Politecnico di Torino

Master's degree in Computer Engineering

Study on the use of social networks for the creation of gaming experiences

Candidate
Klajdi Myftari

Supervisor
Prof. Andrea Sanna

ACCADEMIC YEAR 2020/2021

Acknowledgements

I would like to thank Xhoi Kerbizi for his help in the thesis and his good advices, Alejandra Molina for the moral motivation given during these challenging years. And I want also to thank my sister: Anxhela Myftari, for believing in me and urging me to keep on, and I would like especially to say thank you to my parents, who made many sacrifices to give me the chance to reach this important milestone of my life.

Summary

This thesis is dedicated to the study, design and subsequent development of a Massive Multiplayer Online prototype videogame, that has the purpose to integrate and replicate the most famous social media platform existing: Facebook. But converted in a 3D and playful counterpart, expanded and enhanced with new contents and activities. This work has been commissioned by Orbyta Tech company: one of the branch companies of Orbyta that deals with the design, implementation, delivery, integration and application maintenance of software, hardware and IOT systems. The purpose of the thesis is the interest in developing a metaverse application that stands out thanks to the strong relation with social media.

The main requirements of the application are: login through the social platform, with the account; generation of a semi-procedural village; exploration of the world; minigames and gamification of the social features. The application aims to let users, from all over the world and from different ages, connect into a centralized digital town and spend joyful moments but also interact each other, mixing social media features. There was a research phase involving the quest for the best social platform fitting with the application and at the end it came out to be Facebook. Firstly there was a research on the importance of the Metaverse and, at the same time, there was a study on the most used social platforms to decide which one to integrate to the game. Activate Consulting forecasts that during the next 4 years gaming activity will grow the time spent with technology and media. Every ordinary activity (e. g. social, search, live events, shopping) will little by little take place inside of gaming environments going through a process of gamification. Besides, the 5G is near and it will highly enhance bandwidth speeds that will be determining to expand the use of digital technologies such as cloud gaming and it will favorize the consolidation of the Metaverse. One of the most played mobile online games is The Sims Freeplay: a life-simulation game that is visited monthly by 1 million users and that has a high average score on the app stores too. On Facebook Games Top Charts, instead, there is a Role Playing Game named Worlds FRVR. It is increasing the audience because players can create and share new worlds and events, like parkour races, but they can also make new friends or invite existing Facebook friends into their session. Another great and growing use of Facebook, except socializing, is to see the video game lives of your favorite streamers on the "Facebook Gaming" streaming and cloud gaming service. The user base is ever-growing: more than 800 million people use it every month to play online games and 90 million of them are active members of the Facebook Gaming Groups. The "Group" section lets users and creators make topics into discussions and there is also the "Looking for Players" section to help members of the community find other people among the community to play with. Facebook has earned customer trust by offering them many free-to-play (F2P) games like Candy Crush but it also invested on streamers and influencers since they are the main reason for approaching a huge slice of users towards the application. Facebook Gaming is accessible only with a Facebook account. Metaverse is the definition of a virtual world that people can join with a 3D avatar to create and explore new environments, socialize and play with friends, and this is only the beginning. The Metaverse will allow more people in the future to

embrace digital creativity in their lives. It is leading to the drop of hardware barrier required combined with increased bandwidth, so this means that digital creativity and play will become more accessible to an always increasing amount of worldwide public. There are many titles nowadays concentrated on the metaverse concept that host millions of users daily. Crayta lets you create new maps and games, like Roblox and RecRoom. But there is also Breakroom, that lets employers create the 3D replica of their office to make employees work in an innovative way, or IMVU that allows players to sell digital furniture and earn cryptocurrencies. Facebook bought Oculus Virtual Reality company because it wants to push towards the metaverse too and now Oculus users have to create a Facebook account to use the devices. Facebook's social platform boasts a number of almost 2. 8 billion monthly active users, and that is normal, due to its age in the market. The users can be found at all ages, 72. 8% are within the 18–44 years old range. Besides, it was found that 2. 79 billion people globally use Facebook. 57% of U. S. social media users use Facebook to share content. In a 2019 Statista report it was found that 65% of Facebook users view photos on the social platform, 46% watch videos and 43% share content via direct messages. But there are also 1.4 billion people who use Facebook Groups on a monthly basis.

The tools used for the research of the best social platform to fit in the game were Visual Studio 2019 Community and the packages it could host for that purpose. The possible candidates were Facebook, Instagram, YouTube, Twitter and TikTok. Some of the social platforms needed to join a developers website to grant permissions to extract the data. The research found Facebook as the ideal candidate since it offered a larger variety of content to scrape but also provided useful tools for the purpose. The prototype was developed inside Unity3D engine, with the C# language. Unity is one of the biggest game engines to create indie and AAA games. Facebook makes games creation with Unity easier thanks to the Facebook for Unity Software Development Kit, to exploit its functionalities to get and treat the information of users with standard methods provided by the social. For the online component came in help Mirror asset for Unity to sustain a large number of connections. Mirror uses low-level transport protocols like TCP and UDP to manage players connections and client-server communications via C#. The Design of the application starts with the creation of the Design Document, that is common to every game development process. It states how the different elements of the game will behave and how they will look like. The player will join the game through a login screen with Facebook and then his character will appear in the world and will be able to wander around and interact with the other users, the square mini-games and the houses. The latter will have a Facebook Feed to look at. He can interact with it by just scrolling it or by checking comments and reactions and also by watching videos. A Use Case Diagram and a Storyboard were provided to show all of the actions users can make and a gameplay sequence. The document gathers also ideas to implement in the future, like: house and character customization, atmospheric events and connections to different social networks, a main menu and an in-game menu, opportunity to take photos with the avatar and watch them in a dedicated photo gallery in the menu, invite friends to join the session through Facebook messages, add as a Facebook friend a user met in town, celebration of events and more. All of the work was developed on a machine running Windows 10 and tested on Android and PC devices, but the main target platforms to deploy the final product were Mobile and PC. For the mobile version were designed User Interface controls on the screen to move the player and the camera but also special buttons to trigger temporary actions like opening the door, watching the feed or jumping. Players can interact with the other users by means of a global online chat or by playing mini-games and making the best score, that will be updated on a leaderboard. Users have always a pet that will follow them around and, in the future they are going to be customizable. One critical point is that the company has to sign a stable partnership with Facebook when releasing the game since the policy of the info permission is sensitive to change. The application development had a first part related to data scraping to understand which platform gave more permissions and user content to manage. Then it continued with the semi-procedural environment design to create the square and the houses of the village. Afterwards it included the Facebook integration to grant the permissions and download the social media content. So, in a first part it had to be created on the Facebook Developers portal a section containing important info of the app and then it was bound with Unity thanks to the SDK imported in the engine. Next it dealt with the adaptation and gamification of the downloaded content and finally, there was the execution of a server to host the clients connecting from everywhere. Players could recognize each other thanks to the Facebook account name on their head, that pointed towards the camera of the local player. The camera script was able to make the player visible also in narrow spaces by coming forward. The chat used special events to delegate the server to send the message to the listening clients thanks to the help of Mirror's attributes. The same was made with the score update on the leaderboard. The animations were downloaded with the assets and then they were arranged in the Unity Animator tool and, thanks to the use of the C# script, it was possible to trigger the transitions. A Device Simulator tool in Unity tested the adaptation on different screens of the UI elements to see how they behaved on different resolutions. The analysis was done with the Profiler tool of Unity and it could test both PC and mobile versions by plugging the latter, with a USB cable, to the machine. Both of the platforms kept stable RAM values but the CPU usage on the PC kept a constant 30fps rate, and dropped to 15fps in rare occasions. Instead the mobile stayed low on 15fps mostly because of the scene rendering. The game tried to represent the counterpart of Facebook in almost every aspect: the Home section and some elements of Facebook Gaming were transposed in the square. The houses made the friends' and user's profile section and the wandering around these parts, with avatar was the "translation" of the navigation system. When the player interacts with the feed there is a camera switch to view the posts better. The Posts on the Feed want to resemble as much as possible the direct counterpart of Facebook too. So, they show the name and the profile picture of the publisher, the description and the media, the date and the place of publishing but also the reactions and comments. About the builds, the Facebook SDK for Unity package had a built-in login flow only for Android and iOS devices, so the PC build needed a custom login flow and this postponed it for a future release. The game wants to look low poly for hardware limits but also for aesthetic reasons. The game followed a trial with 12 users. The users were delegated as testers, since the app mode in the portal was in development mode, and it was necessary to let the it gain their Facebook account permissions. The users were given a subjective evaluation test to understand their social media interests and the opinions they had about the game after the trial. Their age matched demographics of worldwide Facebook users as closely as possible. The results said that they think that social feature is very important for the app and that they see Facebook as an ideal candidate for it, together with Instagram. About the activities they want to push on the interaction between players, mini-games and planning of events. They enjoyed playing the app but needed the help of a person to understand it at most. Some implemented features were consistent but others not so much. The interactions were quite easy but others a lot cumbersome. In conclusion, the prototype seems going towards the right way but there are still many changes to make, to add more features and optimize hardware consumption. And it is also very important to sign durable contracts between the company and the social media businesses to ensure the gathering of the user info, since this is the core feature of the app.

Contents

List of Figures

List of Tables

Source Code

Chapter 1

Introduction

1.1 Company presentation

This thesis is dedicated to the study, design and implementation of a social-related video game, commissioned by Orbyta. Orbyta is an Italian company that provides professional consultancy and innovative solutions ranging from the design and implementation of complex information systems to the management and governance of processes, from corporate compliance services to the management of systems and networking services. The company is divided in different branches, each one concerning with different problems:

- **Orbyta Strategy**: in particular, it provides administration, finance and control, accounting, talent acquisition, employer branding, training, commercial development and marketing services.
- **Orbyta Compliance**: it supports its customers by analyzing their compliance with current regulations, remedying non-conformities and providing a continuous monitoring service for the correct management of corporate compliance. In full ORBYTA style, their priority is customer satisfaction, with a dynamic, fast and efficient service, aimed at solving problems quickly and with maximum efficiency.
- **Orbyta Infogest**: ORBYTA Infogest is the company of the ORBYTA Group that deals with designing, supplying and reselling adequate HW and SW, installing and supporting PCs, servers, storage, internetworking in heterogeneous operating environments. ORBYTA Infogest is the IT partner that develops tailor-made infrastructures and offers consultancy with high added value. Infogest interprets its System Integrator mission by aggregating HW and SW technologies to design new processes that impact on the company organization, making it more modern and effective. Infogest makes use of consolidated partnerships with reference players on the market, such as HP, Microsoft, Fortinet, Vmware, Veeam, Arcserve, evaluating from time to time the best technologies to build safe and reliable solutions.
- **Orbyta People**: it offers consultancy on employment matters, human resource administration and management, payroll processing, benefits management, welfare and trade union relations. Orbyta People is the consulting firm specialized in 360 ° in the field of personnel management and administration. It offers consultancy on employment matters, human resource administration and management, payroll processing, benefits management, welfare and trade union relations.

- **Orbyta Tax & Finance**: specialized in the field of tax consultancy, assistance in tax and accounting compliance, corporate consultancy, company valuation and company shares. Orbyta Tax & Finance, was born from the ten-year experience of accountants and tax specialists specialized in Tax and Tax Advice. The company offers all accounting and consultancy services in corporate and tax matters to all types of companies (professionals, sole proprietorships, partnerships and corporations), including the establishment and all the formalities required by law, in addition to out-of-court and judicial assistance on a wide range of corporate and tax legal cases. The company also provides high value-added services such as assistance in Tax Litigation, assistance in setting up a firm or company, company mergers, company demergers, company sales, company rentals, company liquidations, lease agreements and management control corporate. The competence and professionalism of professionals and collaborators is available to solve problems not only in the corporate and business sphere but in all areas of business management. In full style of the group, they try to exploit all the possibilities that technological evolution makes available. By the services it offers there are:

  - **Tax consultancy**: proposal for a solution to tax problems that arises from the constant discussion with the customer about the economic aspects of the company and the sector. The Accountant represents an essential figure capable of guiding the entrepreneur in the complex world of taxation.
  - **Corporate consultancy**: all aspects that affect corporate life affect the final result and performance. Each of these aspects is important for managing the company with a 360-degree view of the business. The role of the accountant is to help the entrepreneur understand these aspects
  - **Corporate consultancy**: they talk about planning, management control, business plan drafting, budget analysis and cost analysis, financial analysis. All tools suitable for businesses of any size and the consultancy of the accountant must guide their application by designing the tools based on needs.
  - **Strategic consulting**: it is the competence of the accountant to provide the tools to take the right step, with greater attention in delicate and critical times such as today. The accountant is a partner of the client in business management.

- **Orbyta Engineering**: design and construction management in the field of civil and mechanical engineering in the civil and industrial sectors, for public or private clients. Orbyta Engineering was born with the aim of offering qualified services in the design and construction management in the field of civil and mechanical engineering in the civil and industrial sector, for public or private clients. In addition to the design from scratch, the company deals with energy diagnosis and certification of buildings, energy management and design of thermal engineering systems with particular attention to the design of renewable energy systems in accordance with current regulations. Orbyta Engineering provides consultancy for mechanical design in the industrial sector, as well as in temporary and mobile construction sites (Legislative Decree 81/08) and has specific instruments and skills for the measurement of physical and environmental parameters (such as acoustic impact) both in the industrial and environmental fields. Orbyta Engineering offers several services too:

  - **Plant engineering works**: design and construction management for the construction of thermal power plants and refrigeration units, distribution systems for civil and industrial heating and cooling, gas adduction and distribution systems, chimneys for the expulsion of burnt gases, water networks, sewage distribution systems , rainwater recovery systems. Design of renewable energy plants (biomass plants, solar plants, cogeneration plants, geothermal plants). Design of photovoltaic and electrical systems.
  - **Energy Efficiency**: energy Efficiency Technical-economic actibility of energy redevelopment interventions, executive design of energy improvement interventions both of the building envelope and of the electrical and mechanical systems with technical and administrative support for access to national or regional incentives.
  - **Fire prevention**: Orbyta Engineering are able to carry out fire prevention design also through the engineering approach with Fire Safety Engineering (FSE), which uses simulation calculation models of fire, smoke and heat, as required by the Fire Prevention Code. This methodology makes it possible to find alternative solutions with considerable savings in time and money resources, in particular for the fire resistance of structures, escape routes, etc.

    To complete the fire prevention services, the design of all fire-fighting systems is carried out: fire-fighting water systems, hydrants and hoses according to UNI 10779, UNI 11292 (rooms intended to house pumping units for fire-fighting systems), UNI / TS 11559 (dry hydrant networks);

    * fire detection and alarm systems according to the UNI 9795 standard;
    * smoke and heat evacuation systems according to the UNI 9494 standard;
    * sprinkler systems according to UNI EN12845 and UNI 11292, American NFPA standards;
    * EVAC systems - sound evacuation.

    The company makes use of the skills of numerous professionals to carry out multiple tasks and to satisfy the requests of diversified clients (both public and private), modulating its staff according to the actual needs and specific skills required for carrying out the tasks.

- **Orbyta Legal**: it is a company between lawyers that offers judicial and extra-judicial legal assistance and advice, even on an ongoing basis, with particular regard to business management and development, contracts, the protection of industrial and intellectual property rights, the protection of personal data and to the law of new technologies.

  Orbyta Legal offers legal advice and assistance in relation to business management and development, also with reference to new business projects born in the form of innovative startups. In addition to supporting companies in the management of external relations, offering consultancy in the drafting of contracts and in the negotiation of commercial agreements, Orbyta Legal also assists companies in internal management in relation to statutory changes and in the definition and execution of extraordinary transactions, such as mergers and acquisitions. The company also provides its Clients with assistance in debt collection, both out of court and in court, from the payment reminder up to the moment of forced execution against the debtor. Our company also provides high value-added services such as assistance in Tax Litigation, assistance in setting up a firm or company, company mergers, company demergers, company sales, company rentals, company liquidations, lease agreements and management control. corporate. The competence and professionalism of professionals and collaborators is available to solve problems not only in the corporate and business sphere but in all areas of business management. In full style of the group, we try to exploit all the possibilities that technological evolution makes available. Orbyta Legal too, brings a series of services:

- **Industrial and intellectual property law (ip)**: Orbyta Legal offers legal assistance with reference to the protection, both in extrajudicial and judicial areas, of industrial and intellectual property rights, relating to trademarks, patents for inventions, design, know-how, copyright, copyright. The consultancy activity also provides assistance for the registration of trademarks both nationally at the Italian Patent and Trademark Office (UIBM), and transnationally by filing with the competent Authorities, in particular the European Patent Office. for patents and the Community Office for Trademarks and Design (EUIPO), as well as the preparation and negotiation of contracts for the sale, license, distribution, franchising, merchandising, sponsorship, advertising and transfer of technologies. The company also provides its customers with assistance in the pre-litigation phase and in any litigation regarding nullity, forfeiture, counterfeiting, unfair competition and consequent compensation for damage.
- **Business crisis management**: Orbyta Legal offers legal assistance in business crises by identifying the tools for regulating the crisis and the solutions that best suit the characteristics of the business. Starting from an examination of the financial situation of the company, Orbyta Legal, with the support of the Tax & Finance Department of the Orbyta Group (Orbyta Tax & Finance), assists companies by preparing strategic plans that allow them to overcome the crisis situation and provides assistance in carrying out those actions that allow the company to overcome difficulties (debt restructuring agreements, tax settlements, recovery plans, settlement of the over-indebtedness crisis pursuant to Law 3/2012).
- **Administrative liability of bodies (legislative decree 231/2001) - ODV activities**: Orbyta Legal carries out consultancy activities, even on an ongoing basis, on the liability of Entities for administrative offenses depending on crimes committed by individuals in the interest or to the advantage of the Entities themselves (Legislative Decree 231/2001), assisting its Clients in all documentary and organizational-procedural activities necessary for improving corporate governance and consequently avoiding the risk of committing the aforementioned crimes. The company employs professionals, with proven experience in the field, who possess the skills and requirements of the Supervisory Body (the so-called "OdV") with the task of supervising the effectiveness of the procedures to oversee the risk areas and their effective implementation. , on the correct functioning and observance of the Model and verifying the correct application of the rules defined in relation to the risk profiles highlighted.
- **Digital law**: Orbyta Legal, also thanks to the strong synergy with the IT Department of the Orbyta Group (Orbyta Tech), dedicates itself daily to the examination of new technologies and in particular to the legal aspects

that they highlight and has therefore acquired the necessary skills to offer consultancy and assistance to its customers, even on an ongoing basis, in the field of new technologies and precisely in the field of e-commerce, cyber security, contracts in the field of ICT (supply of hardware and IT systems; software licenses; development and supply of software applications ; system assistance; supply and development of online IT and information services; website design and development), software protection, legal problems relating to the use of the web. The company also provides consultancy services on the subject of compliance with the PSD2 Directive (Payment Services Directive n.2), supporting the customer in the definition, identification and implementation of security solutions both from a documentary and organizational-procedural point of view, which guarantee the adherence to the reference legislation for the type of activity carried out.

