.0.1
  - A LG Watch R smartwatch, running Android 6.0.1
  - A wired headset
  - A lightweight plastic "sword", designed to add heft to the players' arm movement

Both the questionnaire and informed consent form template are available at the Annex sections F and H, respectively, of this thesis document. The questionnaire is meant to provide insight about player perception of the game and how it may correlate with data from the player's gameplay session. The Informed Consent Form was needed as a means to inform the players of potential health hazards (motion sickness, malaise, fatigue or injuries).

8.1.2 EXPERIMENTAL PROTOCOL

The experimental protocol was designed as follows:

1. The experience is presented to the test subject as an Exergaming in Virtual Reality experience, with no mention of adaptivity in order to avoid bias. Sickness and the risk of possible malaise of minor injuries is explained.
2. Subject signs the Informed Consent Form if willing to participate in the experiment.
3. An internal ID is assigned to the subject, used to link the given questionnaire answers to the game data taken live from their gameplay session.
4. The user fills the first part of the questionnaire: User Profile and Usual Gaming Experience.
5. The researcher creates an experience profile for the player, by registering in the platform the player's internal ID, height, weight, age, sex and measured resting heart rate (as measured at the time by the smartwatch).
6. Subject first tries a tutorial version of the game - "GhostStandT" - in which all data gathered from it is discarded. The user can play the game to its full duration (10 minutes) or quit the game whenever he/she understands the game mechanisms and has gathered enough experience in playing it. "GhostStandT" makes use of the "GhostStandB" adaptivity profile created in Grappher.
7. After being exposed to the "GhostStandT" version of the game, the user is invited to rest for a brief period of time, at least one minute.
8. The user then plays the "GhostStandB" version of the game. This too also makes use of the "GhostStandB" adaptivity profile. The user is expected to play this game for 10 minutes.
9. The user should now fill the "Game 1" part of the questionnaire. A period of time to rest is also provided.
10. The subject now plays the "GhostStandA" version of the game, using the "GhostStandA" adaptivity profile. He/She is expected to play for 10 minutes.
11. The user fills the final part of the questionnaire, "Game 2".
12. Informal and unstructured feedback is collected, if provided by the user.

8.1.3 RESULTS

Twenty-four subjects aged between 18 and 40 years (mean age = 25.3 years, SD=6.05) participated in this test by playing the two versions of the game (non-adaptive and adaptive) and completing the survey. Three of the subjects were females and 21 were males. Of the 24 elements, four people played the non-adaptive game twice. All subjects signed the provided informed consent form.

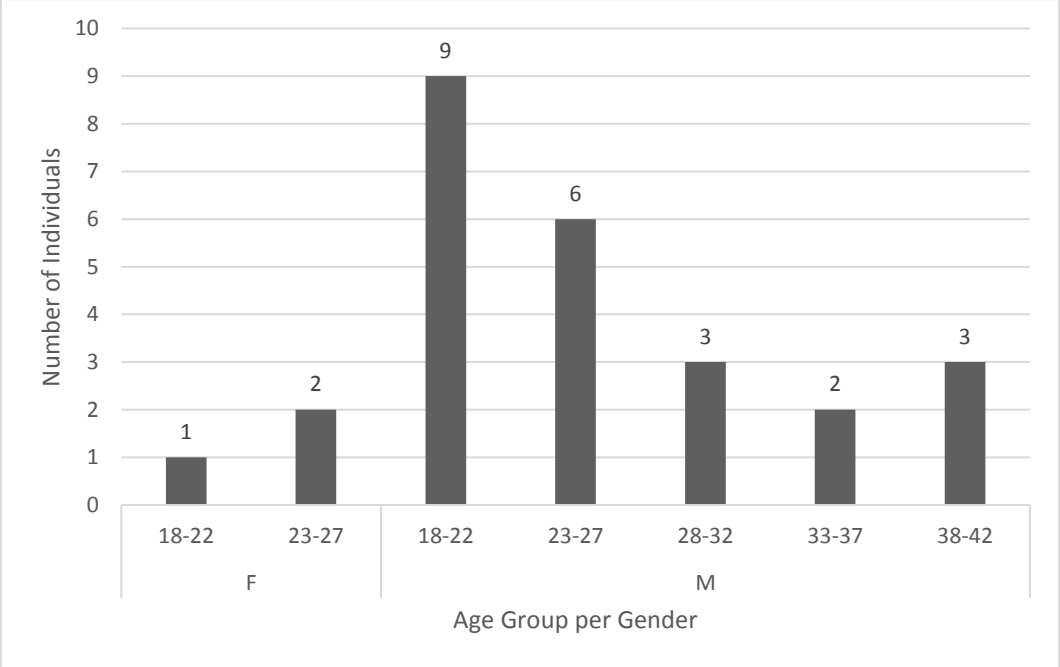


FIGURE 57- NUMBER OF TEST SUBJECTS BY AGE AND GENDER

Subjects were mostly males in their early 20's (Figure 57). Participants were invited to the experience via a broadcasted email to the faculty's students and personnel.

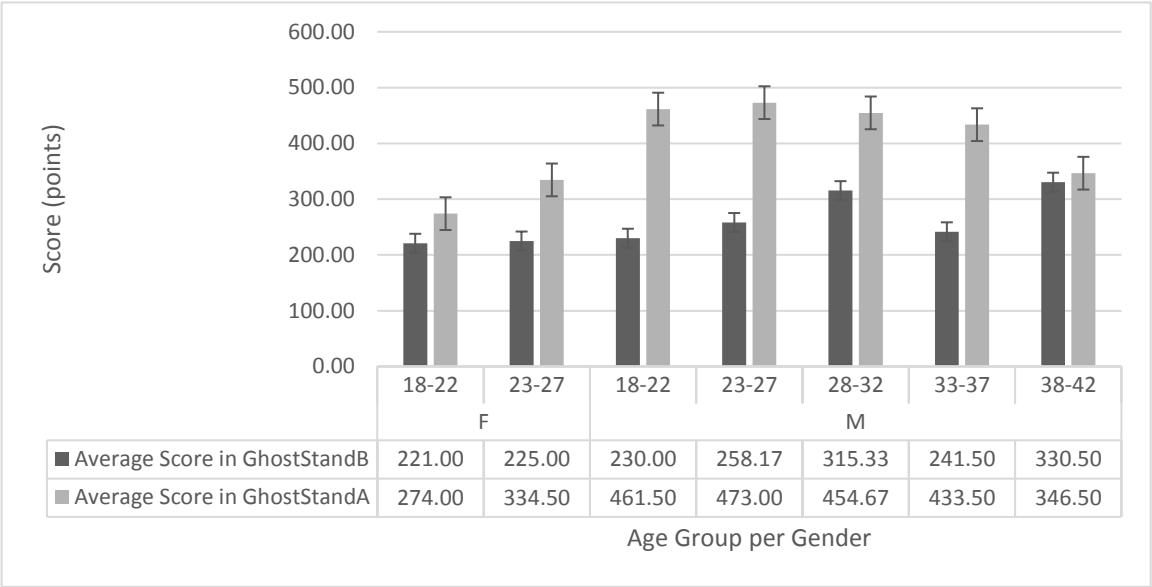


FIGURE 58- AVERAGE SCORES IN GHOSTSTAND SESSIONS PER AGE GROUP AND GENDER

Players attained higher scores when playing the adaptive version of GhostStand (GhostStandA), as it can be seen in the above Figure 58. The difference in the games' average scores is almost negligible for players in the 38-42 years-old range. The higher scores in GhostStandA, when compared to those attained in GhostStandB were expected, since the GhostStandA was designed to have higher ghost spawn rates than that of the GhostStandB game.

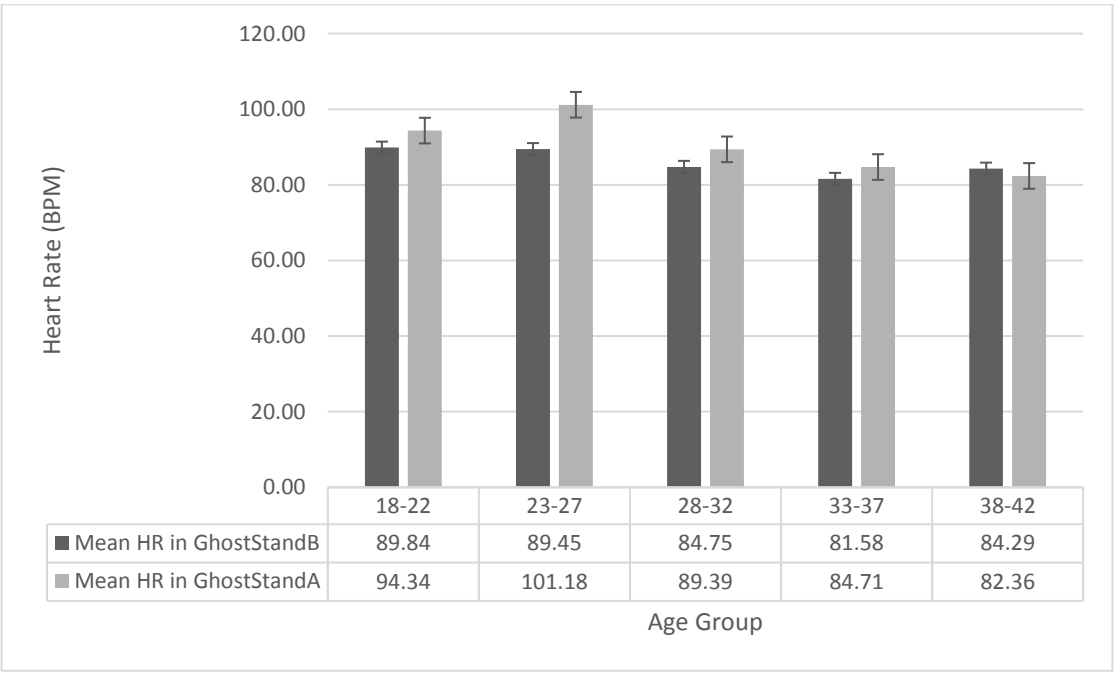


FIGURE 59- AVERAGE HEART RATE IN GHOSTSTANDA AND GHOSTSTANDB PER AGE GROUP

As the GhostStandA version of the game featured a higher ghost spawning rate, it is expected that engaged players will have a higher average heart rate than what they had when playing the GhostStandB game. The above chart, in Figure 59, confirms these expectations. Heart rates were generally higher in the GhostStandA when compared to those of the GhostStandB version of the game. It is significantly more visible in the most expressive age group in the experiment: the 23-27 years-old individuals. In the age group of testers with an age between 38 to 42 years, the expected result is not met. Some possible explanations exist, such as the players being tired, bored or unengaged when playing the second version of the game. However, the difference is negligible, and the group consists only of 3 individuals. As such, no conclusions can easily be drawn for that age group. Similar results were expected when analysing the effort rates, as it is highly dependent on the heart rate.

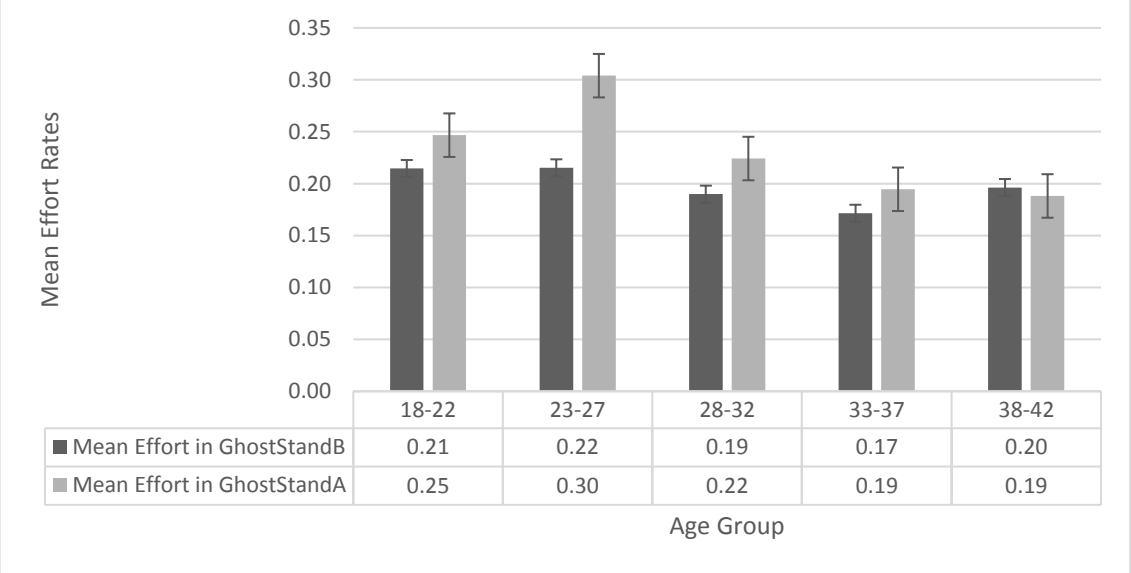


FIGURE 60- MEAN EFFORTS IN GHOSTSTANDA AND GHOSTSTANDB PER AGE GROUP

Effort rates in both games were expected to yield similar results as that of the heart rates, since it is calculated with those values. As Figure 60 presents, GhostStandA generally triggered greater effort rates in the players. The exception were the older players in the 38 to 40 years range. However, the difference is little enough for it to be inconclusive. Another feature of interest is the mean effort rate of players in the 23 to 27 age group attained. This value, roughly 30%, is very close to the value that maximizes the spawn rate in GhostStandA's profile. In fact, it is visible in Figure 45 that the game was designed with the intent of keeping the players in a physical state where the calculated effort is

30%. It is possible that other groups, although reporting higher relative effort rates in GhostStandA, were not stimulated enough to increase their effort rates up to 30%.

TABLE 4- PAIRED SAMPLES STATISTICS

		AVERAGE	N	STANDARD DEVIATION	AVERAGE'S STANDARD ERROR
PAIR 1	EffortMeanB	0.202	20	0.0472	0.0105
	EffortMeanA	0.253	20	0.0745	0.0167
PAIR 2	HRMeanB	87.340	20	6.848	1.531
	HRMeanA	94.089	20	10.336	2.311

5

TABLE 5- PAIRED SAMPLES CORRELATIONS

		N	CORRELATION	SIG.
PAIR 1	EffortMeanB & EffortMeanA	20	0.504	0.023
PAR 2	HRMeanB & HRMeanA	20	0.554	0.011

TABLE 6- PAIRED SAMPLES T-TEST

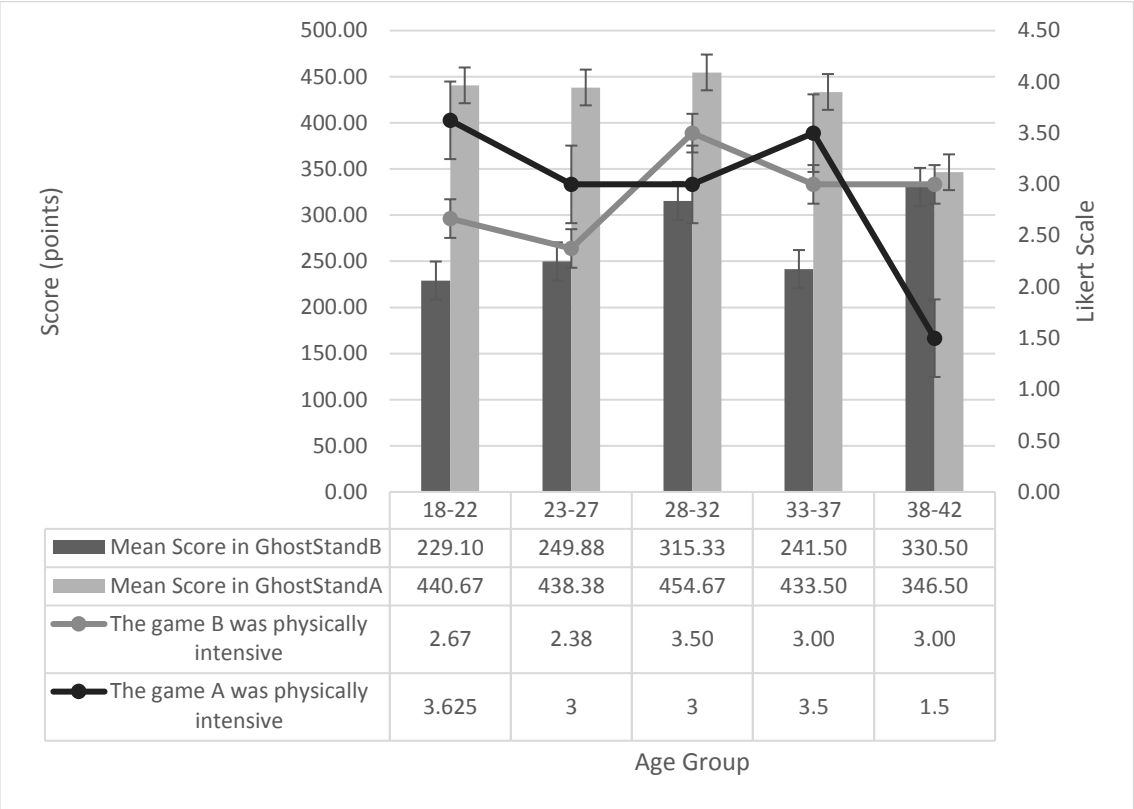
		Paired Differences					t	df	Sig. (2 exts.)
		Average	Standard Dev.	Average's Standard Err.	95% Difference Confidence Interval				
					Lower	Upper			
Pair 1	EffortMeanB - EffortMeanA	-0.051	0.065	0.015	-0.081	-0.020	-3.444	19	0.003
Pair 2	HRMeanB - HRMeanA	-6.749	8.682	1.941	-10.812	-2.685	-3.476	19	0.003

For the heart and effort rates' analysis a t-test for correlated samples was performed (Table 6- Paired Samples T-Test). The t-test for correlated samples is especially useful in research involving human subjects because it can mitigate the differences between individuals.

These results suggest that there is a significant difference in the players' physical effort between the non-adaptive game and the adaptive one ($t(19)=3.44$), $p<0.003$). The confidence interval indicates the effort difference is somewhere between 2 and 8%. As expected, the effort made in the adaptive version of the game was greater than that in the other, non-adaptive version.

The Pearson correlation coefficient (Table 5- Paired Samples Correlations) showed a significant (sig. < 0.05) correlation between both the efforts and heart rates of both versions of the game, albeit with a low "r" value (approximately 0.5).

The questionnaire provided (available in Annex F and its results in Annex G) was based on the Game Engagement Questionnaire [12] and on an adaptive exergame master thesis questionnaire [13]. The GEQ questions provide insight into the psychological state and general gaming experience of the players, useful for determining if the players did enter a state of flow, boredom or stress, while the other questions are specifically tailored for exergames, useful in evaluating the physical state the player was in during gameplay.



	18-22	23-27	28-32	33-37	38-42
Mean Score in GhostStandB	229.10	249.88	315.33	241.50	330.50
Mean Score in GhostStandA	440.67	438.38	454.67	433.50	346.50
The game B was physically intensive	2.67	2.38	3.50	3.00	3.00
The game A was physically intensive	3.625	3	3	3.5	1.5

FIGURE 61- SCORES AND PERCEIVED EXERTION PER GAME AND AGE GROUP

One of the most relevant statements in the questionnaire was “The game was physically intensive” in the section “Exertion” of each game. Younger age groups perceived the game GhostStandA to be physically more intense than the GhostStandB version (Figure 61). The older group, 38 to 42 years old, even though they had similar scores and average heartrates among the games, felt that the GhostStandB version was significantly more physically intensive than the adaptive one. The age group of 28 to 32 year-olds also perceived the GhostStandB version to be slightly more physically demanding. Since this group has one of the smallest effort differences between the game’s versions, it could explain their answer.

