FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Mobile Exergames: Exploring Methods for Generating Challenges based on the Context of Physical Interaction

Pedro André dos Santos Oliveira

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Doctor António Fernando Vasconcelos Cunha Castro Coelho

Second Supervisor: Master João Tiago Pinheiro Neto Jacob

Third Supervisor: Master Pedro Gonçalo Ferreira Alves Nogueira

June 29, 2015

Mobile Exergames: Exploring Methods for Generating Challenges based on the Context of Physical Interaction

Pedro André dos Santos Oliveira

Mestrado Integrado em Engenharia Informática e Computação

June 29, 2015

Abstract

As mobile technology evolves, so does the mobile game industry. With smartphones almost as powerful as desktop computers or game consoles, development of mobile games increases. Inheriting the physical interaction that exergames provide, like those released for Microsoft Kinect, Nintendo Wii or PlayStation Move, mobile exergames aim to deliver physically-interactive experiences through the use of mobile devices.

The immersion in games comes from balancing the skill level at completing a predefined challenge to the difficulty of the mechanics which compose that challenge. On mobile exergames, the player's location and motion information compose the mechanics that are developed.

The primary goal of this project is to design, develop and evaluate a mobile exergames framework that can speed-up the development of new and more immersive exergame mechanics based on user context retrieved from location and motion sensors on a mobile device.

To achieve those goals, I started by performing a review on the topic of exergames, location-based games and mobile games, their design, development and problems. The result of the study provided enough information to define the core elements that should compose a mobile exergames framework. The core elements were then mapped into modules and structures and services that could feed the framework in a way that allowed developers to create mobile exergames based on new physically-interactive mechanics.

To evaluate the results of project, the framework was tested with the development of a mobile exergame. The conclusion is that the expected resulting framework provides enough flexibility to create new exergame mechanics and that it can be integrated with other mechanics, such as storytelling and location-based.

Resumo

À medida que as tecnologias móveis evoluem, a indústria de jogos móveis também evolui. Com telemóveis quase tão poderosos como os computadores ou consolas, a procura pelo desenvolvimento de jogos móveis aumenta. Herdando a interação física que os exergames oferecem, como os lançados para a Microsoft Kinect, Nintendo Wii ou PlayStation Move, os exergames móveis procuram entregar uma experiência de interação física através do uso de dispositivos móveis.

A entrega nos jogos vem do equilibrio que existe entre a abilidade de completar um desafio e a dificuldade das mecânicas que o compõem. Nos exergames móveis, é a localização e movimentação do jogador que compõem as mechanics desenvolvidas.

O objetivo principal deste projeto foi desenhar, desenvolver e avaliar uma framework para exergames móveis que permitesse desenvolver exergames novos e mais imersivos, mais depressa.

Para atingir os objetivos, foi feita uma revisão bibliográfica sobre exergames, jogos baseados na localização e jogos móveis, o seu planeamento, desenvolvimento e os problemas que acarretam. O resultado do estudo permitiu recolher informação suficiente para definir os elementos principais que deveriam compor a framework para exergames móveis. Esses elementos foram, depois, mapeados em módulos e estruturas e serviços que pudessem alimentar framework e permitessem a criação de exergames móveis com base em novas mecânicas de interação física.

Para avaliar o resultado do projeto, a framework foi testada com o desenvolvimento de um exergame móvel. A conclusão é que a framework expectada tem flexibilidade suficiente para criar novas mecânicas de exergames, possíveis de integrar com outras mecânicas, como fábulas e baseadas em localização.

Acknowledgements

First, I would like to thank my supervisors, Doctor António Coelho, Master João Jacob and Master Pedro Nogueira for all the support I was given throughout this dissertation, the inputs, the feedback and the lessons taught.

Second, to all my colleagues and friends that supported me and reminded me to never give up; to have hope in the future and to follow my dreams.

To my family that has taken care of me and supported me so far; they have seen the ups and downs and were always there; a big thank you.

Lastly, I own the biggest thank you to a handful of friends without whom I would not be here, literally: to Professor António Augusto Sousa, for being always there when I needed the most; to Maria Luís Duarte for not giving up on me, for making me realize I will always have someone that cares; to Nuno Ribeiro - who has been there since the very beginning; to Margarida Pereira, the sole reason I don't have regrets; and to João Pedro Marques, for that one miracle that literally saved my life.

Pedro Oliveira

"I dream of things the entire world cannot imagine."

Pedro Oliveira

Contents

List of Figures

LIST OF FIGURES

Abbreviations

API	Application Programming Interface
FEUP	Faculdade de Engenharia da Universidade do Porto
LIACC	Laboratório de Inteligência Artificial e Ciência de Computadores
MIEIC	Mestrado Integrado em Engenharia Informática e Computação
PRODEI	Programa Doutoral em Engenharia Informática

Chapter 1

Introduction

This chapter is an introductory chapter and presents the context and scope of this dissertation project, the goals and contribution, the methodology used, and the structure of the dissertation.

The theme of this dissertation is mobile exergames and the methods for generating challenges based on the context of physical interaction, and so, it addresses the topics of mobile exergames, mobile exergames design and development, and mobile exergame mechanics.

This dissertation is part of the Integrated Master in Informatics and Computer Engineering (MIEIC) programme from the Faculty of Engineering of the University of Porto (FEUP).

1.1 Context

Exergames are video games that require players to be physically active or exercise in order to play the game; common exergames genres are virtual sports, fitness exercise or other physically-interactive activities. So far, game controllers for exergaming range from censored pads, like Dance Dance Revolution, to motion sensor video cameras, like Microsoft Kinect [Exe11].

Mobile Exergames, by definition, are exergames that can be played on mobile devices, such as tablets, smartphones and smartwatches. Unlike home console controllers, like Microsoft Kinect, Nintendo Wii and PlayStation Move, mobile devices don't have the same capability to capture player's motion, therefore, mobile exergames, usually, depend on location and motion data from the devices' location and motion sensors, typically, GPS, accelerometer and gyroscope, which provide inaccurate data.

A major reason for the interest in exergames in the video game industry over the past decade has been health-related issues like obesity and overweight. The World Health Organization's (WHO) states that worldwide obesity has reached more than 1.9 billion adults and over 40 million children since 2013. And the main causes for obesity and overweight are: an increased intake of energy-dense foods, high in fat[WHO15]; and an increase in physical inactivity due to most time-consuming sedentary activities, like TV watching and playing video games [TKWP01].

Another major reason for researching mobile exergames is the expected profit. Accordingly to research specialist Newzoo, in 2015, global mobile game revenues will rise above console game revenues for the first time, with mobile games reaching $.30.3 billion worldwide, and console games taking $.26.4 billion, driven by the steady growth in US and European regions and emerging south-east Asian and China markets. The firm expects sales for the total worldwide gaming market to be $.91.95 billion in 2015 [Gau15, Stu15].

Mobile exergames have the potential to become a major area in the game industry, because almost everyone has a smartphone that takes it everywhere. The inclusion of 3D accelerometers provides users with an innovative game interface, which allows them to interact in novel ways with the environment. Motion sensors can detect gesture recognition through the use of both rotation and translational acceleration as long as the physical position of the phone within actual space is known a priori. In order to deduce the phone's orientation, it is possible to take into consideration the effect of gravity on the device and use it to measure the activity of a player within a game such as jumping or running [VCBE08].

1.2 Project

Although current technology allows developers to create mobile exergames based on data retrieved from position and motion sensors on mobile devices, there is no support to create physical activities based on the retrieved data, which means, all game mechanics must be implemented by hand for each game. Different games implement different mechanics; however, exergaming is based on physical activities like walking, running, jumping, spinning around or climbing stairs; and, currently, there is no support for common mobile exergame mechanics.

This dissertation tries to answer "how to support common mobile exergame mechanics development in order to create better, funnier and more immersive mobile exergames?" "Would a framework, capable of generating challenges based on the user's contextual data (motion and position), benefit the development of physically-interactive mobile exergames?" "Which mechanics should a mobile exergames framework support?"

In order to answer these questions, I believe that a framework that works as an abstract layer is necessary to develop mobile exergames more efficiently. For the framework to be beneficial, it should support easy access to the user's contextual data (position and motion) and support exergame mechanics based on that data. The mobile exergame mechanics the framework should support must be common to most or all mobile exergames, usually: stepping, walking, running, jumping, turning and spinning.

1.3 Goals

Therefore, the goal of this dissertation is to create a mobile exergames framework to support mobile exergame development and mobile exergame mechanics such as stepping, walking and running, and prove the framework allows these mechanics to be implemented on different mobile exergames with different mobile exergames requirements and designs.

1.4 Structure

This dissertation is organized in six chapters:

This chapter (1) introduced the theme and context of this dissertation, the problem it tries to solve, the perspective solution and what goals the solution aims to achieve. On chapter 2, I present a review of the state of the art of mobile exergames and some related work.

In chapter 3, I specify the framework requirements based on the literature review.

Chapter 4 describes the classes and interfaces implemented in order to develop the framework.

Chapter 5 presents GeoCatch, a mobile exergame developed with the framework. followed by the game design of the proof of concept game.

The last chapter (6), discusses the work done and evaluate the framework and its results; then, I present the conclusions extracted from this dissertation and point out the work to be done in the future.

Chapter 2

State of the Art

This chapter tries to define the state of the art on mobile exergames. It starts with a review of the literature on exergame design made in the past decade and the relation between conventional console game design and physically-interactive mobile game design.

2.1 Literature Review

In 1982, Chris Crawford composed The Art of Computer Game Design. He explained his understanding of computer games as "a fundamental part of human existence" due to its interactivity, exclusive to this media art. Within computer games, he presented four common factors to all computer games: representation ("a subjective and deliberately simplified representation of emotional reality""), interaction (the way of changing the cause and effect that defines the game), conflict (the obstacles that prevent the player from achieving its goal), and safety (the relation between psychological experiences of conflict and danger that exclude physical realization). He presented the motivation that leads players into the game (fantasy/exploration, proving oneself, social lubrication, exercise, need for acknowledgement) and declared that it's the designer's job to educate, entertain and edify the game-player. These motivations may, very well, be in the origin of the game taxonomy he proposed, with classes for action games and strategy games, and its subclasses (fighting, sports, race, adventure, and role-playing). He, then, divided the process of game designing into four phases: input/output structuring, game structuring, program structuring, and design evaluation.

In summary, a computer game is a form of entertainment where players interact with elements in order to meet a goal, overcoming challenges set in its way and winning conflicts against the computer. Following his video game analysis, he addresses "Play" and "Fun", and defines their relationship with "Game", shortening it to: "Game" is the activity that the player performs; "Play" is how the player performs it; and "Fun", which is the experience that the player derives from it [CPA+03].

2.1.1 Game Design in Exergames

In exergames, the design of games and its mechanics focuses on physical interaction. Bogost (2005) studied the history of exergaming and the rhetoric of running (games that use "button-mashing for sprinting" and adapt it to the player's feet), the rhetoric of agility (games that rely on multiple physical gestures that disrupt one another), the rhetoric of reflex (games that demand carefully timed physical responses to external stimuli), the rhetoric of training (games that encourage and reward exercise as a consistent balance between physical action and pauses), and the rhetoric of impulsion (games designed to elicit physical response based on the player's gestures) [BS05].

Consolvo et al. (2006) made a pilot study of design requirements for technologies that encourage physical activity, with a prototype composed of a pedometer and a mobile phone, and Houston, the custom software for the phone. The authors discussed four key requirements from their collected data analysis: activity accreditation (eliminate deceptive measurements – lack of sufficient information), personal awareness (provide a history of past behavior, current status and activity level performance), social influence (social pressure, social support and communication), and practical lifestyle constraints (attractiveness – hardware form factor) [CESL06].

Hoysniemi (2006) provided an understanding of methods for physically-interactive game design and evaluation, consisting of case studies that demonstrated the importance of the game control and its responsiveness, intuitiveness and physical appropriateness; and the importance of social, psychological and methodological implications. By using Dance Dance Revolution (DDR) as a base for his study, he concluded that DDR promotes face-to-face socialization and physical exercise, boosting players confidence and health/fitness [Hoy06].

Campbell, Ngo, and Fogarty (2008) reviewed game design in fitness applications and extracted a set of principles relevant to fitness games: core mechanic (the set of essential interactions – easy to learn/difficult to master); representation (aesthetics and narrative – neither too complex nor too shallow); micro goals (conflicts boosting continuous play); marginal challenge (flow and fun – appealing to replay); free play (unrestricted narrative – easing integration in everyday life); social play (internal and external relationship based on social interaction); and fair play (fairness in the rules and the core mechanic - competitiveness) [CNF08].

In 2008, Jesse Schell released The Art of Game Design: A Book of Lenses, which summarizes game design as "the act of deciding what a game should be." In his perspective, the designer's job is to create the experience, which is enabled by the game [Sch08]. He also presents his lenses as different perspectives of looking into the game design. Experience, fun, flow, needs, action, goals, rules, skill, expected value, fairness, challenge, reward and feedback are some of the lenses important for the design of exergames.

In [Hal11b], Robert Hall presented a motivation for mobile exergames and the engineering challenges inherent to them in a five phase discussion: architecture (communication problem, full distribution problem, long range play problem), design (rapid re-coherence (RRC), distributed joint state problem), coding (cross-platform/cross-compilation, hardware performance differences, sensor capability differences, operative system and language differences), requirements ((outdoor

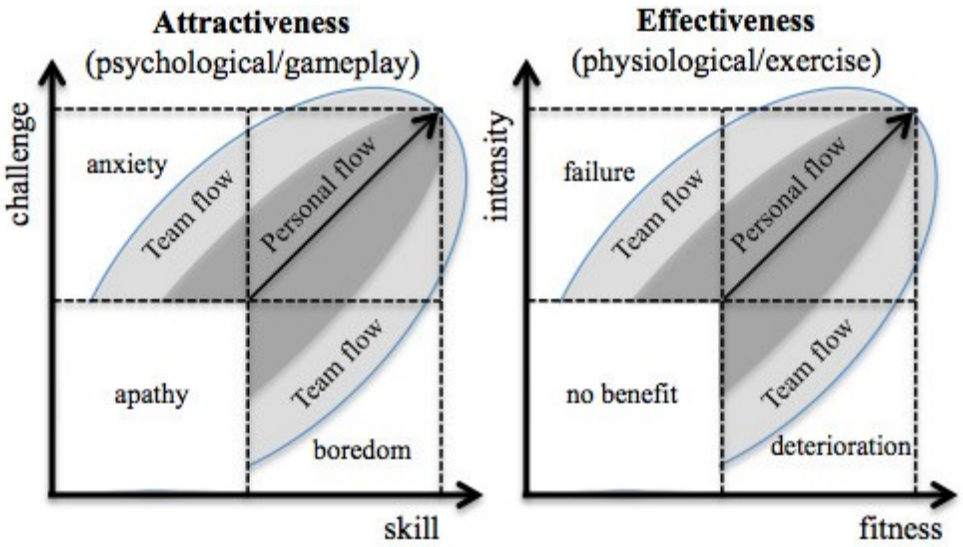

Figure 2.1: A dual flow model for exergaming [SHM07].

games) requirements restriction problem), and validation ((outdoor games) semantic validation problem, (outdoor games) pragmatic testing problem). He then presented a design for a point-and-shoot weapon that included a procedure for engineering relaxed accuracy standards and supporting different types of game experiences [Hal11a].