- **Privacy and data protection**: Orbyta Legal carries out consultancy activities, even on an ongoing basis, on the protection of personal data (Legislative Decree 196/2003 and EU Regulation 2016/679), assisting its Clients in all documentary and organizational-procedural activities necessary for the adaptation of the company reality to the national and community legislation, currently in force. The company employs professionals, with proven experience in the field, who possess the skills and requisites required by law to carry out, in favor of private companies and public bodies, the role of DPO (Data Protection Officer or Head of Protection of Data). Orbyta Legal also supports the data controller in the preparation and updating of "terms and conditions" and the privacy policy of websites, social networks, prize competitions, mobile applications, etc.
- **Labor law**: Orbyta Legal assists companies in the management of labor law aspects both with reference to the drafting of subordinate, parasubordinate and agency employment contracts, to the preparation of internal company regulations, codes of ethics, confidentiality and non-competition agreements, negotiation of individual and collective agreements, both with reference to civil litigation regarding individual and collective dismissals, and with regard to accidents at work and occupational diseases.
- **Real Estate law and Real Estate**: Orbyta Legal offers advice and assistance in negotiations relating to real estate, such as sales, leasing, leases for residential or commercial use, company rentals, rent to buy, loan for use, and contracts for the construction of buildings or the provision of services. , as well as in civil proceedings and compulsory mediation concerning, for example, the purchase of a property for usucapione; defects in the works and damage from infiltrations; the guarantee for eviction; the possessory actions for reinstatement, maintenance, report of new work or report of feared damage; petitorial actions of claim, denial, regulation of borders and for affixing of terms; the action for the division of property in communion. With regard to real estate properties leased, both for residential use and for commercial, artisanal or industrial use, the company provides its legal assistance also in the matter of evictions for arrears or for termination of the lease as well as in the recovery of rents from the defaulting tenant.
- **Family law and inheritance successions**: Orbyta Legal offers advice and assistance in matters of family law and inheritance law, offering assistance in matters of separation and divorce, in the matter of determining

(or modifying) the conditions of custody and maintenance of minor children also with reference to unmarried parents, as well as in the definition of equity issues resulting from the family crisis, managing divisions and liquidations of assets, shareholdings and companies. Legal advice to the person is also offered in the matter of civil unions and cohabitation contracts, as well as in proceedings of interdiction, incapacitation and appointment of support administrator as well as in matters of inheritance rights and shares due to the heirs, in the judgments of ascertainment of the inheritance and hereditary division.

- **Orbyta Tech**: Orbyta Tech carries out highly complex projects with the most modern technologies and using the most innovative methodologies. They deal with the design, implementation, delivery, integration and application maintenance of software, hardware and IOT systems.

This project was involved with the Orbyta Tech branch. It deals with software development and engineering, integration activities and techniques related to information systems. It also provides specialist consultancy related to the management of IT services, process governance, project management, data extraction and analysis, functional analysis and in general consultancy services on business applications and processes.

1.2 The goal of this work

The goal of the thesis is the research and development of a prototype related to the creation of a centralized MMORPG game and to integrate it with a massive social platform where users can connect, play and share social content. In the current decade various applications like this have been published but the difference of this piece is the strong integration level with social platforms. Among the core functionalities there are: free roaming, social interaction, mini-games and friends invitations to join the app. These mechanics are common to many metaverse games. The application is open to all ages of users and it is also a way to thin the boundary between games and social platforms.

Since the application is strongly tied to social media, there are three categories of users: the main user, so the one who is using the application, his friends and the friends of his friends. The main user joins the application after he logs in with his social profile, through the portal where he is redirected. During the development of the prototype I regarded Facebook as the best choice to test due to its vast category of content and to the Facebook Unity SDK package, since I chose to use the Unity Engine for the development. After the login, a semi-procedural world is generated. It creates a town which includes a public square in the middle, visible to all players, and a grid of roads and houses around it. Each house belongs to a Facebook friend. Contrary to the square, each player has a different view on the grid structure, since it shows the 3D counterpart of his own Facebook's friend-list, where at first place is generated the player's house. Each house can be customized but it has the same core functionalities that resemble the features of every Facebook profile: feed, photos, videos, albums, etc... Every player met in the game is a Facebook user. You can interact with them by showing some content thanks to a floating window or by using a built-in chat system or, simply, by waving your avatar's hand. Moreover, players can pause the game, logout, or change audio and video options or rather exit the application. But the menu hosts also a photo gallery of the pictures taken in-game.

Concerning the core technology adopted to achieve the goal of this work, there are Facebook Unity SDK and Mirror Networking, both integrated to Unity Game Engine. In particular, the following frameworks, packages and SDKs have been used with Unity and Microsoft Visual Studio to test the collection of social data and only some of these have been chosen for the development of the prototype: Newtonsoft.Json, Google.Apis, Curl, .NET. A detailed study of these frameworks was necessary to achieve the thesis goal.

Except the introduction, the thesis is split into other 5 chapters, whose contents are briefly described as follows.

Chapter 2 describes the researches that made Facebook the subject for this prototype. Then, it contains also examples of the most popular metaverse applications and it shows how important are social features inside of them.

Next, there is Chapter 3 that explains the goal of the developed application, describing briefly the required functionalities and the technologies used to implement it. But it also explains why Mirror Networking is a fit solution for the multiplayer purpose of this project.

Chapter 4 is dedicated to the design of the developed application. The Design Document and the allowed functionalities for the user are introduced at first and then the main phases of the development are explained subsequently.

In Chapter 5 are listed the attended and actual results of the development. And finally, the thesis ends with the conclusions 6 and the outlook for the future work.

1.3 Project Requirements

As mentioned before, the development of an MMO application, strongly related to social media is an innovative solution to introduce into a newborn market, that brings people interactions inside a video game. This work wants to demonstrate how a renamed and consolidated social platform as Facebook can be translated into a gaming experience inside a 3D world, where users can be avatars and profiles houses. Before choosing Facebook it was fundamental to understand which were its limits against the other platforms. So, in the first phase it was necessary to understand what could be the best social platform to extract most of users info and the most valuable ones. To do so, it was compulsory to install software and APIs to test each of them. The main requirements and goals of this work are:

- **Login through social platform:** In case of Facebook, the player needs a Facebook account to login and the game starts after completing the login process;
- **Generation of a semi-procedural village:** Once the player logs in, the world generates using the data contained into his account.
- **Exploration:** The player is able to explore the world and enter the houses to visit his friends and to do activities.
- **Play minigames:** Inside the main square of the town there are different playful activities to start solo or in competitive mode.
- **Social Platform gamification:** The players are able to look at friends' social feed inside their homes, watch their albums and other published content.

Chapter 2

State of art

The following articles and reports show why metaverse development is important in this decade and why integrating it with a massive social platform like Facebook is an innovative way to stand out from the flourishing market. First of all, consulting companies reports prospect an increase in video game users and in Virtual Reality and Augmented Reality devices adoption. Besides, they claim that cloud gaming and gaming subscription will be more adopted due to better pricing and to internet improvement, thanks to the upcoming 5G technology.

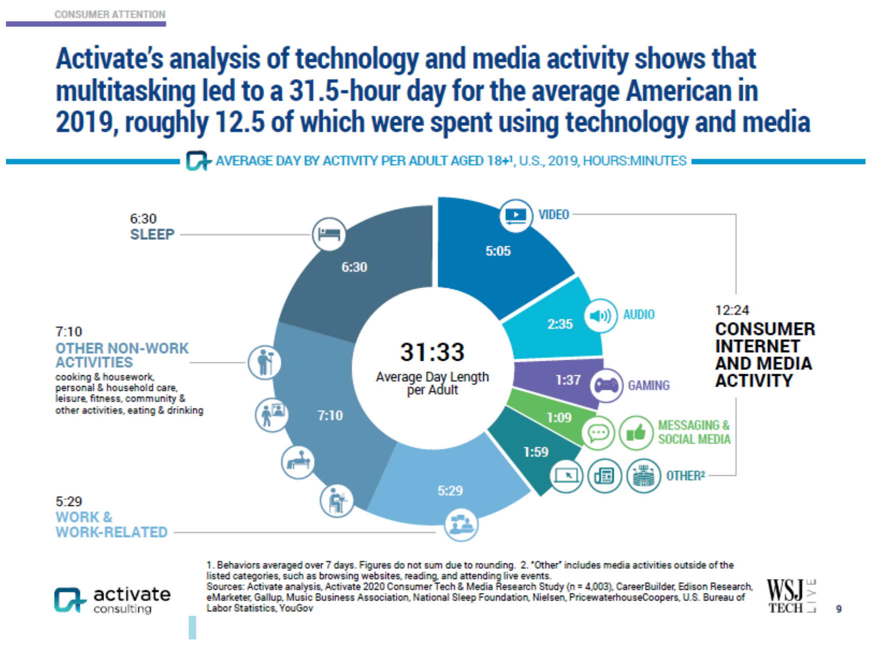
2.1 Activate Consulting 2021 outlook

Activate is one of the leading management consulting firms for technology, media, internet and entertainment industries. Its annual "Activate Tech & Media Outlook" forecasts the key trends and introduces the most important insights in the technology and media industries in the year ahead, in this case, year 2021. About this work, it is important to focus on three important topics in the report: **Video Games, Super Users, Connectivity**.

2.1.1 Video Games

Every ordinary activity (e.g. social, search, live events, shopping) will little by little take place inside of gaming environments going through a process of gamification. Most of the technology platforms will grow their presence in the gaming topic, bringing the paradigm of these activities to video games. Starting with media in general, Activate's analysis of technology and media activity shows that in U.S.A. during 2019 (see Fig. 2.1) the average people used to do multitasking for 31.5 hours a day and 12.5 of which were spent on technology and media. Talking about media, Activate forecasts that during the next 4 years gaming activity will grow the time spent with technology and media. As we can notice from the report (see Fig. 2.2), gaming activity occupies only 1 hour and 44 minutes of the total 13 hours and 13 minutes of multitasking but it is supposed to grow 3.4% in the next 4 years. Furthermore, Activate shows that the average daily internet media use for every adult (18+) in the U.S.A. will increase by 0.2% from 2020 to 2024 and, focusing on gaming, it will increase by 3.4%. Dealing with social media (see Fig. 2.3), Activate gathered the data from Desktop, mobile web and apps and reports that the average time spent with most famous social platforms has increased and TikTok's level of engagement is approaching that of Facebook (see pag. 15/148 on Activate's Outlook: [1]). Videogames are going to be a greater revenue source in the following 4 years. Indeed, if we look back at 2016, going through 2020 to 2024, gaming will lead to a big revenue, the most important platform to lead this growth will be mobiles (see Fig. 2.4). It's not a case that it's the mobiles. Top mobile games provide unique opportunities for players to connect with their friends and the gaming community. Indeed, video games are allowing the creation of a virtual shared space where every digital activity will take place. Gamers will join social activities and events through video games since they are going to extend beyond gameplay. E.g. players will be able to socially interact with other users like they do in the social platforms, or will watch live concerts using their avatar, like in Fortnite. It is going to become the central hub for people's digital lives and later, for their real lives. So they will be able to listen to music, watching tv and video, taking care of their lifestyle and practicing sports and e-sports. This can be mixed with social interactions evolving in celebrations, birthday parties and job meetings or private events. But players would also engage on creating an endearing location to show their skills to other players and friends. The same thing is already happening with their avatars. After analyzing the games market, Activate finds out that the most important social feature for 44% players is to play with existing friends. (see from pag. 31 to 36 on Activate's Outlook: [1]).

Players censorship shows that female players increased more than 66% and there are more than 56% new gamers aged 45 years or above. Another factor that increased video games users, more in particular, cloud gaming, is the COVID-19 pandemics. Since the beginning of shelter-in-place it accelerated gaming subscriptions. Gaming companies will concentrate on keeping the fidelity of these high-value players. Cloud gaming services success depends on access by the players to cloud and streaming technology, highly valued content and favorable pricing models. Another sector that increased adoption due to the lockdown is the AR and VR field. Indeed, 56% of VR headset purchasers bought a headset during the COVID-19 outbreak while 22% of VR headset purchasers bought their first headset during the COVID-19 outbreak. The reason is also given by the fact that VR has replaced a vast set of physical experiences and has become a social platform too, other than video games. Activate forecasts that this way the Augmented Reality (AR) / Virtual Reality (VR) market will reach roughly $20B by 2024 (see pages 38-40-42-43-44-45-75-76-77 on Activate's Outlook: [1]).



1. Behaviors averaged over 7 days. Figures do not sum due to rounding. 2. "Other" includes media activities outside of the listed categories, such as browsing websites, reading, and attending live events.
Sources: Activate analysis, Activate 2020 Consumer Tech & Media Research Study (n = 4,003), CareerBuilder, Edison Research, eMarketer, Gallup, Music Business Association, National Sleep Foundation, Nielsen, PricewaterhouseCoopers, U.S. Bureau of Labor Statistics, YouGov

Figure 2.1: Activate's analysis of technology and media activity (Ref. [1])

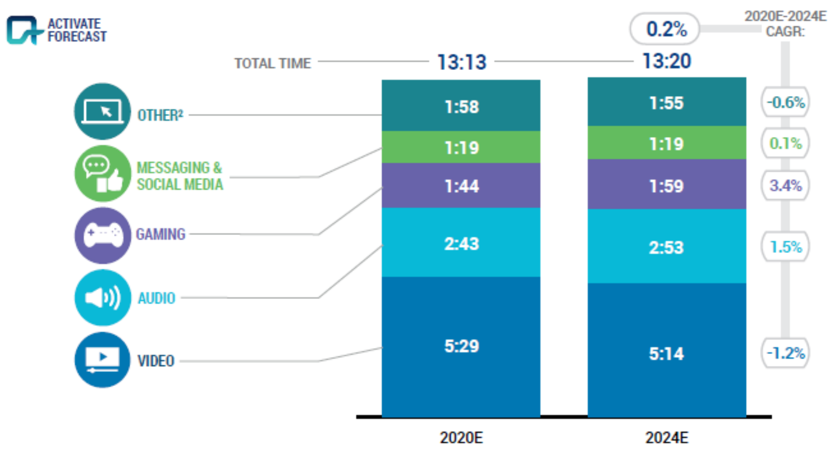


Figure 2.2: Average Daily Internet and Media attention per adult aged 18+1, U.S., 2020E Vs. 2024E, Hours:Minutes (Ref. [1])

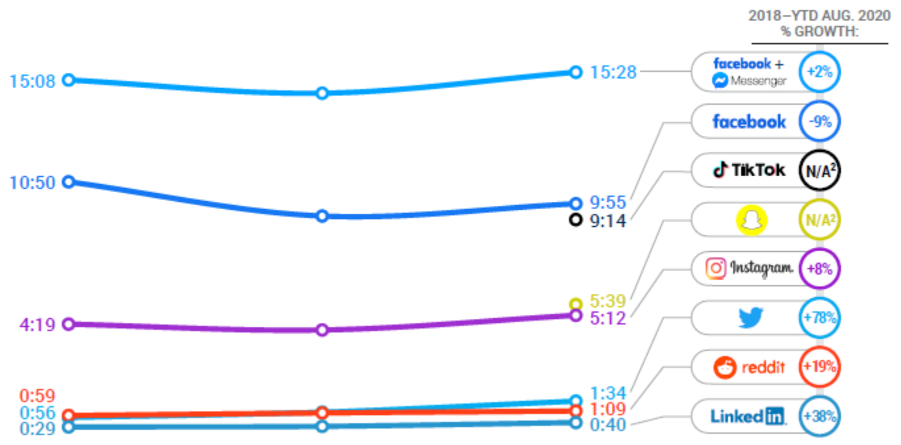


Figure 2.3: Average Monthly Time spent per user' on Desktop, Mobile, Web, and App by social platform, U.S., 2018-YTD AUG. 2020, HOURS:MINUTES (Ref. [1])

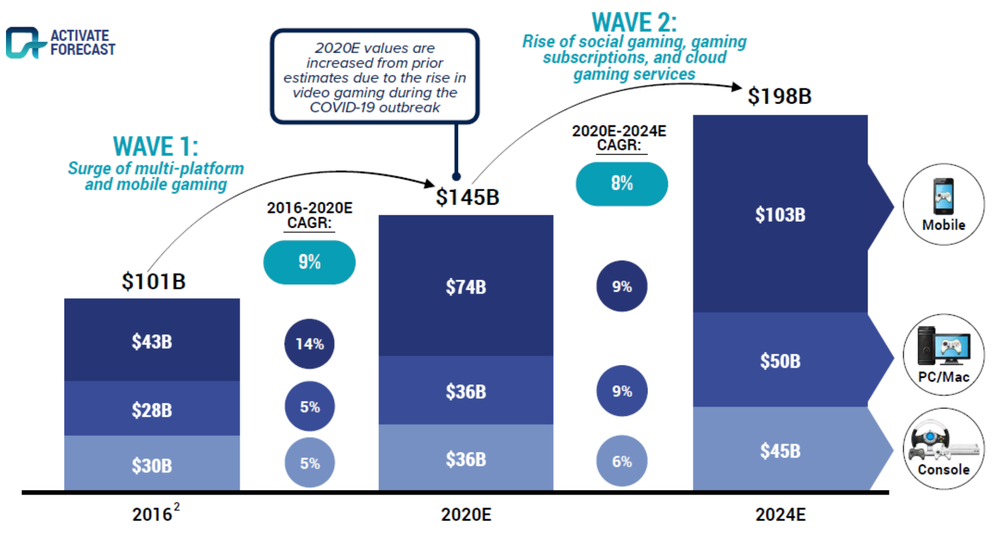


Figure 2.4: Consumer Video Game Revenue by platform, Global, 2016 VS. 2020E VS. 2024E, Billions USD (Ref. [1])

2.1.2 Super Users

To succeed in the change aimed at videogames, technology and media companies will need to concentrate their attention on serving the 23% of all users which are the overwhelming majority of time and money spent on video, eCommerce, gaming, music, and virtual reality. So, they will need to create personalized offerings that fulfill their set of interests. This category is named Super Users or Super Consumers. Super Users are the pioneers on adopting new technologies, such as virtual reality, and are the one to try it to substitute live events and other social experiences. They spend way more time with technology and media than the other users. These users are also younger and with a hgher level of education. Besides they are more connected to internet and have unlimited mobile data access in most of the cases (see pages 17-20-21-22/148 on Activate's Outlook: [1]).

2.1.3 Connectivity

Since the internet is opening the doors to 5G a great percentage of consumers will upgrade to it starting with mobile devices. This will highly enhance bandwidth speeds and it will be determining to expand the use of digital technologies such as video conferencing, telemedicine, digital fitness, video streaming and, of course, cloud gaming.

2.2 The Sims Freeplay

After a research on *gameshunters.com* [2] website it showed in 15-th position a video game named **The Sims Freeplay** [3], a life-simulation game developed by Firemint in 2011 and published by EA Games. It is available on mobile iOS, Android and BlackBerry devices. The game lets you visit your friends cities but also meet your Facebook friends after they registered their account in the game. The user stats of this game are considerable. In "The Sims Freeplay: User Stats" we have the following data:

- Monthly: 1,000,000
- Weekly: 500,000
- Daily: 100,000
- Viral: 3,815

So, there are many users in this moment visiting the game and they are still increasing.

