# Design and Architecture of an Open Multiplayer Exergaming Platform called Move2Play

Flávio Coutinho, Ismael S. Silva, Glívia A. R. Barbosa, Thiago F. Costa,
Luis G. D. Carvalho, Cassiano B. Andrade, João P. S. Pereira, Reynaldo S. G. Pinto and Jonathan H. Silva
Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, MG 30510-000, Brazil
Email: {fegemo, ismaelsantana, gliviaangelica}@cefetmg.br,
{thiagofigcosta, luis.diniz.carvalho, cassiano.b.andrade, joao.psp96, reynaldo.sgp, 01jonathanh10}@gmail.com

2022 21st Brazilian Symposium on Computer Games and Digital Entertainment (SBGames) | 978-1-6654-6156-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/SBGAMES56371.2022.9961109

***Abstract*—Exergaming has been shown to increase the interest in regular physical activity. However, the available options usually involve acquiring specialized hardware, may be tied to a single game, and are generally designed for single player. A gym environment, for example, could benefit from not having to acquire adhoc and expensive hardware for each user. In this scenario, we propose Move2Play, an exergaming platform that relies on commodity hardware – a computer, a screen, and one smartphone per player – to offer single and multiplayer game sessions. In addition, the platform is extensible to new games, which third parties can produce. Also, four toy exergames were developed for demonstration. We believe that Move2Play can foster the practice of physical activities and contribute to promoting health and well-being.**

***Index Terms*—exergames, gaming platform, motion detection**

## I. Introduction

The regularity of physical activities is directly associated with preventing diseases such as diabetes, hypertension, mental disorders, cancer, chronic heart diseases, and obesity [1]. However, despite these benefits, a large portion of the population follows a sedentary lifestyle. For example, in Brazil, 45% of the residents do not practice regular exercises or spend less time than the duration recommended by the World Health Organization (i.e., less than 150 minutes each week) [1].

Gyms are a popular option for practicing exercises thanks to the freedom of choosing schedules and the variety of equipment and activities available. Despite that, evasion rates as high as 70% in the first year have been reported [1].

Exergames are an option for practicing physical activities indoors [2]. They are games that require players to make movements to play, integrating entertainment with physical activity. Single-player exergames focus on the stimulation and performance of an individual to improve their physical condition. In turn, multiplayer exergames focus on stimulating competition or cooperation between groups of people, aiming to engage them in physical activity. Furthermore, using motion detection techniques, exergaming can make regular physical exercises more fun and provide users with different goals to keep a healthy routine [2].

Motivated by this demand, this paper introduces an exergaming platform that allows users to play single or multiplayer games while performing aerobic physical activities. Move2Play focuses on gyms; however, the platform can also be hosted by other facilities, such as long-term care homes, schools, or by standalone users at home. Furthermore, the platform relies only on commodity hardware that may already be available, such as a computer, a screen (e.g., TV, monitor, projector), and the players' smartphones, exempting users and hosts from acquiring specialized hardware.

In addition, the platform includes single and multiplayer games that are simple and free while allowing new games to be developed by third parties. All platform components: Console, Controller, and Games are software and can be digitally distributed. Beyond the availability of the platform, a side contribution is its open architecture, as it can be used in other initiatives or commercial products.

## II. Related Work

Exergaming initiatives can be analyzed considering four dimensions: (a) single experience vs. platform, (b) player collaboration (cooperation or competition), (c) hardware setup, and (d) requirements for acquisition. Table I summarizes some characteristics of the reviewed literature.

In the (a) platform dimension, the works can either propose a single exergame or they can have multiple games, maybe supporting extension to new ones. On the lower end, the works [2]–[4], [6]–[8] present a single exergame each, whereas [5], [9]–[14] introduced two or more. At the high end of the spectrum are the open platforms, i.e. the ones that allow the development of new games, by third parties, or the creators themselves. However, no such initiative was found in the surveyed literature.

In the (b) collaboration dimension, the proposed platforms were classified into single or multiplayer experiences. Most works are single-player, except for [14] and [2].

The first [14] presented a platform for exergames that use the sensors for the players' smartphones to detect their movements. A server component running on a computer receives the actions identified by smartphones and forwards such information to the running game. The results indicated that simple motion detection based only on movement magnitude could engage the players, although it imposed restrictions on the game mechanics. The second work [2] proposed an

This work was partially supported by Fapemig and CEFET-MG.