2.1.1.1 Sinclair's Dual Flow Model

Sinclair, Hingston et al. (2007) review of exergaming history aimed at evaluating the physical and health characteristics of exergames. Their research attempted to identify success factors in exergame design by studying commercial exergaming systems like exercise bikes, foot operated pads and motion sensors. He established two dimensions on exergames success: attractiveness and effectiveness. Attractiveness is the level of entertainment, or fun, which is affected by challenge, curiosity and fantasy; this concept is based on the "flow" model from Csikszentmihalyi (1975) which defines nine components: balance between skill and challenge; merging of action and awareness; clear goals; feedback; immersion (concentration and focus); control; safety (no self-concern or self-doubt); time detachment, autotelic experience. Effectiveness is the correlation between physical fitness and the exercise intensity level; the balance between skill and challenge. The authors proposed the "dual flow" model, which includes the two dimensions of the exercise (see Figure 2.1), and whose key requirements are the balance between skill and challenge, and between fitness and intensity. They concluded that making an exergame attractive to players and, at the same time, effective as an exercise is an important success factor [SHM07].

Later, he validated his model with the development of an exergame based on four modules (see Figure 2.2): User Identification and Exercise Planning Module is used for detail and exercise level input; the Game Play Engine is the core of the video game and the gameplay manager; the Exergame Control Logic Model features the logic algorithms that apply the dual flow model;

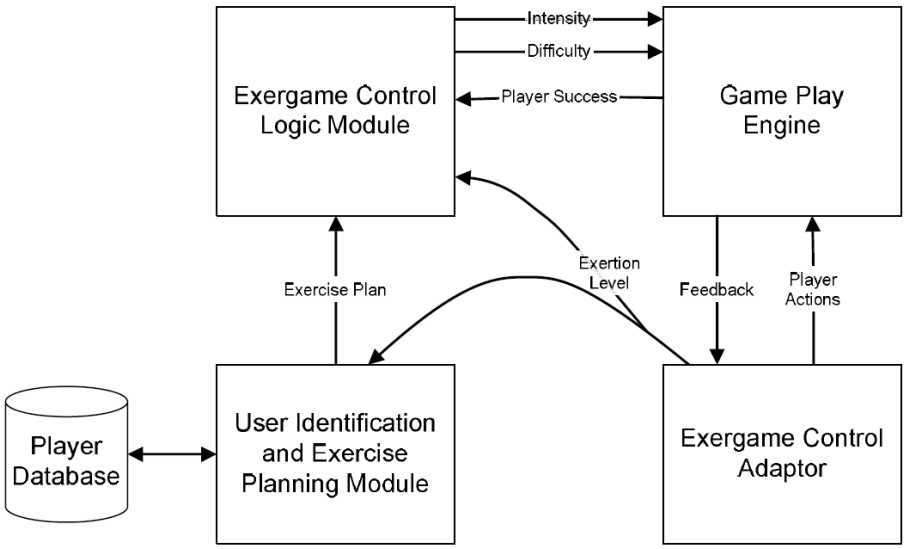

Figure 2.2: An Exergame Framework [SHM09].

and the Exergame Control Adapter reads player motion and button presses, and gives direct and indirect feedback by changing intensity and challenge parameters. The game was tested on a modified exercise bike with an heart rate sensor to measure current heart rate. The outcomes proved that the exercise was effective and engaging as workout [SHM09].

Kiili and Merilampi (2010) developed four exergames in a study of mobile phones as motion-based controllers for physical interactive games, based on an extended version of the dual flow model. The games used mobile phones to interact with a big screen using an accelerometer sensor. The games were tested by focus groups in order to gather data and the results indicated that these games were engaging, intensive, competitive and rewarding. The main issues were: fairness (regarding skill level); challenge-skill balance (from their extended dual flow model); playing time and its relation with performance and intensity; teamwork; rewards and punishment. The results indicated that the extended dual flow model can be used to design engaging exergames with motion-based controllers. They further developed two exergames to motivate children and adolescents to exercise more, thus validating the use of their dual flow model extension for suitable game design in school environment [KM10, KMK11].

Korn et al. (2012) implemented a context-sensitive user-centered scalability (CSUCS) model into an exergame and an assistive system for the elderly that used motion detection and gamification for real-time analysis. The authors stated that games and assistive systems should scale to user's abilities and dispositions, and make changes based on performance and skill by means of gamification. The authors used an Human Activity Assistive Technology Model based on the basic Human Performance Model from Bailey (1989) to describe four basic components: activity, human, assistive technology and context, which were used together with the "flow" model to create an exergame for the elderly and measure their movements and performance [KBS+12].

2.1.2 Mueller's Exertion Framework

Mueller, Gibbs et al. (2008) presented an investigation towards the taxonomy of exertion games with a focus on social aspects. Their research led to the creation of a theoretical framework which was presented later on in [MGV08, MAVG09]. Their framework was designed to encompass meaning among exertion, sociality and engagement. They developed an exergame, Remote Impact, which made use of two sensitive padded playing areas, on which two shadows were projected (the player and its opponent). Remote Impact's gameplay allows both players to execute impacts on each ocher's shadow and earn points based on intensity. Their framework was validated based on the exertion, social and engagement level measured. Mueller also presented a qualitative analysis of player data from "Table Tennis for Three", a mediated exergame that contributed to a better understand of social exergames design and the notion of space, parallel and non-parallel activities in exertion games [MGV10]. The authors also found that exergames require an understanding of the physical body and bodily experience, along with the technological challenges of digital motion capture and interpretation.

Mueller et al. (2010) researched the relation between social and exertion aspects by presenting a qualitative study of "Jogging over a Distance" that illustrated the capability using heart rate data and spatialized sound to create an equitable, balanced experience between joggers of different fitness levels who are geographically distributed. From their study, derive four dimensions that describe a design space for balancing exertion experiences: Measurement, Adjustment, Presentation and Control; they also presented six design tactics for creating balanced exertion experiences described by those dimensions. and created a conceptual theme about the relationship between exertion and social aspects of games [MVG+10, MVG+12]. They introduced the Exertion Framework to consider exergame formative design and summative analysis in a systematic way (see Figure 2.3). Their framework supported four perspectives of the body (the responding body, the moving body, the sensing body and the relating body) and three perspectives on gaming (rules, play and context); and it served as a language for future research on embodied exergame systems with tangible interfaces [MEV+11].

2.2 Exergames Applications

Exergames for health and sports led to many commercial games to be released. One of the most representative systems of exergames is the Nintendo Wii and its Wiimote. The Wii and its health games (Wii Fit) and its sports games (Wii Sports) have contributed to the active lifestyle that motivates the development of exergames, and so, have encourage research on its use for other issues. Voida and Greenberg (2009) presented a qualitative study on the socialization element of console gaming through the Nintendo Wii. Their analysis revealed group gameplay enables social interaction, and therefore, motivates social gaming, independent of age, gender, levels of expertise and preferences for gameplay. They concluded that console games can be designed as a computational meeting place that promotes social interaction [VG09].

	The Responding Body	*The Moving Body*	*The Sensing Body*	*The Relating Body*
Rules	*Uncertainty of exertion* *Awareness of Exertion*			
Play	*Expression of exertion* *Rhythm of Exertion*			
Context	*Risk of exertion* *Understanding of Exertion*			

Figure 2.3: The Exertion Framework [MEV+11].

More studies revealed that social houses can benefit from the use of exergames through the use of consoles like Wii which stimulates physical and social activity in these settings [HFA+10]. Not only that but, health issues like lack of balance and motor skills deterioration can be helped through the use of exergames like the Wii Fit [APP+11]. Other studies concluded that console like Nintendo's Wii can provide support for integrating self-efficacy theory into exergames as a mechanism to encourage older adults to engage physical activity [CSW+13].

Stach (2010) synthesized the literature of multiplayer exergames into four aspects: group formation, play styles, differences in skills and abilities, and quality of exercise. He considered important for the design of exergames to understand human behavior and reviewed the Theory of Planned Behavior, the Transtheoretical Model of Behavior Change and self-efficacy in order to better analyze the influence of the four aspects. He concluded that group-based physical activity has been proven to affect motivation; the feasibility of supporting play over time, space and mode of interaction also affected motivation; one of the most important factors is the quality of exercise – duration, frequency and necessary skill/fitness level [Sta10].

Staiano and Calvert (2011) reviewed measurement of physical health in exergames studies by analyzing the results from built-in exergame measurements and concluded that exergames are promising research tools to measure caloric expenditure, heart rate, and health behaviors as long as caloric expenditure and heart rate algorithms to correlate internal and external measures are developed [SC11a, SC11b]. They examined a 20-week exergame intervention on a public high school using the Nintendo Wii Active game to measure weight loss, calories expenditure, peer support, self-efficacy and self-esteem. They claimed exergames to be an effective technological tool for weight loss among youth [SAC13].

Gobël et al. (2010) described a set of adapted exergames to motivate workout by combining game technology and game-based methods and concepts, mechanisms for personalization and adaptation, and sensor technology, in order to integrate the user's vital state and parameters into the game [GHW+10]. Their framework was divided in three modules (Content Extraction Module, Context Extraction Module and Dynamic Service Adaptation Module), which extracted data from body sensored networks (accelerometers) and multimedia services (weather and calendar services), and mapped them into subsets for serious games depending on exertion level and user context [HESGS11]. They, then, presented BalanceFit, an adaptive prototype game for fall prevention based on the adaptation and exergame analysis framework, StoryTecRT, and Nintendo's Balance Board. Qualitative analysis cared for visual style and sensibility which was made adjustable and succeeded in being playable by young fit people as well as people with impairments [HGS13].

2.2.1 Summary

In this section, we see that mobile exergames can have significant impact on physical and social activity, can help to prevent balance and motor skills deterioration and improve the overall lifestyle of people. In order to motivate people to play mobile exergames, its necessary to design the game in a way that supports balance between skill and challenge, and balance between fitness and intensity; it's also necessary to give players feedback and accreditation for the activities the play; and finally, it's necessary to keep the game fun and allow some social influence through collaboration and competition.

However, it has been pointed out issues in the design of mobile exergames, like how to balance challenge and skill or fitness and intensity, when to reward or punish the player; and how to create teamwork and competitiveness in multiplayer games. Also, development of mobile exergames doesn't come without issues: the architecture of the game, the communication and range; the aforementioned design; and the coding (cross-platform vs single-platform, programming languages, content and context extraction from the sensors and the generation of challenges).

All this issues point out to a need for a framework that supports mobile exergame design and development; a framework that can retrieve information about the user's context and generate challenges based on it; a framework that measures the skill/fitness level of the player and adapt the challenges without losing the immersion and entertainment levels of the game; a framework that rewards the player when they achieve success and motivates them when they fail; and finally, a framework that promotes social interaction: collaboration or competition.

2.3 Technology Review

In this section, I present some of the current technologies for mobile exergame development; based on the survey, two technologies were tested in order to select the development platform.

2.3.1 Survey

When it comes to mobile games, smartphones are the prominent device to play; nowadays smartphones usually come in one of three flavors: Android devices, iOS devices or Windows Phone devices. Matt Haggerty (2012) wrote an article on "The State of Mobile Game Development" which featured some of the most popular game engine/frameworks available on the market [FHH12].

2.3.1.1 Box2D

Box2D is an open source C++ engine for simulating rigid bodies in 2D [Box15]. Some of Box2D features relate to collisions (collision detection, convex polygons and circles, collision groups and categories) and physics (contact, friction, and restitution, reaction forces and momentum decoupled position correction). It has a testbed in OpenGL with Freeglut, a graphical user interface with GLUI, a test framework and builds with CMake.

2.3.1.2 Cocos2D-x

Cocos2d-X is a suite of open-source, cross-platform, game-development tools, written in C++. It has modules to support Data Structures, Debugging, File I/O, User Input Handling, Memory Management, Network, User Interface, Scene Graph Scheduler, Scripting, Threading, Game Controller and Assets Manager. It also features Effects, TileMaps, Particles, a Physics Engine and Third-Party SDK Integration (Android, iOS and Windows Phone). Additionally, there is a Cocos Studio and Cocos Code IDE to fully support the development of games [Coc15].

2.3.1.3 BatteryTech SDK and Engine

Battery Powered Games maintain and develop BatteryTech SDK - a c++ platform which provides connectivity between Android, iOS, Windows, OSX and BlackBerry 10 operative systems; and BatteryTech Engine - a high-level 2D and 3D game engine that runs on Lua and supports graphics and 3D models display and loading, music and sound play, lights and shadows, 2D physics (using Box2D library), custom game behavior scripting, and input via multitouch and accelerometer [Gam15b].

2.3.1.4 libgdx

LibGDX is an open-source java game development framework that provides a unified API for multiple platforms. The framework provides an environment for rapid prototyping and fast iterations.