Doing some app analytics on App Annie I found out that between July and September 2021 the AppStore application received 10 thousand rates more and its average score remained stable on 4.54/5 stars (see Fig. 2.5). While between the same period, more than 4.5 million Android users rated the app and its average score dropped 0.05 stars but the number of 5/5 stars increased from 655k to 3.51 million (see Fig. 2.6).

Besides, The Sims Freeplay team introduced also the AR technology in the game. Indeed, the article of "Next Reality"[4] states that Electronic Arts is inviting players to use this technology to visit their friends digital homes since it supports multiplayer and lets you customize your house and interact with 3D characters using ARKit 2.0.

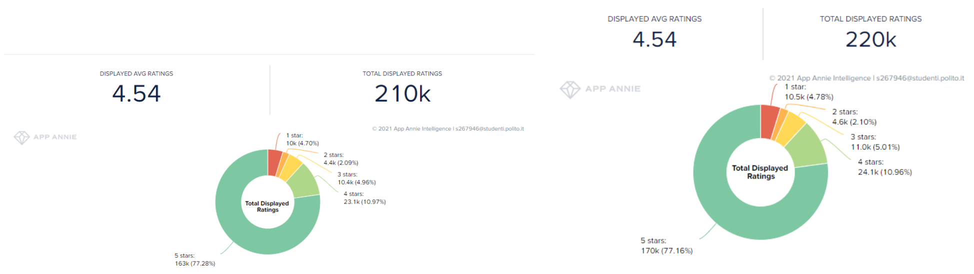


Figure 2.5: The Sims Freeplay AppStore stats between July and September 2021 (Ref. [5])

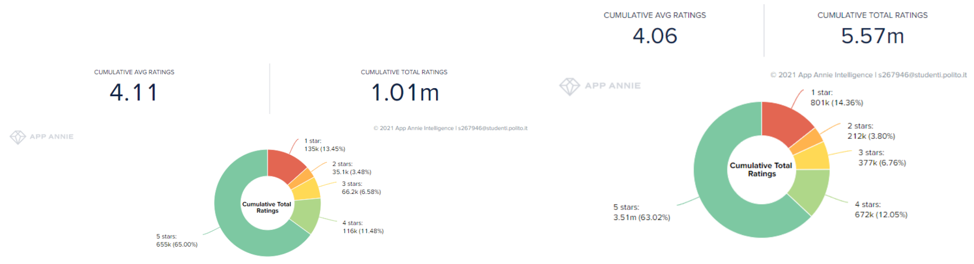


Figure 2.6: The Sims Freeplay PlayStore stats between July and September 2021 (Ref. [6])

2.3 Worlds FRVR

In the Facebook Games Top Charts [7] many of the first games deal with gambling. But if we switch to the "Role Playing Games" category, a considerable name comes out: Worlds FRVR. This is a cloud 3D role playing game published by FRVR. According to Business Wire [8], FRVR is an HTML5 games publisher based in Malta that has a consistent background in making Facebook video games that are easily playable through Facebook Messenger invites between players like the famous Basketball FRVR. On July 2021 it announced a $3 million seed investment led by Accel that believes in FRVR's dynamism and ambition. Coming back to Worlds FRVR, this game is very interesting about the content it offers. It is reported it is increasing its audience (see Fig. 2.8) thanks to the fact that players can create and share new worlds and new events, like parkour races, but they can also make new friends or they can invite existing friends into their session. Indeed, you can also invite Facebook friends through Facebook Messenger and after accepting he spawns in your same session directly. Indeed, I tried so by creating another Facebook user (Klajdi TestApp) and logged in the game with it. Then I invited my original account through Facebook Messenger and it spawned at the beginning of the map of the same session after clicking the button to accept the invite (see Fig. 2.7).

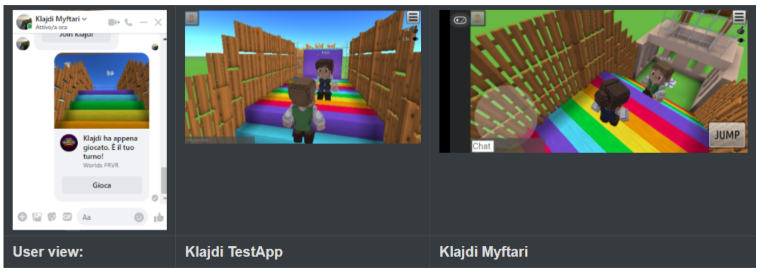
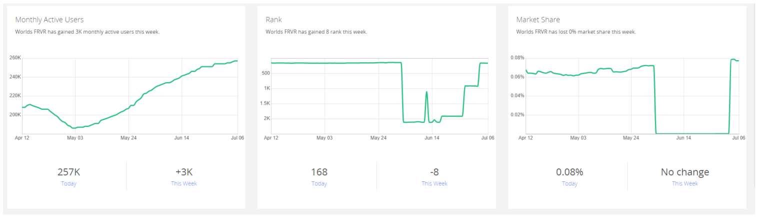

Klajdi Myftari
Attivo/a ora
Klajdi ha appena giocato. È il tuo turno!
Worlds FRVR

Figure 2.7: Facebook invite in Worlds FRVR game session experiment


Worlds FRVR has gained 3K monthly active users this week.

Worlds FRVR has gained 8 rank this week.

Worlds FRVR has lost 0% market share this week.

Figure 2.8: Worlds FRVR game stats April-July 2021 (Ref. [9])

2.4 Facebook Gaming

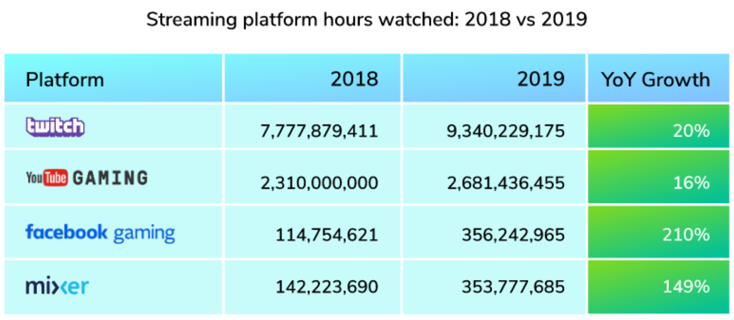
Facebook too started to aim at gaming services. Indeed, they developed a platform only for games: Facebook Gaming. More specifically, it is a Streaming and Cloud Gaming service. According the company, another great and growing use of Facebook, except socializing, is to see the video game lives of your favorite streamers. Facebook decided to invest on streamers and influencers since they are the main reason for approaching a huge slice of users towards the application. The "Influencer Marketing Hub" article (Ref. [10]) states that gaming is a booming sector and it is expected to reach $138 billion by 2021. Except people buying gaming consoles, this growth is due to the increase of users who enjoy watching online gamers playing. It is also becoming a source of opportunities for marketers and influencers alike. This is making the platform successful on this field and it is gaining ground against the old Mixer and Twitch, indeed it has grown at least 210% in one year compared to competitors. In 2019, the platform's content was viewed for 356,242,965 hours, that is 241,488,344 hours more than in 2018. (see Fig. 2.9). Beisdes, Microsoft had shut down Mixer after announcing partnership with Facebook Gaming (Ref. [11]). So, the streamers and audience were be moved to Facebook Gaming platform growing its audience basin. Microsoft had about 30 million viewers on Mixer at the time.

The user base is ever-growing; more than 800 million people use Facebook Gaming every month to play online games. 90 million of them are active members of Facebook Gaming Groups. According to Facebook's internal analytic, 2.6 billion people use Facebook products, including Facebook, Instagram, WhatsApp and Messenger. These users are also keen to playing console games. Facebook has earned customer trust by offering them many free-to-play (F2P) games like Candy Crush, so they trust it more than YouTube and Twitch. It is reported that 82% of Indian gamers between age 18-34 year old say they use Facebook in general to discover new games to try. 75% of them confirm that mid-stream ads helped them in the same way. Facebook's game market was prospected to reach $13.2 billion in North America in 2020 (see Fig. 2.10).

Talking about the workers on the platform, Business Insider (BI) reports [12] that gamers find it easier to make money on Facebook Gaming than on other platforms, owing to less competition and high-converting audience. In fact, many top gamers have ditched their regular jobs to do full-time gaming on the platform.

The Facebook Gaming application was meant to be available both for Android and iOS phones but initially there were some complications. "The New York Times" article (Ref. [13]) reports that Google PlayStore approved Facebook Gaming's app for the Android devices but Apple rejected it at least five versions of it. Apple cited rules that forbid apps with the "main purpose" of distributing games and apps that offer simple games in a store or store-like interface. Lately, Apple approved Facebook Gaming only after Facebook submitted a version with games completely removed, the spokesman said. However, Facebook Gaming's success has been rising during the years. In the "Deloitte" report (Ref. [14]) we see indeed that the European e-sports market grew further successful development in 2019 and it meant that most companies increased their revenues. The market was being watched to see if growth will continue in 2020 despite or perhaps because of the COVID-19 pandemic (see Fig. 2.11).

Facebook Gaming is accessible only with a Facebook account. It is accessible through PC or mobile, in the Gaming section inside the Facebook app or web page. There is also the Facebook Gaming mobile app that forces you to login only through a Facebook account or advising to create one if not created yet.
Conclusions: you can access Facebook Gaming only with a Facebook account (see Fig, 2.12).


Streaming platform hours watched: 2018 vs 2019

Platform	2018	2019	YoY Growth
twitch	7,777,879,411	9,340,229,175	20%
YouTube GAMING	2,310,000,000	2,681,436,455	16%
facebook gaming	114,754,621	356,242,965	210%
mixer	142,223,690	353,777,685	149%

Figure 2.9: Streaming platform hours watched: 2018 vs 2019 (Ref. [10])

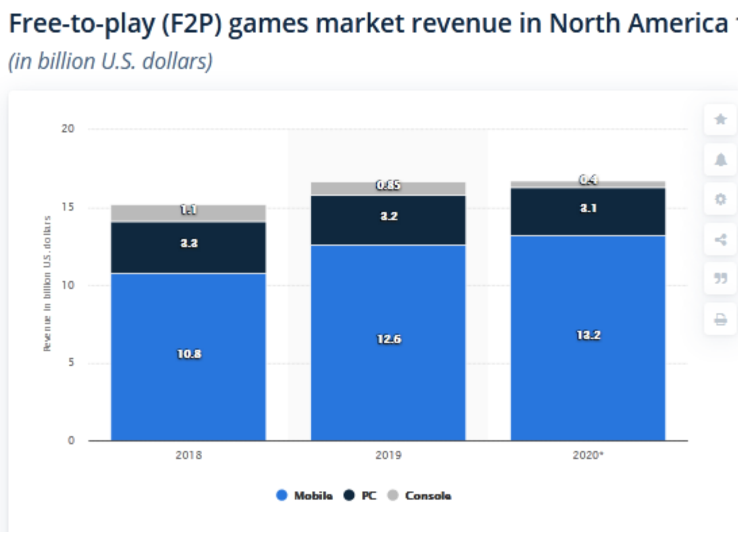


Figure 2.10: Free-to-play(F2P) games market revenue in North America (in billion U.S. dollars) (Ref. [10])

2.5 Towards the Metaverse

Before and especially during the Covid-19 pandemic, many companies began to consider creating metaverses to bring play, creativity and social interaction into a new dimension. What is the metaverse? Metaverse is the definition of a virtual world where people can join with a 3D avatar they can command. It lets them create and explore new worlds, socialize and play with friends, and this is only the beginning that

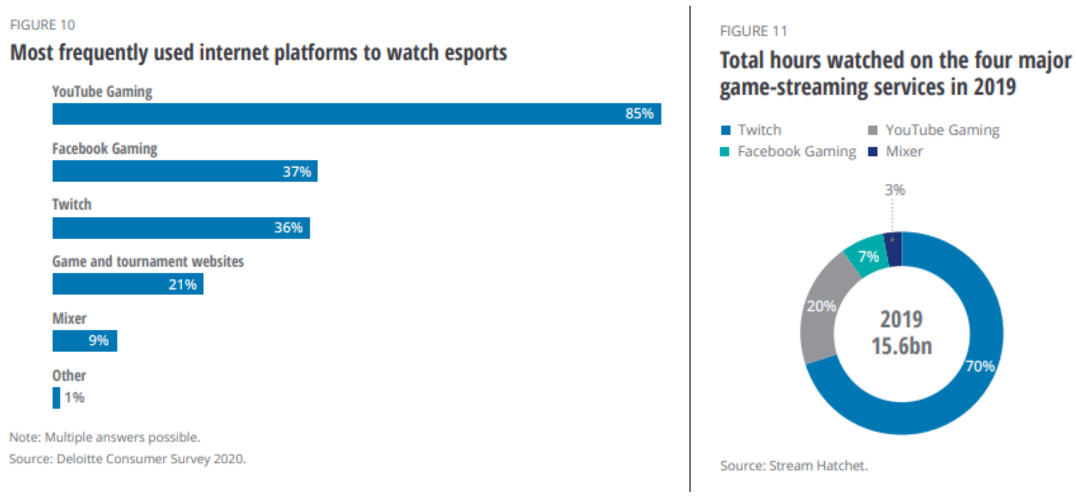
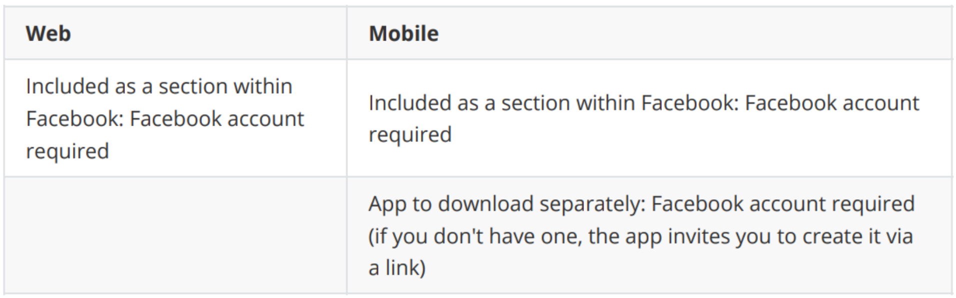


Figure 2.11: Deloitte E-sports statistics (Refs. [15] and [16])



Web	**Mobile**
Included as a section within Facebook: Facebook account required	Included as a section within Facebook: Facebook account required
	App to download separately: Facebook account required (if you don't have one, the app invites you to create it via a link)

Figure 2.12: Facebook Gaming access methods

these virtual worlds can offer. How can the Metaverse affect humans' percetion and creativity? Games Beat article (Ref. [17]), about this, states specifically that some researchers from Drexel College of Arts and Sciences demonstrated, in a recent study, how the reward centers of our brains are activated when we have 'Aha!' insights. They continue saying that these results 'may be a manifestation of an evolutionarily adaptive mechanism for the reinforcement of problem solving, exploration and creative cognition'. So it's part of human being's genetic make-up to be inclined to enjoy being creative. The Metaverse will allow more people in the furture to embrace digital creativity in their lives.

It was demonstrated that peer learning offers students benefits such as more positive learning environment, improved interpersonal relationships, higher academic marks, better personal and social development, and increased motivational levels.

The Metaverse is leading to the drop of hardware barrier required combined with increased bandwidth, so ti means that digital creativity and play will become more accessible to an always increasing amount of the public worldwide.

All of it is also possible thanks to the help of the Super-Users, hooking to Activate's Outlook. Indeed, they are more confident to explore this new technology and they will be the pulling factor for the other users.When it comes to online confidence, Generation Z are leading the way. The Center for Generational Kinetics showed in their third international study that this generation makes no distinction between the real and online worlds, and Gen Z replied that the most important attributes to their generation are tech-savviness (rated 19%) and freedom (rated 22%)'.

2.5.1 Metaverse Games

The followings are some of the most famous Metaverse applications created so far.

1. **Crayta:** Crayta is a cloud platform developed by Unit 2 Games to let the users create and play contents without the need to know programming. It has been recently acquired by Facebook. Crayta is having a great potential in the Metaverse field. Indeed, as Tech Crunch (Ref. [18]) states, Crayta has cornered its own niche and its smartest move was pushing monetization paths like Battle Pass seasons, giving the platform a Fortnite-like vibe. The title feels designed to run on cloud-gaming platforms, with users able to share access to games just by linking other users, and Facebook seems convinced to use Crayta to push forward their own efforts in the gaming world. Facebook Gaming Vice President Vivek Sharma wrote in an announcement post:

   > "Crayta has maximized current cloud-streaming technology to make game creation more accessible and easy to use. We plan to integrate Crayta's creation toolset into Facebook Gaming's cloud platform to instantly deliver new experiences on Facebook".

   According to Routenote (Ref. [19]) Crayta is going to make Facebook earn in the future a considerable amount of money thanks to music industry. Another metaverse game: Roblox, is famous for hosting album launches and livestream concerts of famous stars that were displayed appearing as avatars as part of an in-game awards ceremony and a Lil Nas X virtual concert that attracted 33 million views. Especially with Facebook's power behind the platform, many artists can do the same on Crayta. Indeed, it seems that this topic is something that Facebook is eyeing up; as Facebook Gaming's statement said:

   > "Our next phase of growth will come from bringing the activities of playing, watching and connecting closer together."

   So, with the collaboration of Unit 2 Games, Facebook aims to expand the notion of a creator to include people who build, publish and share games, worlds and new experiences on Facebook collaboratively.' Unit 2 Games' main goal was to make game creation accessible and build communities around the content created by users. So, Music could make a backdrop to the new games created by Crayta users, and artists would get paid in case their tracks were used.

2. **Breakroom:** Developed by Sinespace, Breakroom gamifies the workplace thanks also to Virtual Reality technology. It creates a Metaverse where employees can create their avatar and use it to join meetings and to roam in the virtual office to meet each other for business and leisure time. The Evening Standard (Ref. [20]) states that Sinespace is specialised in gamifying the workplace with 3D scale models of officers and this led to more than 100 enquiries from bosses interested in recreating the digital counterpart of their environment. Someone may wonder why all this work for business corporations. Well, it was shown that simple Zoom video calls can't recreate the same feelings of real life meetings and thus employees are not motivated the same. But there is more. Rohan Freeman, chief executive of St James's-based Sine Wave Entertainment, said that the £400-a-month creations allow companies to customise templates of open-plan offices, canteens and conference rooms.

> "In a virtual world you can be wandering around in an area and you bump into Jeff from accounts and realise: 'Oh, I must have a quick word with him about something'. It's all about creating a sense of community without having to perform all the time in the formalised structure of a video conference call."

Like many other Metaverse applications, Breakroom can be accessed with any kind of hardware. Besides, you can join via webcam video or build an animated avatar to walk around the world (Ref. [21]).