978-1-6654-6156-6/22/$31.00 ©2022 IEEE

TABLE I: Characteristics of the exergaming platforms found in the reviewed literature

| Works | Players | Games | Hardware Requirements (subset) | Motion Sensing (subset) |
|---|---|---|---|---|
| [3]–[7] | single | single | VR kit, exercise bike, camera, computer | headset, exercise bike, kinect, smartband, camera |
| [8]–[13] | | multiple | custom wearable sensors, microcontrollers, smartphones, custom mat | smartbands, pressure |
| [2] | multi | single | smartphones, smartbands | smartbands |
| [14], ours | | multiple | computer, screen, smartphones | smartphones |

asynchronous multiplayer game called StepQuest. It requires a smartphone and a smartband for each player, exempting the need for computers. As it is an asynchronous game, players might become idle and, in that case, the authors noticed that replacing players with higher-performance bots engaged more, highlighting the importance of active competition.

In addition, although still a single-player exergame, in [3], the player can compete with his past and future performances – defined as an extrapolation of the previous and current proficiency. Compared to playing without any competition in focus groups, the players that competed manifested higher engagement.

The (c) hardware setup dimension describes the effort required to prepare the environment for exergaming sessions for either the host (e.g., gym), the player, or both. Systems that require custom hardware, such as in MouvMat [10] (custom-built mat grid with a microcontroller, projector, and computer) [8]–[10], [12] were considered as having a high setup cost. Platforms that involve placement of cameras [4], [7], VR kits [3], [5], [6], or usage of console accessories or computers [4], [11], [14] are deemed medium effort. And those that only require common gear that players might already be using, such as smartphones/bands are considered low effort [2].

Motion sensing is a crucial requirement for any exergaming initiative, and movements can be detected in vision (e.g., camera), inertia (e.g., accelerometers), pressure (e.g., balance boards, mats), or power-based sensors (e.g., pedal). Among other characteristics, the used sensors impact the (d) acquisition dimension, where the hardware requirements for the reviewed platforms, their cost, and ease of access were analyzed. Hard to acquire platforms are the ones that use special-purpose or adapted hardware, such as some exercise bikes or treadmills [3], [5], [8]–[10], [13] usually for each user. For example, [3] requires a Lode Excalibur Sportbike (25,000 USD), and an HTC Vive HMD (800 USD) for each player. The use of VR kits [4], [6], [7], smartbands [2], and gaming consoles or accessories (Wii Remote and Balance Board [11], Kinect [4]) might also require the acquisition of entertainment hardware, but usually at a lower cost than the previously mentioned works. On the other end of this spectrum are platforms that employ only general-purpose hardware [14], such as computers and smartphones. In this case, no acquisition is necessary if hosts and players already possess such equipment.

Move2Play is a new and unique exergame platform that contemplates desirable characteristics from all analyzed dimensions. First, it is an open platform that allows new games to be developed. Such games can be either single or multiplayer. For setup, the host requires a computer connected to a screen, but each user only needs a smartphone. Last, as the platform uses commodity hardware, the acquisition is simple, or the required parts can even be already available.

## III. Platform Components

In this section, the developed platform components are presented: the Console, the Controller, and the Example Games.

### A. Console

Besides managing the state, through the Console's UI, an administrative user can change settings such as the match duration of a game, the Console name that appears for Controllers to connect, and lower-level configuration such as the ports through which Controllers can find and connect to the Console.

The home screen (Fig. 1a) shows the available game library and allows the user to install new games by providing an installable ZIP archive, whose format is described in the Move2Play documentation[1].

When a Game is running, the screen splits into two parts and shows information regarding the active game at the top. When the match is over, the screen goes back to show only the game library, and the console resumes accepting a request to open a new game waiting room.