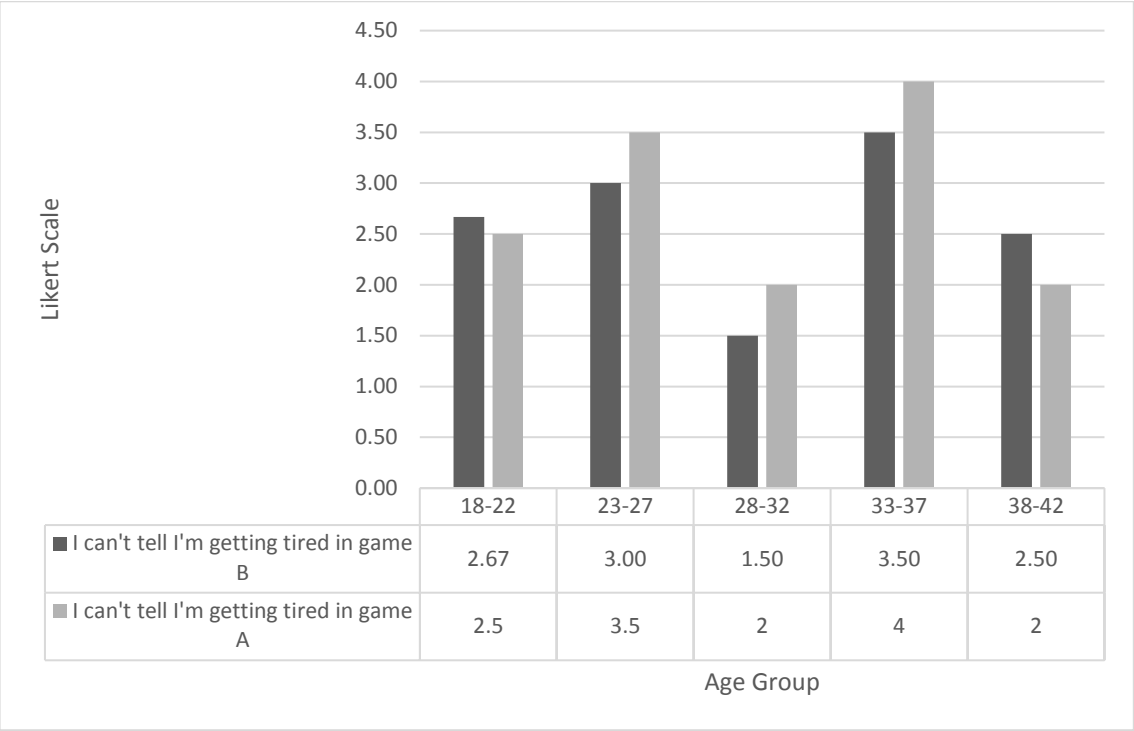


FIGURE 62- PERCEIVED GAME FATIGUE PER AGE GROUP

Concerning the flow state, the statement "I cannot tell I'm getting tired" also provided interesting findings. As it can be seen in Figure 62, most players did not feel as tired in GhostStandA as in GhostStandB, despite having higher effort rates in GhostStandA. This is considered to be a good indicator of being in flow state, as per the Game Experience Questionnaire. These results show that players do have different perceptions of difficulty and physical exertion when playing an adaptive or non-adaptive version of a game, answering **RQ7** positively.

8.2 GHOSTCHASE EXPERIMENTAL RESULTS

Since GhostChase is a location-based exergame, and considering that the GhostStand game was designed with no location-based features in mind, it was decided that the GhostChase experiment would focus in the game adaptivity based on the location and not on the player's current physical state. As such, three tests were devised to ascertain if the SPaCiaL framework, in its current form, is capable of both determining the player's current location context and change a game's behavior depending on it, as designed. Since these tests' results are binary (either the framework is able, or it is not) and no user-specific data other than the location is needed, the tests were simulated by simulating the player's physical movement, by controlling the player's movement via keyboard, and the player's heart rate, using GeoSensors to randomly generate valid values. Controlling the player's movement means manually changing latitude/longitude values by a predefined amount over time. Player speed movement would not exceed 3 meters per second. This was chosen so as to provide a controlled experimental environment that would not risk equipment or a player's health, while being able to test if the designed adaptive mechanisms were functioning as designed.

The game, via the Game Data platform, had its events recorded. The position of the player was saved, every 2 seconds, and the position of where the ghosts spawn was kept as well.

All the recorded events were processed by a JavaScript script that would generate the georeferenced heat maps that are the basis of most images in this chapter's result section.

8.2.1 EXPERIMENTAL PROTOCOLS

Three experiments where devised. They would differ only in location. Each experiment requires the game to be played twice, as one of the versions of the game is adaptive while the other is not.

These versions are:

- **GhostChase #1 (GS1)**
  - In this version, the player starts at the *Faculdade de Engenharia da Universidade do Porto*, and must run to the nearest hospital (highlighted as the goal in the game), going first to a crosswalk in order to assess if the game is context aware.
- **GhostChase #2 (GS2)**
  - In this version the player starts near the *Casa da Música* in Porto and has to go to the nearest hospital. The path contains several crosswalks that should make clear the context awareness capabilities of the adaptive version of the game.
- **GhostChase #3 (GS3)**
  - In the third experiment, the player starts in the west side of the *Île-de-France* in Paris and must go to the hospital near the *Nôtre Dame*. The path contains some crosswalks, close to the initial position of the player. This experiment is also meant to highlight how the proposed context awareness would function in different locations as long as context-relevant information is available.

GHOSTCHASE EXPERIMENT #1 (GC1)

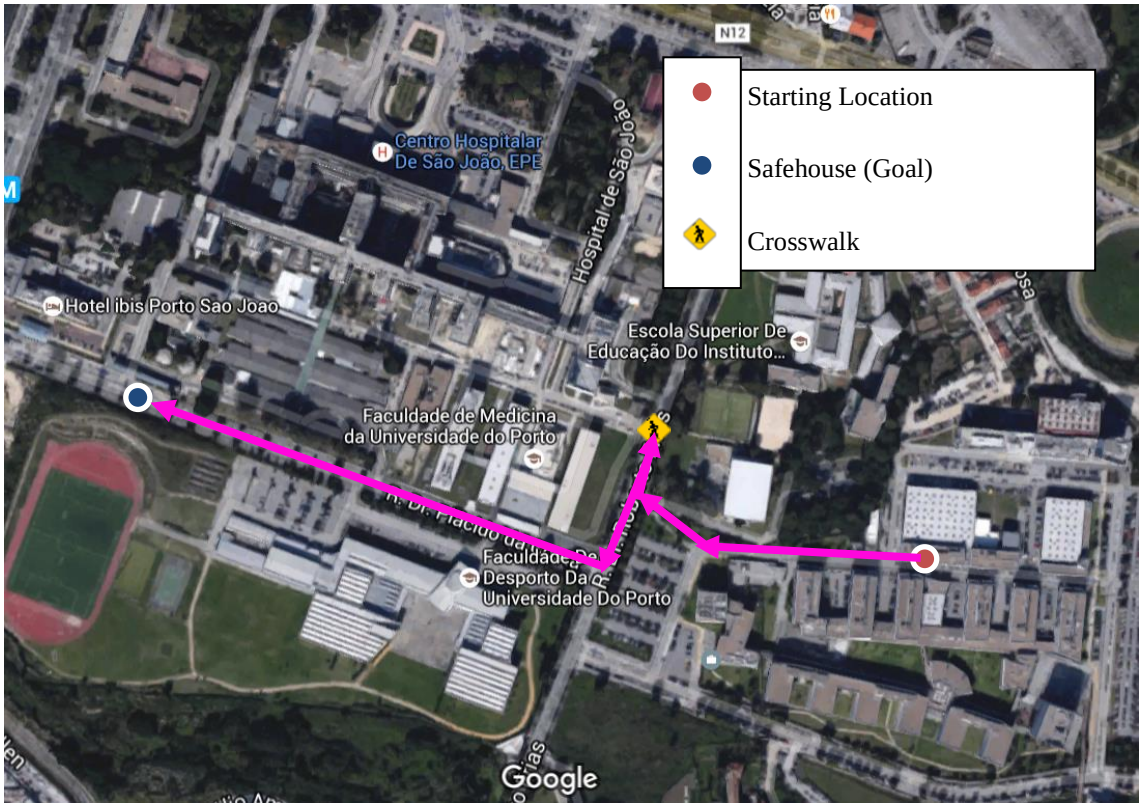


FIGURE 63- GC1 EXPERIMENTAL DESIGN MAP OVERVIEW (PATH IN PINK)

The first GhostChase experiment (GC1) where one player would run to the safehouse (the nearest hospital, *Hospital S. João*) in both the adaptive and non-adaptive game was designed (as per Figure 63). It has:

- The player starts in the college's campus, preferably near the GIG laboratory at the *Faculdade de Engenharia da Universidade do Porto*.
- Once the game starts, the player moves (running or walking) to a specific pedestrian crossing.
- As the player arrives at that crosswalk, he/she spends at least one minute in that location.
- After said minute has passed, the player resumes the game, going to the safehouse.
- The game ends when the player arrives at the safehouse.
- Repeat the protocol for the second version of the game.

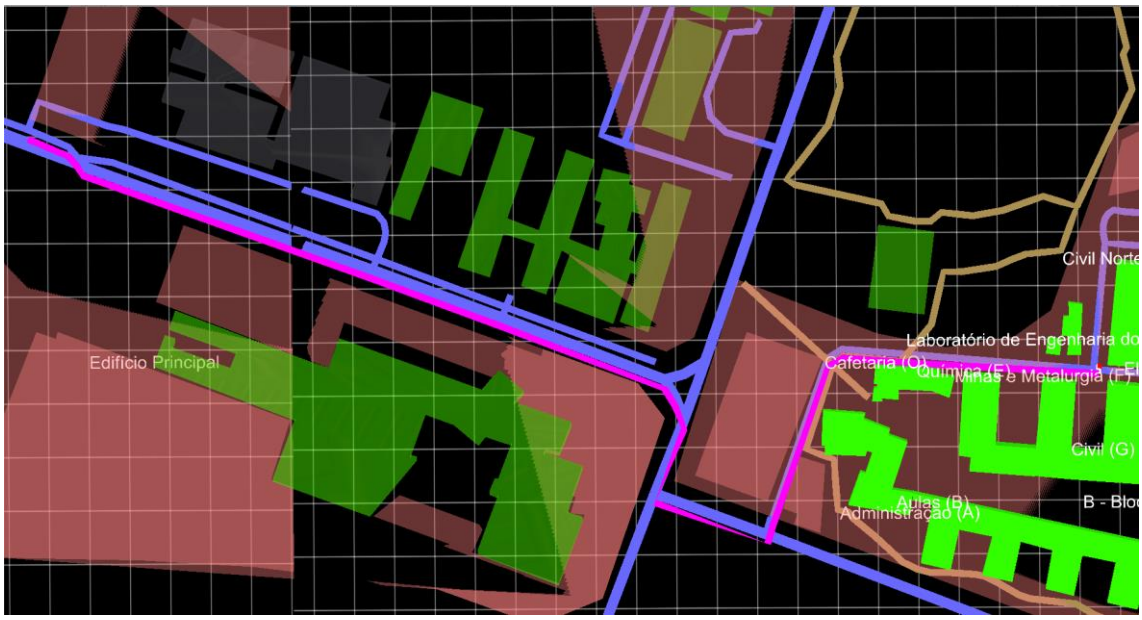


FIGURE 64- SUGGESTED PATH FOR THE GC1 EXPERIMENT (IN PINK)

Even though the application suggested a clear path (from the player's starting position to the safehouse) as per Figure 64- Suggested Path for the GC1 experiment, a different path was chosen. The reason behind this is that the path the application suggested had no crossing information available in OpenStreetMaps, and thus, would lead to little to no differences between the adaptive and non-adaptive versions of the game.

GHOSTCHASE EXPERIMENT #2 (GC2)

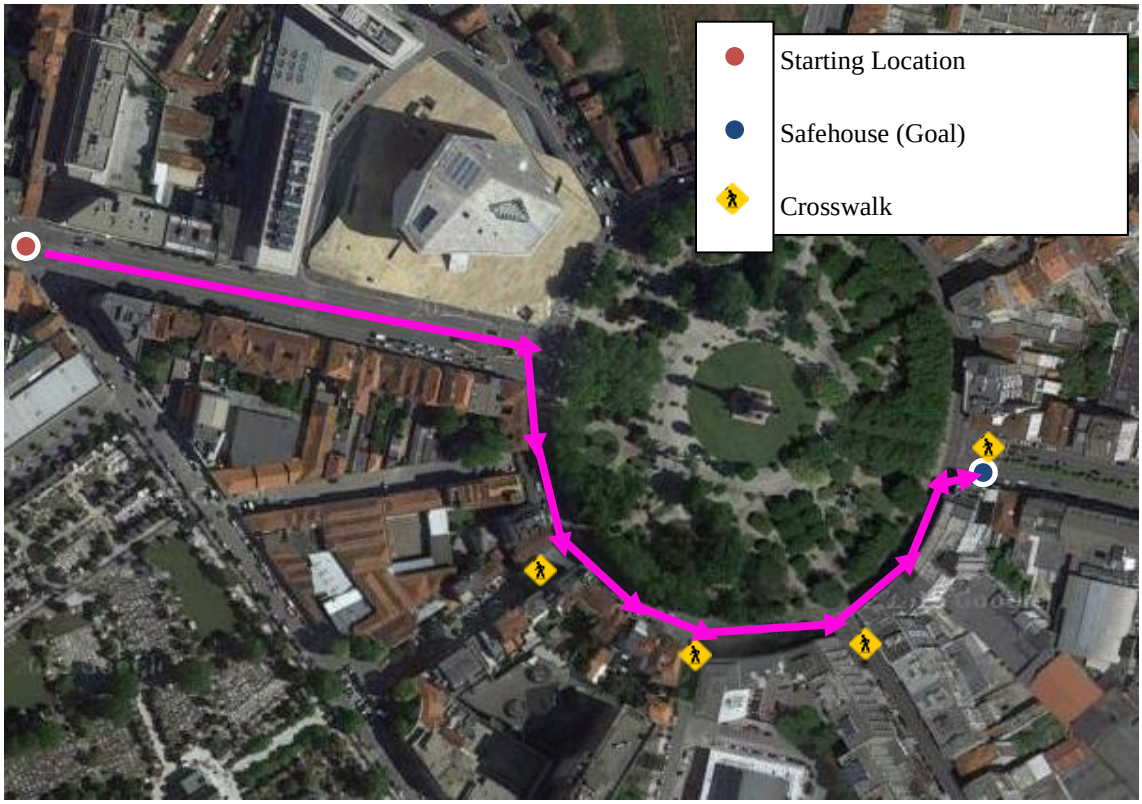


FIGURE 65- GC2 EXPERIMENTAL DESIGN MAP OVERVIEW (PATH IN PINK)

The second GhostChase experiment (GC2) had the player run to the safehouse (the nearest hospital, *Hospital Privado do Porto*) in both the adaptive and non-adaptive versions of the game (as per Figure 65- GC2 Experimental Design Map), starting from a position with no nearby crosswalks and through a path with multiple crosswalks. It was designed as follows:

- The player starts in area with no crosswalks nearby.
- Once the game starts, the player moves (running or walking) according to the game's suggested path.
- As the player arrives at that crosswalk, he/she spends at least one minute in that location.
- After said minute has passed, the player resumes the game, going to the safehouse, following the suggested path.
- The game ends when the player arrives at the safehouse.
- Repeat the protocol for the second version of the game.

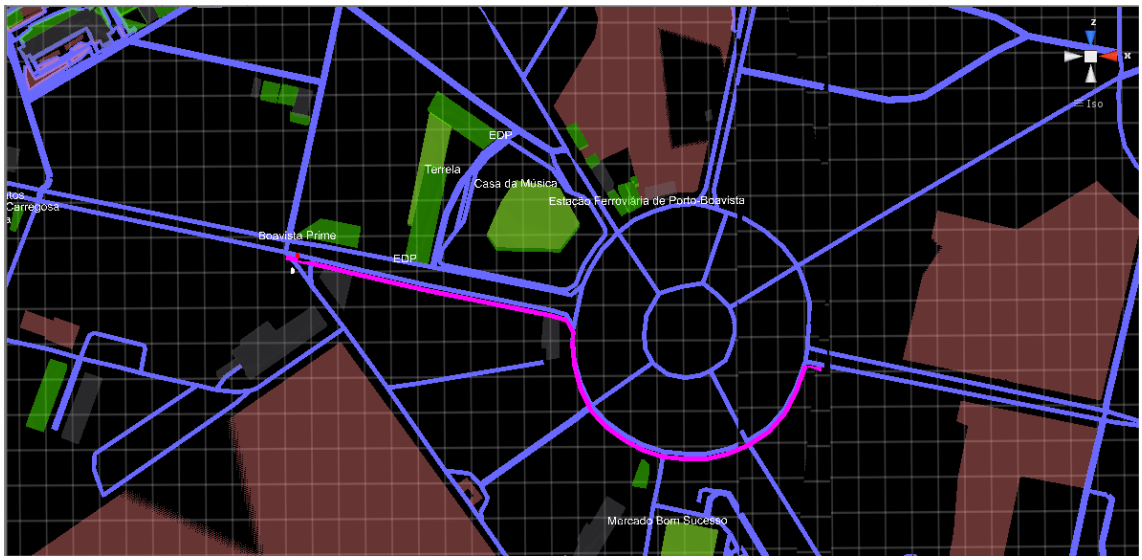


FIGURE 66- SUGGESTED PATH FOR THE GS2 EXPERIMENT (IN PINK)

Contrary to what occurred in the GC1 experiment, in the GC2 experiment the player is supposed to follow the path the application delineated. As the player will have to be near crossroads with crosswalks in this path, there was no need for the player to deviate from it. The suggested path will have the player cross at or near multiple crosswalks that are recognized by the GeoStream API as such.

GHOSTCHASE EXPERIMENT #3 (GC3)

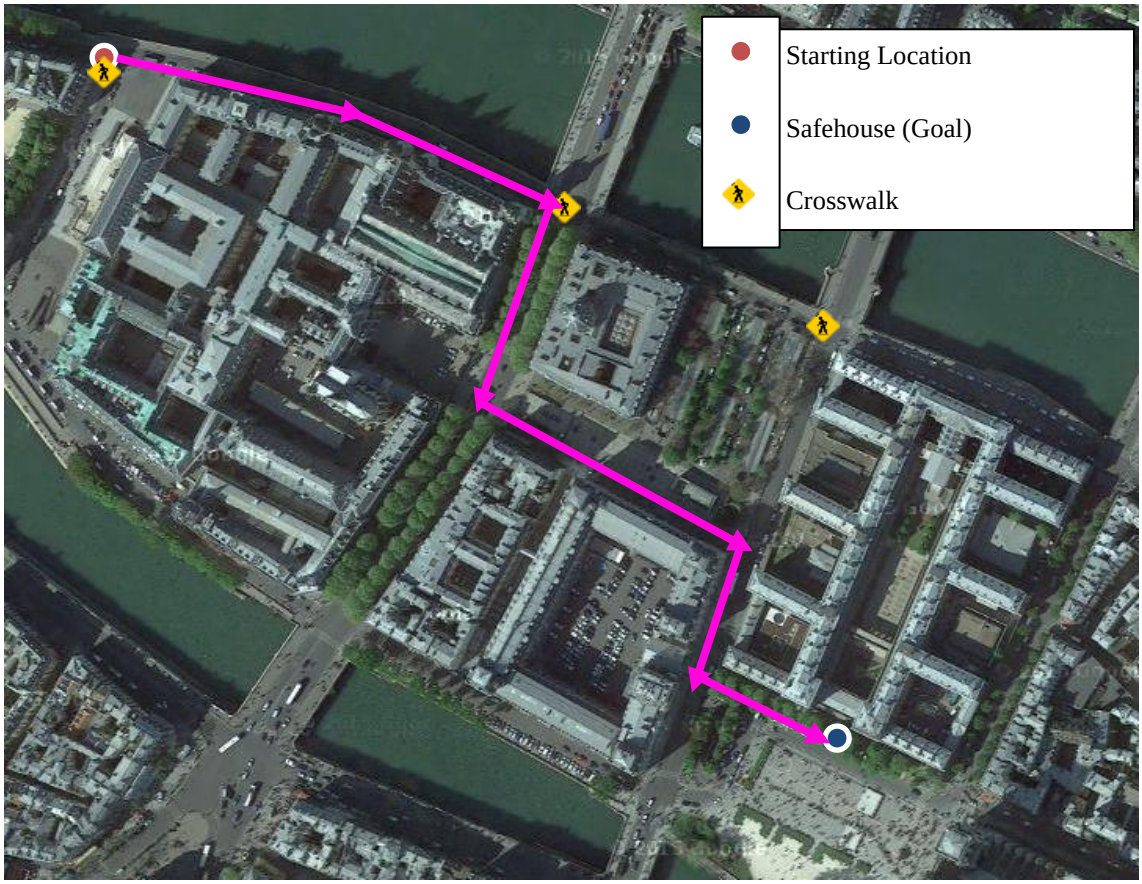


FIGURE 67-GC3 EXPERIMENTAL DESIGN MAP OVERVIEW (PATH IN PINK)

The third GhostChase experiment (GC3) had the player run to the safehouse (the nearest hospital, *Hôtel-Dieu* in Paris, France) in both the adaptive and non-adaptive versions of the game (as per Figure 65- GC2 Experimental Design Map), starting at a crosswalk and passing a few crosswalks . It was designed as follows:

- The player starts at a crosswalk.
- Once the game starts, the player moves (running or walking) according to the game's suggested path.
- As the player arrives at that crosswalk, he/she is not meant to spend any time at the crosswalk.
- The game ends when the player arrives at the safehouse.
- Repeat the protocol for the second version of the game.