3. **IMVU:** IMVU is an online metaverse and social networking website similar to Second Life, where you can also sign up via Facebook. In 2021 IMVU has more than **200 million** registered users and **4 million** monthly users. The huge amount of visits in IMVU's Metaverse are given by the use that players make of its virtual coin. Indeed, Games Beat article (Ref. [22]) reports that users can exchange IMVU's proprietary Credits for digital goods, like a skin for their avatar, or to visit virtual rooms, e.g. a dance club. IMVU has 7 million active users per month who exchange **14 billion** Credits each month and engage in **27.5 million** monthly unique transactions. The market has more than **50 million** products that people can buy today, with the catalog growing by 400,000 items each month.

4. **Roblox:** Roblox is a very famous metaverse game where many players of different ages meet to socialize, create worlds and play together in never ending ways. Robox has become so famous that **32.6 million** people on average join its servers every day and more than **1.25 million** people made money in Roblox by creating worlds that players can visit. At the end of 2020 users spent **30.6 billion** hours playing on the platform, so an average of 2.6 hours per daily active user each day (Ref. [23]).

5. **Rec Room:** Rec Room is a social gaming platform too and it is known for raising **$100 million** at a **$1.25 billion** valuation. The funding is another vote of confidence for user-generated content. Rec Room has more than 5 million rooms to explore, and there are more than 2 million players who built stuff in Rec Room. Rec Room launched as a free-to-play experience in 2016, and now it has **15 million** lifetime users. Revenue grew **566%** in the past year since monetization was introduced in the game, said CEO Nick Fajt in an interview with GamesBeat (Ref. [24]). Fajt states:

   > "We've had strong growth over the last 12 months, we think that Rec Room can become an enduring large business that fuses games and social."

   Dealing with hardware, RecRoom was made available to several platforms, including PCs, game consoles and mobile devices. Besides, the CEO said that their future goal is to move on AR and VR devices too.

   On the table below (see Table 2.1) you can see a summary of the features of the social video games introduced before.

2.5.2 VR & AR: Oculus

Oculus is a Virtual Reality devices company that launched in 2012 and was later bought by tech giant Facebook in 2014 for $2 billion. Even if at the time Oculus co-founder Palmer Luckey said users wouldn't need a Facebook account to use their device, this changed last year to come for all users by 2023. Oculus is more affordable than some of its rivals since with Oculus Quest 2 it catched around 30% of the PC market and sold around a million headsets in the fourth quarter of 2020 alone (Ref. [25]).

2.5.3 5G and the metaverse

5G technology is strongly emerging nowadays and it will bring huge improvements in internet connection, especially in gaming connection. Games Beat about it reports (Ref. [26]) that 5G is enabling new forms of entertainment, favoring the consolidation of the Metaverse.

2.6 Facebook

About Facebook itself, the Sprout Social article states that Facebook boasts a number of almost **2.8 billion** monthly active users, and that is normal, due to its age in the market. Facebook users can be found at all ages, 72.8% are within the 18–44 years old range. Users ages 25–34 years are the largest demographic (see Fig. 2.13). In the distribution of worldwide Facebook users, 13.1% were female accounts between 25 and 34 years old and 19.3% were male in the same age range. Among countries, India has the largest number of users at 310 million, followed by the U.S. with 190 million and Indonesia with 140 million. 32% of the U.S. social media users surveyed

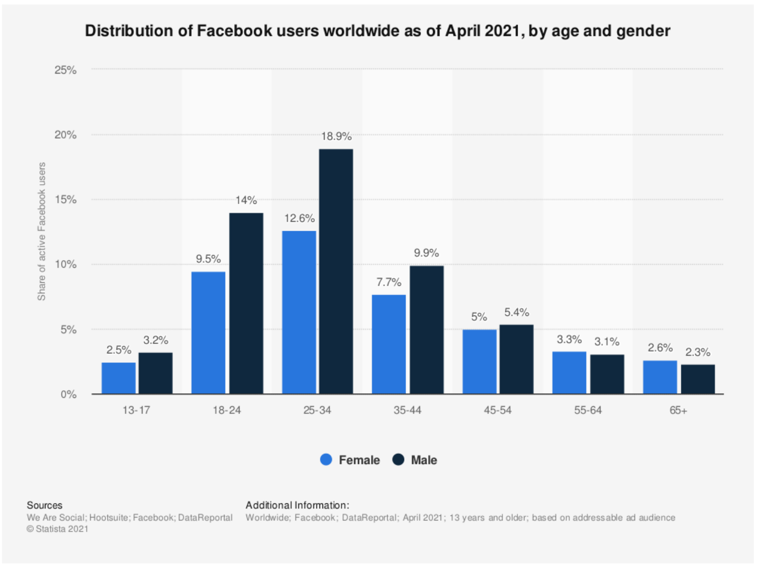


Figure 2.13: Facebook demographics worldwide (Ref. [27])

aged 12–34 recognised Facebook as the social platform that they used the most (see Fig. 2.14). This is a dramatic drop from five years ago when the top usage was at 58%.

For advertisers, there is good news because 86% of internet users with $100k+ income use Facebook. It means that their tailored ads can target different income levels without worrying about a low target count. Besides, it was found that **2.79 billion** people globally use Facebook. In the U.S., it is expected to grow from 228.6 million currently to 237.8 million around 2025. The usage penetration is small, indeed it is increasing 1.9 percentage points over four years.

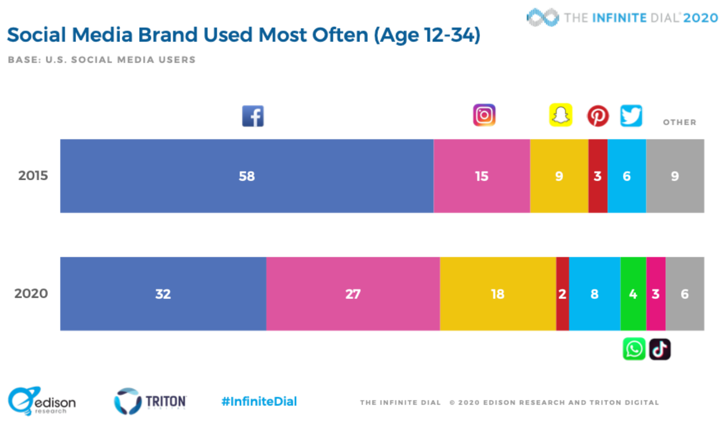
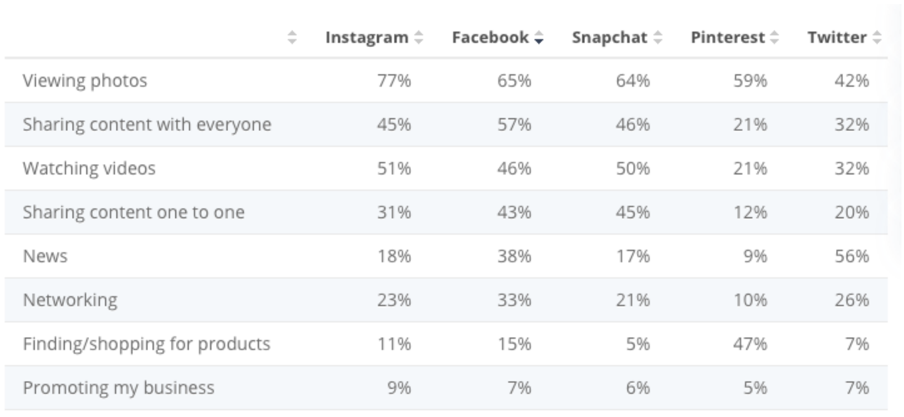


Figure 2.14: Social Media Brand Used Most Often (Age 12-34) (Ref. [27])



	Instagram	Facebook	Snapchat	Pinterest	Twitter
Viewing photos	77%	65%	64%	59%	42%
Sharing content with everyone	45%	57%	46%	21%	32%
Watching videos	51%	46%	50%	21%	32%
Sharing content one to one	31%	43%	45%	12%	20%
News	18%	38%	17%	9%	56%
Networking	23%	33%	21%	10%	26%
Finding/shopping for products	11%	15%	5%	47%	7%
Promoting my business	9%	7%	6%	5%	7%

Figure 2.15: Main reasons people use social media platforms (see Ref. [27])

Facebook users almost only use mobile to access Facebook; 98.3% of them accessed Facebook via any kind of mobile device and 79.9% of users only used mobile. According to **login frequency**, 73% of U.S. Facebook users login daily but he number increases to 93% of users on a weekly basis.

What do they use Facebook for? 57% of U.S. social media users use Facebook to share content. In a 2019 Statista report it was found that **65%** of Facebook users view photos on the social platform. Facebook ranked the highest in use of sharing content with everyone (57%). This is followed by 46% of users who use it to watch videos and 43% who share content one to one via direct messages (see Fig. 2.15).

Facebook has another strong component that makes it community friendly. In its annual Communities Summit, Facebook revealed that **1.4 billion people** are using Facebook Groups on a monthly basis.

2.6.1 Facebook Gaming Groups

Staying on groups topic, Facebook decided to try this formula also inside Facebook Gaming. Indeed, the Facebook Team decided to introduce a dedicated “Facebook Groups” section. As Games Beat article (Ref. [28]) states: within these fan groups, creators can create topics into discussions and threads that users can move into chat channels. Furthermore, Facebook is also entering a new post type, “Looking for Players”, to help members of the community find other people among a streamer’s community to play with.

2.6.2 Summing up

The table below (see Tab. 2.1) is a summary of all the main games cited by far and it shows the advantages and disadvantages with respect to the features they offer. As we can see, the majority of them is projected to be multi-platform since the barrier of technology should be absent to gather users from many fronts. Facebook is a fast way to sign up first time, reducing the effort of registration since many users have a Facebook account. Augmented and Virtual reality devices are prospected to be the future, so many of these applications are already open to adopt them as and alternative. Since these games focus on social interaction, another important feature related to them is the opportunity to let players create and share their inventions, and all of these apps understand it.

Games/Features	Multiplatform	Facebook Login	AR or VR compatible	Create Maps and Events	Socialize
The Sims Freeplay	No	Yes	Yes	Yes	Yes
Worlds FRVR	Yes	Yes	No	Yes	Yes
Crayta	Yes	Yes	No	Yes	Yes
Breakroom	Yes	No	Yes	Yes	Yes
IMVU	Yes	Yes	Yes	Yes	Yes
Roblox	Yes	Yes	Yes	Yes	Yes
Rec Room	Yes	No	Yes	Yes	Yes

Table 2.1: Features comparison of the Social Games presented above.

Chapter 3

Technologies and tools

3.1 Visual Studio 2019 Community

Visual Studio 2019 Community (Ref. [29]) is a free, extensible, full-featured IDE for building modern applications for Android, iOS, and Windows, as well as web applications and cloud services. During the research it was useful to test which social platform gave more info and how easily. The research concentrated on the following platforms: **Twitter, TikTok, YouTube, Instagram, Facebook**. To verify it, the application offers the possibility to download packages, SKDs and frameworks that communicate with these platforms:

- **Facebook (7.0.6):** Download the friends list in a .json file and download user photos and photos of his Facebook friends using an access token and app data;
- **Google.Apis (1.51.0) + Google.Apis.YouTube.v3 (1.51.0.2338):** Download of lists of YouTube videos, channels and playlists related to a name passed to compile-time;
- **Newtonsoft.Json (12.0.3):** Serialization and deserialization of data downloaded from social platforms;
- **Curl:** Query of information to download from social platforms;
- **Microsoft.AspNet.WebHooks.Receivers.Instagram:** Download of a user's posts (username, message, photo/video, one unit/album) and save the related data (url for media) in a .json file, using Instagram permissions;
- **.NET:** Download di risorse tramite URI usando la classe WebClient;

The testing of the social platform could be mostly possible only after registering on their developer section.

3.1.1 Google Cloud Platform

Google Cloud Platform (Ref. [30]) is a high-performance infrastructure used for data analytics, cloud computing, and machine learning. It was a core component to generate an API Key to request YouTube data from Visual Studio platform.

3.1.2 Facebook Developers

Facebook Developers (Ref. [31]) is a platform for creating and testing applications to be associated with Facebook and Instagram features. It was necessary again to obtain the permissions to do the data extraction of Facebook's users.

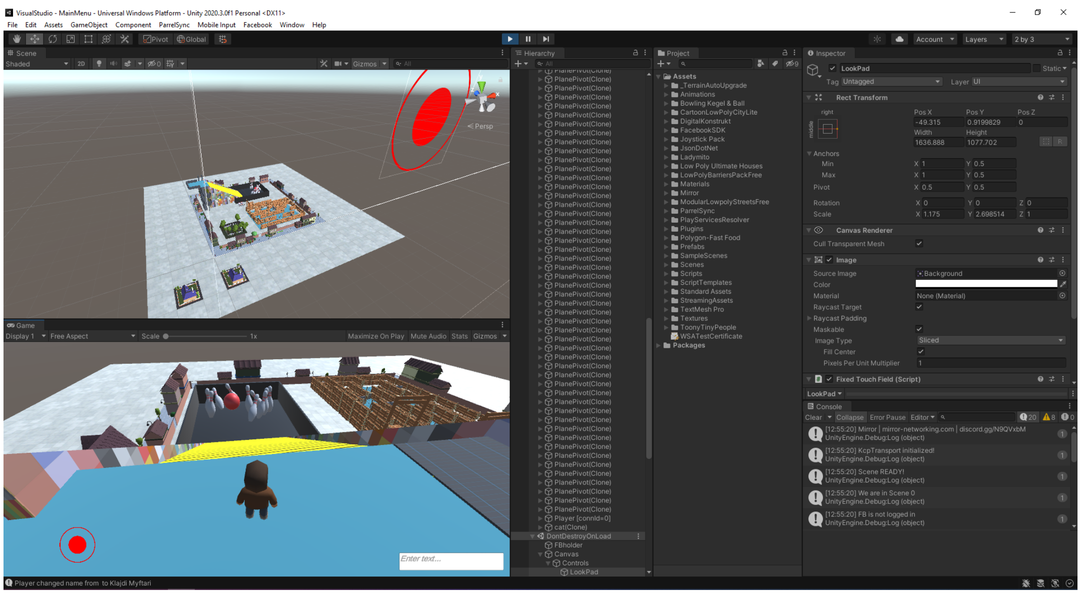
3.1.3 Heroku

Heroku (Ref. [32]) is a platform as a service (PaaS) that lets developers the possibilities to build, run, and make applications entirely operative on cloud. During this work it was used with Facebook to get the timed token after the login redirect.

3.2 Unity

Unity (Ref. [33]) is a cross-platform game engine that is mostly used to create 2D and 3D videogames, or Virtual Reality and Augmented Reality applications such as videogames, and simulations. But lately it has been extended its purpose, for example, it is used for automotive and film production. The engine combines several tools and libraries to make the development of said applications easier, and provides support for deploying the final product to over 25 different platforms. Unity engine has been written using the C++ programming language, but the development of applications using Unity is done through C# or JavaScript scripting. This is the second most used application for this work. To create the prototype, Unity has been chosen among other engines because it provides the opportunity to integrate several packages and assets to interact with external platforms like social media. Besides, it provides access to a consistent Asset Store, where members of the community can share their own packages (free or paid) that implement custom solutions or content ready to be easily reused. Furthermore, 3D models created in modelling applications can be easily exported into an FBX format and imported in the engine, including animations too. Unity layout (see Fig. 3.1) is made very simple and concise to help developers understand and orientate during the making process. As we can see, it has two main windows that show the view of the scene from two different perspective: the default view and the camera view. In the first case, it is a frame that lets the developer move into the scene environment and manipulate objects by using navigation and transformation commands. While the Game Window shows only the view from a virtual camera put in the scene and that is the view that the game will show during run-time. Onto the right we have the Hierarchy Window that shows the elements in the scene under a different way. Then, there is the Project Window that projects indeed the project's folder, in the file-system. The Inspector Window contains the Components of each item the developer selects. The info usually are: transform, which includes the position, scale and rotation coordinates of the item, Mesh Render or the visibility of the Game Object. And the Collider, since Unity manages collisions too. But there is also the possibility to attach always new components and the most commons are the scripts. Thanks to the scripts, the Game Objects can have behaviors and interact in the environment. To debug the scripts or the engine in general there is the Console Window. It prints three categories of messages: Info, Warning and Error. The latter kind of message usually cannot let the application run. To run the application, developer has to press the Play Button on top, he can pause it and resume it or stop it.

- **Facebook SDK (Ref. [34]):** since the prototype was associated to Facebook, from the developers website it was possible to download the SDK and integrate it on Unity. It was necessary to make the API calls to download the relative users' data;
- **Joystick Pack (Ref. [35]):** Since the application is thought to be for mobile platforms too, this asset has the collection of different types of virtual joysticks that can be inserted in the app to implement mobile commands;


VisualStudio - MainMenu - Universal Windows Platform - Unity 2020.3.0f1 Personal <DX11>

```
[12:55:20] Mirror | mirror-networking.com | discord.gg/N9QVxbM
UnityEngine.Debug:Log (object)
[12:55:20] KcpTransport initialized!
UnityEngine.Debug:Log (object)
[12:55:20] Scene READY!
UnityEngine.Debug:Log (object)
[12:55:20] We are in Scene 0
UnityEngine.Debug:Log (object)
[12:55:20] FB is not logged in
UnityEngine.Debug:Log (object)
```

Player changed name from to Klajdi Myftari

Figure 3.1: Screenshot of the prototype project and Unity Editor layout

- **JSON .NET For Unity (Ref. [36]):** Asset that offers better methods to serialize and deserialize .json files and to manage JToken objects;
- **Standard Assets (for Unity 2018.4) (Ref. [37]):** Asset that offers a basic set of characters and vehicles to test on the engine. In this case, the 3d third-person test character was useful to navigate the game environment and interact with the features implemented;
- **TextMeshPro (Ref. [38]):** Asset to manage text fonts and to create text in 3D, with more advanced features than the simple text manager of Unity;
- **Mirror (Ref. [39]):** High-level API for Unity for networking, which supports different types of low-level network transports (TCP, UDP, etc. . .). Built and tested for MMO-scale networks;
- **Bowling: Kegel & Ball (Ref. [40]):** Simple bowling set to build a bowling game. It was used to create a bowling mini game inside the square of the semi procedural world;
- **Cartoon Low Poly City Pack Lite (Ref. [41]):** Simple asset on Unity Store to create low poly cities. It was used into this work to create the streets system.
- **Prototyping Pack (Free) (Ref. [42]):** Asset pack to create easy and fast prototypes for games. In this case it was necessary to make the bowling park, made of a ramp where the bowling ball slides down, a pool where the pins are placed, an automatic elevator and the stairs.
- **Free chibi cat (Ref. [43]):** This asset contains a little animated cat and it was useful to make an avatar pet to follow the player wherever it went.
- **Low Poly Ultimate Home Pack (Ref. [44]):** This asset pack includes five different prefabs and each of them contains one or more home. They were useful to create the surrounding of the square and to customize the houses of the players, like gardens and fences.