Some of its features are: cross-platform (Windows, Linux, Mac OS X, Android (2.2+), BlackBerry, iOS, Java Applet and JavaScript/WebGL); audio (streaming music and sound effects for WAV, MP3 and OGG, and direct access to audio device for PCM sample playback and recording); input handling (mouse and touch-screen, keyboard, accelerometer and compass, and gesture detecting); math &. physics (bounding shapes and volumes, picking and cullum, splines, integration with Box2D physics); file I/O &. storage; and graphics (particle system, textures, meshes, shaders; 2D scene-graph API, orthographic and perspective camera, and others). LibGDX integrates with Eclipse, Intellij IDEA and NetBeans IDEs [Zec15].

2.3.1.5 Marmalade

Marmalade Technologies Ltd has developed Marmalade, a cross-platform native development environment, allowing code in C++ with full support for standard libraries and third-party extensions. It features development for multiple operating systems and devices (iOS, Android, BlackBerry, Tizen, Windows, Mac and others) using a single cross platform C/C++ code base; access to many native platform features such as native UI, in-app purchasing, video playback, and compass; access to many out of the box features such as analytics, ads, and SQLite; development and testing on desktop using the Marmalade simulator; development and deployment to devices, directly from Windows or Mac; development, testing and signing iOS apps on Windows without a Mac; generation of native code; and extend Marmalade with the EDK, providing easy access to third party API's [Tec15a].

2.3.1.6 GameMaker

GameMaker: Studio is a game engine developed by YoYo Games Ltd.

"GameMaker: Studio caters to entry-level novices and seasoned game development professionals equally, allowing them to create cross-platform games in record time and at a fraction of the cost of conventional tools! In addition to making game development 80 percent faster than coding for native languages, developers can create fully functional prototypes in just a few hours, and a full game in just a matter of weeks" [Gam15a].

It has a standard Free version, a Professional version for $. 89.99 and a Master Collection for $. 559.99. The free version requires registration and a GameMarker splash screen. The engine features Unlimited Resources, a GameMaker: Player Export and a Windows Desktop Module. The professional version doesn't require registration nor a GameMaker splash screen, and features Unlimited Resources, Texture Management, Multiple Configurations, Team Features, Developer Services Prtal (DSP), Mobile Testing (Android), the Windows Desktop Module + YoYo Compiler + Export, Windows App Module + YoYo Compiler + Export. The Master Collection adds production for all console, mobile, desktop and web platforms.

The studio has a fully integrated developed environment (IDE) that allows for no-programming development but also supports scripting, through its Game Maker Language (GML). It allows to

import images, animations, audio and fonts, and provides tethered or wireless testing. It also integrates with source control software as SVN, CVS or Git.

2.3.1.7 Unity

Unity is a cross-platform game engine developed by Unity Technologies. *"Unity is a flexible and powerful development platform for creating multiplatform 3D and 2D games and interactive experiences. It's a complete ecosystem for anyone who aims to build a business on creating high-end content and connecting to their most loyal and enthusiastic players and customers"* [Tec15b]. The engine comes in one of three editions: Personal Edition for free; Professional Edition from $.75/MONTH; and Enterprise Solutions based on tailored setup.

The Unity Engine is supported by the Unity Editor, which features: Animation (retargetable animations, full control of animation weights at runtime, event calling from within the animation playback, sophisticated state machine hierarchies and transitions and blend shapes for facial animations); Graphics (enlighten-powered real-time global illumination, physically-based shading, reflection probes, curve and gradient-driven modular particle system, intuitive UI tools); Optimization (advanced memory profiling, umbra-powered occlusion culling, asset bundling, level of detail support, build size scripting, multi-threaded job system); Audio (real-time mixing and mastering, hierarchies of mixers, snapshots and predefined effects); 2D and 3D Physics (Box2D with a comprehensive range of effectors, joints and colliders; NIVIDA® PhysX® 3.3); Scripting (C#., JavaScript or Boo; AI features with advanced automated path finding and Navigation Meshes); and Version Control (full integration support for Perforce and Plastic SCM).

Unity also provides an Asset Store with thousands of ready-made free or for purchase assets and production tools, editor extensions, plug-ins, environments and models, created by Unity Technologies and third-party developers. Additionally, Unity provides integration with its Ads, analytics, cloud and multiplayer services.

2.3.1.8 Android

"Android is the operating system that powers more than one billion smartphones and tablets. Since these devices make our lives so sweet, each Android version is named after a dessert. Whether it's getting directions or even slicing virtual fruit, each Android release makes something new possible" [And15a], and it is currently developed by Google and, as of 2015, is the largest installed base of all operating systems [Gar15]. Android, currently, powers phones, tablets, watches, TV and car, making it the most versatile and ubiquitous operative system in the market.

Android provides an application framework to build innovative apps and games for mobile devices in a Java language environment, using Android's various APIs [And15c]. Android APIs allow developers to build apps with multimedia (audio playback, capturing photos, printing content), graphics and animation (displaying bitmaps efficiently, displaying graphics with OpenGL ES, animating views with scenes and transitions), connectivity and cloud services (connecting devices wirelessly, performing network operations, transferring data without draining the battery,

syncing to the cloud and resolving cloud save conflicts, transferring data using sync adapters and transmitting network data using Volley[1]); location and maps; and user info &. sign-in [And15b].

Android also integrates with Google Play services APK, which contains the individual Google services and runs as a background service in the Android OS. Google Play services features a Location API (high-accuracy location reporting, geofencing, and activity recognition) [Goo15], Maps API (3D rendering, vector tiles, indoor floor plans, draw on the Map, gesture control and location services) [Dev15c], Fitness API (access to raw sensor data, automated storage of fitness data, fitness history) [Dev15a], and Games API (saved games, events and quests, player analytics, real-time and turn-based multiplayer, achievements and leaderboards) [Dev15b].

2.4 Summary

In this chapter, I provided a literature review on mobile exergames; we concluded there are challenges with mobile exergame development based on the difficulties to calculate and balance skill and fitness levels, and adapt the intensity and challenge of gameplay (physical activities); I also concluded there is a need for exergames to be open to collaboration and competition for the sake of social interaction; and finally, I concluded that exergames, even though must have exercise as its core mechanics, must also be fun and immersive, so players keep their motivation to play.

On the technology review, I checked a few game engines and game development frameworks that could be used to develop mobile exergames. Box2D provides a great physics engine that is base for many other engines and frameworks. Cocos2d-X, BatteryTech and LibGDX provide cross-platform support and feature accelerometer input; however, they lack the seamless integration with location and map services. Marmalade generates native code through its Java Native Interface plugin and native iOS framework wrapper libraries; therefore, it has access to all the features in native platforms (Android, iOS and Windows Phone). GameMaker's Free Edition does not export to mobile platforms. Unity Engine framework has access to mobile devices location and accelerometer data. Android is a native mobile device operative system features multimedia, graphics, animation, connectivity, cloud services, location and motion sensors data reading.

[1] Volley is an HTTP library for networking between devices

Chapter 3

Requirements Specification

The process of developing a mobile exergames framework integrates different steps. The objective of this chapter is to introduce the framework requirements necessary to the design and development of the framework.

"Requirements are statements that identify a product or process operational, functional, or design characteristic or constraint, which is unambiguous, testable or measurable, and necessary for product or process acceptability (by consumers or internal quality assurance guidelines)" [WSU06].

The next section presents an overview of the framework. Section 3.3 describes the user requirements elicited through the analysis of the state of the art. Then, I describe the requirements as modules within the framework (see section 3.4).

3.1 Overview

The goal of the mobile exergames framework is to facilitate the development of mobile exergames and the mobile exergame mechanics. In order to achieve that, it is necessary to create an interface for creating mechanics based on data from location and motion sensors. Those mechanics can, then, be implemented on different mobile exergames. The overview of the domain is illustrated in Figure 3.1.

This section provides an description of the main domain objects presented in the domain model:

- **Accelerometer** — The accelerometer is a hardware sensor that measures the acceleration forces applied to a mobile device on all three physical axes, including the gravitational force.

- **Developer** — The developer is the framework's target audience, whose goal is to develop mobile exergames based on mobile exergames mechanics.

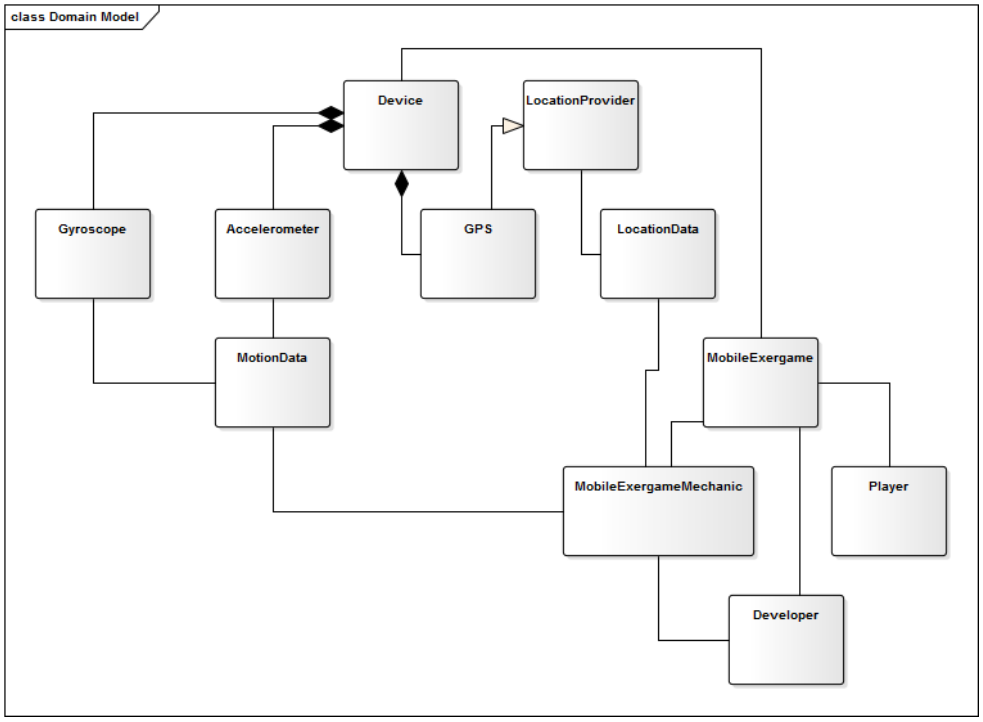

Figure 3.1: The Domain Model.

- **Device** — The device is the hardware component that serves as the console and controller for mobile exergames.
- **GPS** — The GPS is a satellite navigation system that provides location and time information to a mobile device.
- **Gyroscope** — The gyroscope is a hardware sensor that measures the rotational force applied to a mobile device on all three physical axes.
- **LocationData** — LocationData is the latitude and longitude retrieved from a location provider system.
- **LocationProvider** — LocationProvider is a service or system that provides information regarding the device's latitude and longitude.
- **Mobile Exergame** — Mobile Exergame is the application that will implement mobile exergames mechanics and runs on a mobile device.
- **Mobile Exergames Mechanics** — These are the mechanics that will extract location and motion sensor data from a mobile device and accept input from a player's physical interaction.
- **MotionData** — MotionData is the computed data that measures the player's motion regarding the accelerometer and gyroscope three physical axes.

- **Player** — The player is the end-user of all mobile exergames developed with the mobile exergames framework.

3.2 User Characteristics

The framework's end-users are game developers whose aim is to develop mobile exergames with improved mobile exergames mechanics. Therefore, the users require to be developers with knowledge on mobile development, game development and exergame design.

3.3 User Requirements

User Requirements are the requirements that must implemented, described in natural language. They represent the needs of the framework's users and how they are going to interact with the framework. In this section, I present a list of the user requirements as derived from the literature review and related work.

3.3.1 Profile

The framework shall retrieve user information regarding his fitness and skill level, past behavior, current status and activity level performances as Consolvo (2006) proposed [CESL06].

3.3.2 Context

The framework shall define the user's context and availability based on extracted data from sensors and multimedia services (weather and calendar) as proposed in Hardy (2013) [HGS13].

3.3.3 Mechanics

The framework shall define mechanics for developers to create challenges based on running, physical gestures, physical response and training [BS05].

3.3.4 Dynamics

The framework shall allow gameplay to be based on internal or external social interaction, or free to play [CNF08].

3.3.5 Presentation

The framework shall be open for narrative development, allowing developers to create campaigns, side quests and challenges based on the core mechanics [CNF08].

3.3.6 Evaluation

The framework shall allow games to evaluate physical appropriateness , skill and intensity levels based on the user's profile and performance levels as seen in [KM10, KMK11].

3.3.7 Flow

The framework shall define the logic algorithms that apply the dual flow model by changing the intensity and difficult of each challenge based on the user's profile, context and evaluation [SHM09].

3.3.8 Accreditation

The framework shall define player's accreditation through a rewards and punishment system based on player's evaluation [CESL06].

3.4 Framework Requirements

Requirements can be functional or non-functional; functional requirements are statements of services the framework should provide, how the framework should react to particular inputs, and how the framework should behave in particular situations; non-functional requirements are constraints on the services or functions offered by the framework. In this section, I present a more detailed description of the framework's requirements.

3.4.1 Profile Requirements

3.4.1.1 Fitness Information Retrieval

The framework shall create, update, retrieve and delete user's fitness information. Fitness information must store player's age, height, weight and sex.

3.4.1.2 Skill Information Retrieval

The framework shall create, update, retrieve and delete user's skill level. Skill level must be defined for each physical activity (i.e.: running, cycling).

3.4.2 Context Requirements

3.4.2.1 Location Information Retrieval

The framework shall extract user's location through mobile devices' location services (i.e.: GPS, Wi-Fi).

3.4.2.2 Motion Information Retrieval

The framework shall extract user's motion information through the mobile device's motion sensors (i.e.: accelerometer, gyroscope).

3.4.2.3 Agenda Information Retrieval

The framework shall extract user's agenda information through a user-defined web service (i.e.: Google Calendar).

3.4.2.4 Weather Information Retrieval

The framework shall extract weather information through available web services (i.e.: Weather Channel, Yahoo Weather).

3.4.3 Mechanics Requirements

3.4.3.1 Running Activities

The framework shall support running challenges based on user contextual information.

3.4.3.2 Cycling Activities

The framework shall support cycling challenges based on user contextual information.

3.4.4 Dynamics Requirements

3.4.4.1 Collaboration

The framework shall support challenges to be played in collaboration with other players. Challenges in collaboration may be having two players reach two locations at the same time (i.e.: running to two different spots).