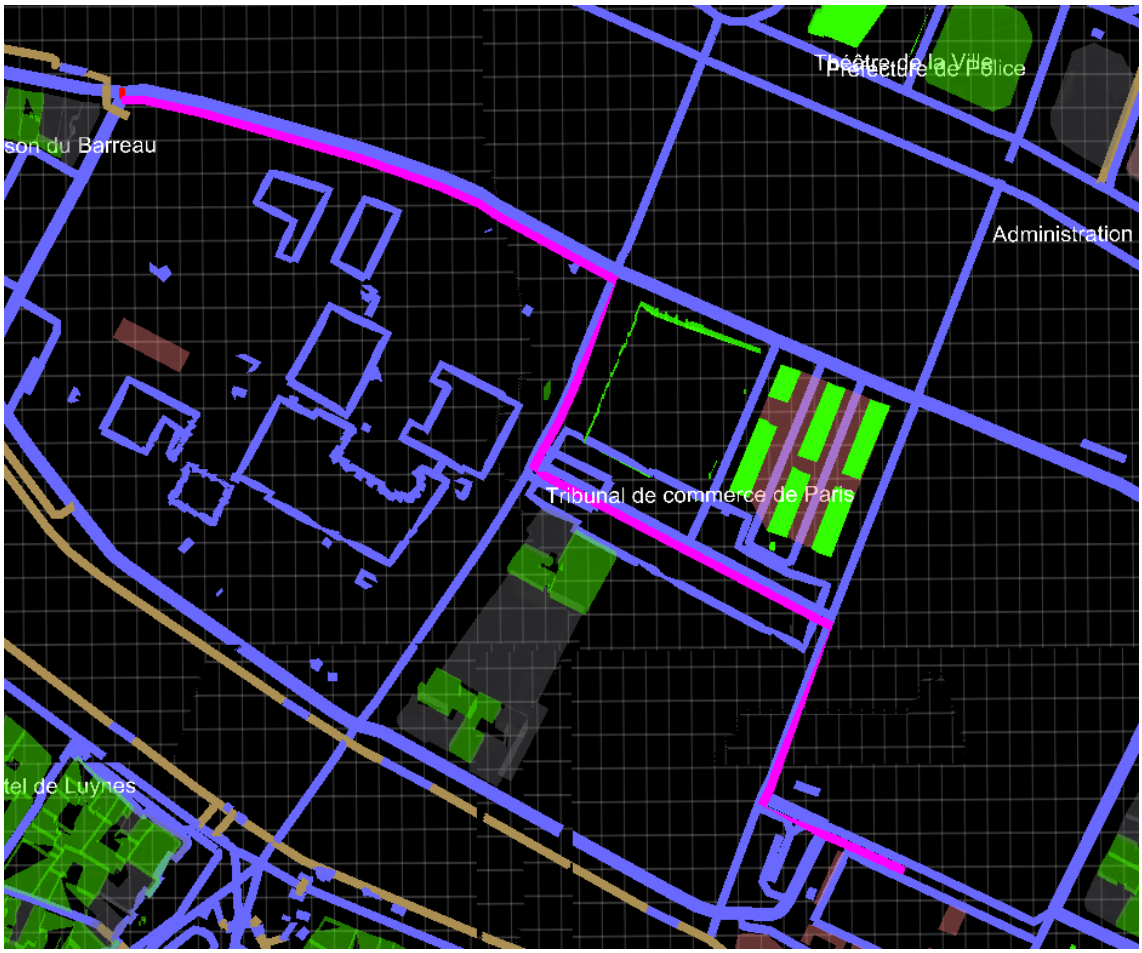


FIGURE 68-SUGGESTED PATH FOR THE GC3 EXPERIMENT (IN PINK)

Similar to the GC2 experiment the player is supposed to follow the path the application delineated, meeting multiple crosswalks (Figure 68-Suggested Path for the GC3 Experiment (in pink)). However, in this experiment (GC3), the player starts at a crosswalk and is not expected to spend any time near the crosswalks that are along the way. The idea is to test different starting conditions (near a crosswalk) and if a fast paced player will notice differences in the game (spending no time waiting to cross the road).

8.2.2 RESULTS

GHOSTCHASE EXPERIMENT #1 (GC1)

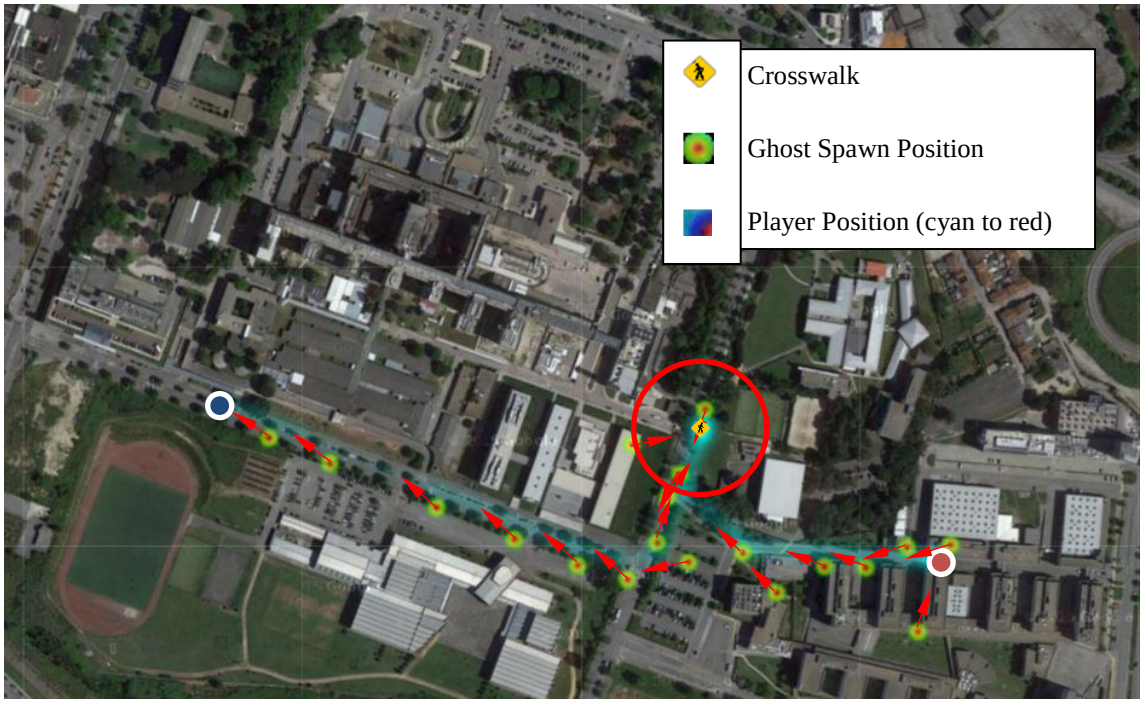


FIGURE 69- GC1 GHOSTCHASEB GAMEPLAY SESSION AND EVENTS

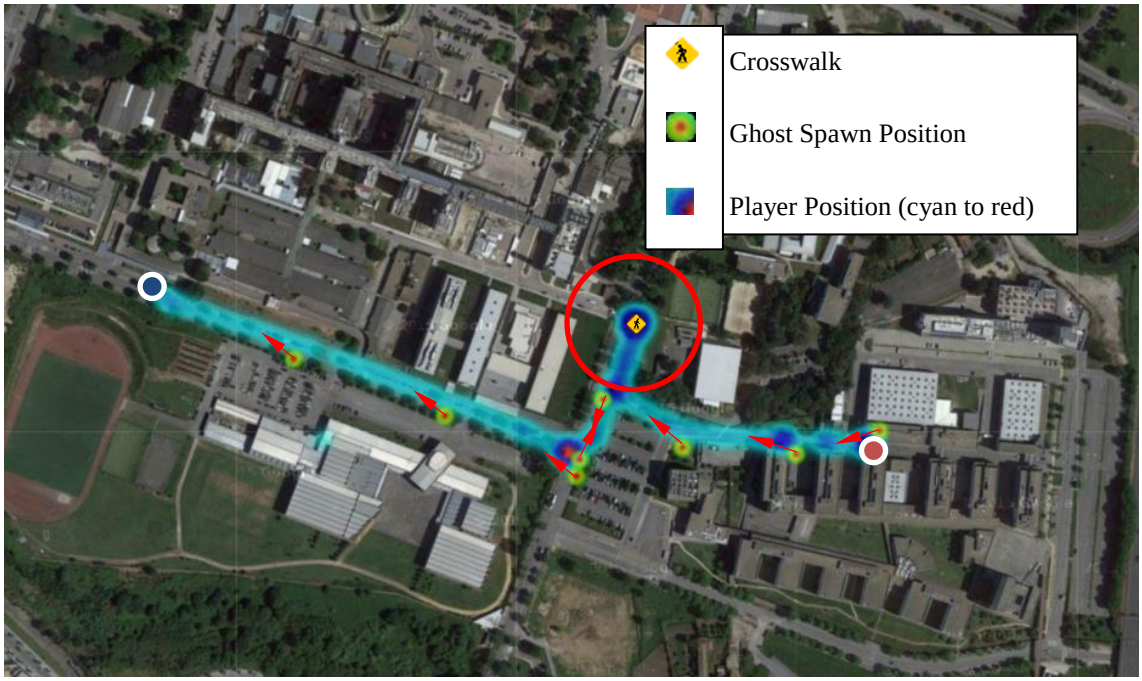


FIGURE 70- GC1 GHOSTCHASEA GAMEPLAY SESSION AND EVENTS

The experiment was done in a simulated environment. Both versions of the game were played as per the experimental protocol.

The stored data was then used to generate the above geo-referenced heat maps (Figure 69 and Figure 70) . The player initially started in the right most position seen in the heatmap, inside the college campus. Then proceeding to go to the crosswalk, signaled in the map. After waiting for about a minute, we walked towards the goal, an entrance of the closest hospital (leftmost player positions in the map). This process was repeated for both versions of the game.

As previously seen, the adaptive version of the GhostChase game, GhostChaseA searches for MapFeatures with the feature "crossing". If it contains said value and if that MapFeature is less than 50 meters from the player's current position, no new Ghosts will spawn. For them to spawn, the player will have to go to a new position where no "crossings" are within 50 meters.

The red arrows represent the approximate direction the ghost was moving towards to when it spawned.

The red circle in both those images has a radius of approximately 50 meters. In the GhostChaseB version of the game, it is possible to see that ghosts did spawn when the player was within 50 meters of the pedestrian crossing. In the other image, of the GhostChaseA version, no ghosts spawned within 50 meters of the crosswalk.

GHOSTCHASE EXPERIMENT #2 (GC2)

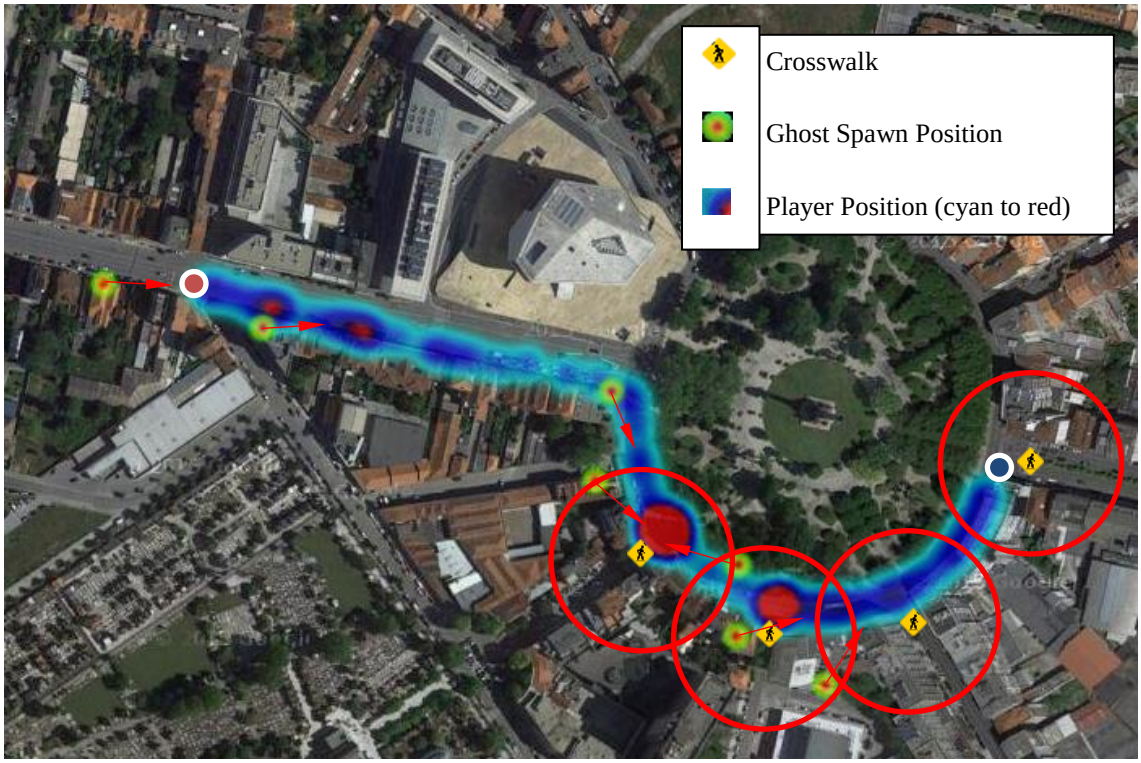
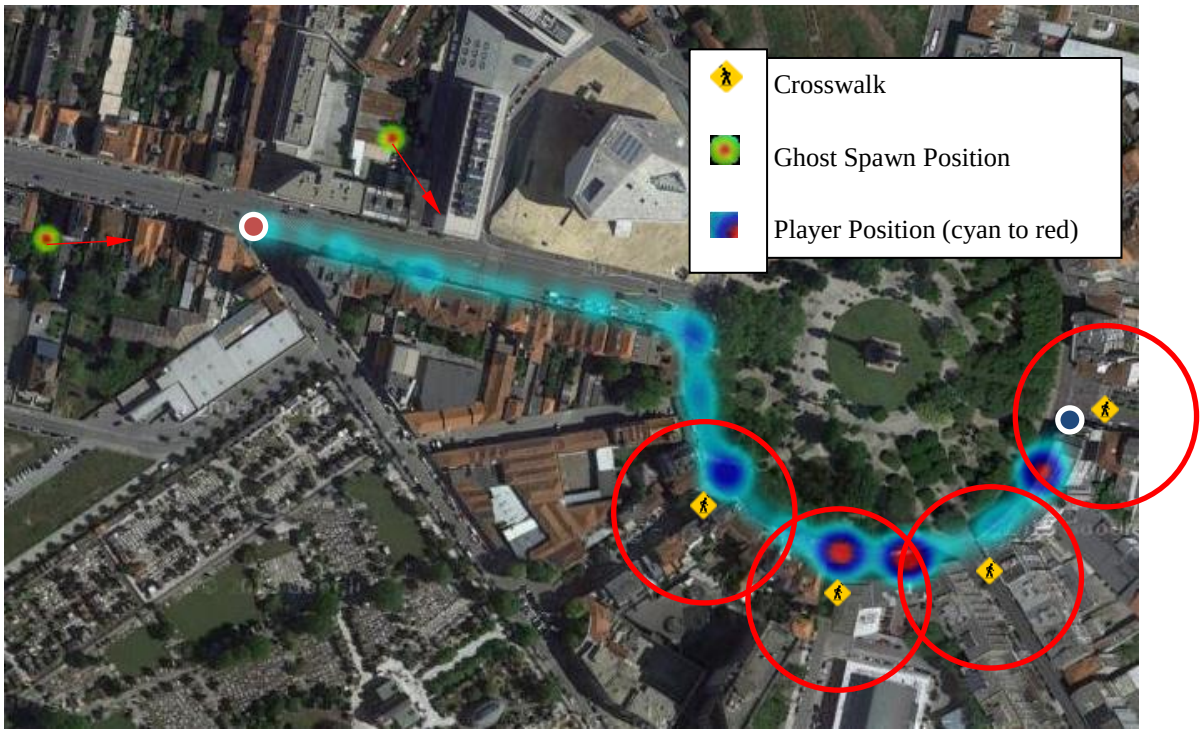


FIGURE 71- GC2 GHOSTCHASEB GAMEPLAY SESSION AND EVENTS



5 *FIGURE 72- GC2 GHOSTCHASEA GAMEPLAY SESSION AND EVENTS*

The experiment was done in a simulated environment. Both versions of the game were played as per the experimental protocol.

The stored data was then used to generate the above geo-referenced heat maps (Figure 71 and Figure 72) . The player initially started in the left most position seen in both heatmaps and proceeded to go to the goal following the path given by the application. The player stopped for approximately one minute in each of the first three crosswalks encountered, in both variations of the GC2 experiment.

The differences between the adaptive and non-adaptive versions of GC2 are even more evident than in the GC1 experiment.

Similar to what occurred with the GC1 experiment, it is visible how the adaptive version of the game differs from the non-adaptive version of the game. In the GC2 non-adaptive version, ghosts would continue to spawn well within 50 meters of crosswalks when the player was standing near them (as it is visible from their movement vectors, in red). However, in the adaptive one, ghost only spawned in the street the player started at, as the application found no crosswalk nearby. When the player entered the roundabout, as it contains several crosswalks nearby, and since their safety areas (circles in red in both images) overlap each other, the player saw no ghosts spawning until the end of the game session. This further proves that the game is well aware of the surroundings of the player and can be used by a game to provide challenges that are both adequate to the player's fitness level and surroundings.

GHOSTCHASE EXPERIMENT #3 (GC3)

5 ***FIGURE 73- GC3 GHOSTCHASEB GAMEPLAY SESSION AND EVENTS***

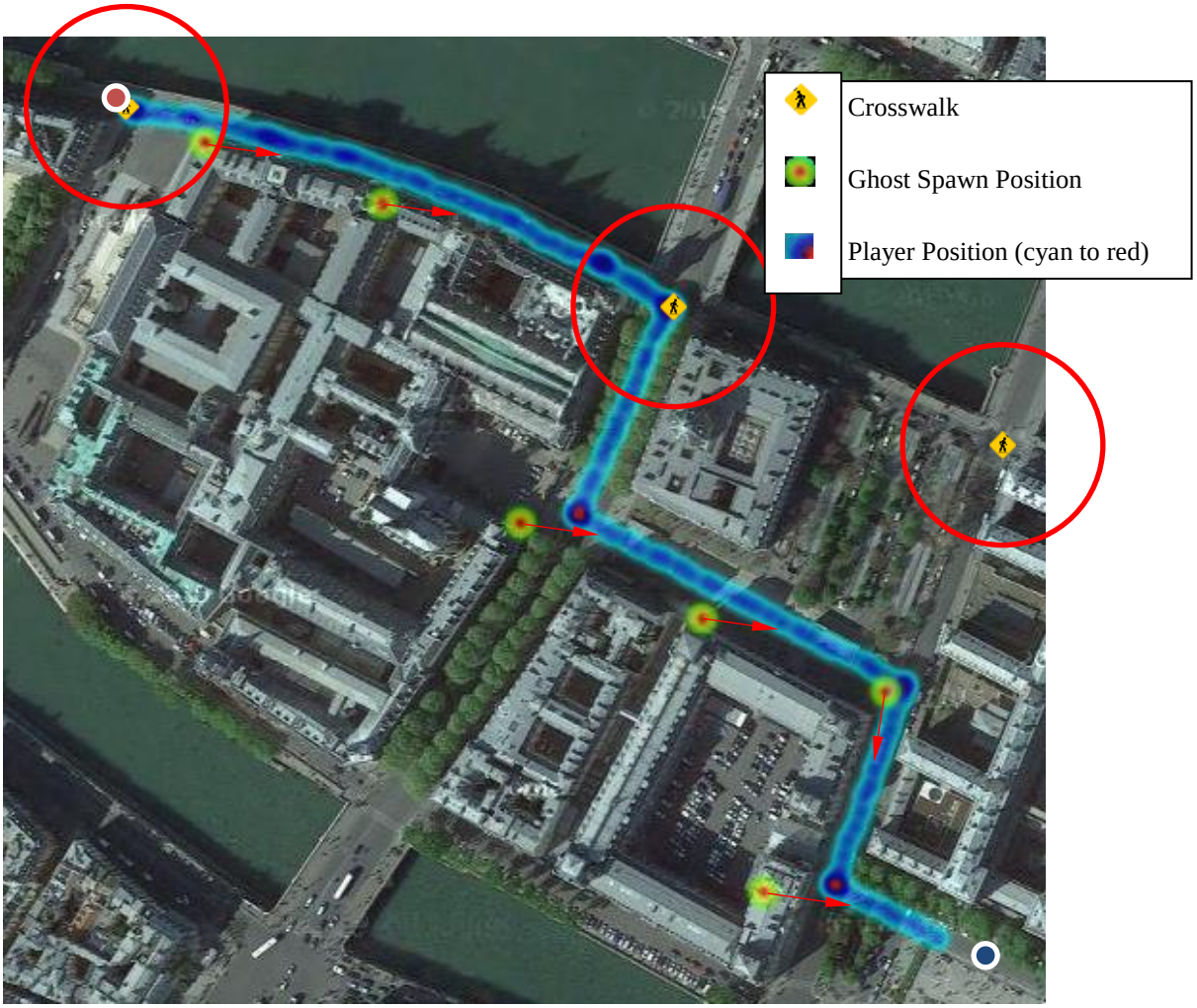


FIGURE 74- GC3 GHOSTCHASEA GAMEPLAY SESSION AND EVENTS

The experiment was done in a simulated environment. Both versions of the game were played as per the experimental protocol.