- **Modular Lowpoly Streets (Free) (Ref. [45]):** Asset that contains a set of streets and street decorations, useful to make the pavement of the square and around it.
- **ParrelSync (Ref. [46]):** Thanks to this asset it is possible to test offline a multiplayer application, using the localhost or another network. It made possible the testing of Mirror using multiple clones of the same project.
- **Polygon - FastFood (Ref. [47]):** Simple asset containing two fastfood carts.
- **Toony Tiny People Demo (Ref. [48]):** It includes a series of character with animations. One of them was used as default skin of main player.
- **Toon Furniture (Ref. [49]):** It contains all of the furniture used to decorate the houses of the players.
- **FREE Suburban Structure Kit (Ref. [50]):** Hosts all of the house models to define players' dwells.

The only programming language used for this work is **C#** and it is supported either for Visual Studio or Unity.

3.3 Mirror

As mentioned before, Mirror is a high-level API for Unity for networking. But, since it is not the only solution, after an effective research it came out that it is the best alternative for this work. As the following table shows (see Tab. 3.1), even though Mirror does not have a user friendly API and can't do host migration, it can make large scale connection, which is vital for this project. Thousands of users shall connect to a single session and Mirror proves worthy to make it happen. Indeed, stress tests were made directly in Unity, during the development and there are also some examples of Mirror under huge stress test (see Refs. [51] and [52]) and they demonstrate (see Fig. 3.2) the high capability of the API to hold a large number of players at the same time.

	Pros	Cons
Azure PlayFab	Dedicate Servers	Free servers only for specified regions and only for 750 hours
Photon PUN 2	User Friendly API	Free upto only 20CCU
	Tutorials Videos	CCU Limit
	Built-in Host Migration	
MLAPI	Unity's Official Solution	Still under development
	Free and similar to Mirror	
	No CCU limits	
Mirror	Totally free	No Host Migration
	No CCU limit	Uneasy API
	Large scale connections	Needs dedicated server

Table 3.1: Pros and Cons of the main multiplayer solutions for Unity Engine: Azure PlayFab (Ref. [53]), Photon PUN 2 (Ref. [54]), MLAPI (Ref. [55]) and Mirror (Ref. [56])

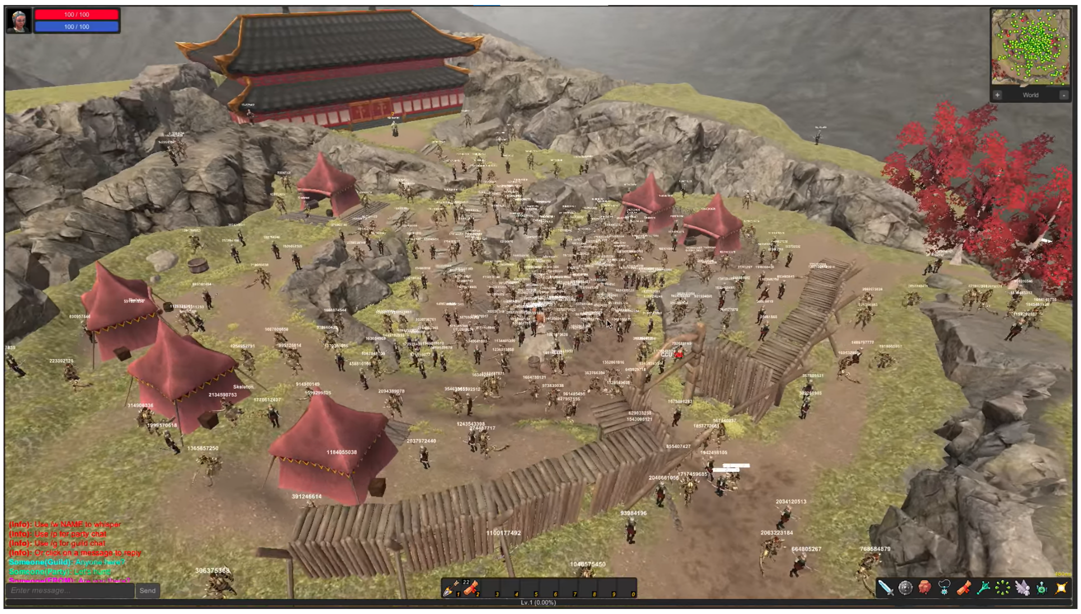

(Info): Use /w NAME to whisper
(Info): Use /p for party chat
(Info): Use /g for guild chat
(Info): Or click on a message to reply
Someone(Guild): Anyone here?
Someone(Party): Let's hunt!
Someone(FROM): Are you there?

Figure 3.2: Mirror stress test from the official website: worst case scenario (Ref. [52])

Chapter 4

Design and development

4.1 Project Design Document

The following documentation provides a detailed description of the purpose of this project and how it was planned, from start to develop. It is structured to be clearly understandable.

1. **Overview:** The main goal of this work, as mentioned above, is to provide an alternative way to browse social networks, in this case, Facebook. When the player logs into the game through the Facebook account (see Fig.4.3), a semi-procedural village is generated, based on the data retrieved from his account. For each friend connection, a small house spawns which, upon entering, it shows his activity (Feed) on Facebook. *If a friend is a close friend of the user and/or active, his home will be shown next to his.* When Players will launch the app, they will be prompted in the login window. Once they chose to login, a redirect window to Facbook login opens and, after the user signs in and gives the permissions requested by the app to work, can freely wander around the village, enter the houses and meet other users (see Fig. 4.1), logged into the platform, and interact with them. Inside the houses players can be able to look at Facebook Feed, just scrolling it or checking comments and reactions and also watching videos. Facebook API calls allow the retrieval of a considerable amount of data from the platform, which could be used to customize everything, as well as to interact with others' published content. The users could then have a certain freedom of customization of their avatar and home and furnish it to their own taste. Besides, players will be able to chat each other with other players who will play online. For a fee, new styles may be added for the village (Medieval, Nordic, Rural, City, Italian, Oriental, and so on), as well as new clothes, new home furnishings, etc. And again: atmospheric events linked to the place, possibility to edit the village, and so on. In two words: an "Animal Crossing/The Sims" (see Refs. [57] and [3]) clone using Facebook data. Along the same lines it would then be possible to have different versions connected to different social networks: for Instagram it could be called "My Art Gallery", for twitter "My Gossip Room" etc. To describe visually a scenario of how the app is meant to be used, there is a storyboard (see Fig. 4.9) showing each step, starting from the login scene until the feed interaction scene.

2. **Story Boards:** The following are a list of storyboards to deeply describe a scenario extraced from the Use Case Diagram mentioned before (see Fig. 4.1).

3. **Architecture:** Figure 4.2 shows the Software layer. All the work was developed on a machine running Windows 10. The research on Visual Studio involved Nuget. As Microsoft official documents state, (see Ref.[58]) Nuget is an essential

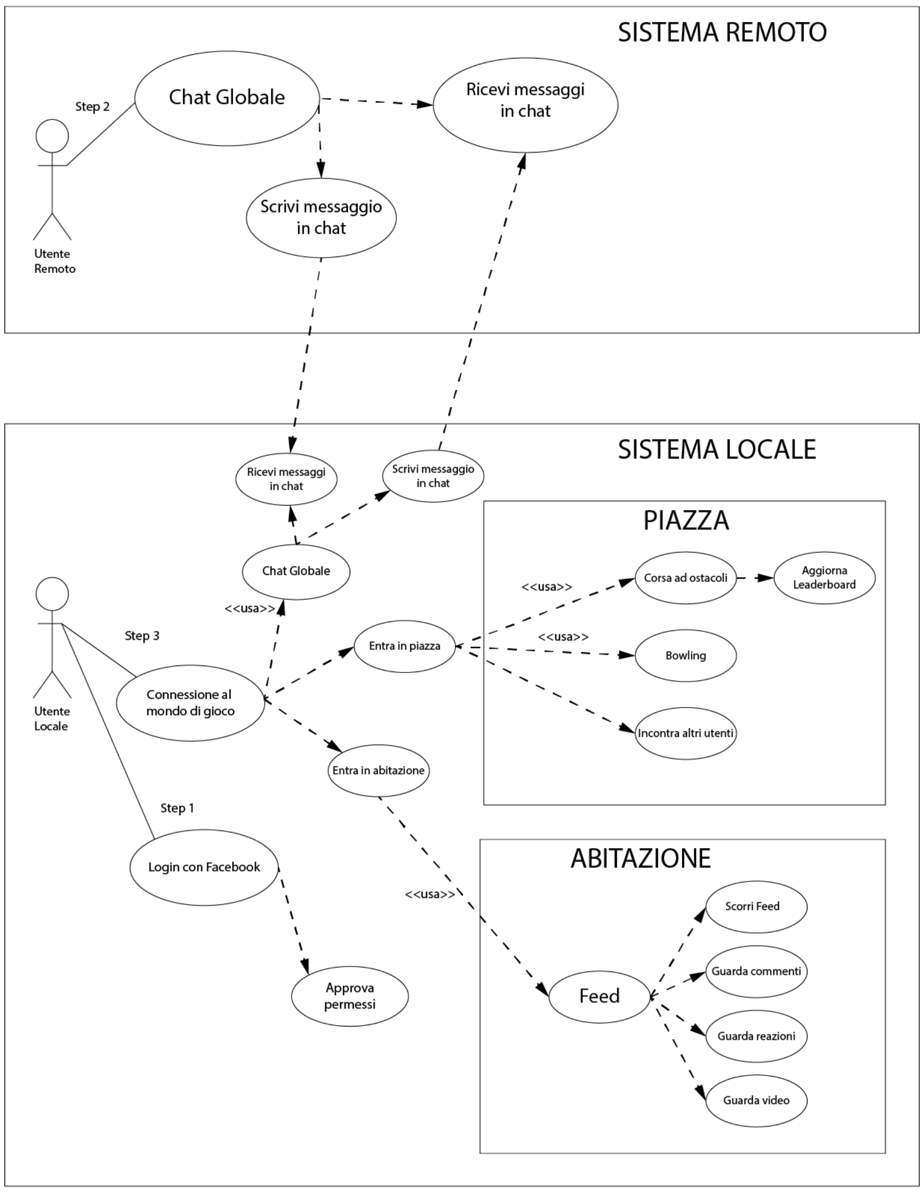


Figure 4.1: Use Case Diagram describing, step-by-step all of the actions players do when using the application

tool for developers to create and share useful code and content. In this case it was used to download packages and SDKs to test the several APIs. Then, .NET was necessary too to exploit the core functionalities to manage the data retrieved with the API calls. But to get the permissions to scrape these data it needed to login to the developer platforms, like Google Cloud Platform, or to redirect on another website (e.g. Heroku) to recover the access token to get the permissions to collect the data. Then, the development moved to Unity, mainly using the external asset Mirror and the Facebook SDK, that was a core feature to get users' data. The Unity project application was built for PC and Android. For the latter, it was tested on a smartphone running Android OS.

4. **Platforms:** The main target platforms for this prototype are Mobile and PC.

5. **Social Platforms:** The social platforms are the feature that distinguishes this

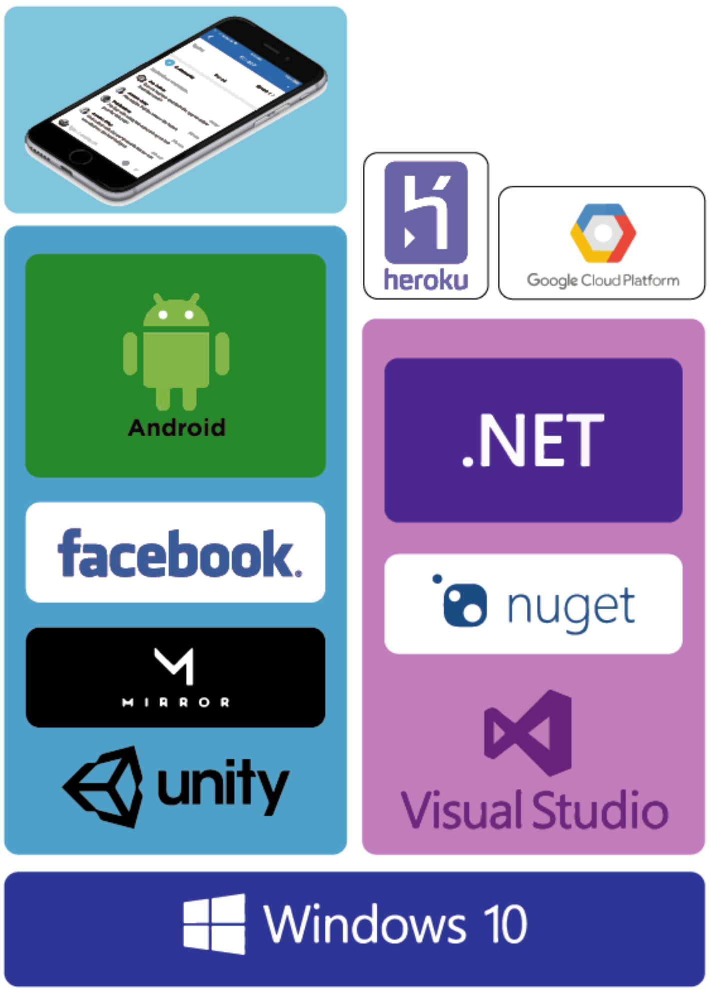


Figure 4.2: Software layer overview

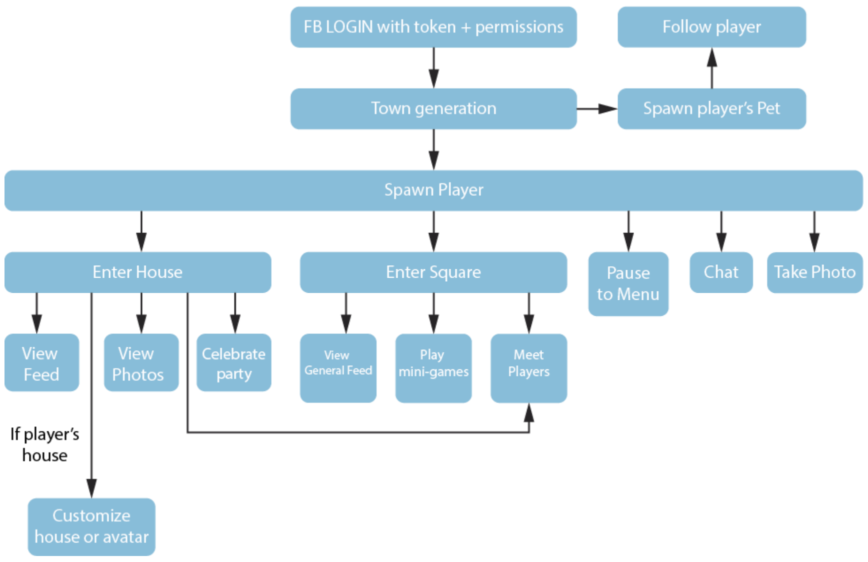
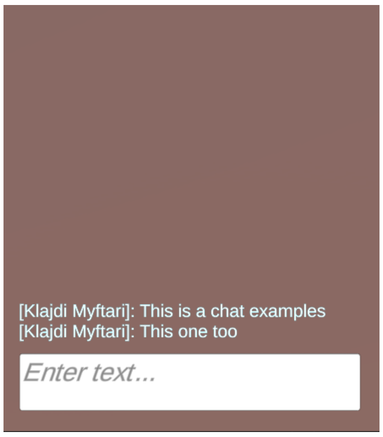


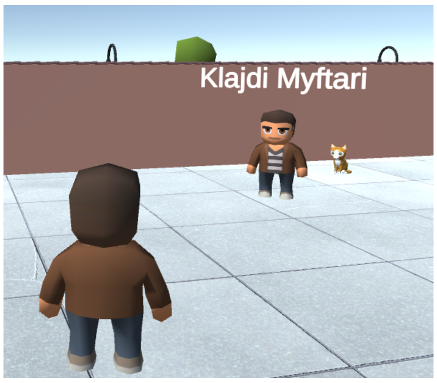
Figure 4.3: Stream Diagram of player's global interactions

application in comparison with other MMO games. The prototype supports Facebook APIs but in the future more are the choices to add, such as Instagram, Twitter, YouTube;

6. **Classic/AR/VR:** The game is initially thought to be Classic but it is open to become AR/VR in the future.
7. **Multiplayer:** Multiplayer is the core feature for the application because it lets friends interact each other and with each other's content through social media.
8. **Content Creation:** The prototype is only focused on gaming and social interaction at this time but in the future it will be open to let players create their own content.
9. **Category:** This is an MMO networking application linked to elements of gamification. The main purpose is to let players socialize and spend their time together, recreating the features of social platforms but in a more engaging way. They can do many activities together, some of these are mini-games.
10. **UI**
    - Both PC and mobile versions will always show on the right side of the screen a global chat (see Fig. 4.4a). The structure of a message is as shown in the figure: we have on the left the name of the sender and on the right his message. Another UI element always present in both of the platforms is the name of the other players on top of their head (see Fig. 4.4b) to help the user recognize them.
    - **PC:** Computer visuals will be almost totally clean from extra elements on screen, indeed only some popups will appear when doing specific actions (indicator to open the door of a house or to interact with the feed). The global chat has an input box that can be focused by pressing a specific key and player can then type directly from the keyboard and send the message with the return key.
    - **Mobile:** In this case there is one main UI element always present on the left inferior corner of the device screen: the "Move joystick" button, while the rest of the screen lets the player move the camera when swiping. The interaction buttons instead, will pop up when an action is triggered (open the door, watch or exit the feed, jump). The global chat has an input field that the user can tap and that opens the mobile keyboard to let him type the message, than a send button below can be pressed. Last but not least, there is also the pause button to enter the in-game menu.
11. **Visuals:** The game will have a cartoon-like low-poly look.
12. **View:** Third person. Camera will be placed behind the player that will be seen entirely. When entering the house or narrow spaces, the camera may zoom in, interactively, and, when interacting with the feed or when focusing on more specific things, it will toggle another camera pointing the target.
13. **Game Locations:** The game is set into a semi-procedural village that resembles the 3D counterpart (see Fig.4.5a) of the Facebook account of a user:
    - **Main square:** Public Feed;


[Klajdi Myftari]: This is a chat examples
[Klajdi Myftari]: This one too

(a) An example of the chat box used in-game

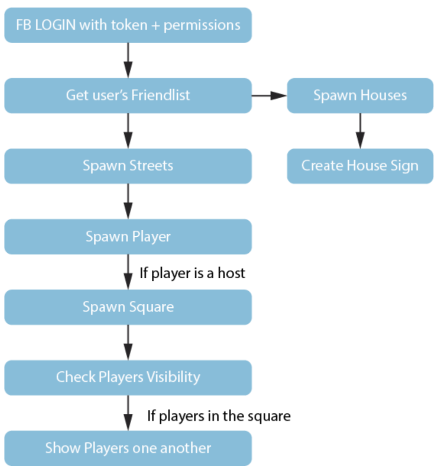


(b) An example of the names on top of the opposite players

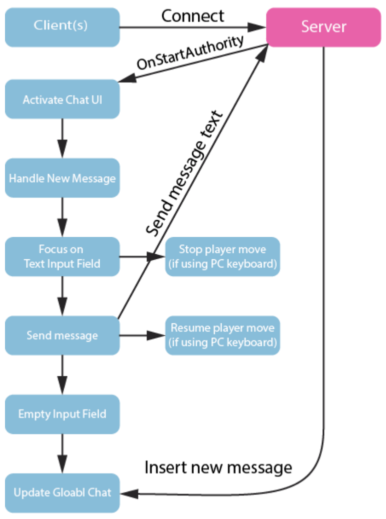
Figure 4.4: Example of chat system (picture 4.4a) and players' name tags (picture 4.4b)

- **Homes:** Users' and his friends accounts;

Each house has a garden to celebrate users' parties (e.g. birthday). Each user will see his own list of friends materializing into houses. So, each one will see a different neighborhood. The main square is the only part of the city common to all, where players can meet and do activities.