3.4.4.2 Competition

The framework shall support challenges between players. Challenges between players may be running further distances, or same distances in shorter time.

3.4.4.3 Free Play

The framework shall allow players to take challenges without following a single-player mode (i.e.: campaigns, story mode).

3.4.5 Presentation Requirements

3.4.5.1 Single Player

The framework shall support single player mode (i.e.: campaigns, story mode).

3.4.6 Evaluation Requirements

3.4.6.1 Fitness Evaluation

The framework shall analyze and rate a player's performance based on the challenge's result and fitness profile.

3.4.6.2 Flow Requirements

3.4.6.3 Flow Control

The framework shall adapt challenges' intensity and difficult level based on the player's last evaluation and current physical condition (i.e.: fatigue level).

3.4.6.4 Accreditation

3.4.6.5 Achievements

The framework shall reward the player's performance based on the challenge evaluation or other criteria. Other criteria must be defined for each challenge.

3.5 Sumary

In this chapter, I presented the requirements the framework shall support as an overview and decomposed structure.

Chapter 4

Design and Implementation

In this chapter, I explain the contributions that external frameworks provide and classes that are adapted and used to develop a mobile exergames framework, namely, the Google Play Services and the Android API. Then, I describe the classes and methods implemented and how they behave regarding the development of mobile exergames and physically-interactive mechanics.

4.1 Overview

In this section, I present an overview of the mobile exergames framework illustrated with a class model diagram that decomposes the framework in separated modules.

Figure 4.1, illustrates how the mobile exergames framework integrates with the Android API and Google Play Services. A Google Play Services more detailed specification can be consulted on Appendix A.

4.2 Android API

The Android API is enormous and contains libraries that perform all kinds of operations. The main classes used in the development of the mobile exergames framework are:

4.2.1 android.hardware

The hardware module of the Android API provides support for hardware features, such as camera and sensors. The main classes and interfaces that integrate the framework aim to get data from the device's sensors or notified the user whenever some sensor broadcasts a triggered event.

Most Android devices have built-in sensors to measure motion, orientation and environmental conditions. One goal of the framework is to track readings from the accelerometer and gyroscope sensors, and infer complex user gestures and motions (tilt, shake, rotation or swing). These gestures and motions are the base for the creation of exergame mechanics.

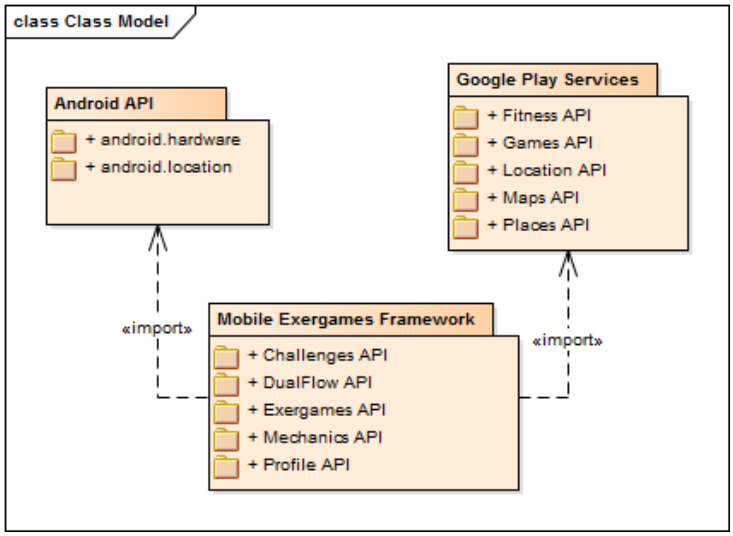

Figure 4.1: Mobile Exergames Framework Class Model

4.2.2 android.location

These Android's classes provide an interface to access the device's location services, GPS services and geocoding services. However, this API has been integrated with the Google Location Services API, which offers higher accuracy, low-power geofencing and other useful location-based services.

4.3 Google Play Services

To start using Google Play Services, it is necessary to setup the development environment and configure the app in Google Play Developer Console. To do that, it is necessary to access the Google Play Developer Console web page, create a new app for the project (game), generate an OAuth 2.0 client ID for the app, and configure the required/following APIs.

4.3.1 Google Play Games Services for Android

The Play Games Services contains classes and methods for creating games for Google Play. The Games API provides classes and interfaces for Players, Games, Quests, Events, Achievements, Leaderboards and Saved Game data. It's the base of support for the exergames API and challenges API created in the mobile exergames framework.

4.3.2 Google Fit APIs for Android

The Google Fit APIs for Android provide classes and methods to access the device's motion sensors and offer physical activity functionality. It also provides access to historical data.

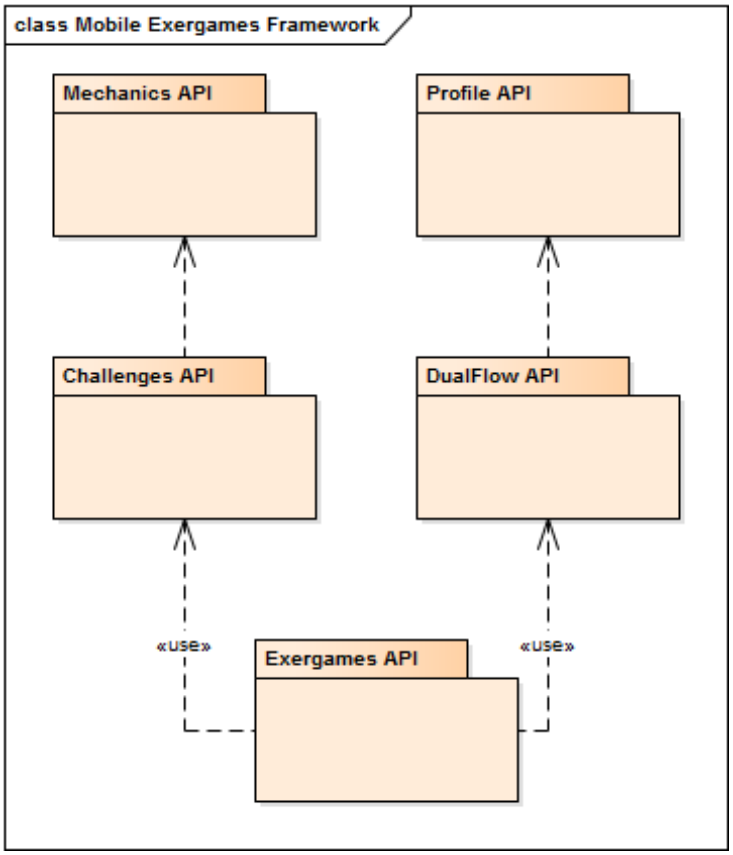

Figure 4.2: Mobile Exergames Framework Components

4.3.3 Google Maps Android API

The Google Maps Android API provides a Google Map to visually display the player's location and any other elements that may become part of any mobile exergame; it allows to place markers and draw circles and polygons which are useful to provide boundary information to the user. It also powers the underlying location API that serves as the base for location-based games.

4.3.4 Google Play services location APIs

The Google Play services location APIs provide features to bring location awareness to applications; features include location tracking, geofencing, and activity recognition. The FusedLocationProviderApi brings methods for getting location updates based on accuracy and energy expenditure criteria. The GeofencingApi adds and removes geofences and triggers an alert when there's a geofence transition. The ActivityRecognitionApi allows registering for updates which yield detected activities by reading short bursts of sensor data; it can detect is the user is on foot, in a car, on a bicycle or still.

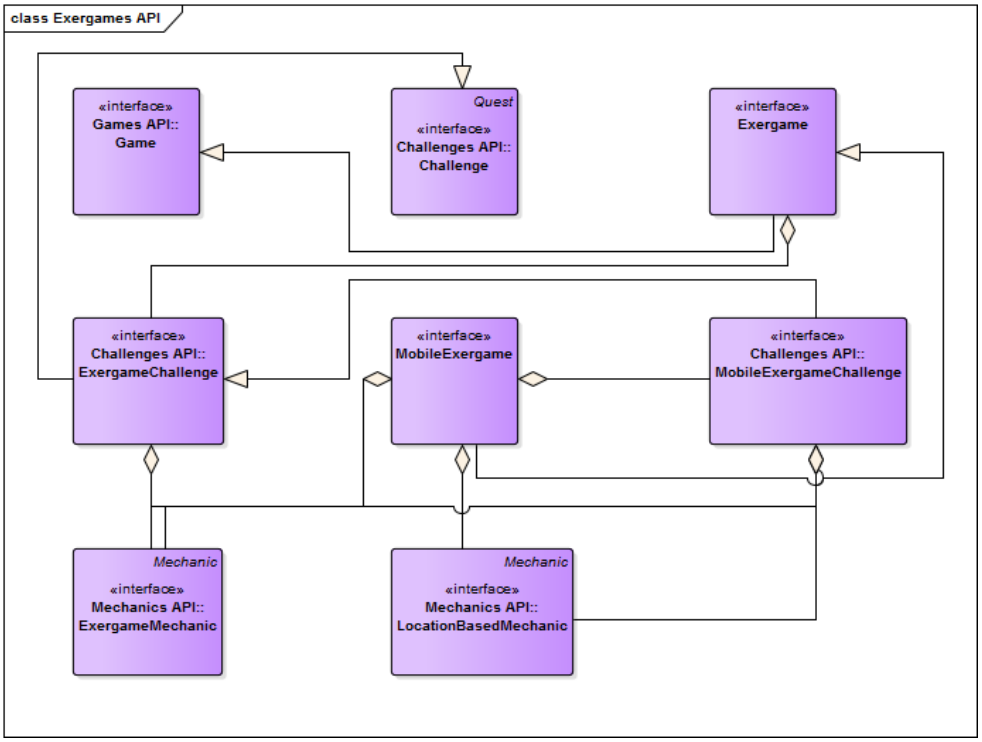

Figure 4.3: Exergames API.

4.3.5 Google Places API for Android

The Google Places API for Android helps to develop location-aware applications that respond contextually to the local businesses and other places. It contains the GeoDataApi which accesses the Google's database of local place and businesses information, and the PlaceDetectionApi which returns an estimate of the place where the device is currently known to be located (i.e.: park).

4.4 Mobile Exergames Framework

In this section I describe the mobile exergames framework design and how each module contributes to the framework.

4.4.1 Exergames API

The exergames API provide classes and interfaces to create exergames and mobile exergames. It extends and implements the classes and interfaces from Google Play Games Services and it provides the base for developing mobile exergames.

4.4.1.1 Exergame

The Exergame interface is a data interface for retrieving exergames information. It declares methods to retrieve game developer, game support, multiplayer modes, achievement and leaderboard

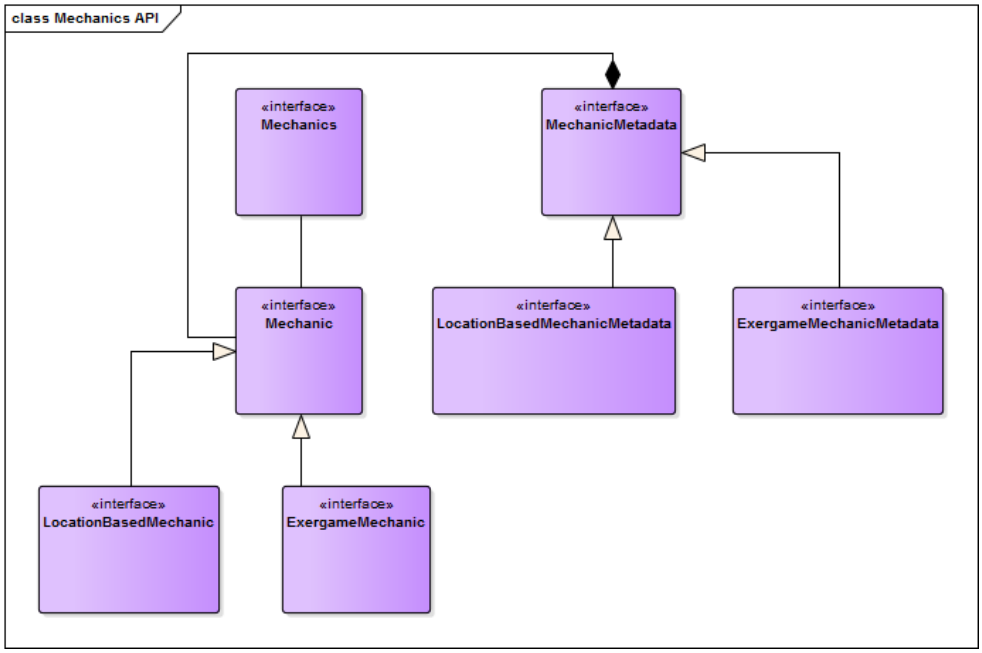

Figure 4.4: Mechanics API.

information. Exergames can contain exergame challenges based on exergame mechanics.

4.4.1.2 MobileExergame

The MobileExergame interface is a data interface for retrieving mobile exergames information. It declares methods to retrieve game developer, game support, multiplayer modes, achievement and leaderboard information. Mobile exergames can contain mobile exergame challenges based on mobile exergame mechanics.

4.4.2 Mechanics API

The mechanics API provide classes and interfaces to create mechanics for mobile exergames. Mechanics are activities required to interact with the game as a form of player's input. This means it can be running, walking, jumping, spinning or any other form of motion that can be detected.

4.4.2.1 Mechanic

The Mechanic interface is a data interface for retrieving mechanics information. It declares methods to retrieve the mechanics' ID, description, name, and user interactions (i.e.: gestures, typing) required to validate the input.