### B. Controller

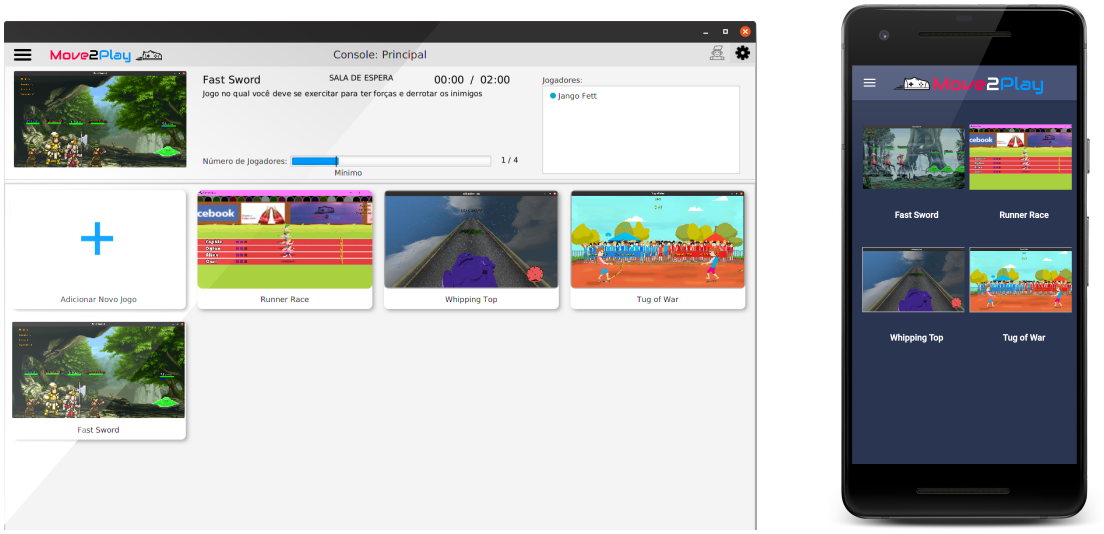
The Move2Play Controller was developed as a native Android application and is responsible for allowing players to browse, join and play Games on the Console, by detecting players' movements during matches.

When open, the Controller shows a list of Consoles discovered in the network. Upon connecting, the home screen (Fig. 1b) displays the Console game library, and, if there is an open waiting room or a game running, the top part shows such information allowing the user to join. If there is no active game, the user can select one to open a waiting room.

When there are enough ready players, the Console starts the match, and all Controllers instruct players to lock the screen and affix the smartphone as the Game is about to start. After a 10s countdown, the smartphone vibrates, and the Controllers start detecting movements.

When the match is over, the Controller shows a results screen indicating the total number of movements detected for the player and his classification among the other participants in the match. Then, the user is redirected back to the home screen, where a new Game can be picked up to be played. At

[1]Installable docs: https://move2play.com.br/docs/game-installable/




(a) Console home screen with an active game

(b) Controller home screen

Fig. 1: Move2Play Console and Controller main screens


(a) Whipping Top (b) Tug of War (c) Fast Sword (d) Runner Race

Fig. 2: The four example Games created for the the platform

any time during a match, a player might leave the game (or waiting room) and go back to the main screen.

### C. Example Games

Along with the Console and Controller app, four example Games (shown in Fig. 2) were created: Whipping Top, Tug of War, Fast Sword, and Runner Race. Each one has different characteristics, sporting a different play style and bringing inspiration from distinct game genres.

In Whipping Top (Fig. 2a), the player is represented by a spinning top gliding over an infinite track. Every movement the player performs (e.g., a step, a pedal, or an elliptical cycle) spins the top a little bit, propelling it further. Some parts of the track require players to slow down momentarily to keep the sliding momentum (e.g., when sliding through traffic cones), while others encourage players to stretch their performance and speed up to acquire a speed boost for the top (e.g., a speed boost track), following a High-Intensity Interval Training (HIIT) protocol [15]. The game can be played from 1 up to 4 players competing for who moves their top further.

Tug of War (Fig. 2b) puts players in a classic tug-of-war match, in which each player belongs to one of two teams pulling on opposite ends of a rope. The moves performed by each player are summed up to their team's movements, and in every interval of 3 seconds, the team with the most movements during that period effectively pulls the rope with the other team a little bit. When a team manages to have a pull balance of 5, it scores a point. When the console determines the end of the match, the team with the highest score wins. A match requires at least 2 players and can host up to 16.