(a) Stream Diagram of Town's semi-procedural generation



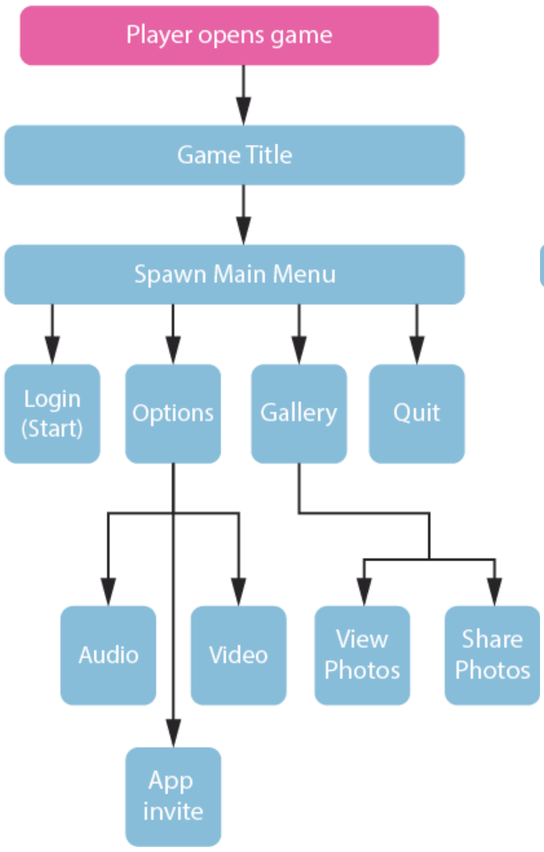
(b) This picture shows how the chat behaviour works

Figure 4.5: Stream diagrams of town generation (picture 4.5a) and chat box behaviour (picture 4.5b)

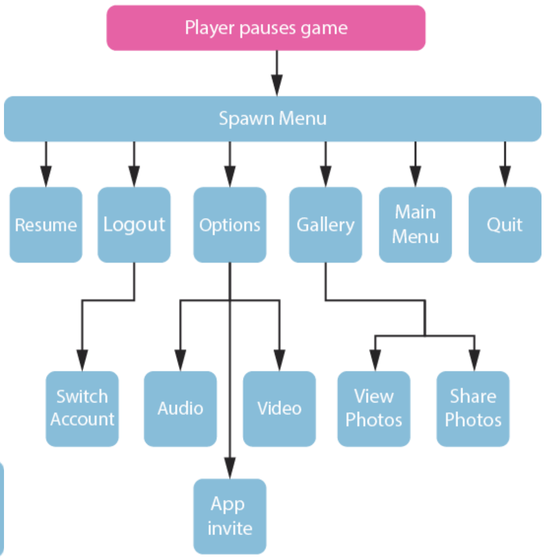
14. **Players Interaction:** The players will be able to see each other and to interact by doing activities together or by chatting. They will see the houses of the town differently because each one will be looking at a different version of the neighborhood, since each friend-list is different for a user.

15. **Menus**

    - **Main Menu:** Main menu will be similar to a Facebook login screen, where players can decide to login to the social platform and start the game, go to the options or quit the application (see Fig. 4.6a).



(a) Stream Diagram of Main Menu



(b) Stream Diagram of in-game Menu

Figure 4.6: Side by side comparison between Main Menu (picture 4.6a) and in-game Menu (picture 4.6b)

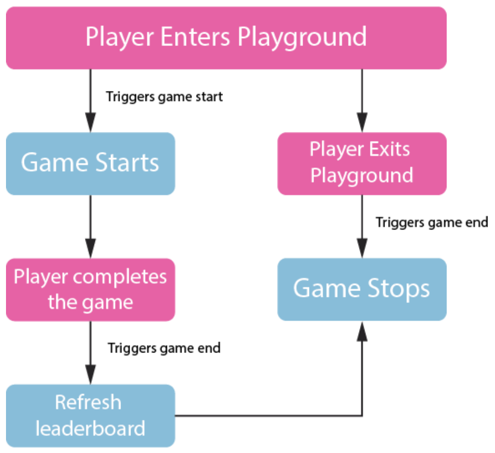
- **In-game Menu:** The menu inside the game session will be triggered by pressing a pause button (see Fig.4.6b) and it will offer the possibility to logout from Facebook, if the player wills to change account, and there is also a button to leave the game. Furthermore, there is the Options button here too, to adjust video and/or audio settings.
- **Gallery:** Players can also view, from the menus, a gallery of the photos they take in the game.
- **Game Saves:** The game saves automatically on regular intervals and when important events occur.

16. **Game Mechanics**

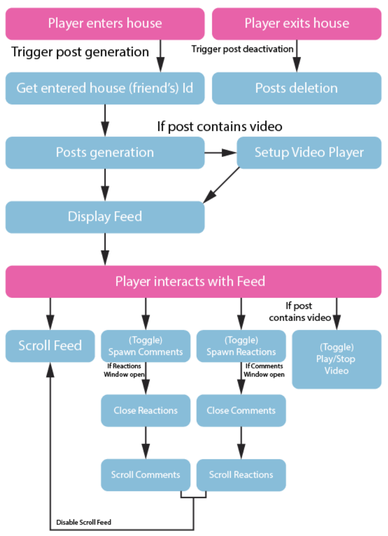
- **Command a customizable avatar to express your profile user in the game:**
  - Users can move the player with the keyboard for movement (PC) or the move joystick UI (Mobile);
  - Users can rotate the camera with the mouse (PC) or with the look joystick UI (Mobile);
- **Interact with other players/users:**
  - You can chat with friends by typing in a messaging box (see Fig.4.5b), always present on screen, or you can simply wave your hand in front of them by pressing a button.

- **Enter/Exit players homes:**
  - when a player goes in front of the door of a friend's house he can decide to enter by pressing a specific button on keyboard (PC) or on the screen (Mobile). In this case he will be moved into the house and can see the activity of the home owner, and if he is inside he can interact with him.
- **Taking pictures:**
  - By pressing a button you can enter photo mode and take stunning pictures! These pictures then can be viewed from the menu (see Fig.4.6) and shared on the platform.
- **Customize your home, avatar and city:**
  - Everything can be customized. The game is meant to propose firstly a version by inspecting the social tastes of the user like: followed pages, likes, music genres, etc...
- **Play always different mini-games into the main square:**
  - The main square shows the general feed of the user and his friends' but it includes also different activities to do, like: bowling and time obstacles race. And the results will be published on a giant leaderboard visible in the square (see Fig.4.7a).
- **Interact with Facebook posts (see Fig.4.7b):**
  - Like;
  - Comment;
  - Share;
  - Tag;
  - View a user's Feed, Photos, Videos and other personal info;
  - Publish content on Facebook from the app;
  - Photos taken in the game with a virtual camera, activities, etc ...

  Everything done using a 3D interaction system.
- **Challenge a user with a mini-game inside the 3D world**
  - Players will be able to challenge each other in different games, like: basketball, tennis, tug of war, volleyball, triathlon, etc...);
- **Invite Facebook friends from the app to connect:** As we saw in figure 4.6, the application will let users invite their friends or just allow them to see their personal info into the game. Since the game needs Facebook permission to work, except the name and the profile pic, the other info needs to be authorized if the user wants to enter the house of his friend to view the content.
- **Make new friends:** Add as a friend a person met in the main square or in other future common areas in the game;
- **Chat with friends:**
  - As shown on the picture before (see Fig. 4.4a), players can chat with everyone anytime;
- **Celebration events:** Inside the game you can participate, with your 3D character, in events organized in the square or at friends' homes. Events can be:



(a) This is how a common mini-game works



(b) Diagram of the Feed Generation and player's interaction with the feed

Figure 4.7: Examples of the mini-game a.i. (picture 4.7a) and Feed generation and interaction (picture 4.7b)

  - Birthday parties;
  - Barbeques;
  - Rallies in the square to vote for the most beautiful avatar (and win prizes);
  - etc...

- **Animals and pets:** Each player can have a pet that will follow him (see Fig.4.8);

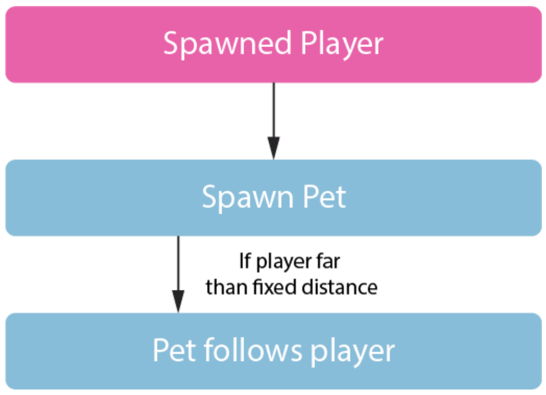


Figure 4.8: Pet a.i. stream diagram

17. **Critical points:** Since the application depends on an external entities, the available functionalities of the app will be children of the APIs available for the relevant platform and may be subject to change anytime, so it is important, when the app will launch, to establish a solid relationship between Facebook and the application.

4.2 Application Development

The overall development of the application can be subdivided into 3 main phases:

1. **Data scraping:** Testing of which social platform had the best API and gave as much permissions to download a good mole and quality of data;

2. **Semi-procedural Environment Design:** set up a Unity scene and test procedural algorithms to create the structure of the virtual town;

3. **Facebook integration:** download and integrate the Facebook SDK and then write C# scripts to make API calls to Facebook servers to download users data after entering a token to grant permissions.

4. **Interaction Design:** "translate" and adapt the downloaded Facebook data into a gamification experience to let players interact with them inside the game. Plus, develop a series of mini-games to experience.

5. **Server Hosting:** description of the execution and maintenance of the server delegated to host the application server-side, to host all of the clients connecting to use the online feature of the application, during the test.

4.2.1 Data scraping

As mentioned in Chapter 3, the first part required the research of the best social platform to test for this prototype. Using Visual Studio 2019, each platform's main packages were downloaded through the NuGet Packages panel. Then, thanks to the official documentations online, it was possible to write the API calls and see what data could be available to download and manipulate later. What follows is a detailed list of all the social platforms that were tested and their results:

1. **Twitter:** it was necessary to complete a specific form and wait for an approval for data scraping. Unfortunately, the approval requires a period of time to be confirmed (in the meantime this platform was left aside, concentrating on others);

2. **TikTok:** the API use request was rejected because it requires that the app using the data of this platform is officially published (requires the application signature and its URL on the Store, both for Android and iOS). Besides, it only allows integration with web or mobile applications;

3. **YouTube:** very immediate and simple use of the platform (only one API Key is required to copy and paste into the code) and many sample codes were available for different types queries as YouTube channels, playlists, videos starting with a desired word, and so on. Possible use in the app to open a video player for posts that host videos shared from YouTube platform;

4. **Instagram:** Instagram divides data collection in 2 categories:
   - **Graph API:** mostly used for business companies and content creators;
   - **Basic Display:** suited up to get basic info of a user profile as: photos, videos and IGTVs. This product was best fitting in opposite to Graph API but it did not provide enough info to use it in the prototype.

5. **Facebook:** the social media offers the “Facebook for Developers” platform which provides a dashboard, hosting all of the apps created in it (see Ref.[59]). There the developer can do all of the operations essential for the app management. The app is identified by a name, an app secret and a temporary token, for granting the users permissions. A useful tool for the data scraping is the “Graph API Explorer”. Developers can make http queries on information relating to a user of Facebook or Instagram, by setting up the Instagram Basic Display product for the latter. But this is possible only by creating an application on the developers platform to get an access token relating to the permissions requested (posts, images, videos, etc...) and by entering the user’s id in the query. On figure 4.10 we can see an example of the “Graph API Explorer”. Looking at the figure, first of all on top there is the query input field. It is preceded by the http method (GET/POST), then by the version of the API and then there is the main part where it is compulsory to specify the id of the user or of the post where we want to get the info from. On the left there are some hints of what we can search in the input field, depending by the context. On the top right there is the access token that lets the developer grant the access to do the queries. This token is temporal and needs to refresh. It is generated after specifying (on the right below) the permissions the developer needs. These permissions can vary and they can belong to: a user, a page, an Instagram account. Finally, in the middle we have the results of the query, in a “.json” file format. Below the results there is another section where the user can export the query method in different proposed languages to adapt it locally in their code. It also offers a debugger to test the sharing and validity of tokens. Besides, every app created in the Facebook for Developers portal, can be switched between Development and Live mode (see Ref.[60]). When the app is created, it is switched automatically to Development mode. Then, the developer tests the app and finally has to gain the needed permission by submitting the app to an overview process by the Facebook team, and it may take some days. After the approval, he can switch the app to Live mode, read for production. In development mode, the developer can delegate maximum 50 Facebook friends to test his app (see Ref.[61]), by asking them the permission to access their social data. They have to register in the developers portal too and confirm the collaboration request. **Testers** don’t have permissions to access insights or any other app settings. If the admin completes the business verification, then he can add more than 50 testers. It is possible to have more than one **administrator** and they can have complete access to the app, like: viewing insights, changing settings of the app and including and removing roles like these mentioned before, or resetting the app secret or even deleting the app. Then we have the **developers**. Except the administrators, they cannot have destructive access to the app. Indeed, they can test it, view insights, view submissions of the app review and change many of the settings but they cannot change the roles, delete the app or reset its secret. If the admin does not have Facebook friends or wants to create fictional users, then he can go to the Test Users section (see Ref.[62]). There he can create up to 2000 simulated Facebook accounts that can help him test various features of the app.

In conclusion, Facebook is the best choice among social networks that can be related more to the gaming application due to the variety of events and information it provides. Furthermore, there is a Facebook SDK package available for Unity mentioned in Chapter 3, that is used in the development of this prototype.

4.2.2 Semi-procedural Environment Design

Then, once the right platform was chosen, the next step was to understand how to integrate it into the game, and that was decided by creating the Design Document provided at the beginning of this chapter. The document says that the setting should be a semi-procedural world (see Fig.4.5a). So, the first step required the study of a grid system that included the generation of a set of houses and roads, which in the middle had a wide square, where all the players could meet. The attendance of a YouTube course (see Ref.[63]) on procedural polygon creation in Unity. It achieved the understanding on how to design a grid-shaped structure to automatically show the streets and the houses of Facebook users. First, the course consisted in the learning of the some of the basic components of a mesh in Unity: the “Mesh Filter” and the “Mesh Renderer”, from the Unity “Inspector” panel, which respectively contain information on the shape of the mesh (e.g.: cylinder, sphere, ...) and make the object enabled to rendering. Afterwards, it led to code the generation of the grid (see Cod.A.1). As we can see from the source code, the grid takes the friends collection, obtained by the Facebook API calls (see Cod.A.7) and starts spawning a squared grid, doing a spiral loop generation movement. On the odd cells it spawns just a plane, representing the road, while, on the even ones, it checks first how many players last in the collection to spawn. If there are still friends to spawn, then it generates a house, otherwise a road, like before. To optimize the code, the grid spawns first one half of the square, more precisely, one vertical side and horizontal side and then the opposites. The grid structure (see Fig.4.11a) was obtained by using two main prefabs:

- House;
- Road;

The prefabs were created using “Unity Standard Assets” first, and then improving them with new assets. The grid keeps always a squared shape. It spawns as many houses as the number of friends the user has. If the houses are finished, the algorithm keeps spawning roads on the empty space until it completes the shape of the square. In figure 4.11b we can see a visual representation of the case.

4.2.3 Facebook integration

Since the generated houses belong to users and their friends, it was necessary to use the API calls belonging to Facebook. Thanks to the “Facebook SDK for Unity” package and by consulting the official documentation (see Ref. [64]), the API calls were made operative. First of all it was necessary the creation of an empty object in the main scene and a script to attach to it. Then, it was built a Canvas element with a button to trigger the Login procedure. Unity provides a slot to call a function from a script when clicking a button. So, the script created before was used to make the API call to start the Login process. Since this is not the final build, Facebook SDK works only with the access token, that needs to be copied from the “Graph Api Explorer” tool, while in the build version, the call prompts to a redirect window where the user has to enter his Facebook credentials to authorize the permissions necessary to make the application work. But, to make the login prompt possible, it was necessary to come back to Developers Portal to install an important product from the Dashboard: **Facebook Login** (see Ref.[65]), more precisely, Facebook Login for Android. After the installation, it was necessary to begin the Quick Start for Android devices and it required to enter the Package Name and the Debug Android Key Hash to make the testing of the app possible. After the login confirm, the script has the permissions needed to proceed forward. Indeed, in the next step, another method downloaded user's name and profile pic to display his name on the house sign (see Figs.4.12a and 4.12b) for the main user and his friends.

Afterwards, the API calls were used to download the "Feed" when entering each house. Facebook's Feed is composed by a series of publications from users and advertisers called "Posts". Each post on Facebook (see Fig. 4.13a) is made of several elements that belong to the author that published it:

- Name, Surname and Profile Picture of the publisher;
- Publishing date;
- Publishing location;
- The media displayed in the post (Photo or Video);
- Post description;
- Reactions and comments of others users towards the post;
- Link to the source, if the post contains a link;
- People tagged in the post;

So, to keep the same user experience it was necessary to recreate, in Unity, a prefab with a very similar aspect (see Fig. 4.13b). On the same way of Facebook, in the game it was implemented a script to let players look at every information about the posts, such as: see the reactions and the comments or watching a video if a post contains a Facebook video (see Fig.4.7b).

Facebook SDK functions

The application had to make API calls with Facebook servers to download the relevant data to provide users the desired content. But first of all, the app prompts to the login page to let users enter their account credentials and allow all of the needed permissions. Consulting the Facebook fo Developers documentation, about the Facebook SDK for Unity (see Ref. [66]) there were some important API calls to insert in the code containing the Facebook calls manager to let the application gain the permissions and download all of the useful user data. First, it was necessary to import the package by typing the "using" directive before the class in the script containing the Facebook API initializer. Then were used the following methods:

- **SDK initialization:** in the "Awake" method, it was important to check, at first sight, if the SDK was already initialized, if so, then it sent the signal to start the application (see Cod. A.2). Otherwise, it calls *FB.Init* to initialize it. FB.Init contains two parameters that are methods as well: *SetInit* and *OnHideUnity*. SetInit checks if the application managed to start the SDK, in case positive it goes ahead with the next interactions with it, otherwise it aborts, leaving an error message. OnHideUnity instead, carries a boolean parameter that simply tells if the game is being shown. In case negative, it freezes the game flow, otherwise the method unfreezes it and user can start playing.
- **Facebook Login:** Once the SDK is initialized. User sees a simple window with a button stating "Continue with Facebook" (see Fig. 4.14). Clicking the button calls the "Login" method (see Cod. A.3). Inside the method, can be

specified a list of permission, these permissions are used from the application to download user content from Facebook to display his content in the game, like the Feed, personal photos, photo locations, comments and reactions and so on. Doing a login with read permissions simply makes the app download content, while the write permissions let the player post stuff on Facebook too, but in this case it concentrates only on read permissions. The *Logout* method instead, makes the menu come back to login screen again.