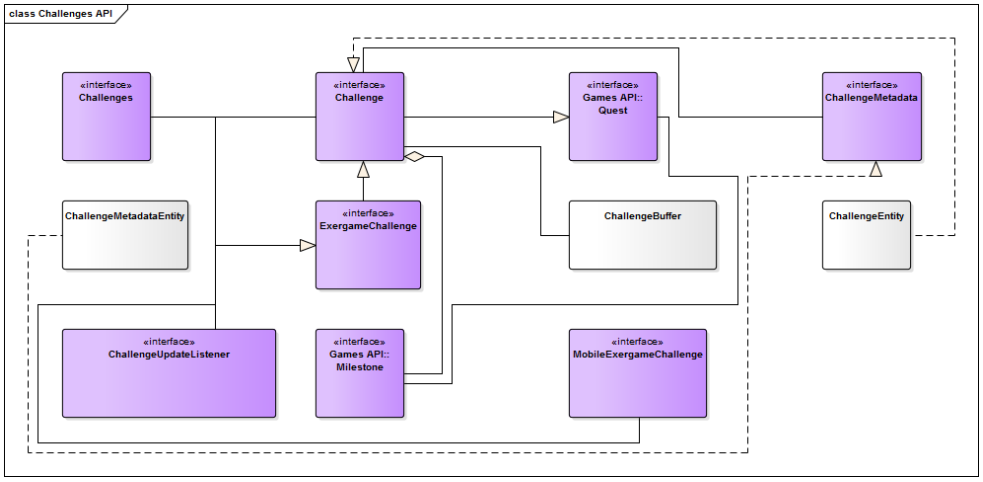

Figure 4.5: Challenges API.

4.4.2.2 ExergameMechanic

The ExergameMechanic interface is a data interface for retrieving exergame mechanics information. It declares methods to retrieve mechanics' ID, description, name, and user physical interactions (i.e.: running mechanics, jumping mechanics) inferred by motion and activity recognition data.

4.4.2.3 LocationBasedMechanic

The LocationBasedMechanic interface is a data interface for retrieving location-based mechanics information. It declares methods to retrieve mechanics' ID, description, name, and user's contextual location information (i.e.: geographical displacement, geofencing transitions).

4.4.3 Challenges API

The challenges API provides classes and interfaces to integrate challenges on a mobile exergame.

4.4.3.1 Challenge

The Challenge interface is a data interface for retrieving challenges information. It declares methods to retrieve challenge's ID, name, description, state, accepted and completed timestamps, and its game information. Challenges implement common mechanics (i.e.: gestures, typing).

4.4.3.2 ExergameChallenge

The ExergameChallenge interface is a data interface for retrieving physically-interactive challenges information. It declares methods for retrieving challenge's ID, name, description, state, accepted and completed timestamps, and its exergame information. An exergame challenge implements exergame mechanics.

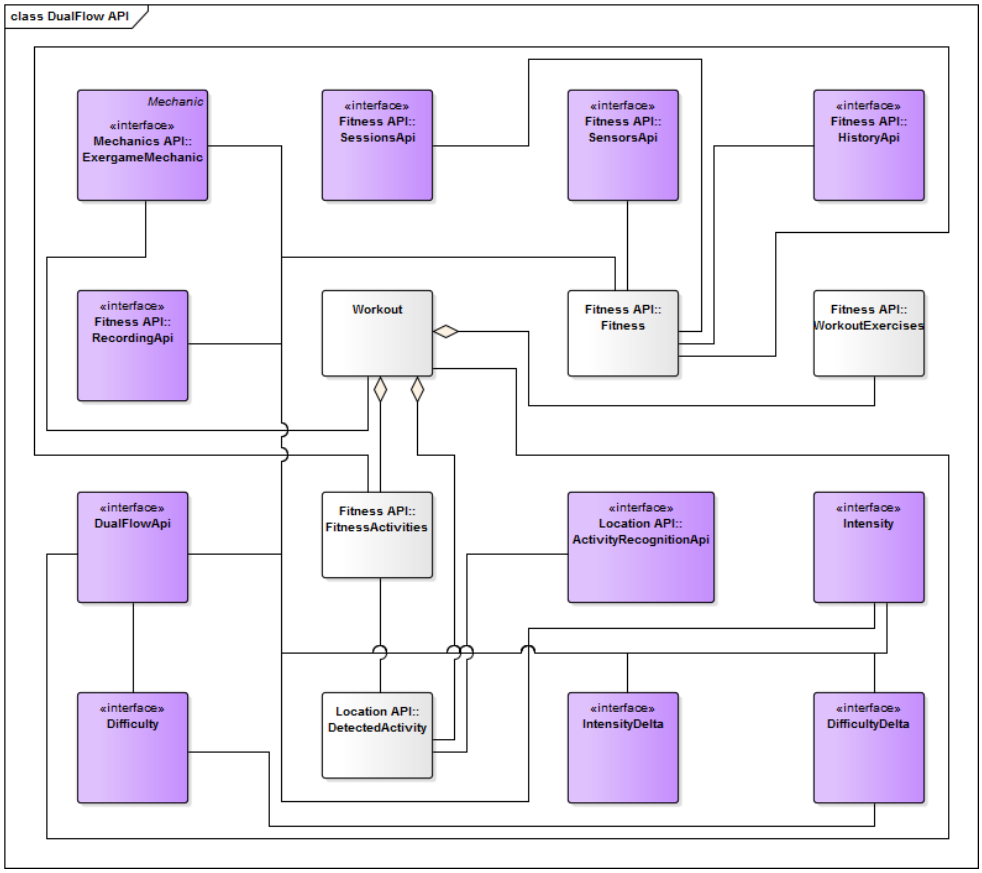

Figure 4.6: Dual Flow API.

4.4.3.3 MobileExergameChallenge

The MobileExergameChallenge interface is a data interface for retrieving location-based physically-interactive challenges information. It declares methods for retrieving challenge's ID, name, description, state, accepted and completed timestamps, and its mobile exergame information. A mobile exergame challenge implements mobile exergame mechanics.

4.4.4 DualFlow API

The DualFlow API provides classes and interfaces to interpret player's performance on exergames and calculate intensity and difficulty levels for each mechanic and challenge.

4.4.4.1 DualFlowApi

The DualFlowApi interface is the main entry point for the DualFlow API and declares methods to retrieve and calculate intensity and difficulty levels according for a specific challenge.

4.4.4.2 Difficulty

The Difficulty interface is a data interface for characterizing a challenge's difficulty. It declares methods to retrieve the challenge and mechanics information and increment or decrement the difficulty level by a DifficultyDelta.

4.4.4.3 DifficultyDelta

The DifficultyDelta interface is a data interface that characterizes the change in a challenge difficulty level.

4.4.4.4 Intensity

The Intensity interface is a data interface for characterizing a challenge's intensity. It declares methods to retrieve the player fitness information and increment or decrement the intensity level by an IntensityDelta.

4.4.4.5 IntensityDelta

The IntensityLevel interface is a data interface that characterizes the change in a challenge intensity level.

4.4.5 Profile API

The Profile API registers the players performance through exergaming and records the fitness profile as well as game results and performance, in order to create a exergame profile that can be used to calculate difficulty/intensity levels.

4.4.5.1 Profiles

The Profiles interface is the main entry point for the Profile API and it features access to the Google Fit API and Exergames API data.

4.4.5.2 Profile

The Profile interface is a data interface for retrieving exergame profile information. It declares methods to retrieve profile's ID, player's data and exergames' data, and to calculate a base fitness level to be used by the DualFlowApi.

4.5 Summary

In this chapter, I provided an insight on the external components that support the mobile exergames framework: the Android API and the Google Play Services. The Android API provides operating system functionality and access to the mobile devices' hardware and raw data; regarding

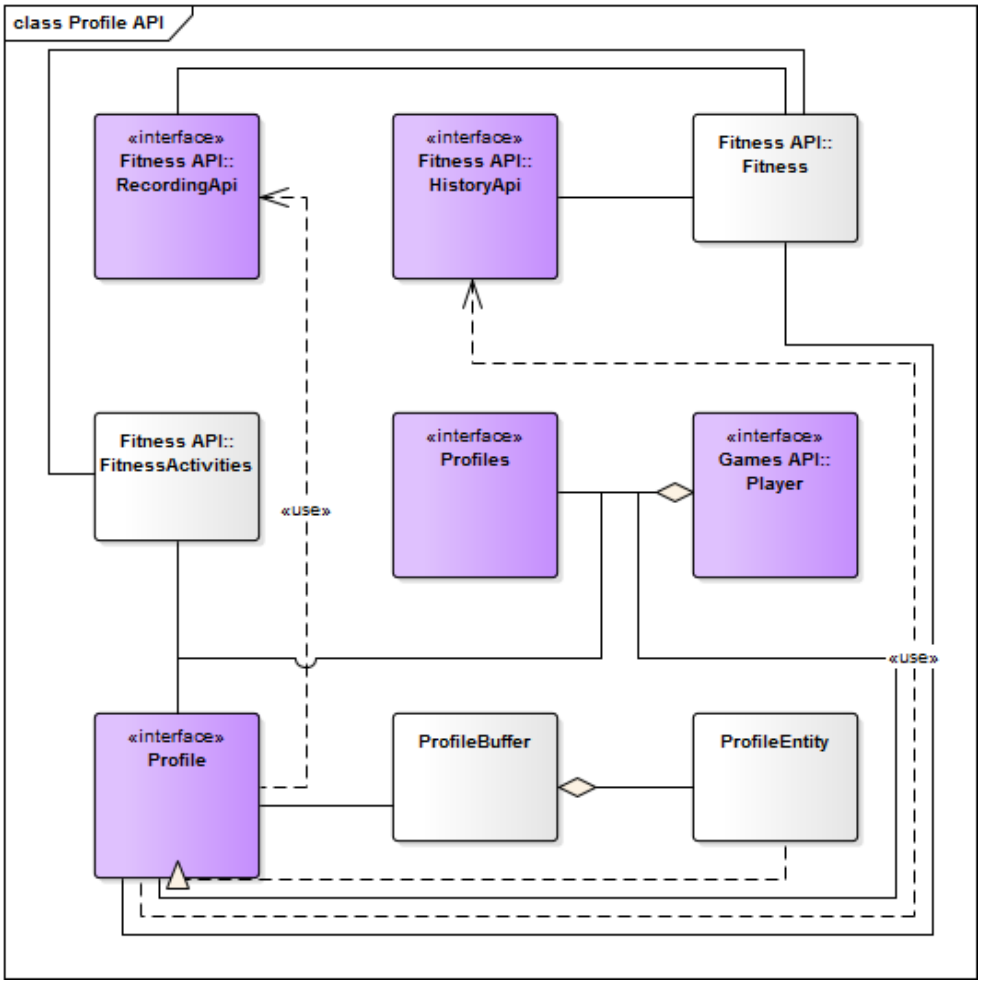

Figure 4.7: Profile API.

exergames, the Android API provides access to the accelerometer and gyroscope motion sensors as well as location-based classes. The Google Play Services represent a bigger part of the external support by offering basic functionality for Google's main services, namely, Google Play Games Service, Location Services, Google Places, Google Maps and Google Fit. All of these services provide resources to support the development of a mobile exergame framework, being on its own (Maps, Location, Places) or by providing useful classes and interfaces upon which to extend into a more specialized category: mobile exergames (exercise/fitness/sports games).

Then, I described the components internal to the framework and how they behave. The mobile exergame framework is to be further extended for each game's design and needs but in general, it offers the ability to create exergames based on exergame mechanics. While mechanics can be tried-out, their goal is to be part of the challenges that make the game's story (events, quests, milestones). The framework uses a Profile to keep tracks on location, motion, fitness and performance data, which is passed to the Dual Flow API in order to calculate intensity and difficulty levels for the game, as well as challenges opportunities.

Chapter 5

GeoCatch

In this chapter, I present GeoCatch, a mobile exergame developed as a proof of concept for the mobile exergames framework. In the following sections, I describe the game's design and technical specifications.

5.1 Game Design

GeoCatch is a mobile exergame for Android that incites players to complete exercise challenges based on the player's fitness level in any place.

5.1.1 Description

In GeoCatch, you decide when and where to take the challenges, the game will do the rest. When you're feeling up to a good sprint, start your engine, let the game check your surroundings and let it show you where to run - but be aware! You must be in the zone before the time is gone! And if you're too tired to run, let it ease up and make it fun! The tag-chasing mobile exergame, GeoCatch, will make your heart beat for a feat.

5.1.2 Key Features

The key features of GeoCatch are:

- **Location-based Context** — GeoCatch lets you play in open spaces, because they're the safest places.
- **User Context** — GeoCatch knows when you're free or when there's somewhere you ought to be.
- **Fitness Profile** — GeoCatch determines how fit you are and how much to raise the bar.
- **Dual Flow Engine** — GeoCatch knows it is only fun when you can run.

5.1.3 Experimental Features

GeoCatch makes use of the mobile exergames framework by exploring and extending the provided APIs in order to create new and experimental location-based exercise mechanics.

5.1.4 Major development tasks

- **Dual Flow Engine** — The Dual Flow Engine receives input from the player's performance and fitness level to calculate the intensity and difficulty of the challenges to be created.

5.1.5 External Code

The game runs and was developed on Android and implements an interface for the Google Play Services and Android API.

5.2 Technical Specification

In this section, I present the technical specifications of the game, including the game mechanics, the core gameplay, the flow of the game and the gameplay elements.

5.2.1 Game Mechanics

GeoCatch is a tag-chasing mobile exergame that challenges the player to sprint across multiple checkpoints within a time limit.

5.2.1.1 Core Gameplay

The core gameplay uses the mobile exergames framework and Google Play services to generate geofences in the area the player is playing. Each geofence has a time limit for the player to trigger a geofence transition into its radius. Each geofence is randomly located on proper space (i.e.: a park) selected from Google Place search results.

The player must sprint towards the checkpoint (geofence) and get within its radius, which will be decreasing accordingly to the time left on the timer. When the player enters a geofence's radius, another one is created nearby.

If the player fails to enter a geofence's radius, the game offers the player the opportunity to try again with new parameters. These parameters help the framework calculate a new intensity and difficulty level, which changes the time and radius of the geofences. All recorded data is stored to the player's profile so the game knows about the fitness level and performance level of the player.

5.2.1.2 Game Flow

The game starts when the player selects a new game; a Google Map is presented to the player with the available challenges. The player selects the challenge it wishes to play and accepts it.