Driving inspiration from RPG games of the 32-bit era, Fast Sword (Fig. 2c) is a battle game in which a group of heroes must sequentially defeat parties of enemies. To make their hero attack, a player needs to keep moving to fill an energy bar that, when complete, triggers an attack on one of the enemies. Enemies attack at regular intervals, and if a hero dies, the player's movements can speed up their resurrection. There are three classes of heroes: gunner, warrior, and healer, and each player receives a random one per match. There are infinite levels, and enemies get progressively harder to kill. This game can be played by 1 to 4 players in cooperation, and its goal is to keep advancing until the match is over.

Lastly, Runner Race (Fig. 2d) puts the player in control of an obstacle course runner in which their real-world moves allow their runner to accumulate the energy required for jumping the obstacles. An energy bar is progressively filled with each detected movement. When it is full, the runner is able to jump, but if an obstacle is reached and the runner lacks energy, they fall to the ground, and it takes some time to get up and resume the course. Runner Race can be played by 1 to 8 players.

## IV. Platform Architecture

Four principles drove the platform design: (a) support for multiplayer interaction; (b) use of commodity hardware, which is typically already available at gyms and schools, such as the user's smartphone, a desktop computer, TVs, or video projectors; (c) openness to the development of new exergames by third parties, abstracting away the motion detection mechanism; (d) freedom in how exergames use the players' movements and in how players interact, being in cooperation, competition or in single-player experiences.

The architecture adopted is similar to the one used in [16]. In their work, the authors envision the organization of a platform that allows using smartphones as interfaces for computer games. It proposes using a controller app – for smartphones, computer games, and a mediator software responsible for handling the platform communication. Based on similar principles, our platform comprises three software components: (i) the (virtual) Console, (ii) the Controller, and (iii) the Games. Fig. 3 depicts an overview of the architecture.

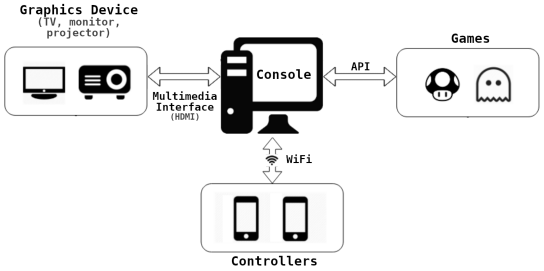



Fig. 3: Overview of the Move2Play architecture

The Console is the core piece of the platform and connects the other components. It runs on a regular computer. During a gaming session, the Console launches one Game of its library in an auxiliary process with its graphics and sound sent to a secondary output device such as a TV, monitor, or projector. Meanwhile, the primary screen keeps showing the Console screen. Each player bears their smartphone on their arm, leg, or pocket, which, working as the Controller, detects their movements and sends the data to the Console. In turn, the Console forwards the movement information to the active Game to be used as the inputs.

The Controller is a smartphone app that acts as both a motion detector and a user interface (UI) for players to browse installed games in the Console and choose one to play. The Controller sends the user interactions to the Console, such as when a user connects to it, joins a game, or makes a movement.

The Console is primarily responsible for keeping the platform state. It allows users of the Controller app to connect and maintains a list of connected users who might be currently playing a game or not, possibly waiting to join the next game session. When users connect to the Console through the app, they receive the list of installed games. A user can join the match if there is one running and room for one more player. If no match is being played, the user can pick a game and open a waiting room for it.

When the first player opens a waiting room for a game, the Console boots it up on the graphics device (e.g., TV), which shows the names of the players who have joined. Each game can have a minimum and a maximum number of players. When enough users in a waiting room declare to be ready to start through the Controller, the Console starts the match.

### A. Communication Protocol

The Console and Controllers communicate through WiFi. Hence, an application-level network protocol [17] was created through which both components exchange information and their state. The protocol[2] is loosely based on a request-response model in which the Controller (🎮) sends request messages to the Console (📱), which responds with a success/error status and some optional payload. The contents of the messages are text encoded in JSON. There are 10 different types of requests, divided into three groups: (i) Console discovery and connection, (ii) user interaction, (iii) and state management. Table II presents all the messages.