The stored data was then used to generate the above geo-referenced heat maps (Figure 73 and Figure 74) . The player initially started in the left most position seen in both heatmaps and proceeded to go to the goal following the path given by the application. In this case, the player did not stop when near any crosswalk and continued following the given path as if no time was spent waiting to cross a road.

As expected, in this scenario the differences between the adaptive and non-adaptive versions are less evident. As the player spends less time inside the safety area of a crosswalk, the amount of ghosts that spawn in the non-adaptive version of the game are not many more than those that spawn in the adaptive counterpart. However, it is still visible, particularly at the starting point, that ghosts do spawn well within the vicinity of the cross-

walks, in the case of the non-adaptive version of the game, while none appear in the adaptive counterpart. Even in cases where the player may have no problems with crossing quickly, the system will prevent ghost spawning to ensure the player is not too focused in the game while crossing the road.

So, considering all three experiments, in different locations and so, with different surroundings, one can determine that the adaptive version of the GhostChase game is more aware of the player's real life surroundings and is capable of changing the game's behavior whenever the player is close to some locations (in the experimental cases, crosswalks). As such, it is definitely possible to reinforce a player's safety regarding real world surroundings through location-based game adaptivity (see **RQ8**).

9 CONCLUSIONS

The literature review presented showed that there is both the need and the possibility for the creation of an unobtrusive solution that is capable of capturing a representation of the player's current physical state and the current real-world context with the required accurate for an adaptive game. The game can then progressively adapt to that new context or attempt to induce a new state in the player (changing mechanics or challenges to drive the player experience). The SPaCiaL framework, proposes the integration of the Flow model and adaptive gameplay for location-based games and exergames, allowing for semantics to drive a game to adapt to the player's context and to induce new states. All four major components of the SPaCiaL Framework (GeoStream, GeoSensors, Grappher and Game Data Tool) have been tested as a pipeline to allow for the creation of adaptive profiles for location-based games or exergames. This shows that it is possible to create an adaptivity framework generic enough to create such profiles for different games using different sources of information. Furthermore, this contribution gives answers to the research questions defined in section 1.3:

- **RQ1**: **Is it possible to standardize a source of information to be used in a game?**
  Multiple sources of information can be used to provide a more reliable and precise context definition for a developer to create an adaptive game, as was shown in Chapter 3.3 and exemplified in Chapter 4, where GeoStream makes use of OpenStreetMaps, Google Altitude API, Google Maps Static API and OpenWeather API, centralizing the information obtained from each.
- **RQ2**: **How can information relevant to a POI (point of interest) be inferred from different sources of data?**
  Merging information from different sources relative to the same POI can be done using a generic schema for representing the data, although the issue of merging conflicting information from different sources is still an open issue (Chapter 4).
- **RQ3**: **How can a game flow theory model be applied to location-based exergames?**
  The game flow theory model, applied to location-based games required several

adjustments so as to take into account the context awareness of these games. The proposed SPaCiaL model, detailed in Chapter 3, provides a possible solution as to how to make the flow state dependant on the surroundings of the player, which was tested in Chapter 8.

- **RQ4**: **How can geo-information be used as an input for a location-based game?**

  The concrete usage of location-specific information in a game was shown in section 7.2.1. In it, it is shown how Grappher can be used as a means to bridge the contextual information gathered by GeoStream with the already defined game mechanics. E.g. in GhostChase, nearby hospitals and crossroads are integrated into gameplay elements.

- **RQ5**: **Is it possible to associate information gathered from different sources to game mechanics?**

  The possibility of taking into account any type of contextual data (be it intrinsic or extrinsic to the player and its location) is a more generic problem than that found in RQ4. Grappher, by allowing any type of information to be processed and to be used in parametrizing game mechanics (as seen in Chapters 6.1 and 7.1.1) showed how this can be done. Although this question is particularly relevant for context-aware games, it can be extended to adaptive games in general.

- **RQ6**: **Is it possible to define a common game adaptivity semantics for both location-based games and exergames?**

  Even though location-based games and exergames may share different types of contextual information, it is possible to provide a solution broad enough for location-based games with no exergame components and, conversely, for exergames with no location awareness. The proposed SPaCiaL model highlighted the relevance of this issue, and section 7.2.1 showed how information relevant to both exergames and location-based games can be used to create adaptive profiles in a similar mode.

- **RQ7**: **Do players perceive the differences in adaptive exergames when compared to non-adaptive ones?**

  Players have shown to have different perceptions of their gameplay session, when comparing the results between the non-adaptive version of a game and its adaptive version. Section 8.1.3 presented how players showed different biological responses when playing the game, as well as different opinions regarding the

perceived difficulty altogether. Interestingly enough, players reported the adaptive game as being easier, despite showing greater effort values in it when compared to the non-adaptive version of the game.

- **RQ8**: **Can player safety in exergames be reinforced through adaptivity?** The issue of safety when playing location-based games is relevant, as players may be distracted by the game and not pay attention to their surroundings. This problem can be mitigated to some extent thanks to the contextual awareness. Section 8.2.2 presents a way to adapt a location-based game to avoid challenges to the player when near crosswalks. By not engaging the player in potentially dangerous contexts, the player is not pressed into splitting focus between the game and the real world, potentially averting accidents.

The hypothesis presented in section 1.2 has been supported by the tests conducted on two adaptive exergames based on the proposed SPaCiaL model. Since the results in sections 8.1.3 and 8.2.2 show a clear distinction in context and player state awareness between the adaptive and non-adaptive versions of these games, it is reasonable to conclude that the proposed SPaCiaL model is capable of supporting the creation of adaptive, location-based exergames. The two games - GhostStand and GhostChase – created to test the proposed framework, provided results that showcase how the framework can benefit the development and the user experience of location-based games and exergames specifically, by improving the player's experience of a game while adapting the challenge to the player's abilities, as demonstrated in Chapter 8. The results support the hypothesis that it is indeed possible to adapt a location-based exergame's challenge to a player's skill, physical ability and location context while also influencing the player's and location context, e.g. by forcing the player to rest or to change his/her location.

9.1 LIMITATIONS

Some limitations, both technical and design wise, have already been identified. Those technical in nature (GPS coverage, data connection availability, sensor precision and device's battery life) cannot be easily circumvented. Still, at its worse, the solution should not hamper traditional game adaptivity solutions, being at most simply unavailable or unsuited. Other limitations will most likely be due to the nature of this solution being a generic approach to performance and adaptivity in games, meaning that it may potentially be sub-optimal for particular scenarios or simply inadequate for very specific gaming scenarios and sensors. This can be circumvented by a more thorough review and analysis of what constitutes a good adaptive profile (or profiles) in those gaming scenarios, through interviews with physical performance professional analysts and through the implementation of additional specialist nodes in the framework. Additionally, location-based information may be incorrect, unavailable or incomplete for the location where the game will be player. This, however, goes beyond the scope of this work.

9.2 CONTRIBUTION

The contribution of this work is manifold. The main contribution is the design of the SPaCiaL model, capable of addressing the adaptivity gap currently affecting location-based exergames that other flow models were unable to address. Additionally, this work showed that the aggregation of multiple sources of geographical information is useful in representing real world information in a game, creating game mechanics around it and in providing sources of information for adaptivity, potentially allowing for location-based games to be better adapted to the player and the game's real world surroundings. The bridging of the multiple sources of information (sensors, game data, player data, etc.) and game mechanics through the usage of graphs showed that it is possible to design adaptivity profiles with little to no programming needed, allowing a game designer to create, test and iterate over explicitly-defined adaptivity semantics.

9.3 FUTURE WORK

Ideally the SPaCiaL framework should be tested in other location-based games, exergames and games to test its robustness. The possibility of testing GhostStand and GhostChase with more participants is also being considered, so as to further consolidate the current results. Releasing the framework's components (GeoStream, GeoSensors, Grappher and the Game Data Tool) to other developers and researchers can help test the limits of their implementations. These tools may also be of use in other areas (simulations, rapid application development and visual programming) other than game design. Regarding the research questions identified and studied in this work, many provide possible lines of research:

- **RQ1**: **Is it possible to standardize a source of information to be used in a game?**
  Identifying even more sources of information (both local, via sensors, and remote, via web services) can help further the design of novel and more context-aware games and application. The solution presented was scalable to new sources of information (specifically GeoSensors and GeoStream), but other sources that fall outside the classification of real-time sensors or location-based services may require specific needs not covered in this work.
- **RQ2**: **How can information relevant to a POI (point of interest) be inferred from different sources of data?**
  The issue of merging information from different sources is a difficult problem on its own, and one which, if solved efficiently and optimally, could benefit not only context-aware games, but context-aware applications and services in general.
- **RQ3**: **How can a game flow theory model be applied to location-based exergames?**
  The application of the flow theory to location-based exergames proved to be doable through the proposed SPaCiaL model. However, other types of games may require a different approach from the available flow models (Dual Flow, Flow, SPaCiaL). Research into the limitations of the SPaCiaL model when applied to

games other than exergames and location-based games will help test the limits of this model, and suggest possible revisions.

- **RQ4**: **How can geo-information be used as an input for a location-based game?**

  Other means of making use of gathered contextual information in a game should be explored. The current method was not designed for nor tested with the integration of geo-referenced multimedia formats in mind (such as images, videos, sounds), gathered from remote sources, with a game's mechanics.

- **RQ5**: **Is it possible to associate information gathered from different sources to game mechanics?**

  The proposed approach of processing flows of context-relevant information through Grappher and parametrizing game-specific mechanics is general enough to warrant the design of other scenarios that might benefit from this approach. Informal tests suggest that this approach can also be used in the area of RAD (Rapid Application Development) or remote design/development. As reported, this approach (of creating profiles via Grappher) had the beneficial side effect of not needing the application itself to be recompiled and redeployed, most of the time. Additionally, adapting this methodology for the collaborative design of game profiles, could benefit the game industry.

- **RQ6**: **Is it possible to define a common game adaptivity semantics for both location-based games and exergames?**

  Since the proposed methodology provides a common method of defining both exergame and location-based game adaptivity profiles, this research question has been answered to in full. However, it may be interesting to see if this methodology is applicable in the design of adaptivity profiles for completely different games (or even applications).

- **RQ7**: **Do players perceive the differences in adaptive exergames when compared to non-adaptive ones?**

  Although it was identified that players had both different biological responses and experiences when playing the adaptive and non-adaptive versions of GhostStand, it remains to be determined the full extent of that difference and how heavily dependent on the game's mechanics it is. Other variables, beyond experience, flow and physical state could also be identified and evaluated.

- **RQ8**: **Can player safety in exergames be reinforced through adaptivity?** Context-aware games were shown to be able to adapt their mechanics in order to safeguard the player. However, more sources of information (such as traffic, weather) coupled with a more automated method of adapting these games for safety should be researched.

Almost all lines of research can be further investigated or expanded to other areas beyond location-based games and exergames and, in some cases, beyond games and into the domain of context-aware applications and services.

(Page intentionally blank)

10 REFERENCES

1. Gregory D. Abowd and Elizabeth D. Mynatt. 2000. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction* 7, 1: 29–58. http://doi.org/10.1145/344949.344988

2. Maayan Agmon, Cynthia K Perry, Elizabeth Phelan, George Demiris, and Huong Q Nguyen. 2011. A Pilot Study of Wii Fit Exergames to Improve Balance in Older Adults. *Journal of Geriatric Physical Therapy* 34, 4: 161–167. http://doi.org/10.1519/JPT.0b013e3182191d98

3. Miguel M Almeida, Helena Rodrigues, and Rui José. 2010. Bluetooth Hotspots for Smart Spaces Interaction.

4. Julian Alvarez, Damien Djaouti, Olivier Rampnoux, and Véronique Alvarez. 2011. Serious Games market : Some key figures. *LudoScience & IDATE*.

5. P. Attiel, M. Singh, E. Emerson, Amit Sheth, and Marek Rusinkiewicz. 1996. Scheduling workflows by enforcing intertask dependencies. *Distributed Systems Engineering* 3, 4: 222–38. http://doi.org/10.1088/0967-1846/3/4/003

6. Ronald Azuma. 1997. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments* 6, 4: 355–385. http://doi.org/10.1.1.30.4999

7. Gillian Barry, Brook Galna, and Lynn Rochester. 2014. The role of exergaming in Parkinson's disease rehabilitation: a systematic review of the evidence. *Journal of neuroengineering and rehabilitation* 11, 1: 33. http://doi.org/10.1186/1743-0003-11-33

8. G. Begis. 2000. Adaptive gaming behavior based on player profiling. 1–5. http://doi.org/US6106395 A

9. Steve Benford, Chris Greenhalgh, Tom Rodden, and James Pycock. 2001. Collaborative virtual environments. *Communications of the ACM* 44, 7: 79–85. http://doi.org/10.1145/379300.379322

10. Florian Berger. 2012. Evaluating an Implementation of an Adaptive Game-Based

Learning Architecture. *User Modeling, Adaptation, and Personalization* 7379, January. http://doi.org/10.1007/978-3-642-31454-4

11. Schell J Bishop J, Kalson A, Strickland K and Technology Center et Al. 2003. Biohazard. *Carnegie Mellon Entertainment*. Retrieved from Bishop J, Kalson A, Strickland K, Schell J, Biohazard, Carnegie Mellon Entertainment

12. Jeanne H. Brockmyer, Christine M. Fox, Kathleen A. Curtiss, Evan Mcbroom, Kimberly M. Burkhart, and Jacquelyn N. Pidruzny. 2009. Journal of Experimental Social Psychology The development of the Game Engagement Questionnaire : A measure of engagement in video game-playing. *Journal of Experimental Social Psychology* 45, 4: 624–634. http://doi.org/10.1016/j.jesp.2009.02.016

13. Christopher Burt. 2014. Having Fun , Working Out : Adaptive and Engaging Video Games for Exercise.

14. Abraham G Campbell, Michael J O Grady, and Noel E O Connor. 2010. FreeGaming : Mobile , Collaborative , Adaptive and Augmented ExerGaming Categories and Subject Descriptors. *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia*: 173–179.

15. PH Carstensen and Kjeld Schmidt. 1999. Computer supported cooperative work: New challenges to systems design. *In K. Itoh (Ed.), Handbook of Human Factors*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.142.972

16. Fernando Cassola, Leonel Morgado, Fausto de Carvalho, Hugo Paredes, and Benjamim Fonseca. 2014. Online-Gym: A 3D Virtual Gymnasium Using Kinect Interaction. *Procedia Technology* 13: 130–138. http://doi.org/10.1016/j.protcy.2014.02.017

17. D Charles, a Kerr, and M McNeill. 2005. Player-centred game design: Player modelling and adaptive digital games. *... of the Digital Games ...* 285, 6: 285–298. http://doi.org/10.1.1.97.735

18. Guanling Chen and David Kotz. 2000. A Survey of Context-Aware Mobile Computing Research. *Time* 3755, TR2000-381: 1–16. http://doi.org/10.1.1.140.3131

19. The Aerospace Corporation. How GPS Works. Retrieved July 20, 2011 from http://www.aero.org/education/primers/gps/howgpsworks.html

20. Jolla P Silva Cunha, Bernardo Cunha, William Xavier, Nuno Ferreira, Luis Meireles, and S A Biodevices. 2010. Vital-Jacket : A wearable wireless vital signs monitor for patients â€™ mobility in Cardiology and Sports. *4th International Conference on Pervasive Computing Technologies for Healthcare*: 1–2. http://doi.org/10.4108/ICST.PERVASIVEHEALTH2010.8991

21. Barney Dalgarno. 2002. The Potential of 3D Virtual Learning Environments: A Constructivist Analysis Barney Dalgarno. *Virtual Reality*: 1–19.

22. Farjana Z. Eishita and Kevin G. Stanley. 2010. Theempa. *Proceedings of the International Academic Conference on the Future of Game Design and Technology - Futureplay '10*: 219. http://doi.org/10.1145/1920778.1920811

23. K Erenli. 2012. The impact of gamification: A recommendation of scenarios for education. *Interactive Collaborative Learning (ICL), 2012 15th International Conference on*: 1–8. http://doi.org/10.1109/ICL.2012.6402106

24. Umer Farooq, John M. Carroll, and Craig H. Ganoe. 2007. Supporting Creativity with Awareness in Distributed Collaboration. *Proceedings of the 2007 International ACM Conference on Supporting Group Work (GROUP '07)*: 31–40. http://doi.org/10.1145/1316624.1316630

25. Ronaldo Domingues Filardo, Rosane C. Rosendo da Silva, and Edio Luiz Petroski. 2008. Validação das equações metabólicas para caminhada e corrida propostas pelo American College of Sports Medicine em homens entre 20 e 30 anos de idade. *Revista Brasileira de Medicina do Esporte* 14, 6: 523–527. http://doi.org/10.1590/S1517-86922008000600010

26. SM Fox and WL Haskell. 1970. The exercise stress test: needs for standardization. *Cardiology: Current Topics and Progress*, 149: 54.

27. Levent G??rg??, Abraham G. Campbell, Kealan McCusker, et al. 2012. Freegaming: Mobile, collaborative, adaptive and augmented exergaming. *Mobile Information Systems* 8, 4: 287–301. http://doi.org/10.3233/MIS-2012-00147

28. Stefan Göbel, Sandro Hardy, and Viktor Wendel. 2010. Serious games for health: personalized exergames. *Proceedings of the ...*: 1663–1666. Retrieved December 9, 2013 from http://dl.acm.org/citation.cfm?id=1874316

29. F. J. Gonzalez-Castano and J Garcia-Reinoso. 2003. Survivable Bluetooth location networks. *Communications, 2003. ICC '03. IEEE International Conference on 2*, 1014–1018 vol.2. http://doi.org/10.1109/ICC.2003.1204504

30. Mordechai Haklay and Patrick Weber. 2008. OpenStreet map: User-generated street maps. *IEEE Pervasive Computing* 7, 4: 12–18. http://doi.org/10.1109/MPRV.2008.80

31. Sandro Hardy, Abdulmotaleb El Saddik, Stefan Gobel, and Ralf Steinmetz. 2011. Context aware serious games framework for sport and health. *MeMeA 2011 - 2011 IEEE International Symposium on Medical Measurements and Applications, Proceedings*: 1–5. http://doi.org/10.1109/MeMeA.2011.5966775

32. Hamilton a. Hernandez, Zi Ye, T.C. Nicholas Graham, Darcy Fehlings, and Lauren Switzer. 2013. Designing action-based exergames for children with cerebral palsy. *Human Factors in Computing Systems: Proceedings of the SIGCHI Conference, (CHI 2013)*: 1261–1270. http://doi.org/10.1145/2470654.2466164

33. Jen Wen Hung, Chiung Xia Chou, Yen Wei Hsieh, et al. 2014. Randomized comparison trial of balance training by using exergaming and conventional weight-shift therapy in patients with chronic stroke. *Archives of Physical Medicine and Rehabilitation* 95, 9: 1629–1637. http://doi.org/10.1016/j.apmr.2014.04.029

34. J.T.P.N. Jacob and A.F. Coelho. 2011. Geo Wars–The development of a location-based game. *Revista Prisma. Com*, 14: 1–13. Retrieved July 11, 2011 from http://revistas.ua.pt/index.php/prismacom/article/view/972

35. João Tiago Pinheiro Neto Jacob and António Fernando Coelho. 2011. Issues in the Development of Location-Based Games. *International Journal of Computer Games Technology* 2011: 1–7. http://doi.org/10.1155/2011/495437

36. Rodrigo Rodrigues Joao SV Goncalves, Rosaldo JF Rossetti, Joao Jacob, Joaquim Goncalves, Cristina Olaverri-Monreal, Antonio Coelho. 2014. Testing Advanced Driver Assistance Systems with a serious-game-based human factors analysis

suite. In *IEEE Intelligent Vehicles Symposium Proceedings*, 13–18.

37. M Carmen Juan, Mariano Alcañiz, Calos Monserrat, Cristina Botella, Rosa M Baños, and Belen Guerrero. Using augmented reality to treat phobias. *IEEE computer graphics and applications* 25, 6: 31–7. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/16315475

38. M J Karvonen, E Kentala, and O Mustala. 1957. The effects of training on heart rate; a longitudinal study. *Ann Med Exp Biol Fenn 35*, 307–315.

39. Tim Kindberg and John Barton. 2001. Web-based nomadic computing system. *Computer Networks* 35, 4: 443–456. http://doi.org/10.1016/S1389-1286(00)00181-X

40. U KYLE. 2004. Bioelectrical impedance analysis?part I: review of principles and methods. *Clinical Nutrition* 23, 5: 1226–1243. http://doi.org/10.1016/j.clnu.2004.06.004

41. Giel Van Lankveld, Pieter Spronck, Jaap Van Den Herik, and Arnoud Arntz. 2011. Games as personality profiling tools. *2011 IEEE Conference on Computational Intelligence and Games, CIG 2011*, September: 197–202. http://doi.org/10.1109/CIG.2011.6032007

42. André Lemos. 2008. Mídia locativa e territórios informacionais. *Estéticas Tecnológicas - novos modos de sentir*: 207–230.