- **Authentication:** The FB.LogInWithReadPermissions method mentioned before, calls another function which contains a parameter of type *ILoginResult* (see Cod. A.4). It is just the result that tells if the login procedure succeeded. in case negative, the method displays the API error, otherwise it calls the *DealWithFBMenus* method to prepare the game UI, stating that the user managed to login.

- **Menu managing:** The *DealWithFBMenus* method checks if the player is logged in (see Cod. A.5). In case he is, the conditional branch starts to make a set of API calls to Facebook, requesting some of the user fields, thanks to the permissions gained at the beginning of the login process. The calls exploit the HTTP protocol, and more in the specific, the GET asynchronous method, and then it delegates a function that manages the response. Afterwards, it calls two other async methods, by using the *coroutine* method provided by Unity (see Ref. [67]). The first method awaits until user's name and profile picture are downloaded and set, then the other one disables the login menu to finally show the game. There is another important method later, necessary to spawn all of the houses of the friends of the user. It uses the field "friends" and the first name of them. Later on it will be discussed in deep. In case the user fails to log in, or decides to log out, the app does the opposite, i.e. it hides the game UI and shows the login menu again.

- **Facebook Name and Profile Picture:** The last method, exploited the Facebook API call and the user permission "first_name" to get the name of the account. Looking at the method in detail (see Cod. A.6) we notice that it contains a parameter of type *IResult* named "result". This is another element belonging to the Facebook SDK for Unity (see Ref. [34]) and it gathers the result of the async HTTP call to get the name of the user. Now the method checks if there is an error, and this operation is a standard for all of the API calls that return an IResult value. Before the statement, though, there is a variable that is linked to the *Text* UI Unity element. If result value is correct, the UserName Text element will add the name after the statement "Hi there, " to greet the user. The subsequent boolean value is important for the coroutine that awaits for it to be set, to show it on screen. The method to display the profile picture works in a similar way. This time, though, the result is of type *IGraphResult*; it is used to manage media elements. If the picture has been downloaded succesfully, then the UI component holding it creates a sprite texture with the desired size and holds it. Then, the boolean value is set to true and the coroutine shows the "Login Succesful" screen, containing a welcome message to the name of the user and showing his profile picture just set.

- **User's friends:** As mentioned before, the method *GetFriends* is necessary to spawn the houses around the center of the town. As the other Facebook API calls, this one has an IResult value too (see Cod. A.7), named result. If it is correct, then the method gets the *json* file from it and parses all of the data. Then it takes the total number of friends directly from the *summary* field. Then it creates a Dictionary to save, for each entry, the name and the id of each friend. First of all it tries to save the name of the main user: if these data are absent, then it catches the exception and clears the friendlist and then logs out the user to retry the login correctly. But if it goes ahead correctly, then it checks if there are friends and in case positive it saves the id and the name of each of them in the Dictionary. Lately, it calls the method *UpdateFriendlist* of the class *GridGen* to start spawning the houses in the game map.

- **House sign generation:** When the generator spawns a house, then it takes its script and orders to create the sign standing outside of it (see Fig. 5.7a). The *CreateHouseSign* method (see Cod. A.8) sets the profile picture and the name of the user on the sign. Both of the methods work in the same way of the methods used to set user's anme and profile pic in the login screen. The *DisplayPic* method, in this case differs on the fact that it takes the texture value of the IGraphResult and creates a new material to associate the profile picture as the main texture. Then, it takes the *Material* component of the *ProfilePic* element of the house sign and sets the new material to it. The *SetUsername* method will be used to get the name of the user to show the Feed when players will enter his house. The name on the sign instead is taken from the parameter passed to the *CreateHouseSign* method.

- **Feed creation:** To get the data of the Feed, the program exploits always the Facebook API call that uses the HTTP Get method (see Cod. A.9). It requires the posts of the user and then it specifies which fields of the posts (description, date, comments, etc...). Then, the *getPosts* method takes the result of the HTTP query and de-serializes the json file. Then it gets the user name from his house, as mentioned before, and starts spawning the posts inside a List named *_postsInstances*. For each post created, it calls the method *print_feed*, present in each Post prefab, to assign the downloaded properties to all of its fields (see Fig. 5.10a).

4.2.4 Interaction Design

The core feature that gives a huge importance to interaction is the multiplayer component. Players should be able to enter the town square and meet other users that are wandering around there. Since **Mirror** is the correct candidate to manage a large scale of connections, the package was imported into the project. First of all, reading the official documentation (see Ref.[68]), to implement the multiplayer feature, it was necessary to create and empty object and to attach the "NetworkManager" script to it. The "NetworkManager", as the name states, manages all the connections and network interactions between Clients and Server. When a player spawns in the game, it is duty of the "NetworkManager" to make him connect, as a client, to the server. The developer has to specify the player prefab to spawn and the spawn position. Another component that is automatically attached to the "NetworkManager" empty GameObject is the KCP Transport. The kcp2k - KCP Transport (see Ref.[69]) is Mirror's new default Transport. It is one of the low level transports available on Mirror. It is extremely fast and simple and has a test coverage around 83.5%. (see Fig. 4.15) The KCP has a good performance in wifi and mobile network (3G, 4G). If the network is never congested, it has a performance similar to TCP.

Each Unity “GameObject” that has a multiplayer behaviour must have attached one or all of the following components, depending on the use we have to make with it:

- **Network Identity:** this component is the core component of the Unity networking high-level API. It confers the game object that owns it, a unique ID, to make the networking system aware of it. It gives the possibility to spawn the game object only server-side on not on the clients, by checking the “Server only” checkbox (see Fig.4.16). The component was applied to every element that had to deal with the networking system;
- **Network Transform:** to let the other players watch the space transformations of an element, like: position, rotation and scale, it is necessary to attach this element to the GameObject. The component contains a checkbox option called “Client Authority” whose activation makes the client send information to the server about the changes on the object’s transformations. Then the server propagates these changes to all the other clients to refresh its last state, from their point of view.
- **Network Animator:** as the name says, thanks to this component, the other clients can synchronize the animation info of the other clients, thanks again to the server, that sends it. Even in this case it was used the “Client Authority” option;

The previous three elements introduced above were used to implements the avatar the user commands in the game.

The following, is the description of the interaction of the player with the several activities to do around the town.

House and square visibility behaviour

When the player arrives in front of one of the houses in town, after looking at the door, an interaction icon appears and it invites him to enter the house. Player’s door detection is developed by writing a script that generates a “RayCast” (see Ref.[72]), therefore a ray of a specific distance, that projects from the front of the player, in this case. And when it collides with the door it checks if the door “Collider” has the desired tag to trigger the enter option, by using a layer mask representing the door, on a cube shaped mesh, checked as a trigger, to avoid collisions with the player but only detect the raycast (see Ref.[73]). Once entering the house, the player teleports on the first building and its sign cartel replaces the name and profile picture with the infos belonging to the owner of the house he decided to enter. Same thing happens with the feed. This decision is fundamental to make players, that entered that house, meet at the same place. Since the players are outside of the square but need to see each other, it was necessary to add another option on the “Interest Management” script (see Ref.[74]). The “Interest Management” (see Fig. 4.17) is used to broadcast the world state to the clients. Instead of sending the full world state to every player, it’s optimizing sending to the player only what’s around him, given a fixed distance. So, for this work, it was attached a script to the square, that inherits from the “Interest Management” class and it managed the distance between the players with it or with the house they entered, and it was necessary to render their mesh. It was important also to activate and deactivate players’ collisions since when wandering outside the interest areas they could not collide each other. Notice that only one “Interest Management” script at the time can be present in the scene.

Feed interaction

Once inside the house, the avatar can wander around, looking at all the rooms (see Fig. 5.7), go in the garden and kick a soccer ball or he can also watch the Feed of the house owner. As described in the stream diagram (see Fig. 4.7b). To interact with the feed, the user has to press a specific key or button, in that case he can trigger the activation of the Feed of the house owner, that may be his or the one of his friend. The house script recognizes that the Feed has been triggered when the player gets inside the house and sends an API call to Facebook to retrieve all the necessary data mentioned in subsection 4.2.3. Then it displays the Feed on one of the house walls. Players can interact with it by going nearby and pressing a specific key or button to enable the feed view. Then, the game switches the camera: deactivating player's one and activating the one belonging to the feed. Then he can interact with it thanks to the following actions : scrolling, toggling comments and reactions or by playing videos, if the post contains one. More precisely, on the PC version, the feed camera activates the mouse cursor visibility and allows scrolling, to roll the page to watch all of the posts, the likes and the reactions contained in each one, or he can play videos too (see Fig. 5.10). The standard mouse cursor is the default arrow, but when the player hovers it on the clickable elements it turns into a hand cursor pointing a finger (see Fig. 5.10c). In the mobile version, instead, the user can swipe the screen to go up and down with the posts, and reactions, comments and videos can be displayed directly by tapping the screen. Once the user presses the escape key, or the escape button, he stops the Feed interaction and the camera comes back to player and he can wander again. The feed shuts down when the player leaves the house and refreshes whenever he decides to come back.

Minigames interaction

Players can decide to play mini-games in the public square or other future games that will be present inside the houses, during celebrations. As shown in figure 4.7a, when the user enters the game area or presses a specific button, it triggers the game start interaction, and the game system and user interface set up. In case the game foresaw a race, it activates a stopwatch, otherwise if it concerns a certain score reached, then it displays the points achieved. If the player leaves prematurely the game, nothing happens, but if he completes it, the game a.i. checks if he did a new record, and in case positive, it does a refresh on the global leaderboard, by demanding the server to refresh it on all connected clients. Some of the mini-games also unlock temporary special abilites. E.g. in case the player decided to try the obstacles race, it would enable the "jump" button on the screen (Mobile) or would enable the Space key to make him jump (PC) (see Fig.5.8b).

Elevator interaction

Inside the square, to access one of the games, there is a stair but also an elevator, doing a loop movement, up and down. The movement was achieved with an animation, thanks to Unity's "Timeline" component, where developers can key-frame an animation, by specifying also the animation curves. In this case, the elevator does a linear loop. A trigger is linked on top of the elevator. It has a specific function: when the player climbs on it, the trigger recognizes him and then calls a method that makes the player have the same position of the elevator along the ride, and when he exits the trigger, his position becomes independent (See Cod. A.11).

Gamertags display

Before proceeding with the explanation of the development of this feature. It is important to introduce three Mirror's fundamental Remote Actions (see Ref.[75]). The network system can perform actions across the network in many ways (see Fig. 4.18). These actions are often called "Remote Procedure Calls". For this work, in general, and specifically, for the gamertag behaviour, two important actions were used:

1. **Command:** The "[Command]" tag typed in the code, above the declaration of a specific function, makes it send a signal from the client's script running that code to the Server;

2. **ClientRpc:** ClientRpc calls are sent from elements on the server to objects on clients. They can be sent from any server object spawned, with a NetworkIdentity. To make a function into a ClientRpc call, users add the tag "[ClientRpc]" to it, and add the "Rpc" prefix. The functions owning this tag, will be run on clients when it is called on the server. All of the arguments will be passed automatically to the clients with the ClientRpc call.

Concurrently with the "Command" action, there was an element tied to the player name variable, to synchronize its visibility to all the other clients, and that is the **SyncVar Hook**. A SyncVar Hook (see Ref.[76]), except a Sync Var, is a property of classes that inherits from NetworkBehaviour (see Ref.[77]) and it is synchronized between Clients and Servers. More precisely, when the value of that property changes, e.g.: the name of the player is downloaded and displayed, it updates the Server, that refreshes the value on the other Clients. The "hook" attribute can be used to *link a function* to call when the variable's value changes. In this case, the script displays on top of the player the name, downloaded from the Facebook API call. The hook method must have only two parameters of the same type of the property linked to SyncVar: one for the old value and one for the new one. For this feature was used the SyncVar Hook. In the class representing the player (see Cod. A.12), were introduced the variables and the methods to get the name and place it on the TextMeshPro UI element on top of the head of the other players, by checking, with the NetworkBehaviour class if the player running the code was local or remote. So, in the other case, if it was the local player, he would disable his name but from the remote players' perspective it would remain visible. Then, in the script attached to his camera, the names of the remote players were gathered together and every one would look at the camera every frame, to make their names clearly visible, every direction the player looked at.

Camera behaviour

Every players' camera will constantly look at the own player parent. When it rotates, it rotates also the player. When the player moves into narrow spaces, like the house (see Cod. A.13), to keep him always visibile and avoid other objects pop between him and the camera, it was typed a script to make the camera always look at him, by coming forward when there is a visible obstacle, or go back to the maximum set distance, in a wide area. The camera detects the presence of the obstacles by projecting a Linecast (see Ref.[78]) from the center of the camera until the player. If in between something different is detected, the camera moves forward, smoothly, until it finds the player.

Chat behaviour

The chat behaviour works with the same logic of the leaderboard update (see Cod. A.10). To communicate each other, players can use a global chat window where they can type and send several messages. To send a message (see Fig. 4.5b), every client, must firstly own an authority, and it does when both client and server initialize. Then, the chat user interface is activated and the player can tap on the input field to enter the message. After clicking the "Send" button (Mobile) or pressing the "Return" key (PC), the script checks if the input field contains a text or there are no errors. In case positive, it sends the message from the client to the server, through a "Command" directive. If the client disconnects it sends a callback named: "Client-Callback". This tag is powered by Mirror's NetworkBehaviour scripts (see Ref.[77]). Client and ClientCallback tags are placed on top of methods to make them return immediately when the server is not active. When the server receives the message correctly, then it calls a method with the "[ClientRpc]" attribute to refresh the chat box in all of the clients connected and the input field of the sender is cleared. Notice that when playing from PC, the script freezes player's movement when focusing on input field to prevent him from moving while pressing sensitive keys when typing the message.

Pet interaction

When the player spawns, after the login, if he had chosen a pet, it spawns subsequently after him, nearby. Every pet has a simple artificial intelligence (see Fig. 4.8) that does the following actions: it checks if the player is more far than a fixed distance. In case positive, it follows his owner until it is next to him, otherwise it stands next to him.

Animation design

The scripts linked to the player, his pet and the elevator involved the help of an "Animator Controller" component. The Animator Controller (see Ref.[79]) is a Unity legacy component that allows the developer to set and keep a collection of Animation Clips and associated Animation Transitions for a game object. In many cases, it may have multiple animation and switch between them when certain conditions occur during the game. During this work it was used with the pet (see Fig. 4.19a) to switch between the animations "walk" and "sit" whenever the condition of player's distance was from it was refreshed. Another case is when the player is moving or not (see Fig. 4.19b), and it is linked to the condition by the value of his speed. The player can also switch from the walk to the run animation when the user moves the joystick UI more far than the centre (Mobile) or when he presses the "run" key (PC). However, even if we have only one animation clip, e.g. the elevator loop in the square (see Fig. 4.19c), it is compulsory to place it into an Animator Controller to enable it in the application. In most of the cases, the animation clips were already present in the assets downloaded but the elevator's animation was created in the "Timeline" component (see Fig. 4.20) and whenever the player climbed on it, a trigger placed above made him have the same position all of the time the player stood on.

Mobile and PC controls

The UI had to be matching for different device screens, so it was important to check that the elements could fit correctly. This was made simple thanks to the Preview

Package “Device Simulator”, to check directly the response of the UI according to the different screen sizes of each device (see Fig. 4.21). Besides, thanks to the platform directives (see Ref.[80]), it helped to test also the gameplay features available only in one device or in the others, like the UI buttons and joystick.

4.2.5 Server Hosting

When the prototype was ready, it was necessary to create a server and execute it to let players meet each other, in the multiplayer session. Mirror comes in help with it thanks to the documentation to create a server to host the clients connecting through the app (see Ref.[81]). First of all, it was necessary to create an account and register a payment method. Then, in the AWS Management Console, it was chosen the option to launch a virtual machine with EC2 (see Fig. 4.22) and was specified the Microsoft Windows Server 2019 Base AMI (Amazon Machine Image) (see Fig. 4.23). Afterwards it was selected the instance type with “free tier eligible” (see Fig. 4.24) and the instance did not need any tyoe of configuration. So, in the next step it was added a storage of minimum 30 gigabytes, and later the tags were left untouched. Once in the Security Groups section, it was iportant to enter the right values to make my server instance reachable from outside clients. So, I had to specify: the protocol type, in this case Custom UDP and protocol UDP, since it is used by the KCP component in the Mirror NetworkManager of the project. Then, the port of value 7777, which is the same set in the KCP component but it can be changed accordingly. Lately, the range of IP addresses that can reach the instance, in this case every IPV4 address, and it was to be inserted a description for this security group (see Fig. 4.25). On the next step where there is the review and the conferm of all of the choices just made before. Once it all is correct, it is possible to launch the instance (see Fig. 4.26). Now, a window popped up to let the choice to create a new keypair or to chose an existing one. So a new RSA keypair was created and it was given an appropriate name and then it was donwloaded into a safe place (see Fig. 4.27). Now the instance can be really launched. The Keypair is important afterwards to access the created instance. Going back to the EC2 dashboard we can access the Services panel and then EC2 to see all the EC2 running instances (see Fig. 4.28). By clicking on our instance and then “Connect” it gives the opportunity to download a Remote Desktop File (RDP), necessary to interact with the server instance. First of all, it is important to get the password used to connect to the server. It can be achieved by retrieving the keypair generated before, then we can decifer it and save it (see Fig. 4.29). Now the RDP can be dowloaded. Before opening it, I made possible to let the server see the main drive of the local machine (see Fig. 4.30). Then, in Unity, a Windows Standalone Server Build of the application was made (see Fig. 4.31). It was after zipped and moved to the drive mentioned before. Now we ca run the RDP file and enter the password saved before. Once the Windows Server communication was open, it had to be set an inbound rule in the firewall settings to let the incoming UDP connection, from port 7777, enter the device (see Fig. 4.32). Afterwards, going to “This PC” section, the drive of the local machine could be seen (see Fig. 4.33). Now, the file stored before could be unzipped on the instance and then it could be run and set ready to listen for any possible connection.