TABLE II: Messages from the network protocol

| Group | Message | From | To | Requires Response |
|---|---|---|---|---|
| i | `findConsole` | 🎮 | 📱 | ✓ |
| | `connectConsole` | 🎮 | 📱 | ✓ |
| ii | `gameInfo` | 🎮 | 📱 | ✓ |
| | `chooseGame` | 🎮 | 📱 | ✓ |
| | `ready` | 🎮 | 📱 | ✓ |
| | `move` | 🎮 | 📱 | |
| | `leaveGame` | 🎮 | 📱 | ✓ |
| iii | `activeGame` | 📱 | 🎮 | |
| | `finishingGame` | 📱 | 🎮 | |
| | `heartbeat` | 📱 | 🎮 | ✓ |

When the Controller boots up, it sends a `findConsole` broadcast to the local network. Any Console that receives it replies with information about its name, IP address, and port. The Console name is important as there might be more than one Console in the same network, and it allows users to know which one they want to connect to.

To connect to the chosen Console, the Controller sends a `connectConsole` message specifying the player's universally unique identifier (UUID), name, and identifying color. Upon receiving the request, the Console establishes a TCP connection and responds with a success/error flag, information about the available games, and the running game or waiting room. The `gameInfo` message requests detailed information about a particular game, and the response contains, for example, a description, the number of players, and game images.

When a user chooses a game to play, the Controller sends a `chooseGame` message and, depending on the Console state, it might open a waiting room or just add the player to the currently active one. When a player informs to be ready, the Controller sends a `ready` message, and, if there are enough ready players, the Console starts the match. When

[2]Protocol documentation: https://move2play.com.br/docs/network-protocol

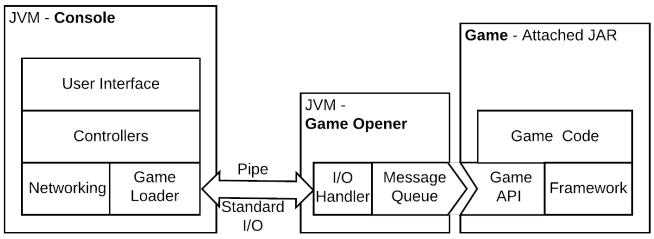



Fig. 4: Connection between Console and Game

the Controller detects a player's movement, it sends a `move` message to the Console, and to avoid network congestion, the Console does not respond in this case. At any time, if a player leaves, the Controller sends a `leaveGame` request.

To communicate its state to the connected Controllers, the Console might send `activeGame` and `finishingGame` messages, respectively, indicating the status of the current game and that a match has ended. Neither messages require responses from the Controllers. Lastly, at regular intervals, the Console might send a `heartbeat` request to both announce that it is active and to probe the connection with the Controller. The protocol documentation[2] further describes the formats and provides more information regarding each of those messages.

The communication between the Console and Game is unidirectional, from the first to the latter. The interface comprises an API that the Game must implement. At this time, the game has to run on the Java Virtual Machine (JVM) and implement the `Move2PlayGame` interface which is detailed later.

For the platform to be extensible to new games, the Console and Game are separate and complete applications. To do so, a small software called *Game Opener* acts as a bridge between both components. Instead of the Console booting up the Game, it creates a process for the *Game Opener* which, in turn, attaches itself to the Game Java Archive (JAR). The console acquires process pipes, i.e. a form of Interprocess Communication (IPC), to the *Game Opener* and writes messages corresponding to the remote procedure calls, i.e. a call to a software function located in another binary, it wants the Game to execute. The *Game Opener* reads its standard input to fetch the messages from the console, parses and adds them to a queue. The enqueued messages are forwarded to the Game through API calls. Even though they are isolated components from a software perspective, the *Game Opener* and the Console can be considered a single component in a broader context.

As the *Game Opener* attaches the game JAR in its process (Fig. 4), it communicates with the game through regular method calls. Such a contract is described by the Java interface `Move2PlayGame` (described later), comprising methods that instruct the game to open a waiting room, start a match, and finish it, among others. Lastly, this model can be extended to allow games developed in different languages, running on non-Java environments, by having other *Game Openers* written for each desired runtime, such as Unity or Lua.

### B. Motion Detection

The Controller app detects each player's movements. By holding their smartphone either in their hand, armlet or in their pockets, the application uses the device's sensors to detect movements through accelerometers. The Move2Play platform was designed to support several games while abstracting away the motion detection task and using a single commodity sensor. Therefore, the algorithm that makes the detection must be efficient, precise and versatile to different movements, making it a crucial part of the platform.