43. Fotis Liarokapis and Sara de Freitas. A Case Study of Augmented Reality Serious Games. . IGI Global. Retrieved June 15, 2012 from http://www.igi-global.com/chapter/case-study-augmented-reality-serious/40733/

44. Ricardo Lopes and Rafael Bidarra. 2011. Adaptivity Challenges in Games and Simulations: A Survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 2: 85–99. http://doi.org/10.1109/TCIAIG.2011.2152841

45. Brian Magerko. 2008. Adaptation in digital games. *Computer* 41, 6: 87–89. http://doi.org/10.1109/MC.2008.172

46. Thomas W. Malone and Kevin Crowston. 1990. What is coordination theory and how can it help design cooperative work systems? *Proceedings of the 1990 ACM*

conference on Computer-supported cooperative work - CSCW '90: 357–370. http://doi.org/10.1145/99332.99367

47. Pedro Meleiro, Rui Rodrigues, João Jacob, and Tiago Marques. 2014. Natural User Interfaces in the Motor Development of Disabled Children. *Procedia Technology* 13: 66–75. http://doi.org/10.1016/j.protcy.2014.02.010

48. David Michael and Sande Chen. 2006. *Serious Game : Games thar Educate, Train and Inform*. Premier-Trade.

49. Bobak Mortazavi, Mohammad Pourhomayoun, Hassan Ghasemzadeh, Roozbeh Jafari, Christian K. Roberts, and Majid Sarrafzadeh. 2015. Context-Aware Data Processing to Enhance Quality of Measurements in Wireless Health Systems: An Application to MET Calculation of Exergaming Actions. *IEEE Internet of Things Journal* 2, 1: 84–93. http://doi.org/10.1109/JIOT.2014.2364407

50. Kenton O'Hara. 2008. Understanding geocaching practices and motivations. *Proceedings of the SIGCHI Conference on Human ...*: 1177. http://doi.org/10.1145/1357054.1357239

51. Veljo Otsason, Alex Varshavsky, Anthony Lamarca, and Eyal De Lara. 2005. Accurate GSM Indoor Localization. iv: 141–158.

52. Veljo Otsason, Alex Varshavsky, Anthony LaMarca, and Eyal de Lara. 2005. Accurate GSM Indoor Localization. *UbiComp'05 Proceedings of the 7th international conference on Ubiquitous Computing*: 141–158. http://doi.org/10.1007/11551201_9

53. São Paulo. 2015. Exergames : Jogos digitais para longeviver melhor.

54. Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. 2000. The Cricket location-support system. In *Proceedings of the 6th annual international conference on Mobile computing and networking - MobiCom '00*, 32–43. http://doi.org/10.1145/345910.345917

55. Alberto B Raposo, Adailton J.a da Cruz, Christian M Adriano, and Léo P Magalhães. 2001. Coordination components for collaborative virtual environments. *Computers & Graphics* 25, 6: 1025–1039.

http://doi.org/10.1016/S0097-8493(01)00156-X

56. Philippe Renevier and Laurence Nigay. 2001. Mobile Collaborative Augmented Reality : The Augmented Stroll 2 Augmented Reality : Mobility and Groupware. *Proceedings of the 8th IFIP Working Conference on Engineering for HumanComputer Interaction* 2254: 299–316. http://doi.org/10.1007/3-540-45348-2

57. Daniel Salber, Anind K Dey, and Gregory D Abowd. 1999. The Context Toolkit : Aiding the Development of Context-Enabled. *CHI '99 Proceedings of the SIGCHI conference on Human Factors in Computing Systems*: 434–441. http://doi.org/10.1145/302979.303126

58. Gerhard Schall. 2009. Handheld Augmented Reality in Civil Engineering. *4th Conference on Computer Image Processing and its Application in Slovenia 2009 (ROSUS 2009)*: 19–25.

59. Juan M. Silva and Abdulmotaleb El Saddik. 2011. An adaptive game-based exercising framework. *2011 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems Proceedings*: 1–6. http://doi.org/10.1109/VECIMS.2011.6053847

60. Jeff Sinclair, Philip Hingston, and Martin Masek. 2007. Considerations for the design of exergames. *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia - GRAPHITE '07* 1, 212: 289–295. http://doi.org/10.1145/1321261.1321313

61. Jeff Sinclair, Philip Hingston, and Martin Masek. 2009. Exergame development using the dual flow model. *The 5th International Conference on Intelligent Environments - IE'09*: 1–7. http://doi.org/10.1145/1746050.1746061

62. Jan Smeddinck, Sandra Siegel, and Marc Herrlich. 2013. Adaptive difficulty in exergames for Parkinson's disease patients. *Proceedings of the 2013 Graphics …*: 141–148. Retrieved December 9, 2013 from http://dl.acm.org/citation.cfm?id=2532154

63. Hayeon Song, Wei Peng, and Kwan Min Lee. 2011. Promoting exercise self-efficacy with an exergame. *Journal of health communication* 16, 2: 148–62.

http://doi.org/10.1080/10810730.2010.535107

64. Olli Sotamaa. 2002. All The World's A Botfighter Stage: Notes on Location-based Multi-User Gaming. *Cultures*: 35–44.

65. Fernando Henrique Sousa. 2011. Uma revisão bibliográfica sobre a utilização do Nintendo® Wii como instrumento terapêutico e seus fatores de risco. *Revista Espaço Acadêmico* 8, 123: 155–160.

66. Pieter Spronck, Iris Balemans, and Giel Van Lankveld. 2009. Player Profiling with Fallout 3. *Proceedings, The Eighth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*: 179–184.

67. Andrew J Stapleton. 2004. Serious Games : Serious Opportunities. *Health Care*: 1–6.

68. Stefan Steiniger, Moritz Neun, and Alistair Edwardes. 2006. Foundations of Location Based Services. *Cartography* 1: 1–28. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.1844&rep=rep1&type=pdf

69. Penelope Sweetser and Peta Wyeth. 2005. GameFlow: a model for evaluating player enjoyment in games. *Computers in Entertainment (CIE)* 3, 3: 3. http://doi.org/10.1145/1077246.1077253

70. Z. Szalavari, Dieter Schmalstieg, Anton Fuhrmann, and Michael Gervautz. 1998. "Studierstube": An environment for collaboration in augmented reality. *Virtual Reality* 3, 1: 37–48. http://doi.org/10.1007/BF01409796

71. Kazumoto Tanaka, Jim Parker, Graham Baradoy, Dwayne Sheehan, John R Holash, and Larry Katz. 2012. A Comparison of Exergaming Interfaces for Use in Rehabilitation Programs and Research. *Loading* 6, 9: 69–81.

72. Zachary O Toups. 2007. Location-Aware Augmented Reality Gaming for Emergency Response Education : Concepts and Development. *Proc. ACM Computer Human Interaction 2007 Workshop on Mobile Spatial Interaction*: 1–4. Retrieved from http://ecologylab.net/research/publications/MobSpatialWorkshop.pdf

73. Minh Quang Tran and Robert Biddle. 2008. Collaboration in serious game development. In *Proceedings of the 2008 Conference on Future Play Research, Play, Share - Future Play '08*, 49. http://doi.org/10.1145/1496984.1496993

74. Anthony Whitehead, Hannah Johnston, Nicole Nixon, and Jo Welch. 2010. Exergame effectiveness: what the numbers can tell us. *Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games - Sandbox '10*, October: 55–62. http://doi.org/10.1145/1836135.1836144

75. Yun Zhang and Linda Candy. 2007. An in-depth case study of art-technology collaboration. *Proceedings of the 6th ACM SIGCHI conference on Creativity & Cognition*: 53–62. http://doi.org/10.1145/1254960.1254969

(Page intentionally blank)

11 ANNEXES

A. SPACIAL GAME DATA SERVICE

Game Analytics tools are used as an aid to help developers pinpoint issues with their games and potential opportunities that may arise. They usually track how players play the games and the issues they encounter.

Recently, recommendations systems have been used with some success as means to match game configurations with similar players, as seen in section 2.4. This approach allows for players with similar characteristics to have optimal game experience.

Combining both could result in a service capable of, on one hand, providing the statistics developers need to manually tune their game's mechanics and, on the other, continuously recommending specific developer-made game configurations to players that would benefit the most from them.

The usage of distinct pre-made game configurations allows for the game to more quickly adapt to the player's needs. Consider the following fictional scenario (Figure 75).

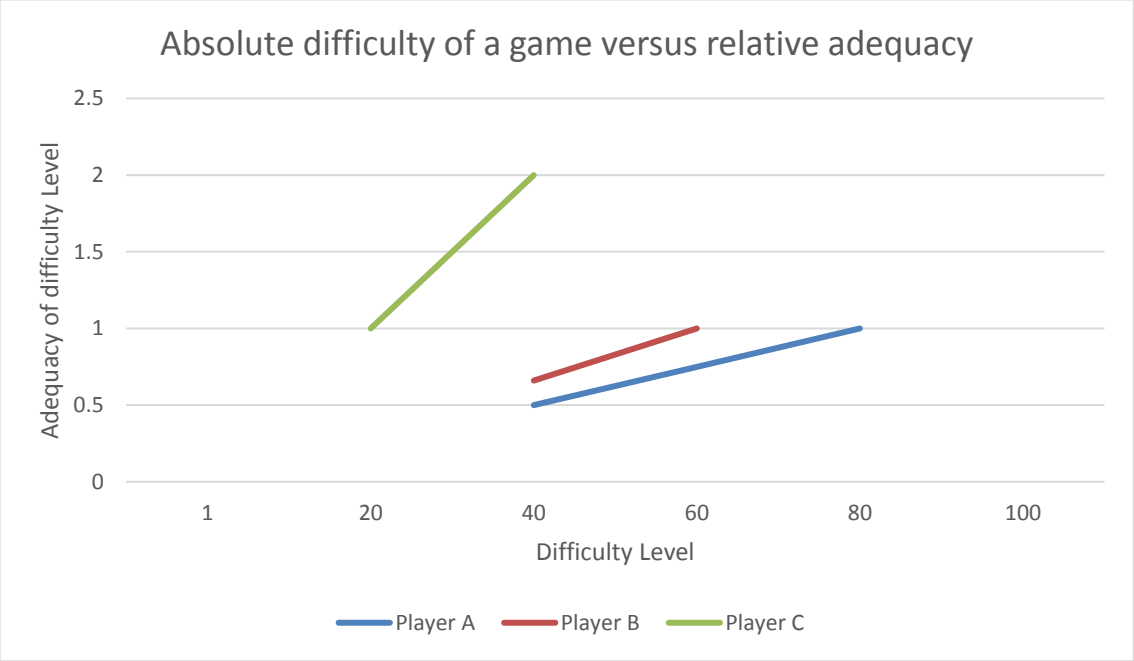


FIGURE 75- ABSOLUTE DIFFICULTY OF A GAME VERSUS RELATIVE ADEQUACY

This scenario describes a game with an initial difficulty level setting of 40. Lower values translate into an easier game, whereas greater values mean a more difficult game. Three players start playing the game. Here we define "adequacy of difficulty level" as being a numerical value that translates how adequate a difficulty level is for that player. A value of 1 means that it fits that player's current abilities (skill). A value lower than 1 means that the game is currently too easy, whereas a value greater than 1, that it is too difficult. So, an adaptive game would shift its difficulty back and forth so that it is deemed adequate i.e. with adequacy of difficulty level as close to 1 as possible.

Since all players start the game with a difficulty of 40, the game is not necessarily adequate for any of them. In the above case, the ideal difficulty levels, for the Players A, B and C, are, respectively, 80, 60 and 20. This means that the game will have to progressively adapt its difficulty to these players' skill, making it more difficult, slightly more difficult or significantly less difficult, in these particular cases. Since all adaptive games require some time to "sample" the players' skill, this means that while the game will, in the end, match the players' skill, it will take time and player interaction to do so.

However, if multiple game configurations of different initial difficulties could be made, if they could be matched to player profiles, and if these players (A, B and C) could be placed as being more or less similar to those profiles, a specific game configuration could be "recommended". This would result in a game with an initial difficulty closer to that of the adequate difficulty for that specific player. The game's own adaptivity mechanisms would work like before, but the initial condition would be closer to the ideal condition, resulting in a quicker, smoother adaptation.

As such, a solution capable of making such recommendations and holding those different game configurations would greatly benefit an already adaptive game.

The proposed Game Data Service aims to provide developers with the needed tools to tackle this issue in the manner described. It is meant to be a decentralized service, as it would only be responsible for holding the games' and players' data and configurations, and for recommending game configurations, matching players' preferences.

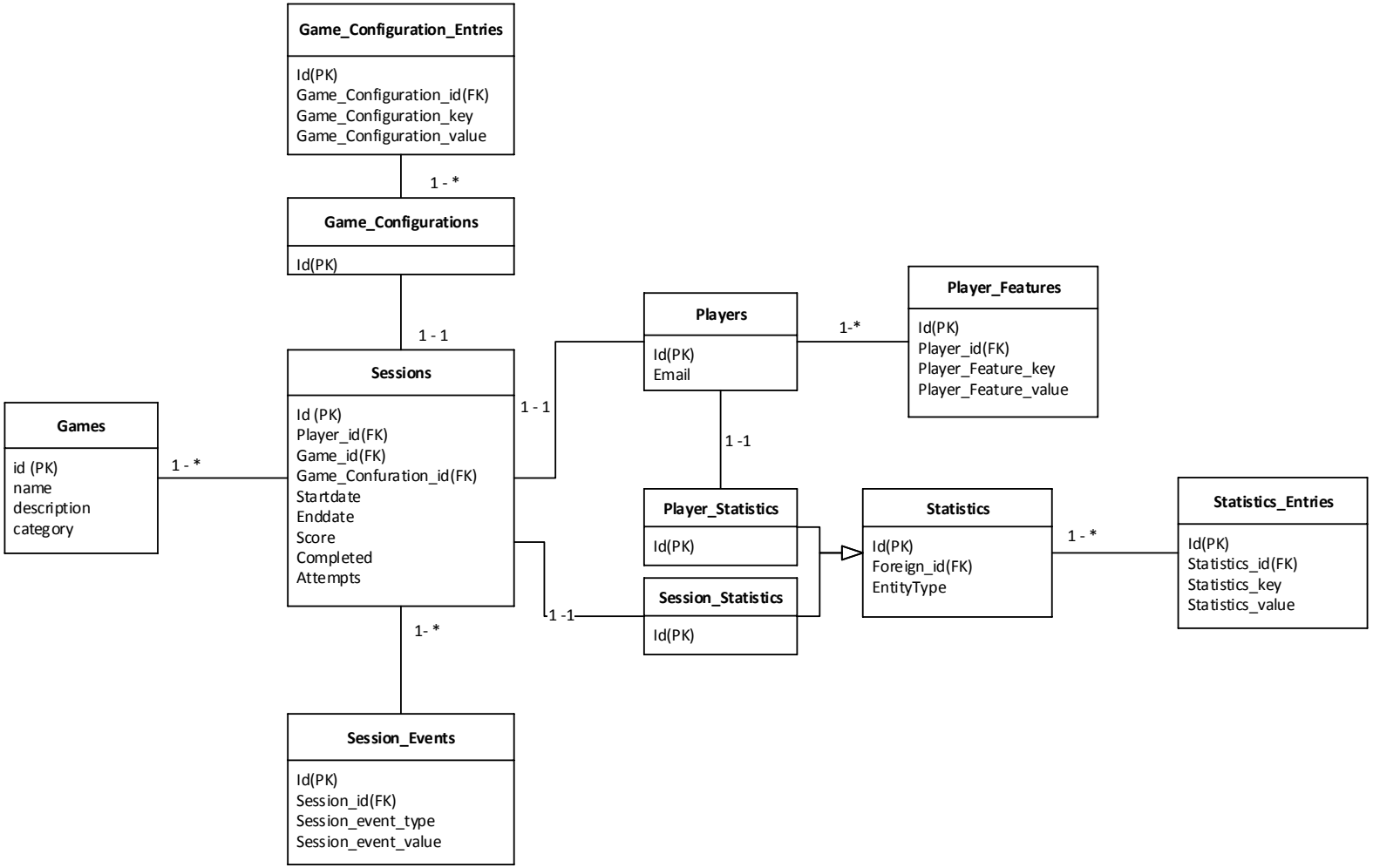


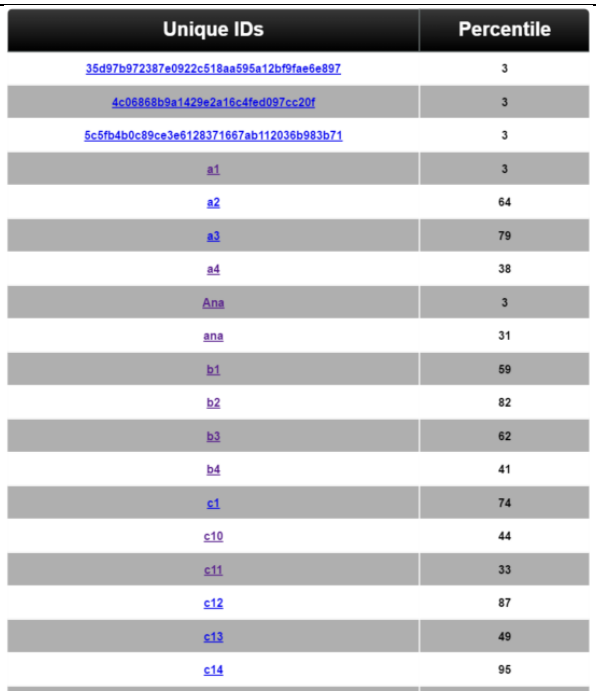
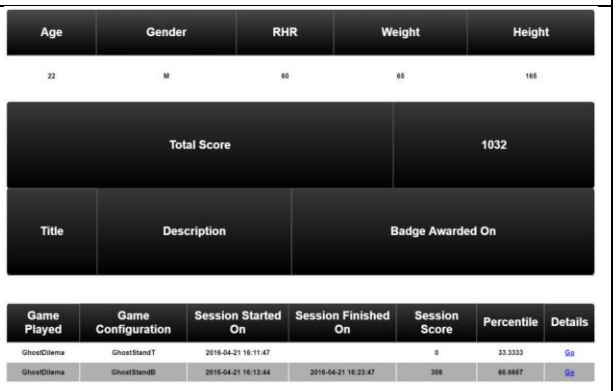
FIGURE 76- UML DATABASE DIAGRAM OF THE GAME DATA SERVICE

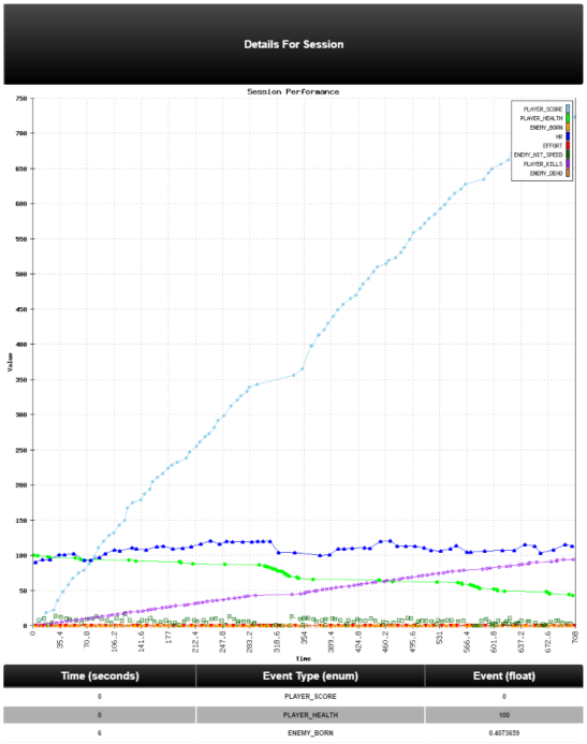
As the UML diagram in Figure 76 illustrates, the Game Data Service revolves around the notion of a game Session. This session, resulting in the interaction between a player and a game, will hold information relative to it, such as the span of time, significant game events that occurred and the configuration (or game settings) for that session. Statistics associated with a game session are also stored (distance traveled, number of enemies defeated, for instance). As for the Player table it stores a variable amount of information that can be used by different games (age, sex, weight, height). The only data that is expected to be mutable are relative to the player and his/her statistics, as they contain present facts. Sessions and game configuration related data are facts and are not meant to be changed or deleted. So, it is expected that the endpoints the web-service makes available are only for insertion of information (starting a game session, ending a game session, adding game session events and statistics, setting the game configuration for that session and its characteristics, updating player information or statistics). This data can then be used, manually, for developers or game analytics tools as means to determine what where the issues (if any) in specific gameplay sessions, or automatically by recommendation systems as means to recommend specific game configurations for players with similar traits.

IMPLEMENTATION DETAILS

This platform, comprised of a Frontend, Web-Services and MySQL database aims to provide means to record real-time game telemetry, useful for both statistical analysis and game adaptivity. The database is similar to that specified in Figure 76. The Frontend displays players, their game sessions and game session telemetry.