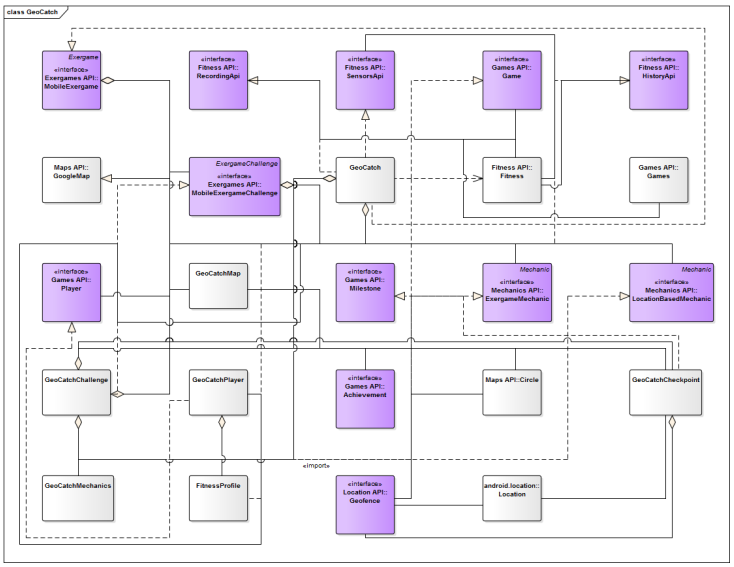

Figure 5.1: The GeoCatch class diagram.

The game creates the first geofence somewhere near the player and "listens" for geofence's trigger events; the map displays a circle representing the geofence radius as it decreases, providing visual feedback to the player.

5.2.1.3 Gameplay Elements

The main elements of GeoCatch gameplay are illustrated in the following class diagram (see Figure 5.1):

GeoCatch makes use of an extended version of the mobile exergames framework. It is a mobile exergame that combines exergame and location-based mechanics in order to create multiple-checkpoint sprint challenges (GeoCatchChallenge). Each checkpoint is a milestone (see Appendix A), within the challenge.

The challenges receive the player's profile information (location, current activity, motion, fitness level and performance history) to create a geofence at a minimum distance and an expiration date/time limit for an transition to occur.

To help the player see the geofence, it is used a Google Map with the player's current position, the current geofence to "catch" (represented with a gradual-shrinking circle on the map) and an estimated position for the next geofence (represented with a small circle/dot on the map).

When the player transitions into the geofence radius, the Dual Flow API computes the performance and returns a new intensity level for the next geofence, which translates in bigger/smaller radius and/or longer/shorter expiration date. The player corroborates the game adaptation by selecting the difficult level.

At the end of the run, all performance values and current fitness information are used to update the fitness level of the player and stored (via Google Fit) for future use.

5.3 Summary

In this chapter, I introduced GeoCatch, a mobile exergame developed with the mobile exergames framework; I described it's design and technical specification, the key features, core gameplay, game flow and elements that interact with the player.

Chapter 6

Conclusions and Future Work

In this chapter, I present the final thoughts on the project and point out the direction to follow in the future.

When I first started this dissertation, the work on mobile exergames devleopment was starting to grow; the tools provided by the big companies were not quite as rich as they are now and the development of mobile exergames was full of challenges and gaps. Particularly, the ability to read motion data and monitor player's fitness levels and heart-rate were not available without third-party wearable devices and hard coded applications.

Nowadays, the big technological companies, like Apple, Google and Microsoft, have developed wearable devices and ubiquitous technology to monitor end-users health and fitness levels. Gadgets, like smartwatches and band fits, with embedded heart-rate monitors and temperature sensors, are now able to measure end-users health and fitness levels without intrusive technologies and without changing their lifestyles. These technologies triggered the development of frameworks for analysis of user information, aiming to improve user's physical condition.

Those framework made easy to develop physically-interactive applications which tracked the users' contextual information and provided feedback and suggestions for improving their lifestyles. The entertainment industry gained from these new technologies as they provided easy access to new frameworks, which, in turn, led to the development of new mobile games that implemented exergame mechanics.

Mobile Exergames development increased with these frameworks; but the overall interest in exergames is divided between sports and fitness applications, with no particular emphasis on gaming experiences, and home-based exergames developed for conventional consoles like Nintendo Wii. I believe that mobile exergame development's future must shift and start working on bringing the virtual part of gaming experiences to the real world; for once, I think that improving augmented reality technologies and increasing network access points would lead to mobile exergames that would make use of virtual objects in real world environments. Nevertheless, I think that developing a mobile exergames framework is getting one step closer to the future of mobile exergames

technology.

The next steps for improving mobile exergame development, would be to integrate new ubiquitous technology such as smartwatches and other wearable devices, car technology, and create a framework that could take those devices into a bigger environment with access to more than just location and motion, both in real and virtual worlds.

Regarding the framework, the next step would be to apply finishing touches and study the migration for other mobile platforms (iOS, Windows Phone) or a generalization that could be used in cross-platform development. It's also necessary to test multiple game designs and flexible the framework is.

Finally, it is necessary to finish developing the concept game (GeoCatch) in order to validate the framework.

References

[And15a] Android. Android - History, 2015.

[And15b] Android Developers. Develop Apps | Android Developers, 2015.

[And15c] Android Developers. Introduction to Android | Android Developers, 2015.

[APP+11] Maayan Agmon, Cynthia K. Perry, Elizabeth Phelan, George Demiris, and Huong Q. Nguyen. A Pilot Study of Wii Fit Exergames to Improve Balance in Older Adults. *Journal of Geriatric Physical Therapy*, page 1, 2011.

[Box15] Box2D. About | Box2D, 2015.

[BS05] Ian Bogost and Cherry St. The Rhetoric of Exergaming. In *The Georgia Institute of Technology*, page 10, 2005.

[CESL06] Sunny Consolvo, Katherine Everitt, Ian Smith, and James A Landay. Design requirements for technologies that encourage physical activity, 2006.

[CNF08] Taj Campbell, Brian Ngo, and James Fogarty. Game design principles in everyday fitness applications, 2008.

[Coc15] Cocos2dx. Cocos2d-x - World's #1 Open-Source Game Development Platform, 2015.

[CPA+03] Chris Crawford, Sue Peabody, The Art, World Wide Web, and Donna Loper. *The Art of Computer Game Design by Chris Crawford*. Washington State University Vancouver, 2003.

[CSW+13] Ying Yu Chao, Yvonne K. Scherer, Yow Wu Wu, Kathleen T. Lucke, and Carolyn a. Montgomery. The feasibility of an intervention combining self-efficacy theory and Wii Fit exergames in assisted living residents: A pilot study. *Geriatric Nursing*, 34(5):377–382, 2013.

[Dev15a] Google Developers. Android APIs | Google Fit | Google Developers, 2015.

[Dev15b] Google Developers. Get Started with Play Games Services for Android | Play Games Services for Android | Google Developers, 2015.

[Dev15c] Google Developers. Google Maps Android API | Google Developers, 2015.

[Exe11] What I S Exergaming. Exergaming. *ACSM*, 2011.

[FHH12] G I Fair, By Matt Haggerty, and Matt Haggerty. The State of Mobile Game Development. *GamesIndustry.biz*, pages 1–5, 2012.

REFERENCES

[Gam15a] Gamemaker. Workflow | GameMaker: Studio | YoYo Games, 2015.

[Gam15b] Battery Powered Games. BatteryTech - Mobile Game Development Platform | BatteryTech, 2015.

[Gar15] Gartner. Gartner Says Smartphone Sales Surpassed One Billion Units in 2014. Technical report, Gartner, 2015.

[Gau15] John Gaudiosi. Mobile game revenues set to overtake console games in 2015. *Fortune.com*, pages 1–5, 2015.

[GHW+10] Stefan Gobel, Sandro Hardy, Viktor Wendel, Florian Mehm, and Ralf Steinmetz. Serious games for health: personalized exergames, 2010.

[Goo15] Google Developers. FusedLocationProviderApi | Google APIs for Android | Google Developers, 2015.

[Hal11a] Robert J Hall. A point-and-shoot weapon design for outdoor multi-player smartphone games, 2011.

[Hal11b] Robert J Hall. Software engineering challenges of multi-player outdoor smartphone games, 2011.

[HESGS11] S Hardy, A El-Saddik, S Gobel, and R Steinmetz. Context aware serious games framework for sport and health. In *Medical Measurements and Applications Proceedings (MeMeA), 2011 IEEE International Workshop on*, pages 248–252, 2011.

[HFA+10] Dave Harley, Geraldine Fitzpatrick, Lesley Axelrod, Gareth White, and Graham McAllister. Making the Wii at Home: Game Play by Older People in Sheltered Housing. In Gerhard Leitner, Martin Hitz, and Andreas Holzinger, editors, *HCI in Work and Learning, Life and Leisure*, volume 6389, chapter 10, pages 156–176. Springer Berlin Heidelberg, 2010.

[HGS13] S Hardy, S Göbel, and R Steinmetz. Adaptable and personalized game-based training system for fall prevention. pages 431–432, 2013.

[Hoy06] Johanna Hoysniemi. *Design and Evaluation of Physically Interactive Games*. Academic dissertation, University of Tampere, Tampere, Finland, 2006.

[KBS+12] O Korn, M Brach, A Schmidt, T Hörz, and R Konrad. Context-sensitive user-centered scalability: An introduction focusing on exergames and assistive systems in work contexts, 2012.

[KM10] Kristian Kiili and Sari Merilampi. Developing Engaging Exergames with Simple Motion Detection. *Focus*, pages 103–110, 2010.

[KMK11] Antti Koivisto, Sari Merilampi, and Kristian Kiili. Mobile exergames for preventing diseases related to childhood obesity, 2011.

[MAVG09] Florian 'Floyd' Mueller, Stefan Agamanolis, Frank Vetere, and Martin R Gibbs. A framework for exertion interactions over a distance, 2009.

[MEV+11] Florian 'Floyd' Mueller, Darren Edge, Frank Vetere, Martin R Gibbs, Stefan Agamanolis, Bert Bongers, and Jennifer G Sheridan. Designing sports: a framework for exertion games, 2011.

REFERENCES

[MGV08] Florian 'Floyd' Mueller, Martin R Gibbs, and Frank Vetere. Taxonomy of exertion games, 2008.

[MGV10] Florian Mueller, MartinR Gibbs, and Frank Vetere. Towards understanding how to design for social play in exertion games. *Personal and Ubiquitous Computing*, 14(5):417–424, 2010.

[MVG+10] Florian 'Floyd' Mueller, Frank Vetere, Martin R Gibbs, Stefan Agamanolis, and Jennifer Sheridan. Jogging over a distance: the influence of design in parallel exertion games, 2010.

[MVG+12] Florian Mueller, Frank Vetere, Martin Gibbs, Darren Edge, Stefan Agamanolis, Jennifer Sheridan, and Jeffrey Heer. Balancing exertion experiences, 2012.

[SAC13] A E Staiano, A A Abraham, and S L Calvert. Adolescent exergame play for weight loss and psychosocial improvement: A controlled physical activity intervention. *Obesity*, 21(3):598–601, 2013.

[SC11a] A E Staiano and S L Calvert. Exergames for Physical Education Courses: Physical, Social, and Cognitive Benefits. *Child Development Perspectives*, 5(2):93–98, 2011.

[SC11b] Amanda E Staiano and Sandra L Calvert. The promise of exergames as tools to measure physical health. *Entertainment Computing*, 2(1):17–21, 2011.

[Sch08] Jesse Schell. *The Art of Game Design: A Book of Lenses*, volume 54. 2008.

[SHM07] Jeff Sinclair, Philip Hingston, and Martin Masek. Considerations for the design of exergames. 1(212):289–296, 2007.

[SHM09] Jeff Sinclair, Philip Hingston, and Martin Masek. Exergame development using the dual flow model, 2009.

[Sta10] Tadeusz B Stach. Design aspects of multiplayer exergames. Technical report, 2010.

[Stu15] Keith Stuart. Mobile games revenue to overtake consoles in 2015. *The Guardian*, page 2015, 2015.

[Tec15a] Marmalade Technologies. The fastest cross-platform C++ game development SDK | Marmalade, 2015.

[Tec15b] Unity Technologies. Unity - Game Engine, 2015.

[TKWP01] S G Trost, L M Kerr, D S Ward, and R R Pate. Physical activity and determinants of physical activity in obese and non-obese children. *International journal of obesity and related metabolic disorders : journal of the International Association for the Study of Obesity*, 25(6):822–829, 2001.

[VCBE08] Tamas Vajk, Paul Coulton, Will Bamford, and Reuben Edwards. Using a Mobile Phone as a Wii-like Controller for Playing Games on a Large Public Display. *International Journal of Computer Games Technology*, 2008:6, 2008.

[VG09] Amy Voida and Saul Greenberg. Wii all play: the console game as a computational meeting place, 2009.

[WHO15] WHO. WHO | Obesity and overweight. *World Health Organization*, (311), 2015.

REFERENCES

[WSU06] R J Wieringa, Computer Science, and Vrije Universiteit. Requirements Engineering : Frameworks for Understanding. 2006.

[Zec15] Mario Zechner. libgdx, 2015.

Appendix A

Google Play Services

In this chapter, I will provide an overview of all classes and interfaces supported by Google Play Services.

A.1 Google Play Game Services for Android

Android game development was made easy with the integration into Google Play games services. The Play Games SDK provides cross-platform Google Play games services that provides achievements, leaderboards, saved games and real-time/turn-based multiplayer, events and quests and location-based features to mobile games [Dev15b]. The following are the main classes and interfaces of the Google Play Games Services for Android:

A.1.1 Games

The Games module contains the games client class.

A.1.1.1 Games

The Games class is the main entry point for the Games API and provides APIs and interfaces to access the Google Play games services functionality. It stores data regarding achievements, events, leaderboards, players and quests, and methods to setup the game options.

A.1.1.2 Game

The Game interface is a data interface for retrieving game information and declares methods to retrieve game developer, game support, multiplayer modes, achievement and leaderboard information.

A.1.1.3 GameEntity

The GameEntity class is a data object representing a set of Game data and implements the Game interface methods.