In a previous work [18], we evaluated the Google Pedometer algorithm [19] and the Android Step Detector [20], assessing their suitability to identify movements during physical activity based on acceleration magnitude. While the former is based on accelerometer sensors present in all Android smartphones, the latter is a software sensor available in 98.1% devices [20]. Tests revealed neither approach could detect movements with high accuracy, precision, and recall (above 95%). In addition, they were unable to detect more subtle, lower-speed movements, such as when users were doing sit-ups. Therefore, we proposed the Castor algorithm, which uses an adaptive speed threshold value above which movements were detected. Comparative tests conducted with Castor and Google revealed slightly better results for accuracy and precision, but Castor performed significantly better in terms of recall, as it could detect movements in exercise bikes, treadmills, and elliptical, like Google's, but also sit-ups, which Google's could not.

In addition to slow and fast movements, Castor was also able to detect accelerated movements, which Google's algorithm could not. When tested in different hardware, Castor outperformed Google as well, as it was able to detect movements even on devices that presented error accumulation from the accelerometer's sensors. Due to the better results, robustness, and versatility this algorithm was implemented.

## V. Game Creation

A common gameplay pattern is to make the number and speed of the detected movements directly influence the player's avatar in the game. For example, in a running game, the faster the player exercises, the quicker their avatar runs. Another pattern is to introduce quick challenges in which players need to change their exercise speed for a short period. For example, in a space shooting game, to unleash a special power, the player has to abruptly increase their activity frequency for 2 seconds. This pattern might not be suitable for treadmills, though, as quickly changing the velocity can not be safely done. In another pattern, the match can be split into micro rounds (e.g., 5s) for games with competitive teams. Inside each round, the team whose players exercise the most scores one point. At the end of the match, the team with the best score wins. Another pattern is to let them accumulate energy to do some activity, such as attacking an enemy or avoiding an obstacle.

Currently, all games must run on the JVM and implement an interface called `Move2PlayGame` from our Game API (see Fig. 5). The methods `initGame()` and `closeGame()`

are called respectively when the Console boots up the Game and when it wants the Game to close itself. After it finishes loading, the Game should enter the `WAITING_ROOM` state. At that time, the Console might issue `addPlayer(...)` or `removePlayer(...)`. When enough players are in the waiting room and declared to be ready from their Controller apps, the Console calls `startMatch()` and, for each movement it receives from a Controller, it executes `move(...)`. When the match duration is over, the Console tells the Game to show the `MATCH_RESULTS` for some time by calling `finishMatch()`. At any time, the Console might ask the Game what is its current state with `getState()`.

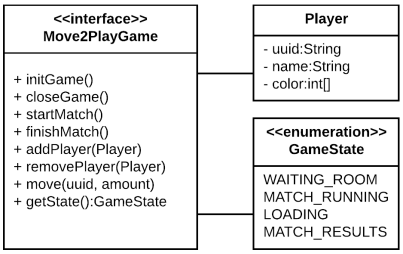



Fig. 5: Components of the Game API implemented by games

The `game-executable.jar` is the self-contained game executable comprising its assets, example screenshots, compiled code, all of its dependencies, and a `game-settings.json` file, which describes the Game, providing some metadata. In addition, it must be distributed with an installable ZIP archive. The installable format, details regarding the `game-settings.json` file and more information are described on the Move2Play documentation[3].

## VI. Final Remarks

This work presented the design and architecture of an open multiplayer exergaming platform. By being wholly constructed as software running on commodity hardware, it broadens the range of potential users by not imposing acquisition barriers. Move2Play is the first platform to operate with commodity hardware with a high precision detection algorithm, a management Console and an open architecture that allows third party development, therefore fostering the practice of exercises through exergames and serving as testing grounds for new research in exergaming.

Complementarily, the platform architecture, as well as the involved technology, can be explored in other studies. Other initiatives related to exergaming can be implemented and evaluated using the proposed architecture, protocol, and algorithms. Besides, this work contributes to the design, evaluation, and propagation of exergames, as it offers the possibility to include new games developed by third parties.

As the next step of this work, we plan to conduct user evaluations inside gyms, assessing the whole user experience and the user engagement with the platform, and the promoted physical activity. Also another future step is to investigate how games can encourage different training styles and goals.

[3]Game documentation: https://move2play.com.br/docs/game-description/

## References


[1] "Análise qualitativa dos motivos de adesão e desistência da musculação em academias," *Revista Brasileira de Ciências do Esporte*, vol. 38, no. 3, pp. 267 – 274, 2016.