TABLE 7- FRONTEND VIEWS OF THE GAME DATA PLATFORM

View Description	View Screenshot
Players View: The initial view simply lists alphabetically all the players that have a profile in the platform. It additionally displays the percentile each player is at, based on the total score. By clicking a player's id the user is taken to the Player Profile.	
Player Profile View: In this view, one can see the stored parameters of the player's profile in the first row, and the total score in the second. Finally, all the player's gaming sessions ensue. On display are, for each session, the game played, the variation of the game (if any), when did the session begin, if the session was finished correctly and when, its score, the percentile of the session relative to all the others the player logged, and a link to the detailed view.	

Session Detail View: All of a session's logged gamed events are displayed in this view. An automatically generated 2d plot (via phplot) rests on the upper part of the view. Note that only numerical values are on display, and no value is normalized. This means that game events with very small ranges in values will be difficult to see alongside values with great numerical ranges. Bellow the plot is a table listing events (type and value), ordered chronologically.	

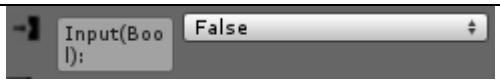
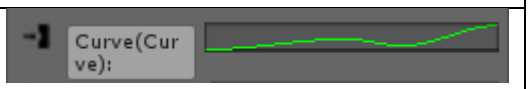
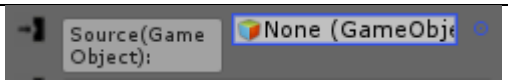
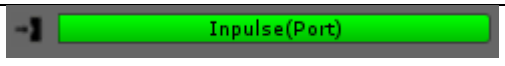
This platform also provides a series of endpoints for applications to communicate with, providing the following features:

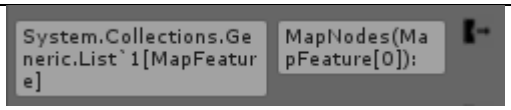
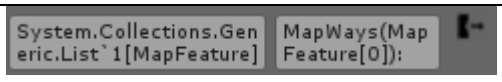
- Create/Login Player
- Set Player Parameters
- Create Game
- Create/End Session for a Player
- Add Session Event to a Session
- Add Session Statistics to a Session
- Add Game Configurations
- Add Parameters to a Game Configuration

Note that the Game Configuration features are what actually supports Grapphers' cloud loading and saving features. Each graph is stored in this service as a Game Configuration with only two parameters: the serialized value of the graph and the timestamp of its serialization. The addition of other parameters to game configurations was not deemed necessary during the development of the prototypes for the case studies, as the graph itself contained all the parametrization and semantics needed for changing the game's behaviour.

B. GRAPPHER NODE TYPES

TABLE 8- GRAPH NODE VALUE TYPES DESCRIPTION AND GRAPHICAL REPRESENTATION IN GRAPPHER

Node Value Types: Description	Appearance in Editor
GraphNodeAction: This is an unassignable Node Value Type. It represents a visual button with a certain behaviour (of type System.Action) that will manipulate the node's content. Usually it is used to add extra Node Values to a Node.	
GraphNodeArray: This represents a generic Array of Objects. In some instances, explicit Array types are recommended to avoid casting / conversion problems	
GraphNodeBool: An input with the only possible values of "False" or "True"	
GraphNodeCurve: This represents a 2D function, based on Unity's AnimationCurve.	
GraphNodeDropDown: It contains a series of Strings with only one selected value.	
GraphNodeFloat: Represents a floating point value.	
GraphNodeGameObject: Contains the reference of a Scene's GameObject or Prefab.	
GraphNodeImpulse: An impulse can be an Inpulse (if added to the Inputs) or an Out-	

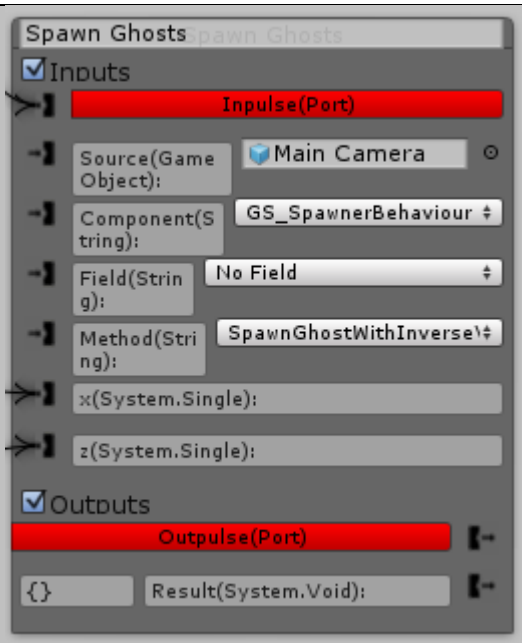
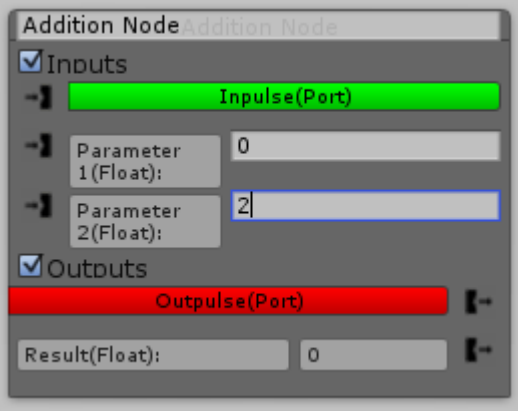
pulse (if added to the Outputs). These represent if a Node can execute (a "true", green Inpulse port) and if a Node has finished its execution (a "false", red Outpulse port). It is used to enforce precedence and synchronization between nodes.	
GraphNodeInt: Represents an integer value.	
GraphNodeBooleanOperator: It consists of a dropdown containing several binary infix boolean operators (And, Or, XOr, Like, Greater, Less, Equal, LoE, GoE).	
GraphNodeMapFeature: Represents a GeoStream MapFeature.	
GraphNodeMapFeatureArray: An Array of MapFeatures.	
GraphNodeMapNodeArray: An Array of MapNodes.	
GraphNodeMapWayArray: Contains the value of an Array of MapWays.	
GraphNodeMaterial: Contains the reference to one Unity Material definition. Since these are not serializable, the ID reference of a material is stored when the value is saved,	

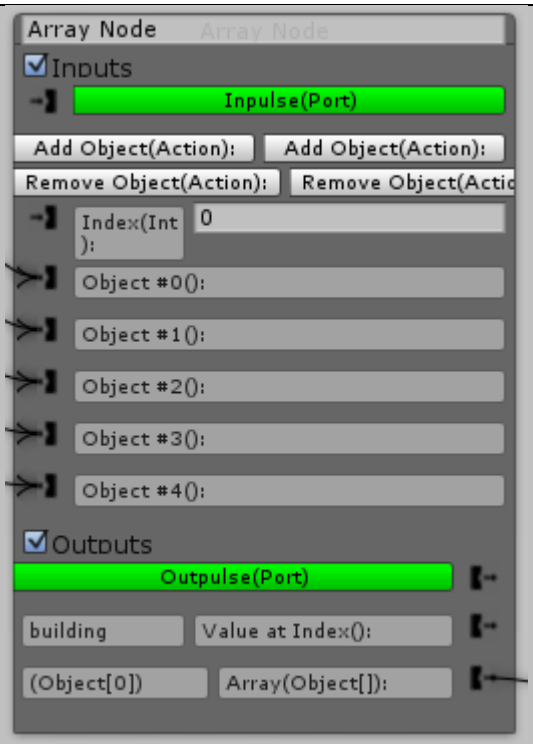
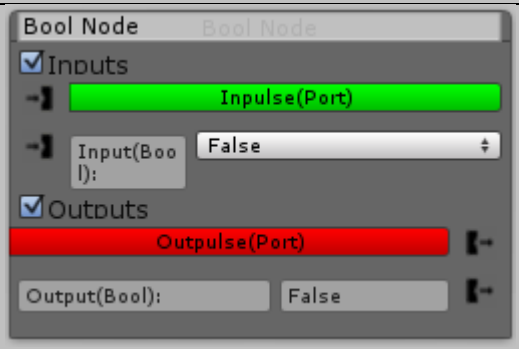
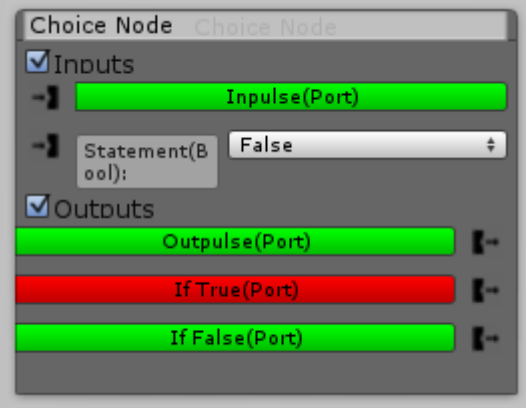
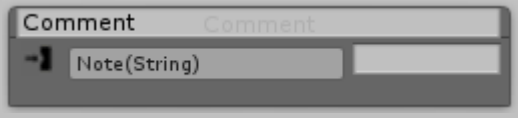
and is used to look it up once it is deserialized.	
GraphNodeObject: Holds the value of a generic System.Object value.	
GraphNodeString: Holds the value or reference of a String.	
GraphNodeVector2: Allows for the using of Unity's Vector2 definition. As this class is not serializable, it is converted to and from an internal Vector2 definition as needed.	
GraphNodeVector3: Allows for the using of Unity's Vector3 definition. As this class is not serializable, it is converted to and from an internal Vector3 definition (CustomVector3) as needed.	

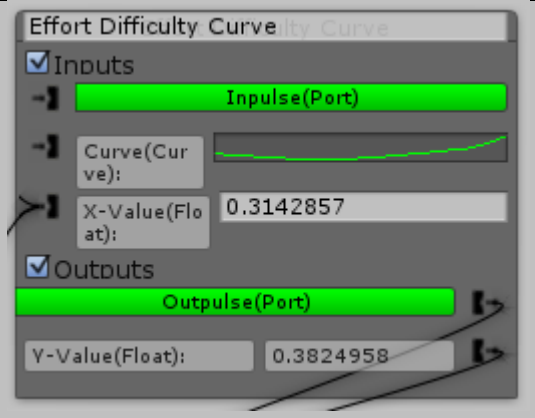
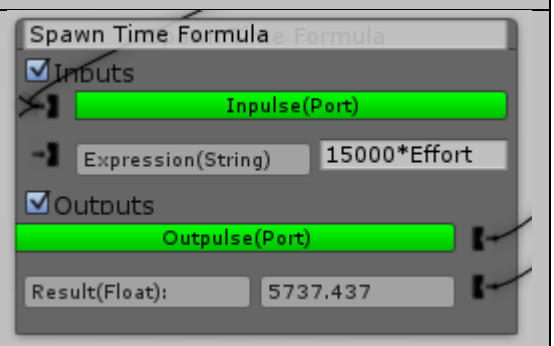
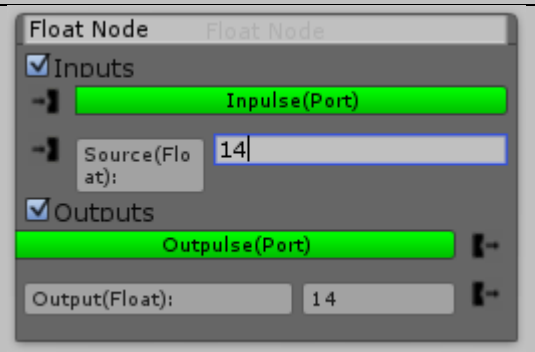
C. GRAPPHER CORE NODES

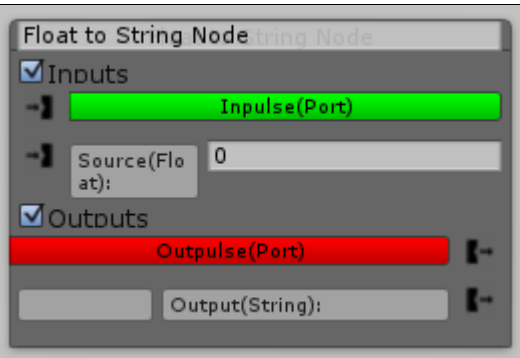
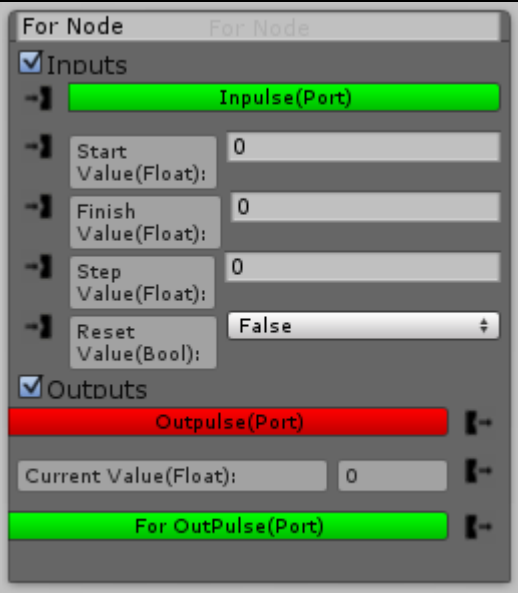
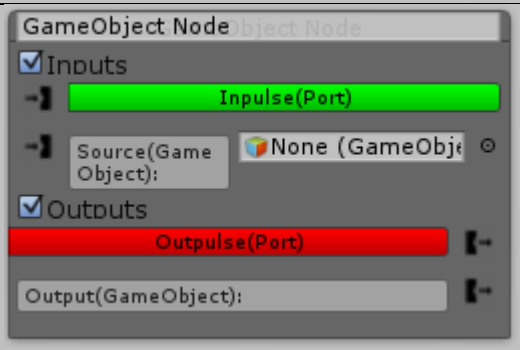
TABLE 9- GRAPPHER'S NODE TYPES PRESENT IN THE CORE PACKAGED

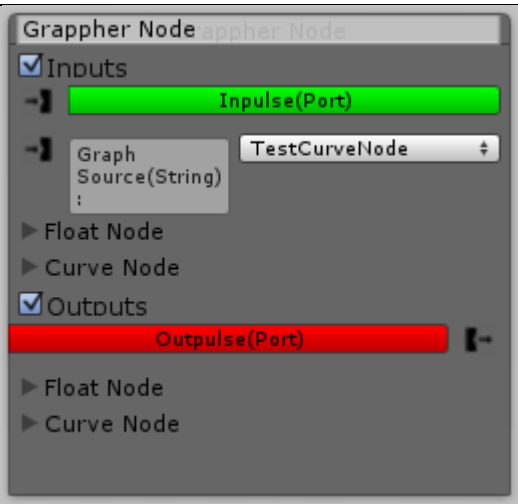
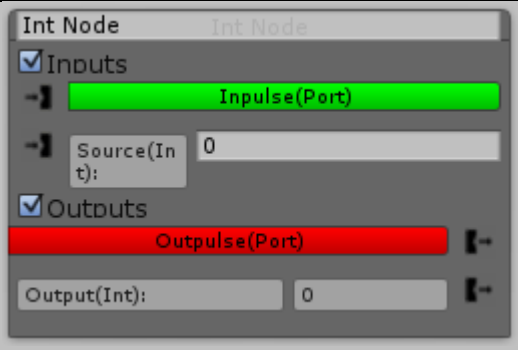
Node Types	**Appearance in Editor**

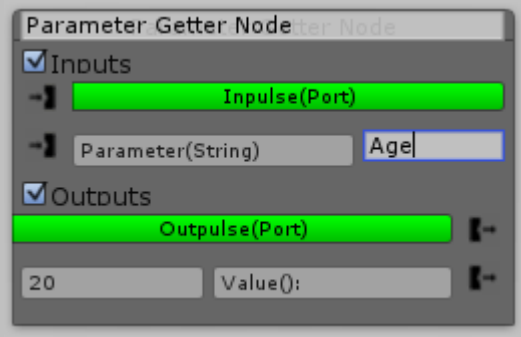
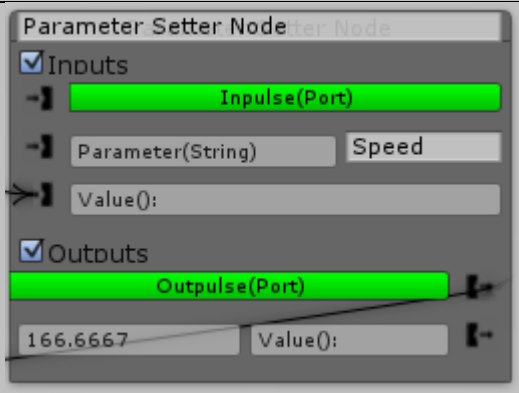
ActionNode: The ActionNode can be used to bridge the gap between Grappher Nodes and Unity MonoBehaviours, present in any given GameObject. It can be used to invoke a function and set or get a field of any given Unity MonoBehaviour. It makes use of C#'s Reflection design pattern to dynamically load functions, fields and parameters of any MonoBehaviour of a GameObject. If a specific GameObject is specified, the action will only affect that instance. If a Prefab is referenced, the Action will affect one Instance per Tick (restarting with the first instance after the last instance's action has been executed). This allows for complex or critical code to be written in a Unity MonoBehaviour and for it to be freely parametrized or used to feed Node Values. However, if a stored ActionNode refers to a Function/Field/Component that is no longer available, it will result in an Exception.	
AdditionNode: One of the first nodes that was developed, it is capable of taking two parameters and storing the resulting sum.	

ArrayNode: An ArrayNode allows for the creation of Object arrays in design and runtime. The array contains only references to objects and has a length that can only be specified in design time.	
BoolNode: The BoolNode allows the user to define a True/False value.	
ChoiceNode: A ChoiceNode is used for flow control during the execution of a Node. It contains two additional Impulse Ports that are enabled or disabled depending on the value of the given Statement.	
CommentNode: As Graphs can easily grow in complexity, allowing a User to annotate certain sections of graph to clarify	

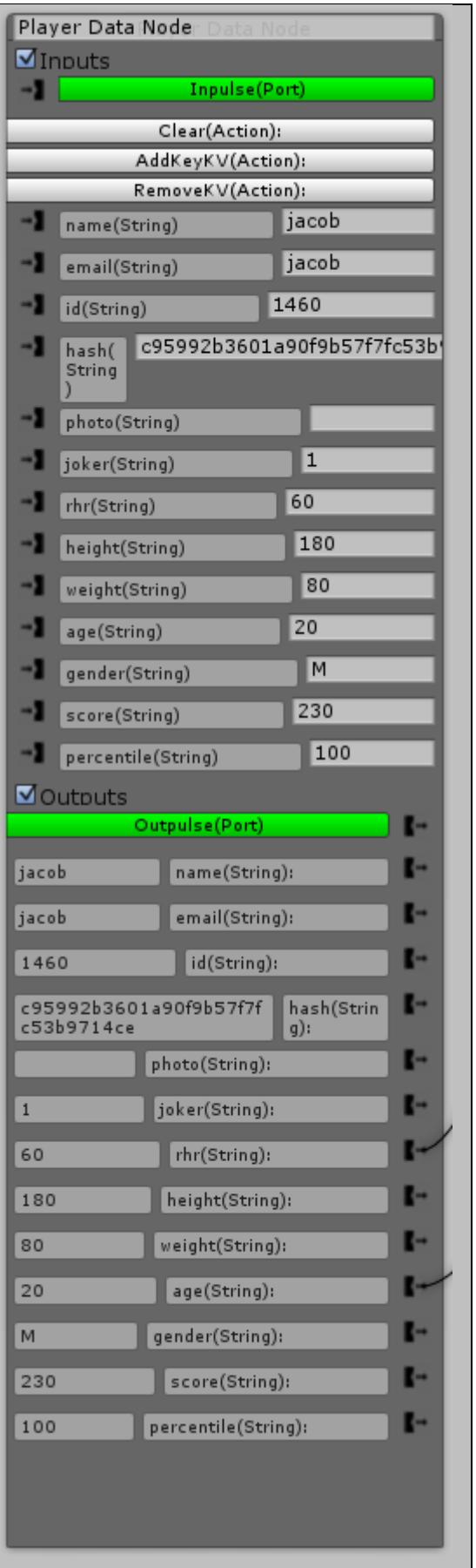
its purpose. These CommentNodes are not executed and simply store a String value.	
ComparisonNode: This node takes two parameters and performs an infix Boolean operation with them. The resulting Boolean value is then outputted.	
CurveNode: This node takes a 2D function, represented as a Curve, and outputs the Y-Value of that function at a given X-Value. If the curve contains no definition for that X-Value, it will throw an exception.	
EvaluatorNode: The EvalutatorNode is capable of evaluating an arithmetic or logical expression. The resulting value may be a string, float or bool. It is based on the NCalc library, making use of its expression evaluation engine.	
FloatNode: This Node holds the value of or refers to a single precision floating point number.	