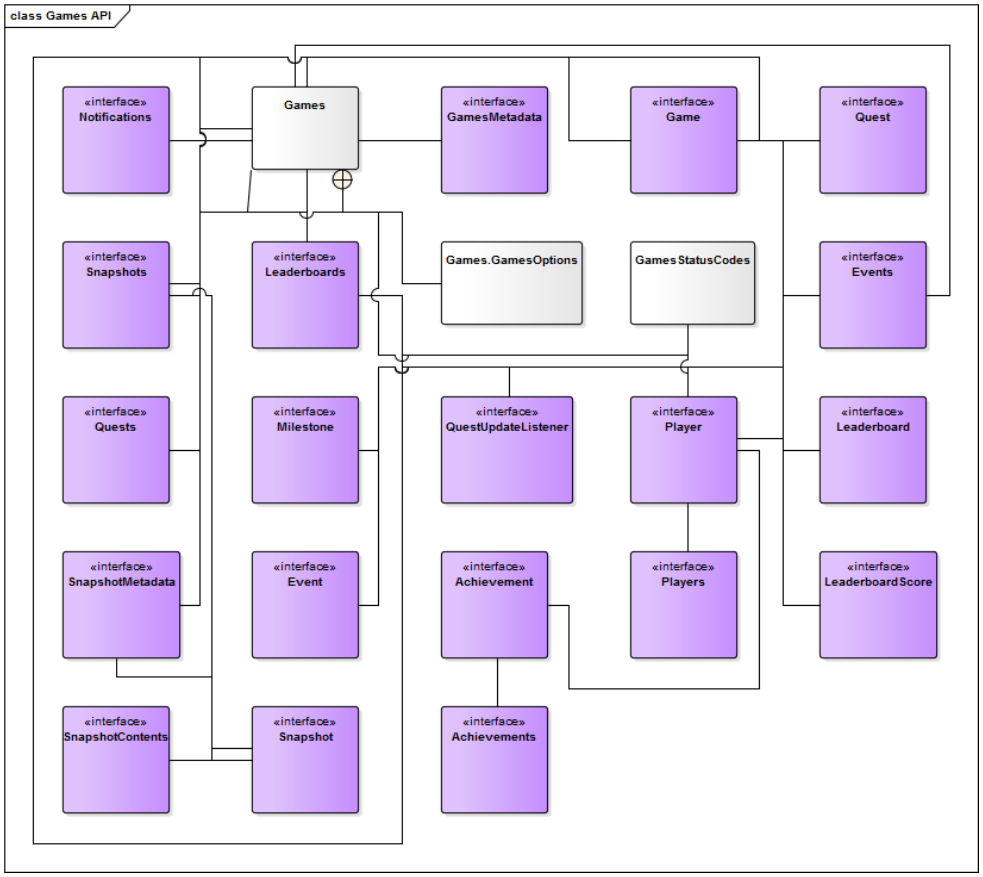

Figure A.1: Google Play games services for Android.

A.1.1.4 Players

The Players interface is the entry point for player functionality and provides methods to retrieve player's information.

A.1.1.5 Player

The Player interface is a data interface for retrieving player information and declares methods to retrieve player's ID, name, and level information.

A.1.1.6 PlayerEntity

The PlayerEntity class is a data object representing a set of Player data and implements the Player interface methods.

A.1.1.7 PlayerLevel

The PlayerLevel class is a data object representing a level that a player can obtain in the metagame and is composed of three components: a numeric value (level), and a range of experience totals.

The player is given a level when they earn experience between the minimum experience and the maximum experience for that level.

A.1.1.8 PlayerLevelInfo

The PlayerLevelInfo class is a data object representing the current level information of a player in the metagame and is composed of four components: the player's current experience points, the timestamp of the player's last level-up, the player's current level, and the player's next level.

A.1.2 Events

This module contains data classes for Play Games events.

A.1.2.1 Events

The Events interface is the entry point for events functionality. In Google Play games services, events are created though the Google Play Developer Console.

A.1.2.2 Event

The Event interface is a data interface for retrieving event information. It declares methods to retrieve an event's ID, description, name, visibility, and player's information.

A.1.2.3 EventEntity

The EventEntity class is a data object representing the data for an event and it implements the Event interface methods.

A.1.3 Quests

This module contains data classes for Play Games quests.

A.1.3.1 Quests

The Quests interface is the entry point for the Quest functionality, which include loading and accepting quests, and claiming milestones.

A.1.3.2 Quest

The Quest interface is a data interface for retrieving quest information. It declares methods for retrieving a quest's ID, name, description, state, accept and end timestamps, and its game information.

A.1.3.3 QuestEntity

The QuestEntity class is a data object representing the data for a quest and it implements the Quest interface methods and methods for retrieving milestone information.

A.1.3.4 Milestone

The Milestone interface is a data interface for retrieving milestone information. It declares methods for retrieving a milestone's ID, state, current progress, target progress and completion reward data.

A.1.3.5 MilestoneEntity

The MilestoneEntity class is a data object representing the data for a milestone and implements the Milestone interface's methods.

A.1.4 Achievement

This module contains classes for loading and updating achievements.

A.1.4.1 Achievements

The Achievements interface is the entry point for achievements functinality. It contains methods to retrieve achievement information, load, reveal and unlock achievements.

A.1.4.2 Achievement

The Achievement interface is a data interface for retrieving achievement information. It declares methods to retrieve achievement's ID, name, description, state, type, experience points, and player's information.

A.1.4.3 AchievementEntity

The AchievementEntity class is a data object representing a set of Achievement data and implements the Achievement interface methods.

A.1.5 Leaderboard

This module contains data classes for leaderboards.

A.1.5.1 Leaderboards

The Leaderboards interface is the entry point for leaderboard functionality. It contains methods for loading and submitting leaderboards' information.

A.1.5.2 Leaderboard

The Leaderboard interface is a data interface for leaderboard metadata. It declares methods to retrieve the leaderboard's ID, name, order of score and leaderboard variants.

A.1.5.3 LeaderboardVariant

The LeaderboardVariant interface is a data interface for a specific variant of a leaderboard; a variant is defined by the combination of the leaderboard's collection (public or social) and time span (daily, weekly, or all-time). It declares methods to retrieve the player's rank and score, and time span for this variant.

A.1.5.4 LeaderboardScore

The LeaderboardScore interface is a data interface representing a single score on a leaderboard. It declares methods to retrieve score, score's holder and rank information.

A.1.5.5 ScoreSubmissionData

The ScoreSubmissionData class is a data object representing the result of submitting a score to a leaderboard. It defines methods for retrieving the leaderboard's ID, the player's ID and the score result.

A.1.6 Snapshot

This module contains data classes for snapshot functionality.

A.1.6.1 Snapshots

The Snapshots API allows to store data representing the player's game progress on Google's servers; this can be used to restored saved state from previous game sessions and provide a visual indicator of progression to the player. The interface defines methods to commit, delete, load and open snapshots.

A.1.6.2 Snapshot

The Snapshot interface is a data interface for a representation of a saved game. This includes both the metadata and the actual game content. It declares methods to retrieve snapshot metadata and contents.

A.1.6.3 SnapshotEntity

The SnapshotEntity class is a data object representing the data for a saved game and it implements the Snapshot interface's methods.

A.1.6.4 SnapshotContents

The SnapshotContents interface is a data interface for a representation of Snapshot contents. It declares methods to load and save game content through parcelable operations.

A.1.6.5 SnapshotMetadata

The SnapshotMetada interface is a data interface for the metadata of a saved game. it declares methods to retrieve the metadata description, progress value, played time, changes, and game and player information.

A.1.6.6 SnapshotMetadataEntity

The SnapshotMetadataEntity class is a data object representing the metadata for a saved game and it implements the SnapshotMetada interface methods.

A.1.7 Multiplayer

This module contains data classes for multiplayer functionality. There are two types of multiplayer: real-time multiplayer and turn-based multiplayer.

A.1.7.1 Multiplayer

The Multiplayer interface contains common constants for multiplayer functionality.

A.1.7.2 Participant

The Participant interface is a data interface for multiplayer participants. It defines values for participation status (i.e.: declined, finished, invited, joined, left).

A.1.7.3 ParticipantEntity

The ParticipantEntity is a data object representing a Participant in a match and it implements the Participant interface's methods.

A.1.7.4 ParticipantUtils

The ParticipantUtils class is a utilities class for working with multiplayer participants.

A.1.7.5 ParticipantResults

The ParticipantResults is a data class used to report a participant's result in a match (i.e.: disconnect, loss, tie, win).

A.1.7.6 Participatable

The Participatable interface defines methods for object that contain participants.

A.1.7.7 Invitations

The Invitations interface is the entry point for invitations functionality and it declares methods for loading invitations.

A.1.7.8 Invitation

The Invitation interface is a data interface for an invitation object and it implements the Participatable interface's methods.

A.1.7.9 InvitationEntity

The InvitationEntity class is a data object representing the data for a multiplayer invitation and it implements the Invitation interface's methods.

A.1.8 Real-Time Multiplayer

This module contains data classes for real-time multiplayer functionality.

A.1.8.1 RealTimeMultiplayer

The RealTimeMultiplayer interface is the entry point for real-time multiplayer functionality. It defines methods for sending reliable messages, as well as create, join or leave real-time rooms for games, decline and dismiss invitations.

A.1.8.2 RealTimeMessage

The RealTimeMessage class is a data object representing a message received from participants in a real-time room.

A.1.8.3 Room

The Room interface is a data interface for room functionality. It declares methods to retrieve the room's ID, status, variant and description, as well as the creator and the participants.

A.1.8.4 RoomConfig

The RoomConfig class is a data object representing the configuration for a new room.

A.1.8.5 RoomEntity

The RoomEntity class is a data object representing the data for a room and it implements the Room interface's methods.

A.1.9 Turn-based Multiplayer

This module contains data classes for turn-based multiplayer functionality.

A.1.9.1 TurnBasedMultiplayer

The TurnBasedMultiplayer interface is the entry point for turn-based multiplayer functionality. It defines methods for initiating, loading, updating, canceling and leaving matches.

A.1.9.2 TurnBasedMatch

The TurnBasedMatch interface is a data interface for turn-based specific match functionality. it declares methods for retrieving data, description, number, status, variant and version of the match, as well as the creator's ID, the associated Game, the Participant's information and the rematch availability.

A.1.9.3 TurnBasedMatchEntity

The TurnBasedMatchEntity class is a data object representing the data for a turn-based match and implements the TurnBasedMatch interface's methods.

A.1.9.4 TurnBasedMatchConfig

The TurnBasedMatchConfig class defines the configuration for creating a new turn-based match.

A.2 Google Fit APIs for Android

The Google Fit APIs for Android consists of the following APIs:

- **Bluetooth Low Energy API** — The Bluetooth Low Energy API provides access to Bluetooth Low Energy sensors in Google Fit. This API enables your app to look for available BLE devices and to store data from them in the fitness store.
- **Config API** — The Config API provides custom data types and additional settings for Google Fit. For more information, see Custom Data Types and Disconnect from Google Fit.
- **History API** — The History API provides access to the fitness history and lets apps perform bulk operations, like inserting, deleting, and reading fitness data. Apps can also import batch data into Google Fit.

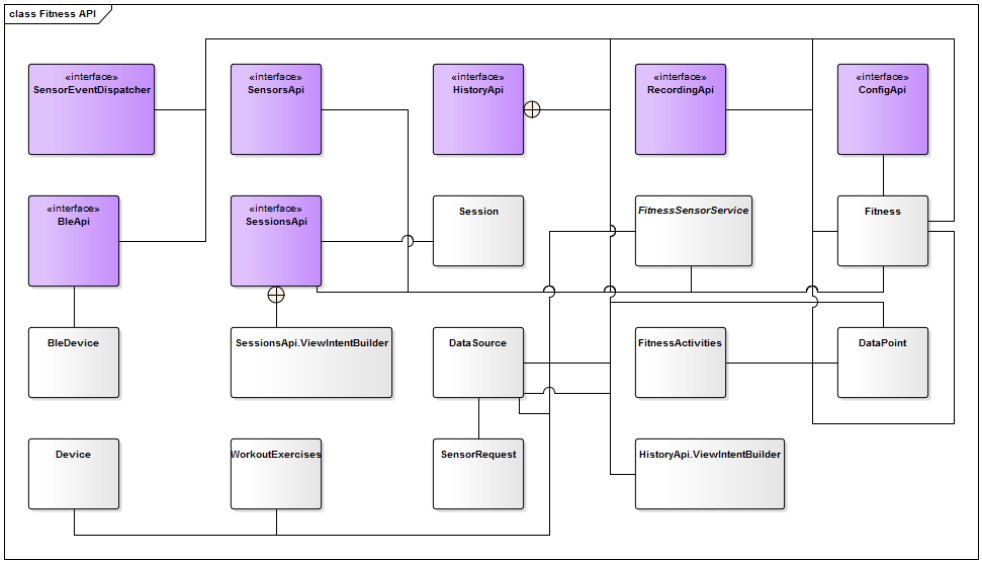

Figure A.2: Google Fit Android API.

- **Recording API** — The Recording API provides automated storage of fitness data using subscriptions. Google Fit stores fitness data of the specified types in the background and persists app subscriptions.

- **Sensors API** — The Sensors API provides access to raw sensor data streams from sensors available on the Android device and from sensors available in companion devices, such as wearables.

- **Sessions API** — The Sessions API provides functionality to store fitness data with session metadata. Sessions represent a time interval during which users perform a fitness activity.

A.2.1 Fitness

The main entry point for Google Fit APIs is the Fitness class. It supports different intents in order to help fitness apps collaborate (i.e.: view, track and view goal actions), and it contains tokens to pass to the following APIs:

A.2.1.1 BleApi

The BleApi is an API for scanning, claiming, and using Bluetooth Low Energy devices in Google Fit. It declares methods to claim, list and unclaim Bluetooth devices, as well as starting and stopping their scan processes.

Most BLE devices will accept connections from any other device, without the need for pairing. To prevent Google Fit from using data from a device the user does not own, it's required for a device to be claimed before it can be used in the platform. The API supports scanning and

claiming devices. Once a device is claimed, its data sources are exposed via the Sensors and Recording APIs, similar to local sensors.

A.2.1.2 ConfigApi

The ConfigApi is an API for creating and reading custom data types and disabling Google Fit in applications.

A.2.1.3 HistoryApi

The HistoryApi is an API for inserting, deleting, and reading data in Google Fit. It should be used whenever historical data is needed and it can be combined with a subscription in the Recording Api to collect data in the background and query it later for displaying. It can also be used for batch insertion of data that was collected outside of Google Fit. It can be useful when data is entered directly by the user or imported from a device that isn't supported by the platform.

A.2.1.4 RecordingApi

The RecordingApi is an API which enables low-power, always-on background collection of sensor data into the Google Fit store. Sensor subscriptions are active when the requesting app is not running, and persisted through system restarts. Collected data can be queried using the History API.