[2] J. Zhu, Y. Feng, A. Furqan, R. C. Gray, T. Day, J. Nebolsky, and K. Caro, "Towards extending social exergame engagement with agents," in *Companion of the 2018 ACM Conference on Computer Supported Cooperative Work and Social Computing*, ser. CSCW '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 349–352.

[3] A. Michael and C. Lutteroth, "Race yourselves: A longitudinal exploration of self-competition between past, present, and future performances in a vr exergame," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ser. CHI '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1–17.

[4] C. Ioannou, P. Archard, E. O'Neill, and C. Lutteroth, "Virtual performance augmentation in an immersive jump and run exergame," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1–15.

[5] Y. Wang, K. Ijaz, D. Yuan, and R. A. Calvo, "VR-Rides: An object-oriented application framework for immersive virtual reality exergames," *Software: Practice and Experience*, vol. 50, no. 7, pp. 1305–1324, 2020.

[6] N. Farič, L. Smith, A. Hon, H. W. Potts, K. Newby, A. Steptoe, and A. Fisher, "A virtual reality exergame to engage adolescents in physical activity: Mixed methods study describing the formative intervention development process," *Journal of medical Internet research*, 2021.

[7] J. T. de Souza, C. Valentini, E. L. M. Naves, and E. A. Lamounier Jr, "A virtual reality exergame with a low-cost 3d motion tracking for at-home post-stroke rehabilitation," in *Anais do XXIII Simpósio de Realidade Virtual e Aumentada*. SBC, 2021, pp. 171–175.

[8] B. S. Oliveira, S. Nesteriuk, and F. Moreno, "Exergames usando tecnologia do movimento maker: protótipo de jogo tipo plataforma," in *Proceedings of the XVI Brazilian Symposium on Computer Games and Digital Entertainment - Art and Design Track*, 2017.

[9] A. J. Moshayedi, S. K. Sambo, and A. Kolahdooz, "Design and development of cost-effective exergames for activity incrementation," in *2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*. IEEE, 2022, pp. 133–137.

[10] C. H. Chu, R. K. Biss, L. Cooper, A. M. L. Quan, and H. Matulis, "Exergaming platform for older adults residing in long-term care homes: user-centered design, development, and usability study," *JMIR serious games*, vol. 9, no. 1, p. e22370, 2021.

[11] E. I. Konstantinidis, P. D. Bamidis, A. Billis, P. Kartsidis, D. Petsani, and S. G. Papageorgiou, "Physical training in-game metrics for cognitive assessment: evidence from extended trials with the fitforall exergaming platform," *Sensors*, vol. 21, no. 17, p. 5756, 2021.

[12] A. Pardos, A. Menychtas, and I. Maglogiannis, "On unifying deep learning and edge computing for human motion analysis in exergames development," *Neural Computing and Applications*, pp. 951–967, 2022.

[13] V. Guimarães, A. Pereira, E. Oliveira, A. Carvalho, and R. Peixoto, "Design and evaluation of an exergame for motor-cognitive training and fall prevention in older adults," in *Proceedings of the 4th EAI International Conference on Smart Objects and Technologies for Social Good*, ser. Goodtechs '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 202–207.

[14] K. Kiili and S. Merilampi, "Developing engaging exergames with simple motion detection," in *Proceedings of the 14th International Academic MindTrek Conference: Envisioning Future Media Environments*, ser. MindTrek '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 103–110.

[15] P. B. Laursen and D. G. Jenkins, "The scientific basis for high-intensity interval training," *Sports medicine*, vol. 32, no. 1, pp. 53–73, 2002.

[16] T. F. Costa, I. S. Silva, G. A. R. Barbosa, and F. R. S. Coutinho, "An architecture for using smartphones as interfaces for computer games," *Proceedings of SBGames*, pp. 611–614, 2018.

[17] A. Tanenbaum and D. Wetherall, *Computer Networks*. Pearson, 2010.

[18] C. Andrade, I. Silva, G. Barbosa, and F. Coutinho, "Real-time motion detection for android smartphones," in *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, 2019.

[19] Google. Simple pedometer. (2019). [Online]. Available: https://github.com/google/simple-pedometer

[20] Android. Motion sensors. (2020). [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_motion