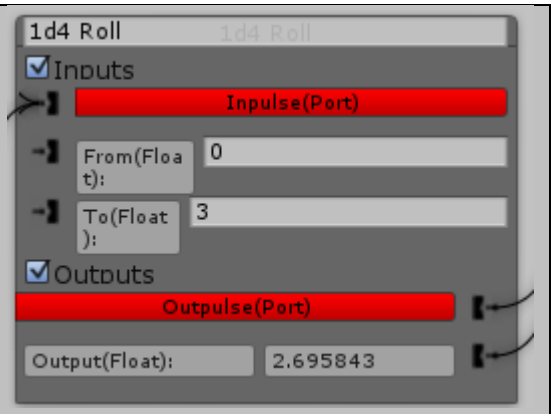
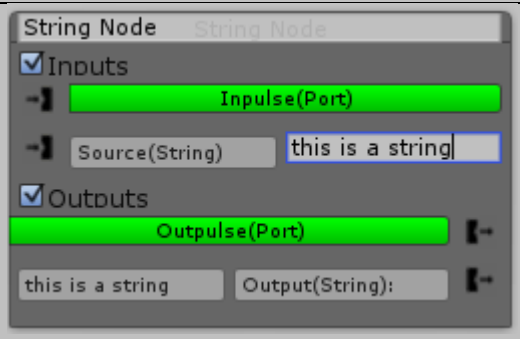
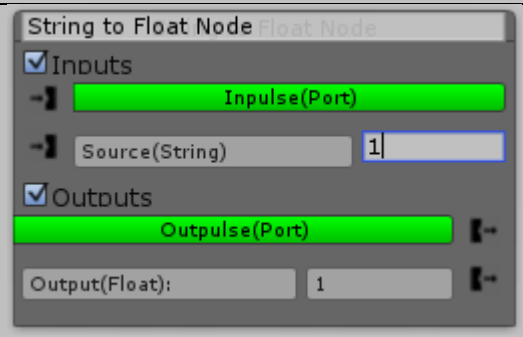
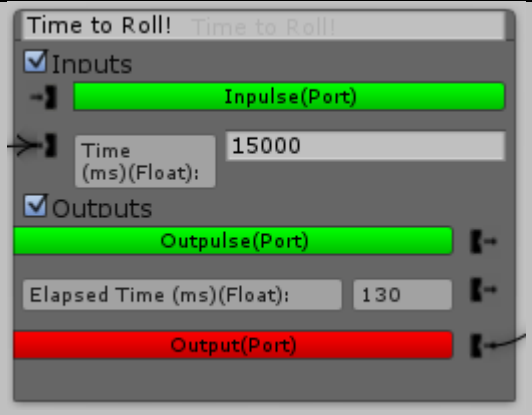
FloatToStringNode: Performs an explicit conversion between a Float and a String. It is currently not used, as Grappher no longer enforces explicit type conversion.	
ForNode: This node represents a "For" cycle. The initial, step and finishing conditions are defined, as well as if the "For" cycle should be reset to its starting value whenever the finishing is reached. Otherwise, it will not run at all. At each Tick, the for node will apply the step value to its current value. If it has not yet reach the finishing value, its Outpulse port is set to false and For Outpulse port is set to true. Once it finished, the Outpulse port is set to true.	
GameObjectSelectNode: This Node provides the user a way to select a GameObject or Prefab and pass it along.	

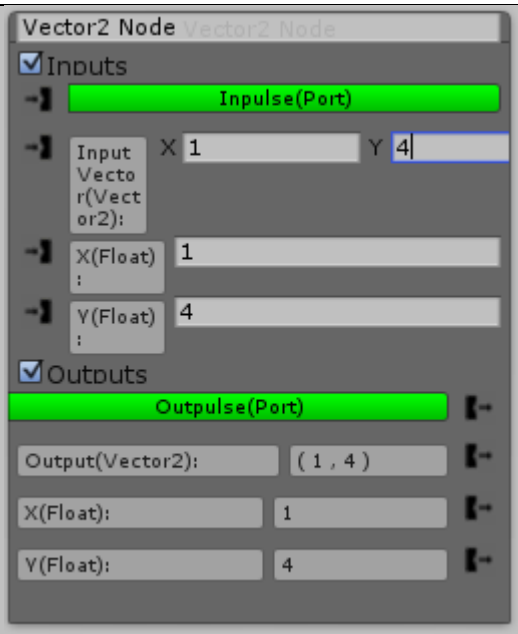
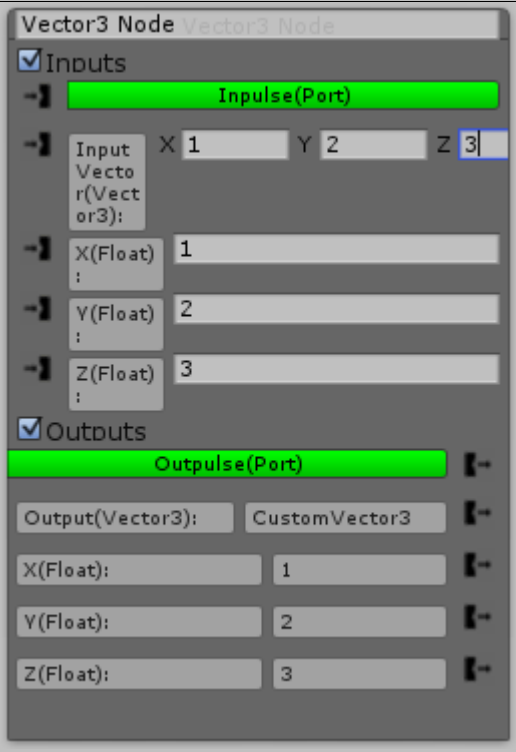
GrappherNode: The Grappher Node is used to invoke a previously defined Graph in a node. This form of graph encapsulation allows for new graphs to reuse sections of previously defined graphs. All nodes and their values are exposed in the Grappher-Node (their inputs and outputs in the node's respective sections) allowing for the invoked graph to be parametrized or to parametrize other nodes of the current graph. It is also used whenever the user decides to create a "Group" when selecting multiple nodes in the current graph. In this circumstance the Graph Source parameter is hidden, as no external graph is present in this context. Internally, the Grappher Node contains the definition of another Graph and its nodes. If a user makes changes to an external graph that is referenced in some GrappherNode, the definition of said changed graph is not propagated to the GrappherNodes in question, unless the user explicity reselects that node. This is by design so as to avoid breaking other graphs that may depend on a now deprecated version of the graph.	
IntNode: Holds the value, or reference to an integer.	

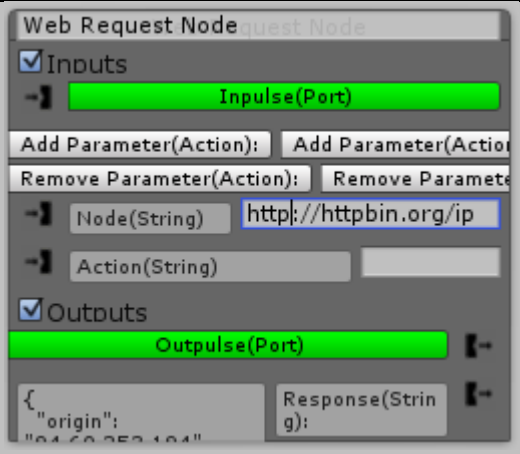
MultipleChoicesNode: This Node has a behaviour similar to that of a ChoiceNode. It simply allows for multiple statements to be added or removed, serving as a "switch" for multiple inputs, controlling the flow of several parts of a graph. Its behaviour can be replaced by multiple Choice Nodes.	
ParameterGetterNode: Allows for accessing a stored System.Object graph parameter by type.	
ParameterSetterNode: Defines a global Parameter. Parameters can be redefined multiple times inside a Graph (even inside other Graphs). The current Parameters and their values can be seen in the Parameters Toolbar in the Graphher Editor Window. Parameters are not serialized when a Graph is saved, being recreated when its ParameterSetterNode is executed.	

PlayerDataNode: This node allows for user data to be set or accessed. This data is made available via a static PlayerData class that contains a dictionary of strings. New Key-Value pairs can be added or removed at will. The PlayerData can be uploaded to (or previously downloaded from) the Game Data Tool webservice. This node is used for adapting a game's behaviour in accordance to its current user data.



RandomNode: This Node allows for the creation of a random float between two value, at each Tick.	
StringNode: This node holds the value of or reference to to a String.	
StringToFloatNode: Another explicit conversion Node, this time from String to Float values. This node is no longer used and remains only for legacy purposes.	
TimerNode: This node has a port that only is activated when a certain time has passed. Whenever the specified time limit is reached or exceeded, it proceeds to reset its internal clock value to zero.	

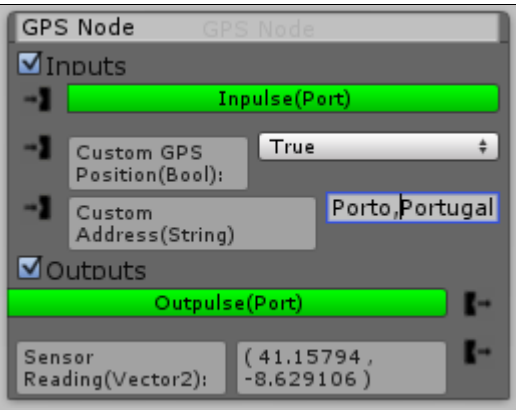
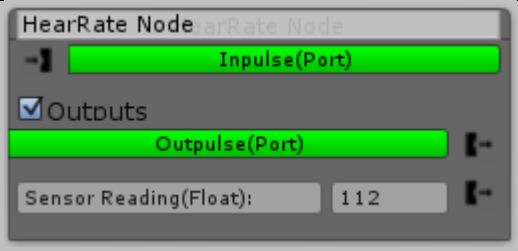
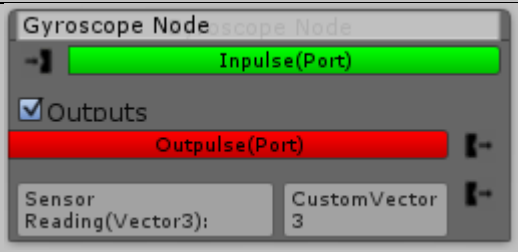
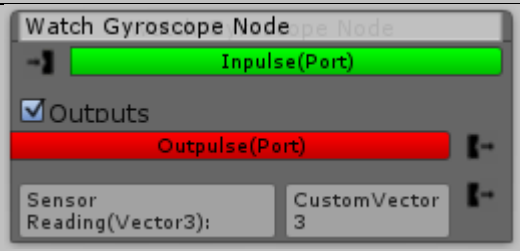
Vector2Node: This node allows for the user to create a new UnityEngine.Vector2 or access the X and Y values of an existing one.	
Vector3Node: This node allows for the user to create a new UnityEngine.Vector3 or access the X, Y and Z values of an existing one.	

WeatherNode: This node makes a WebRequest to the OpenWeatherMap API, retrieving weather data from a given Latitude/Longitude pair of coordinates. The request is made in a separate thread. While the request has no response, its Outpulse port is set to false. Whenever the Latitude or Longitude values are changed, a new request is made.	
WebRequestNode: This node allows for GET HTTP Requests to be made. The request is made in a separate thread, so as to not block the execution of a node. Whenever the Node, Action or Parameters are changed, the request is executed again.	

D. GRAPPHER GEOSENSORS NODES

TABLE 10- GRAPPHER'S NODE TYPES PRESENT IN THE GEOSENSORS PACKAGE

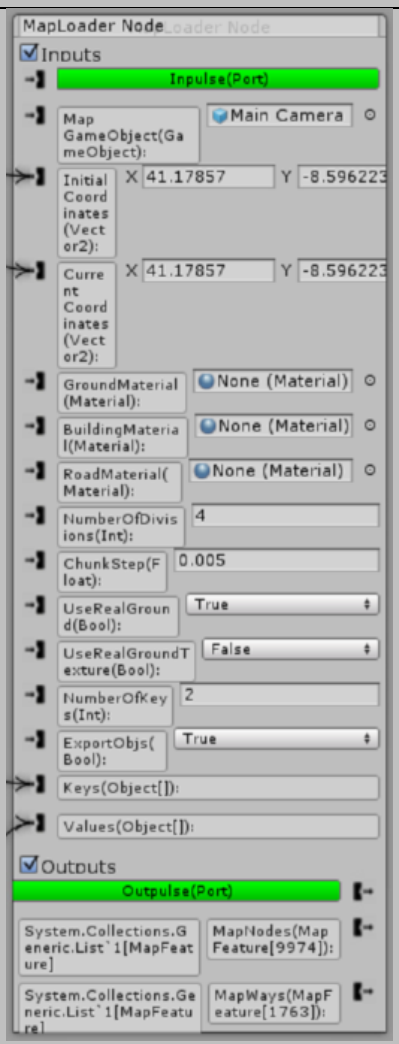
Node Types	**Appearance in Editor**

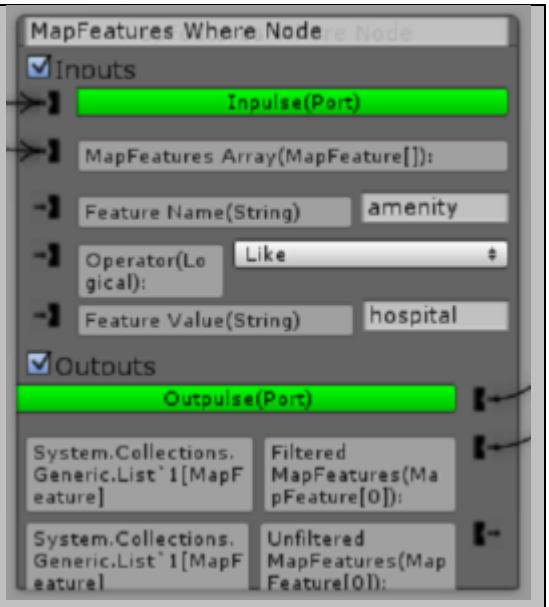
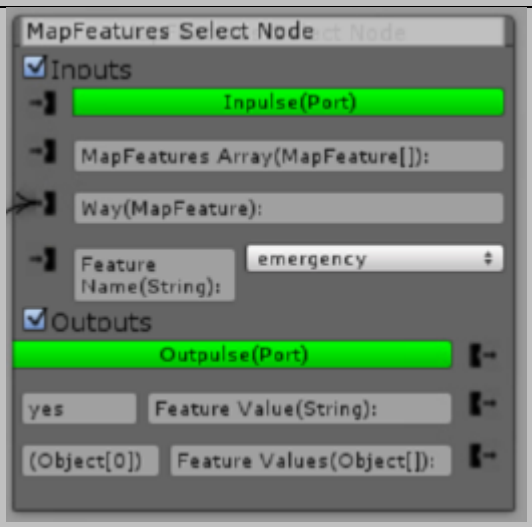
GPSNode: This Node makes use of the GeoSensors definition of the GPS Sensor, providing the pair of coordinates (Latitude, Longitude) of the latest Sensor Reading. If the Custom GPS Position value is set to True, it will not access the device's GPS sensor. Instead, it will geocode a given address, converting it to a pair of Latitude/Longitude coordinates, through Google Maps Geocoding API.	
HeartRateNode: Provides the latest Heart Rate value (in BPM) that the Smartwatch Heart Rate Sensor reported. If no Smartwatch Heart Rate Sensor is available, GeoSensors provide a Random Value, from 50 to 200.	
GyroscopeNode: Provides the current Euler Angles reported by the device's Orientation Sensor data. In some devices, this value may be the result of the post-processed and fused data from several sensors (Accelerometer, Gyroscope and Compass). If no Orientation sensor is available, it will report the data from the Gyroscope sensor. If no Gyroscope sensor is available, it will provide random (0 to 360) values for all three axis.	
WatchGyroscopeNode: Similar to the GyroscopeNode, this Node will provide the latest reading from the connected smartwatch's Orientation sensor, if available. Otherwise it will provide the data of the	

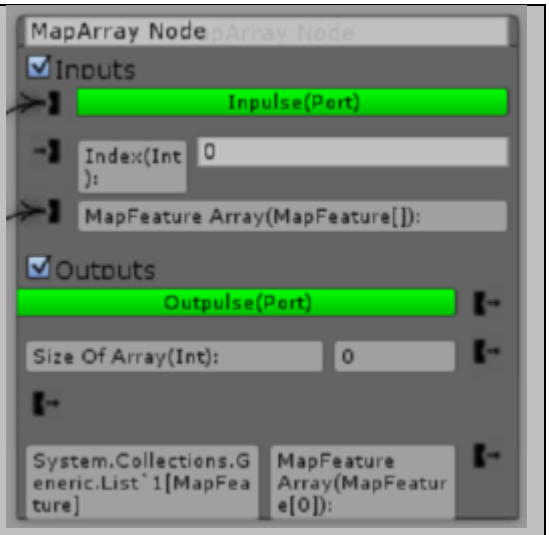
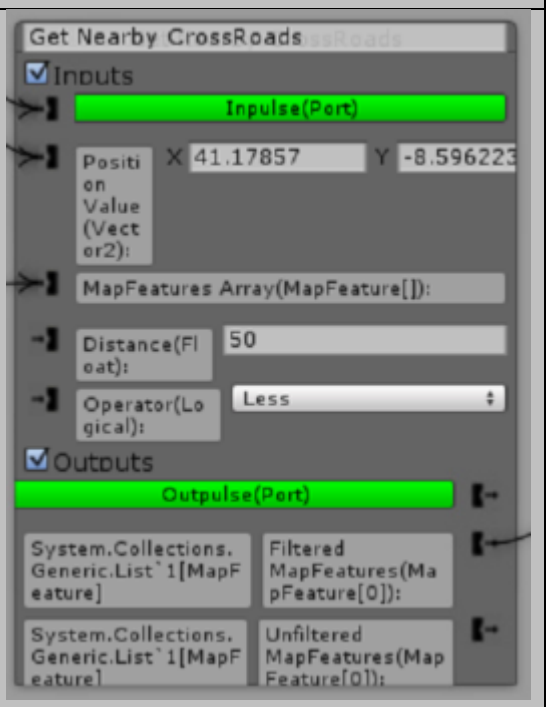
smartwatch's Gyroscope sensor. If none is available, it will provide random values (0 to 360) for all the three axis.	

E. GRAPPHER GEOSTREAM NODES

TABLE 11- GRAPPHER'S NODE TYPES PRESENT IN THE GEOSTREAM PACKAGE

Node Types	Appearance in Editor
MapLoaderNode: This Node is tightly integrated with GeoStream. It allows for the graph designer to attach a MapChunkLoader Component to a new GameObject (if no GameObject is provided) or to the provided one. All other input parameters are the same as that of the MapChunkLoader Component. It does, however, provide the user with two outputs: two arrays of the MapNodes and MapWays that instance of a MapChunkLoader is currently managing. As the MapChunkLoader behaves independently from the execution of the graph, these outputs' length and content can vary as new MapFeatures are loaded or unloaded when needed.	

MapWhereNode: This Node provides a filter for selecting MapFeatures that share some common property. As such, it allows for MapFeatures to be filtered based on one feature, and operator and a value. It provides the user with two ports, one for the MapFeatures that passed the filter's condition and another for those that did not. The limit of only selecting based on one filter condition can be overcome, by linking two MapWhereNodes in series (providing an "AND or AND NOT" clause) or in parallel (serving as functional "OR" and "OR NOT" statements). This allows for queries to be modularly constructed and each partial query statement to be inspected visually.	
MapSelectNode: As MapFeatures have many possible features that characterize them, a Node that allows the user to select these values is also needed. The MapSelectNode allows for the user to provide either a MapFeature instance or array, select from a dropdown of possible feature's names and outputting either the value of the MapFeature instance, or an array of the feature's value in each of the MapFeatures present in the array, depending on the input given.	

MapArrayNode: Allows for selecting a single MapFeature at a given index of a provided MapFeature array. If the MapFeature has a "name" feature, it will also be present as a string in the output.	
MapDistanceNode: This Node serves the purpose of selecting MapFeatures from a given MapFeature array that are at a certain distance from the provided pair of coordinates. Other operations can also be selected, so as to select MapFeatures that are within, outside or at the limit of the given distance.	