Unlike the Sensors API, the Recording API does not support delivery of live sensor events. When live data is needed (such as when the app is open in the foreground), a listener should be added with the Sensors API. An app can have both an active subscription and an active listener at the same time. Unlike listeners, subscriptions don't need to be removed and re-added periodically.

A.2.1.5 SensorsApi

The SensorsApi is an API which exposes different sources of fitness data in local and connected devices, and delivers live events to listeners. It exposes data sources from hardware sensors in the local device and in companion devices and it also exposes data sources from applications. The API supports adding and removing listeners to live data streams from any available data source and it allows listening on a DataType from one or more data sources.

The Sensors API should be used whenever live updates from a sensor stream need to be pushed to the application (for instance, to update a UI). The History API can be used to query historical data in a pull-based model for scenarios where latency isn't critical.

A.2.1.6 SessionsApi

The SessionsApi API is an API for creating and managing sessions of user activity in Google Fit. Sessions are a way of storing user-visible groups of related stream data in a useful and shareable manner, and allow for easy querying of the data in a detailed or aggregated fashion.

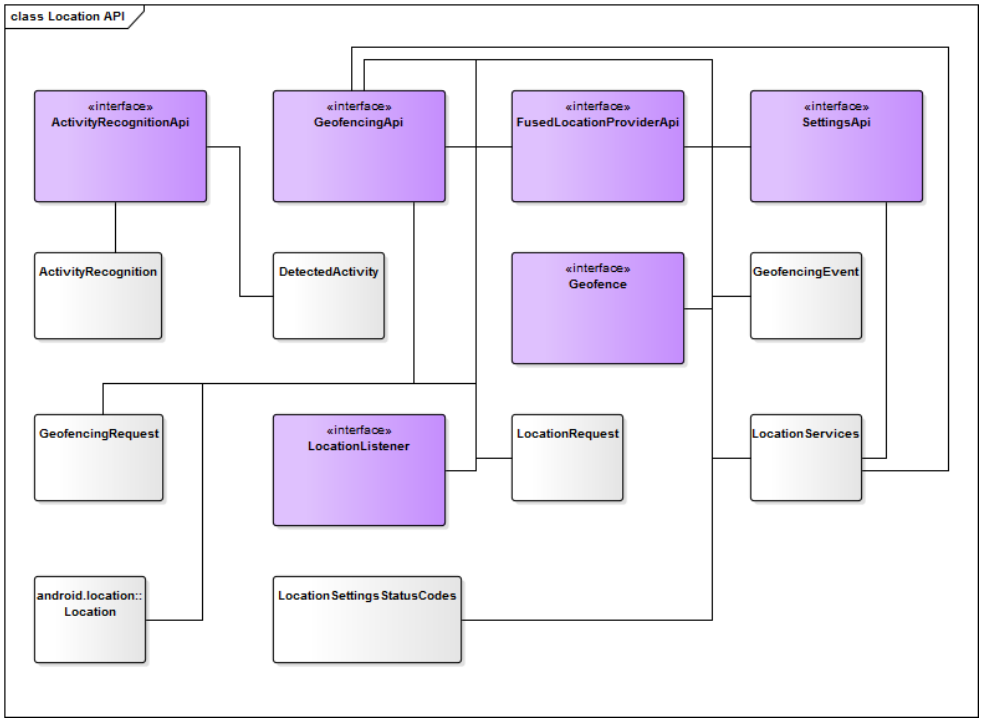

Figure A.3: Google Play services location APIs for Android.

The API contains methods to mark the time range of the session. Data inserted using the HistoryApi during the period where the session is in progress will be associated with that Session once it is stopped. All data in Google Fit that falls within the time range of an inserted session will be implicitly associated with that session.

A.2.2 FitnessActivities

The FitnessActivities class contains constants representing different user activities, such as walking, running, and cycling. Activities are used in Sessions, DataTypes and in read queries. A Session associates an activity with all data that was recorded during a time interval.

Activities be stored and read using the activity sample and activity segment data types. When samples are stored, these are automatically converted into segments by the platform's default data source. When reading data, the activity segment and activity type bucketing strategies can be used to aggregate data by the activities happening at the time data was collected. A subset of the activities can be detected by Activity Recognition from Google Play Location Services API.

A.3 Google Play Location Services APIs

The Google Play Location Services APIs provide features to request the last known location of the user's device, recognize the current user activity, and set alerts to be notified when the device enters or exits an geographical region (geofence).

A.3.1 FusedLocationProviderApi

The FusedLocationProviderApi interface is the main entry point for interacting with the fused location provider. The API provides methods for retrieving the location availability and last known location, as well as receiving location updates.

A.3.1.1 LocationRequest

The LocationRequest class is a data object that contains quality of service parameters for requesting location data to the FusedLocationProviderApi. The quality of service is defined by the priority in magnitude and frequency of the request (i.e.: High Accuracy, Low Power, Balanced).

A.3.1.2 LocationResult

The LocationResult class is a data class representing a geographic location result and contains a list of locations from the fused location provider.

A.3.2 ActivityRecognitionApi

The ActivityRecognitionApi is the main entry point for interacting with activity recognition.

A.3.2.1 ActivityRecognition

The AcitivityRecognition class is the main entry point for activity recognition integration.

A.3.2.2 ActivitiyRecognitionResult

The ActivityRecognitionResult class is the result of an activity recognition and it contains a list of activities that a user may have been doing at a particular time, with a confidence level associated.

A.3.2.3 DetectedActivity

The DetectedActivity class is a data class with constants representing the detected activity of the device with an associated confidence.

A.3.3 GeofencingApi

The GeofencingApi interface is the main entry point for interacting with the geofencing APIs. It declares methods to add and remove geofences.

A.3.3.1 Geofence

The Geofence interface is a data interface that represents a geographical region that can be monitored by the geofencer service and generates alerts when users cross its boundaries.

A.3.3.2 GeofencingRequest

The GeofencingRequest class specifies the list of geofences to be monitored and how the geofence notifications should be reported.

A.3.3.3 GeofencingEvent

The GeofencingEvent class represents a transition or an error from the GeofencingApi API.

A.3.4 SettingsApi

The SettingsApi is the main entry point for interacting with the location settings-enabler APIs. It serves to ensure proper settings configuration for location needs.

A.3.4.1 LocationSettingsStates

The LocationSettingsStates class stores the current states of all location-related settings and defines methods to verify the presence and usability of BLE, GPS, and network location provider services.

A.3.4.2 LocationSettingsRequest

The LocationSettingsRequest class specifies the types of location services the client is interested in.

A.3.4.3 LocationSettingsResult

The LocationSettingsResult class represents the result of checking the location settings states with a LocationSettingsRequest.

A.4 Google Places API for Android

The main entry points for the Google Places API for Android are the GeoDataApi and the PlaceDetectionApi.

A.4.1 Place

The Place interface represents a physical space and encapsulates information about the location, including name, address and type, which can be local business, point of interest, or a geographic location.

A.4.1.1 PlaceBuffer

The PlaceBuffer class is a data structure providing access to a list of Places.

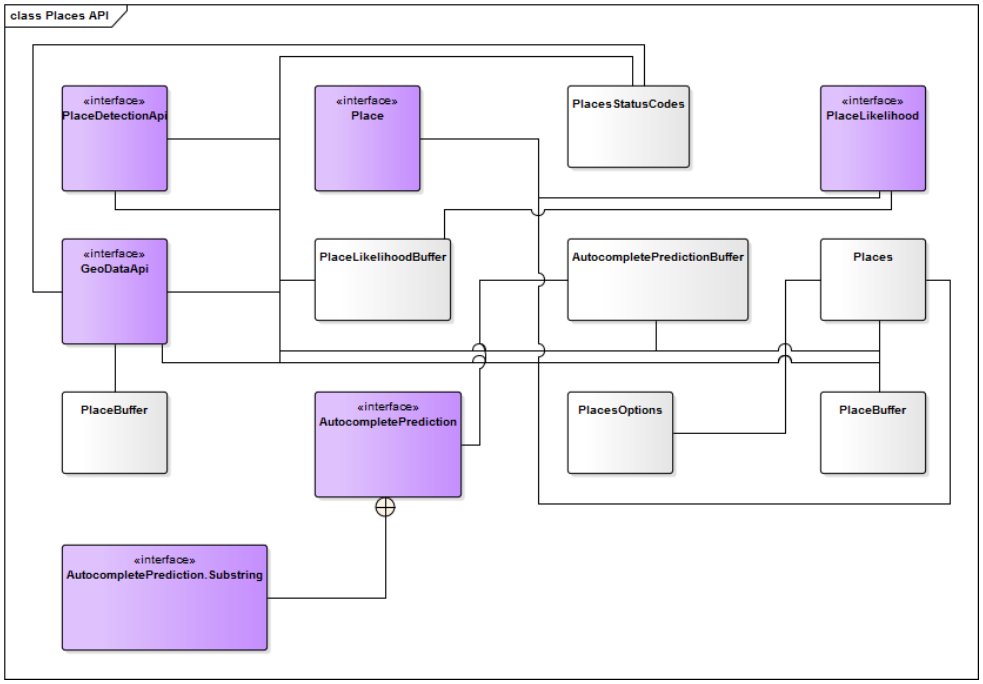

Figure A.4: Google Play games services for Android.

A.4.2 PlaceDetectionApi

The PlaceDetectionApi interface is the main entry point for the Google Place Detection API. It provides quick access to the device's current place, and offers the opportunity to report the location of the device at a particular place (i.e.: check in).

A.4.2.1 PlaceLikelihood

The PlaceLikelihood interface represents a Place and the relative likelihood of the place being the best match within the list of returned places for a single request.

A.4.2.2 PlaceLikelihoodBuffer

A PlaceLikelihoodBuffer class is a data buffer representing a list of PlaceLikelihood's.

A.4.3 PlaceReport

The PlaceReport class is an indication from the client that the device is currently located at a particular place.

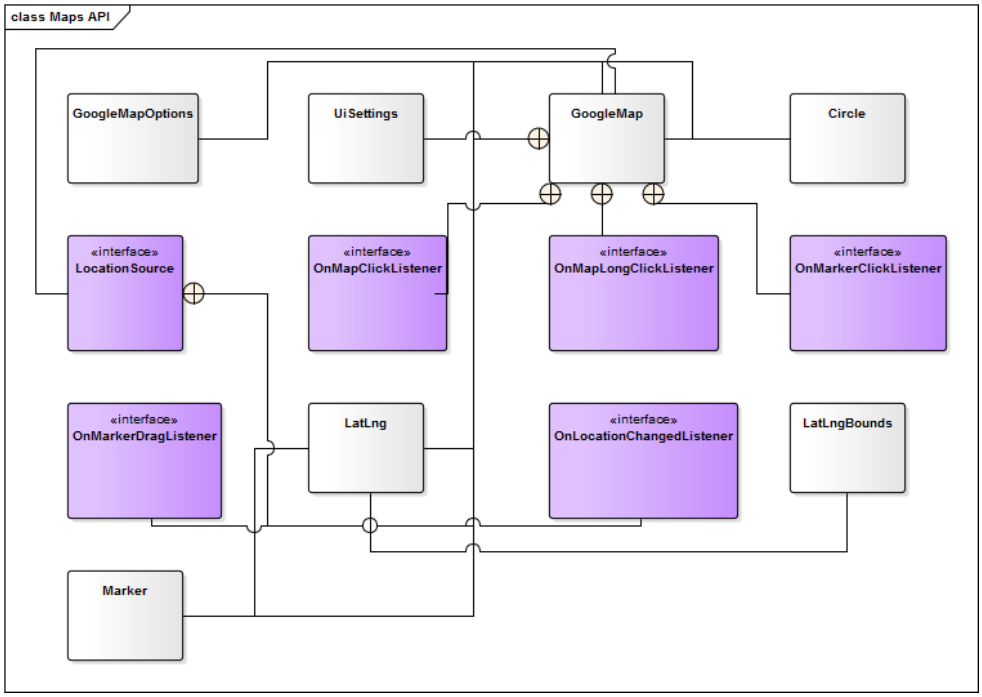

Figure A.5: Google Maps Android API.

A.4.4 GeoDataApi

The GeoDataApi interface is the main entry point for the Google Geo Data API. It provides access to Google's database of local place and business information, and it allows to add Places via AddPlaceRequest class.

A.4.4.1 AddPlaceRequest

The AddPlaceRequest class represents a Place to add to Google's Places database. It is necessary to pass the name, location, address, type and phone number (if available).

A.5 Google Maps Android API

The Google Maps Android API provides Google Maps to the framework. The API automatically handles access to Google Maps servers, data downloading, map display, and response to map gestures. It's also possible to add markers, polygons, and overlays to a basic map, and to change the user's view of a particular map area. These objects provide additional information for map locations, and allow user interaction with the map[1].

[1]To use the Google Maps Android API, it is mandatory to include the Google Play Services attribution text as part of a "Legal Notices" section in the application. The attribution text is available by making a call to GooglePlayServicesUtil.getOpenSourceSoftwareLicenseInfo.

A.5.1 GoogleMap

The GoogleMap class is the main class of the Google Maps Android API and is the entry point for all methods related to the map. A GoogleMap cannot be directly instantiated, rather, it must be obtained from a MapFragment or a MapView added to the application.

A.5.1.1 Marker

The Marker class represents a particular point on a map's surface and is displayed with an icon. A marker has the following properties: Alpha, Anchor, Position, Title, Snippet, Icon, Drag Status, Visibility, Flat or Billboard and Rotation.

A.5.1.2 Polygon

The Polygon class represents a convex or concave polygon on earth's surface, and it has the following properties: Outline, Holes, Stroke Width, Stroke Color, Fill Color, Z-Index, Visibility and Geodesic status.

A.5.1.3 Circle

The Circle class represents a circle on the earth's surface, and it has the following properties: Center, Radius, Stroke Width, Stroke Color, Fill Color, Z-Index and Visibility.

A.5.1.4 LatLng

The LatLng class represents a pair of latitude and longitude coordinates, stored as degrees.

A.5.1.5 LatLngBounds

The LatLngBounds class represents a rectangular boundary defined by a northeast and southwest LatLng coordinates.