F. GHOSTSTAND EXPERIENCE QUESTIONNAIRE

Exergaming experience **This experience pretends to test Exergames (games that require the player to exert themselves physically) in Virtual Reality. Please sign the Informed Consent Form prior to this questionnaire.**
ID **(Please fill this field only with the ID given to you by the researcher)**
Player Profile
Physical and cognitive skills * **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**
1. I am athletic
2. I am an experienced player of video games
3. I am proficient in using computers
4. I am capable of interpreting maps
5. I am able to listen to sounds clearly
6. I am capable of walking and running
Usual Gaming Experience (PG) **When I play my usual games:**
Absortion **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**
7. I feel scared
8. I lose track of where I am
9. I feel different
10. Time seems to stand still or stop
11. I feel spaced out
Flow **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**

12. I don't answer when someone talks
13. I can't tell I'm getting tired
14. If someone talks to me I don't hear
15. I feel like I can't stop playing
16. The game feels real
17. I get wound up
18. Playing seems automatic
19. I play without thinking how to play
20. Playing makes me feel calm]
21. Playing the game was easy
Presence **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**
22. Things seem to happen automatically
23. My thoughts go fast
24. I play longer than I meant to
25. I lose track of time
Immersion **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**
26. I really get into the game
27. I felt in total control of what I was doing
28. I had a strong sense of what I wanted to do while playing
Exertion **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**
29. The game was physically intensive
30. The game was physically challenging to play

31. I easily kept up with the physical demand of the game
32. The game's physical demand was too much at times
33. The game stopped at the right time
34. Physically, I could have kept playing even after the game was over
35. Playing the game was fine for a while, but then the physical demand became troublesome
Game (B) **When I play this game:**
Absortion **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**
7. I feel scared
8. I lose track of where I am
9. I feel different
10. Time seems to stand still or stop
11. I feel spaced out
Flow **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**
12. I don't answer when someone talks
13. I can't tell I'm getting tired
14. If someone talks to me I don't hear
15. I feel like I can't stop playing
16. The game feels real
17. I get wound up
18. Playing seems automatic
19. I play without thinking how to play

20. Playing makes me feel calm]
21. Playing the game was easy
Presence **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**
22. Things seem to happen automatically
23. My thoughts go fast
24. I play longer than I meant to
25. I lose track of time
Immersion **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**
26. I really get into the game
27. I felt in total control of what I was doing
28. I had a strong sense of what I wanted to do while playing
Exertion **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**
29. The game was physically intensive
30. The game was physically challenging to play
31. I easily kept up with the physical demand of the game
32. The game's physical demand was too much at times
33. The game stopped at the right time
34. Physically, I could have kept playing even after the game was over
35. Playing the game was fine for a while, but then the physical demand became troublesome
Game (A) **When I play this game:**

Absortion **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**
7. I feel scared
8. I lose track of where I am
9. I feel different
10. Time seems to stand still or stop
11. I feel spaced out
Flow **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**
12. I don't answer when someone talks
13. I can't tell I'm getting tired
14. If someone talks to me I don't hear
15. I feel like I can't stop playing
16. The game feels real
17. I get wound up
18. Playing seems automatic
19. I play without thinking how to play
20. Playing makes me feel calm]
21. Playing the game was easy
Presence **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**
22. Things seem to happen automatically
23. My thoughts go fast
24. I play longer than I meant to
25. I lose track of time

Immersion **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**
26. I really get into the game
27. I felt in total control of what I was doing
28. I had a strong sense of what I wanted to do while playing
Exertion **Consider 1 as "No", 3 as "Maybe" and 5 as "Yes".**
29. The game was physically intensive
30. The game was physically challenging to play
31. I easily kept up with the physical demand of the game
32. The game's physical demand was too much at times
33. The game stopped at the right time
34. Physically, I could have kept playing even after the game was over
35. Playing the game was fine for a while, but then the physical demand became troublesome
Comments and Suggestions
If you have any comments or suggestions please write them below

G. Questionnaire and Game Results

Game	ID	Age	Gender	RHR	Weight	Height	Score	Finished Test	Dominant H.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
PG	A1	40	M	65	100	179	?	N	R	2	4	5	5	5	5	3	4	3	4	5	2	4	2	4	4	4	4	4	3	4	4	4	5	5	4	4	5	4	5	3	4	4	4	2
PG	A2	29	M	60	72	182	?	Y	L	3	4	5	5	5	5	1	1	2	2	2	3	4	1	2	1	4	4	5	3	2	4	4	5	5	4	4	4	2	2	5	1	2	4	2
PG	A3	34	M	60	75	174	?	Y	R	2	4	4	4	5	4	1	3	3	4	4	3	2	2	4	2	3	4	4	4	4	4	4	4	5	4	4	4	4	1	1	5	1	3	5	2
PG	A4	40	M	60	71	171	?	Y	R	3	1	4	4	5	5	2	3	2	2	2	2	2	2	2	2	2	3	2	2	3	3	3	3	2	3	2	3	2	2	4	2	3	4	2
PG	B1	24	F	60	62	172	?	Y	R	4	1	2	5	5	5	2	2	1	1	3	1	1	1	1	3	2	2	2	1	3	2	2	2	1	3	2	2	1	3	4	2	4	4	2
PG	B2	20	M	60	65	176	?	Y	R	5	5	5	4	5	5	1	1	1	2	2	2	2	1	2	2	4	4	4	3	4	2	4	3	3	4	4	4	2	2	5	1	3	5	1
PG	B3	38	M	62	81	170	?	Y	R	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

PG	B4	23	M	60	60	164	?	N	R	4	5	4	3	4	5	3	3	1	4	5	4	3	5	2	2	3	4	2	3	3	4	3	5	4	4	3	4	1	1	5	1	3	5	1
PG	C1	22	M	60	74	172	?	Y	R	5	5	5	4	4	5	3	3	2	4	4	3	2	3	3	3	2	2	2	3	3	2	4	4	4	4	5	4	5	5	4	3	3	4	3
PG	C2	33	M	60	82	174	?	Y	R	2	5	5	5	5	5	2	3	3	5	4	4	5	4	4	4	3	4	4	3	3	4	4	4	4	4	3	3	1	1	2	2	3	4	1
PG	C3	27	M	60	69	173	?	Y	R	5	3	4	4	4	5	1	1	3	1	1	4	1	3	2	2	2	2	3	2	3	2	4	4	3	3	3	3	3	3	4	2	2	4	2
PG	C4	29	M	60	86	182	?	Y	R	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
PG	C5	26	M	60	57	171	?	Y	R	3	5	5	5	4	5	2	1	3	4	4	4	4	4	5	3	2	5	5	4	4	5	5	5	5	5	5	5	3	3	3	4	5	4	4
PG	C6	22	M	60	62	170	?	Y	R	4	5	5	4	4	5	2	3	3	3	4	2	2	3	4	4	4	4	4	4	4	4	3	4	5	4	4	4	3	2	4	2	4	4	3
PG	C7	26	F	64	52	165	?	Y	R	4	3	5	5	5	5	2	3	3	4	3	2	2	4	3	3	2	3	4	3	2	3	3	4	4	4	3	3	1	1	5	1	3	5	1
PG	C8	18	F	60	50	158	?	Y	R	2	2	3	5	5	5	4	2	3	5	4	1	4	2	3	3	3	4	2	4	5	4	5	3	3	5	3	3	2	3	3	3	3	5	3
PG	C9	23	M	60	70	180	?	Y	R	3	5	5	5	4	4	1	2	4	4	3	2	2	1	3	3	2	4	4	2	2	4	5	3	4	4	3	4	1	1	4	2	4	4	2

PG	C10	18	M	60	75	182	?	Y	R	3	3	5	5	5	5	1	1	1	1	1	1	3	1	3	3	1	5	5	3	1	5	3	3	1	3	5	3	3	5	5	1	3	3	1
PG	C11	18	M	60	80	185	?	N	R	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
PG	C12	28	M	60	75	180	?	Y	L	2	3	5	5	5	5	1	1	1	2	1	2	2	2	4	3	1	3	3	4	4	4	4	4	2	4	5	5	2	1	5	1	4	5	1
PG	C13	18	M	60	59	170	?	Y	R	4	4	4	4	5	5	1	1	3	1	1	1	3	3	4	1	1	4	4	3	3	4	4	5	4	4	3	4	1	1	5	1	2	5	1
PG	C14	22	M	60	65	165	?	Y	R	3	4	5	4	4	4	2	1	3	4	4	2	2	3	3	3	2	2	3	4	4	4	4	4	3	4	4	4	4	3	4	2	3	4	2
PG	D11	21	M	60	73	175	?	Y	R	4	3	5	5	5	5	1	1	2	3	2	2	3	2	1	3	4	4	5	2	4	4	5	3	3	4	2	4	1	1	5	1	3	5	1
PG	D22	23	M	60	63	176	?	Y	R	3	3	3	3	5	4	3	2	3	2	4	2	3	3	3	3	3	4	4	4	4	4	4	4	5	4	4	4	4	3	4	4	3	4	3
PG	D33	23	M	61	64	165	?	Y	R	3	4	5	3	3	4	2	3	3	2	3	2	1	2	4	2	3	2	4	2	3	4	4	4	4	3	3	3	3	3	4	2	3	4	2
PG	D44	20	M	60	80	180	?	Y	R	4	5	5	4	5	5	1	1	3	4	4	3	1	3	3	2	1	3	1	1	3	1	5	3	4	4	4	4	3	5	3	4	4	4	2
B	A1	40	M	65	100	179	?	N	R	2	4	5	5	5	5	4	3	4	4	4	4	2	2	3	4	3	5	4	2	5	5	4	3	1	5	5	4	4	3	4	4	2	3	4

B	A2	29	M	60	72	182	420	Y	L	3	4	5	5	5	5	1	2	3	2	3	2	2	2	3	2	3	3	4	2	4	5	4	2	2	3	4	5	3	3	5	1	2	4	2
B	A3	34	M	60	75	174	314	Y	R	2	4	4	4	5	4	1	3	2	3	4	2	2	3	3	2	2	3	3	2	3	4	3	3	3	4	3	4	3	3	4	2	4	5	2
B	A4	40	M	60	71	171	354	Y	R	3	1	4	4	5	5	1	3	4	2	2	2	3	1	2	2	2	2	2	2	4	2	3	3	2	3	2	3	2	2	4	2	3	4	2
B	B1	24	F	60	62	172	276	Y	R	4	1	2	5	5	5	2	2	2	2	3	4	2	4	1	2	2	1	2	1	3	3	2	2	2	2	2	2	1	2	4	1	4	3	2
B	B2	20	M	60	65	176	483	Y	R	5	5	5	4	5	5	1	3	4	4	4	3	4	2	4	3	5	5	5	3	5	5	5	5	5	5	4	5	2	1	5	1	5	5	1
B	B3	38	M	62	81	170	307	Y	R	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
B	B4	23	M	60	60	164	397	N	R	4	5	4	3	4	5	1	5	3	3	5	5	2	5	1	1	5	4	5	1	5	4	3	3	5	3	5	5	2	1	5	1	3	5	1
B	C1	22	M	60	74	172	207	Y	R	5	5	5	4	4	5	1	3	3	3	4	3	2	4	3	3	3	3	2	3	4	3	3	3	5	3	4	5	4	3	5	2	5	5	1
B	C2	33	M	60	82	174	169	Y	R	2	5	5	5	5	5	2	3	4	5	3	5	5	5	5	5	4	4	4	2	4	4	4	5	5	5	3	4	3	3	3	2	5	5	5
B	C3	27	M	60	69	173	350	Y	R	5	3	4	4	4	5	1	3	3	3	4	2	4	2	3	3	3	4	5	2	5	4	4	4	4	4	4	4	4	2	4	2	3	4	2

B	C4	29	M	60	86	182	265	Y	R	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
B	C5	26	M	60	57	171	247	Y	R	3	5	5	5	4	5	1	1	3	4	4	4	4	4	4	4	3	5	4	5	5	5	4	3	5	5	4	5	2	2	5	1	4	5	1
B	C6	22	M	60	62	170	184	Y	R	4	5	5	4	4	5	2	5	4	4	5	4	3	3	4	3	4	4	4	4	4	4	4	4	4	4	3	4	3	3	4	2	3	5	2
B	C7	26	F	64	52	165	174	Y	R	4	3	5	5	5	5	1	2	2	5	5	4	5	5	5	4	2	4	4	2	4	4	4	4	5	5	2	5	2	1	5	1	2	5	1
B	C8	18	F	60	50	158	221	Y	R	2	2	3	5	5	5	1	3	1	3	3	1	3	1	2	1	2	3	3	1	3	3	5	3	5	3	5	5	4	3	5	3	2	5	3
B	C9	23	M	60	70	180	135	Y	R	3	5	5	5	4	4	1	3	4	4	3	1	2	2	3	3	2	3	4	4	4	4	2	4	4	4	3	5	2	2	4	2	4	4	1
B	C10	18	M	60	75	182	116	Y	R	3	3	5	5	5	5	2	1	1	1	3	1	3	1	1	1	1	5	5	3	5	5	3	1	1	3	5	5	1	1	5	1	1	1	1
B	C11	18	M	60	80	185	255	N	R	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
B	C12	28	M	60	75	180	261	Y	L	2	3	5	5	5	5	1	2	3	2	1	3	1	1	3	3	4	4	4	2	4	4	4	4	1	5	5	5	4	3	5	1	4	5	1
B	C13	18	M	60	59	170	109	Y	R	4	4	4	4	5	5	1	2	3	3	1	1	1	2	2	3	1	3	2	2	2	3	4	3	3	4	4	5	2	2	5	2	5	5	1

B	C14	22	M	60	65	165	309	Y	R	3	4	5	4	4	4	2	4	3	4	3	4	2	3	2	3	4	3	4	2	5	4	3	4	4	3	3	4	2	2	4	2	4	4	2
B	D1	21	M	60	73	175	202	Y	R	4	3	5	5	5	5	2	4	1	2	5	2	5	4	3	3	2	4	5	3	5	5	4	2	4	4	1	4	1	1	5	1	5	5	1
B	D2	23	M	60	63	176	219	Y	R	3	3	3	3	5	4	1	4	2	3	4	1	1	1	2	2	3	4	3	3	4	4	3	2	3	3	4	4	3	3	3	2	2	3	3
B	D3	23	M	61	64	165	201	Y	R	3	4	5	3	3	4	2	5	4	4	4	4	4	4	4	4	4	4	3	5	2	2	3	2	4	5	4	2	4	3	4	5	2	3	4	2
B	D4	20	M	60	80	180	205	Y	R	4	5	5	4	5	5	1	1	3	3	4	1	1	1	4	4	1	4	1	4	4	4	4	4	3	2	5	4	5	5	5	3	4	2	4	1
A	A1	40	M	65	100	179	?	N	R	2	4	5	5	5	5	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
A	A2	29	M	60	72	182	197	Y	L	3	4	5	5	5	5	1	3	4	3	3	2	3	3	1	2	3	3	4	2	4	4	4	3	4	3	4	4	2	2	4	2	2	4	2
A	A3	34	M	60	75	174	506	Y	R	2	4	4	4	5	4	3	4	3	4	3	4	3	3	3	2	4	4	4	3	4	4	4	4	4	4	4	4	4	3	3	2	3	4	3
A	A4	40	M	60	71	171	391	Y	R	3	1	4	4	5	5	1	3	4	3	2	2	3	3	3	2	2	2	2	2	4	2	2	3	3	3	3	3	2	2	4	2	3	4	2
B	B1	24	F	60	62	172	294	Y	R	4	1	2	5	5	5	1	3	2	2	2	4	2	4	1	3	2	2	2	1	3	4	2	2	2	4	2	2	2	4	4	2	4	4	2

B	B2	20	M	60	65	176	370	Y	R	5	5	5	4	5	5	1	5	5	4	5	3	4	3	4	4	4	5	5	4	5	5	5	5	5	5	5	5	5	1	1	5	1	5	5	1
B	B3	38	M	62	81	170	302	Y	R	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
B	B4	23	M	60	60	164	288	N	R	4	5	4	3	4	5	1	5	5	3	5	5	1	5	1	2	4	4	5	2	5	4	3	2	4	3	5	5	2	1	5	1	5	1	3
A	C1	22	M	60	74	172	491	Y	R	5	5	5	4	4	5	1	3	3	4	3	3	2	4	3	3	2	2	1	3	4	3	4	3	5	4	3	5	4	4	5	2	3	5	2
A	C2	33	M	60	82	174	361	Y	R	2	5	5	5	5	5	2	5	5	5	5	5	5	5	5	5	5	4	4	3	4	4	4	3	4	5	3	4	3	3	3	2	4	5	1
A	C3	27	M	60	69	173	736	Y	R	5	3	4	4	4	5	1	3	4	5	4	4	4	4	4	4	4	5	5	2	5	4	4	4	5	5	4	5	4	3	2	3	3	3	3
A	C4	29	M	60	86	182	519	Y	R	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
A	C5	26	M	60	57	171	440	Y	R	3	5	5	5	4	5	2	4	3	5	5	5	5	5	5	4	4	5	4	4	5	5	5	5	5	5	4	5	2	2	5	1	4	5	1
A	C6	22	M	60	62	170	334	Y	R	4	5	5	4	4	5	2	3	3	4	4	3	4	4	3	4	4	4	4	3	4	4	5	5	4	5	3	3	3	3	4	2	4	5	2
A	C7	26	F	64	52	165	375	Y	R	4	3	5	5	5	5	1	5	3	5	5	5	5	5	5	3	1	5	5	2	4	4	4	4	5	5	3	5	2	2	5	2	2	5	2

A	C8	18	F	60	50	158	274	Y	R	2	2	3	5	5	5	1	5	3	4	3	1	2	1	3	1	3	5	4	1	3	5	4	4	5	5	4	5	4	4	4	5	3	2	4
A	C9	23	M	60	70	180	229	Y	R	3	5	5	5	4	4	1	2	2	2	2	1	2	2	3	3	3	4	4	3	4	4	4	3	3	4	3	5	3	2	3	3	4	3	2
A	C10	18	M	60	75	182	291	Y	R	3	3	5	5	5	5	2	2	3	4	4	2	2	1	1	4	3	2	2	2	3	2	3	3	4	4	3	4	3	2	4	2	5	4	2
A	C11	18	M	60	80	185	?	N	R	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
A	C12	28	M	60	75	180	648	Y	L	2	3	5	5	5	5	1	3	1	2	2	2	1	2	2	4	2	5	5	3	4	4	4	3	2	5	5	5	4	3	4	1	4	5	1
A	C13	18	M	60	59	170	308	Y	R	4	4	4	4	5	5	1	3	3	1	1	1	3	2	2	4	1	4	2	2	2	4	4	1	3	4	4	4	4	3	4	2	3	5	2
A	C14	22	M	60	65	165	723	Y	R	3	4	5	4	4	4	3	4	4	4	4	2	3	2	3	3	2	4	4	3	3	4	4	3	4	4	3	4	4	4	3	3	4	4	3
A	D1	21	M	60	73	175	467	Y	R	4	3	5	5	5	5	1	4	2	4	4	3	3	4	3	2	2	4	5	3	4	4	4	4	4	4	1	5	3	1	4	1	4	5	2
A	D2	23	M	60	63	176	658	Y	R	3	3	3	3	5	4	1	1	2	2	3	1	2	1	2	2	2	3	2	3	3	3	3	3	3	3	4	3	3	3	3	2	2	3	2
A	D3	23	M	61	64	165	487	Y	R	3	4	5	3	3	4	3	4	4	4	4	4	3	4	4	4	4	3	4	2	2	3	4	4	4	4	3	4	4	4	4	3	3	4	3

A	D 4	2 0	M	6 0	8 0	1 8 0	7 0 8	Y	R	4	5	5	4	5	5	1	1	4	3	4	2	1	1	5	4	1	5	1	3	5	3	4	3	4	5	5	5	4	4	4	2	2	4	1

Estatísticas de amostras emparelhadas

		Média	N	Desvio Padrão	Erro padrão da média
Par 1	EffortMeanB	,20224265	20	,047171961	,010547971
	EffortMeanA	,25236245	20	,074530222	,016665464
Par 2	HRMeanB	87,340155	20	6,8487674	1,5314310
	HRMeanA	94,088987	20	10,3364600	2,3113027

Correlações de amostras emparelhadas

		N	Correlação	Sig.
Par 1	EffortMeanB & EffortMeanA	20	,504	,023
Par 2	HRMeanB & HRMeanA	20	,554	,011

Teste de amostras emparelhadas

		Diferenças emparelhadas					t	df	Sig. (2 extremidades)
		Média	Desvio Padrão	Erro padrão da média	95% Intervalo de Confiança da Diferença				
					Inferior	Superior			
Par 1	EffortMeanB - EffortMeanA	-,050119800	,065082276	,014552839	-,080579243	-,019660357	-3,444	19	,003
Par 2	HRMeanB - HRMeanA	-6,7488315	8,6819702	1,9413476	-10,8121186	-2,6855444	-3,476	19	,003

FIGURE 77- T-TEST AND PEARSON CORRELATION COEFFICIENT IN SPSS

H. CONSENT FORM

DECLARAÇÃO DE CONSENTIMENTO

(Baseada na declaração de Helsínquia)

No âmbito da realização da tese de Doutoramento do Programa Doutoral em Engenharia Informática da Faculdade de Engenharia da Universidade do Porto, intitulada **Estimating Player Performance and Adaptivity in Exergames And Location-Based Games**, realizada pelo estudante **João Tiago Pinheiro Neto Jacob**, orientada pelo Prof. António Coelho e sob a co-orientação do Prof. Rui Rodrigues, eu abaixo assinado, _____, declaro que compreendi a explicação que me foi fornecida acerca do estudo irei participar, nomeadamente o carácter voluntário dessa participação, tendo-me sido dada a oportunidade de fazer as perguntas que julguei necessárias.

Tomei conhecimento de que a informação ou explicação que me foi prestada versou os objectivos, os métodos, o eventual desconforto e a ausência de riscos para a minha saúde, e que será assegurada a máxima confidencialidade dos dados.

Explicaram-me, ainda, que poderei abandonar o estudo em qualquer momento, sem que daí advenham quaisquer desvantagens.

Por isso, consinto participar no estudo e na recolha de imagens necessárias, respondendo a todas as questões propostas.

Porto, _____ de ____ de 2016

_____

(Participante ou seu representante)