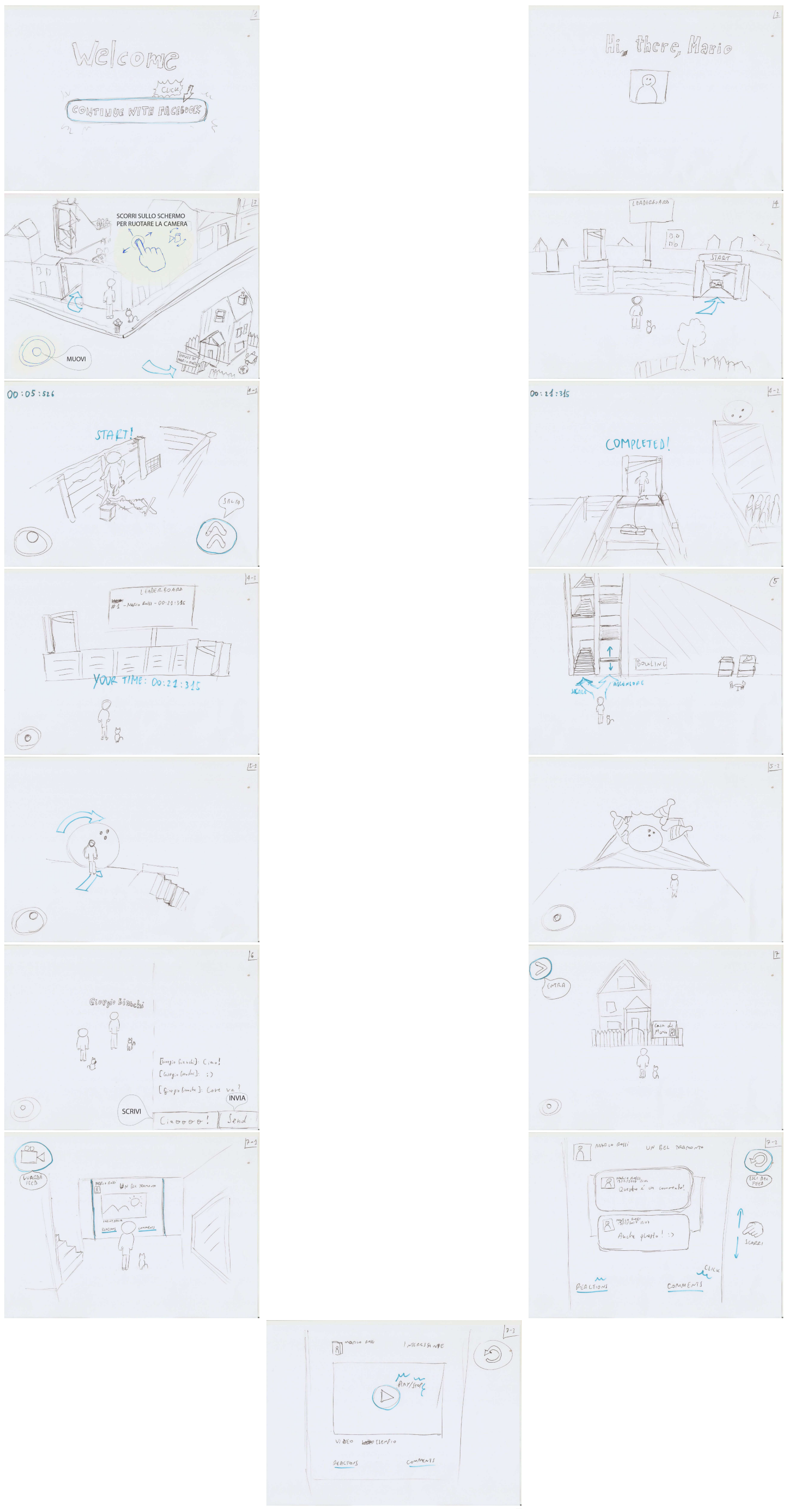
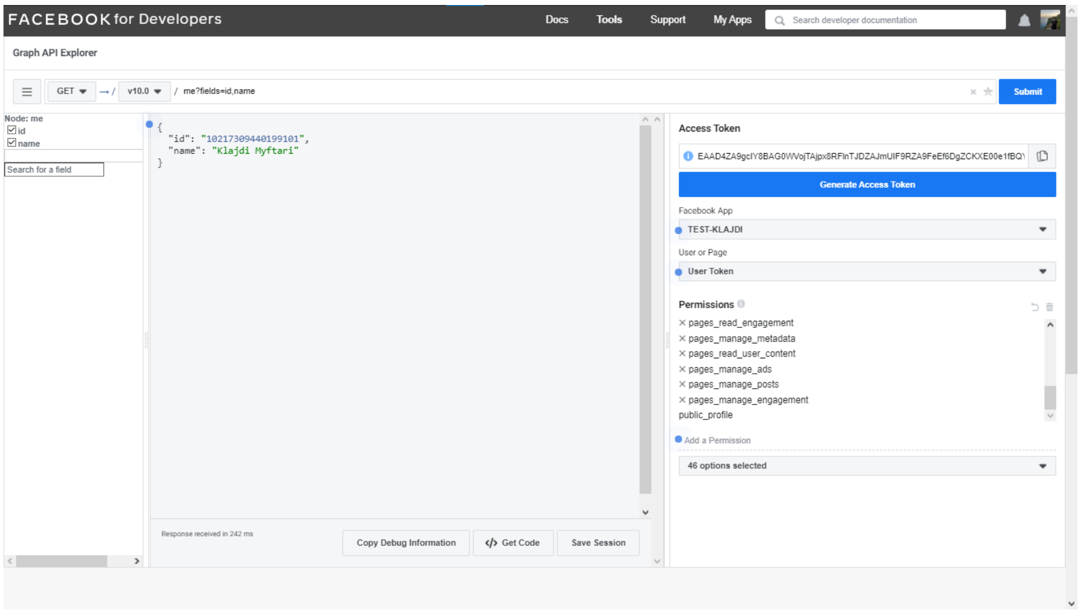


Figure 4.9: Storyboard (read left to right) describing one user scenario extracted from the Use Case Diagram (see Fig. 4.1)



GET → / v10.0 / me?fields=id,name

Node: me
☑ id
☑ name

```
{
  "id": "10217309440199101",
  "name": "Klajdi Myftari"
}
```

Access Token
EAAD4ZA9gcIY8BAG0WVojTAjpx8RFInTJDZAJmUIF9RZA9FeEf6DgZCKXE00e1fBQ

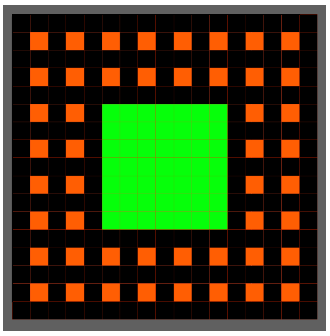
Facebook App
TEST-KLAJDI

User or Page
User Token

Permissions
× pages_read_engagement
× pages_manage_metadata
× pages_read_user_content
× pages_manage_ads
× pages_manage_posts
× pages_manage_engagement
public_profile

46 options selected

Figure 4.10: Graph API Explorer layout on “Facebook for developers” platform


(a) Spiral Grid concept: in orange the houses and in black the roads, while the square in green

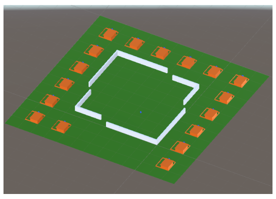
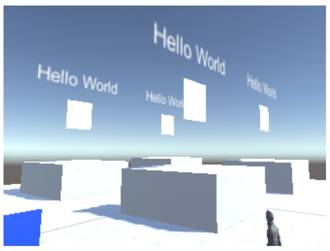
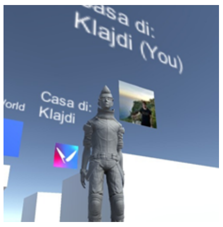

(b) Grid generation in case the number of houses could not fit all the space

Figure 4.11: Comparison between town’s grid concept (picture 4.11a) and perspective view of it (picture 4.11b)

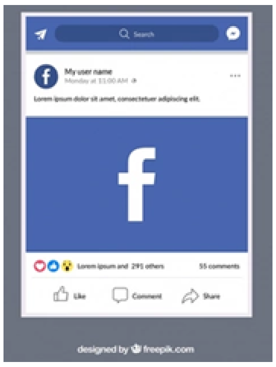


(a) before

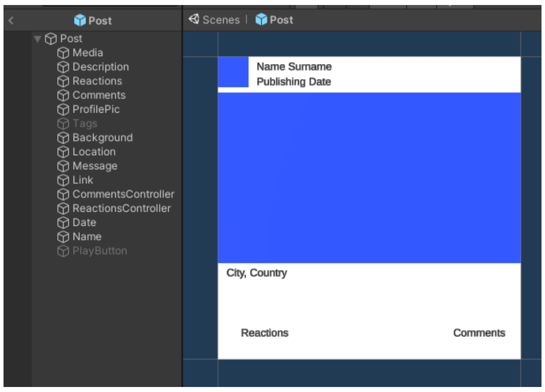


(b) after

Figure 4.12: Houses in-game before and after the download of info through the Facebook SDK package



(a) Facebook Post



(b) Unity Post

Figure 4.13: Side by side comparison between a generic Post on Facebook (picture 4.13a) and the prefab Post created on Unity (picture 4.13b)

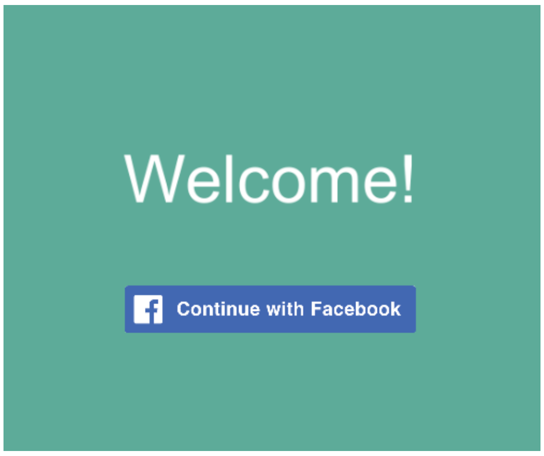


Figure 4.14: Facebook Login Menu when opening the prototype application

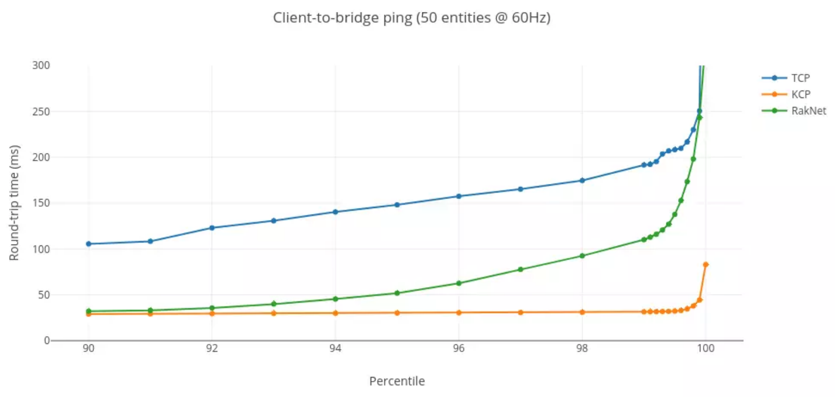
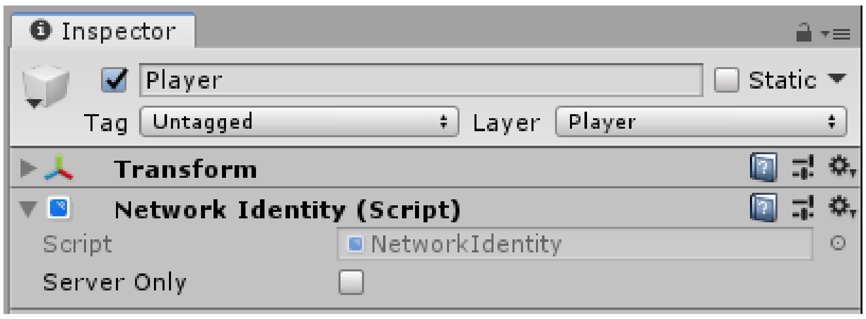


Figure 4.15: Round-trip time percentile of KCP transport against TCP and RakNet transports (see Ref.[70])



Script: NetworkIdentity
Server Only: ☐

Figure 4.16: Mirror's Network Identity component in Unity Inspector window (see Ref.[71])

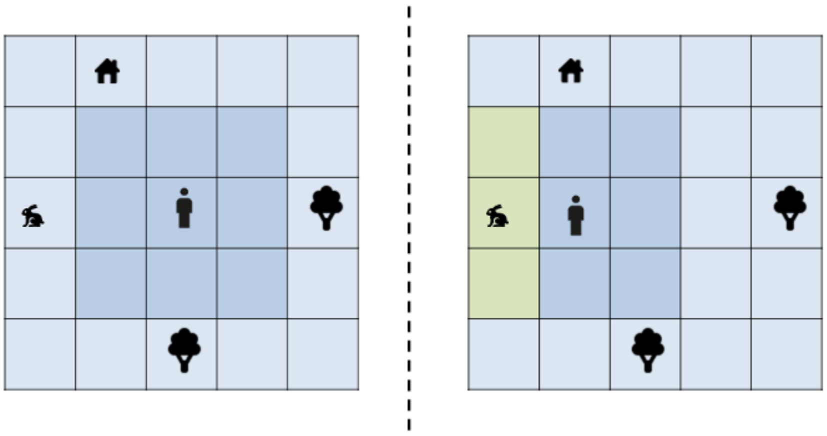


Figure 4.17: Mirror's Interest Management behaviour concept. As we can see: when the player approaches a specific target, the latter becomes visible (see Ref.[74])

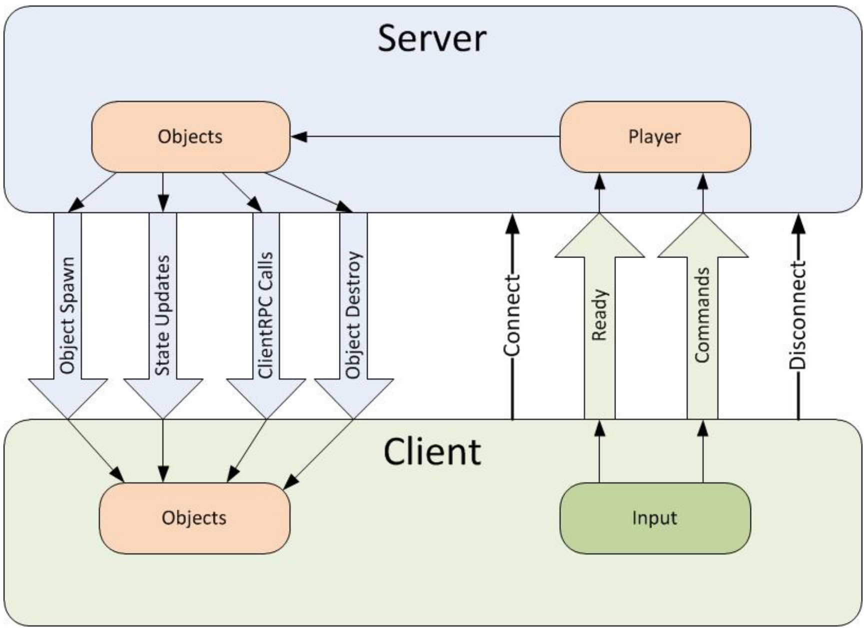
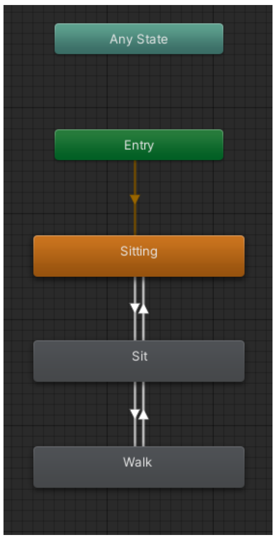
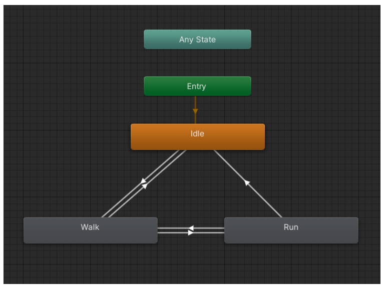


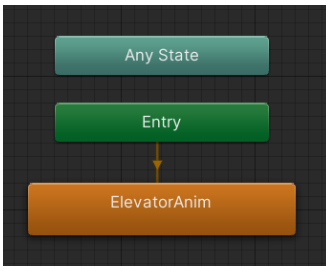
Figure 4.18: Diagram, from Mirror Documentation, that shows the possible remote actions between Clients and Server (see Ref.[75])



(a) Pet's Animator Controller



(b) Player's Animator Controller



(c) Elevator's Animator Controller

Figure 4.19: Side by side comparison between the Animator Controllers used to switch between the animations of the cat (picture 4.19a), of the player (picture 4.19b) and of the elevator (picture 4.19c)

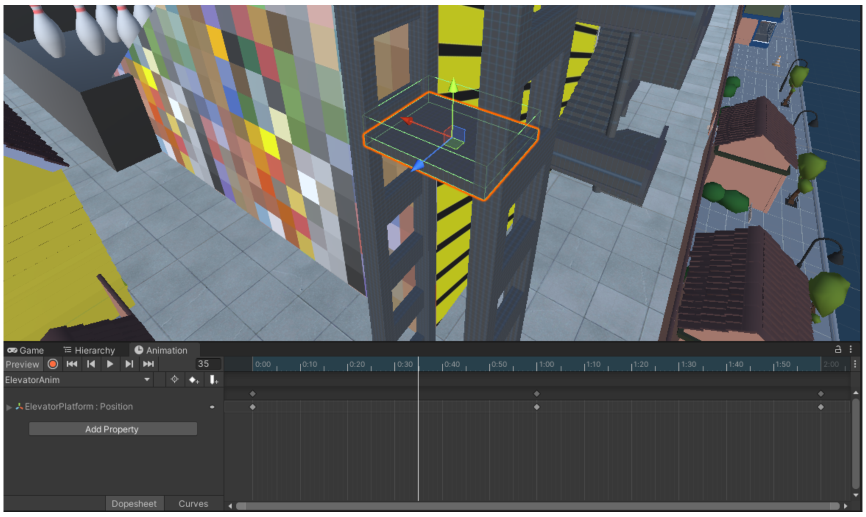
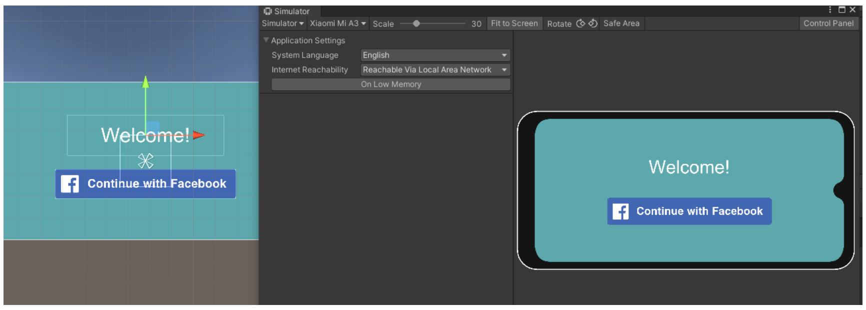
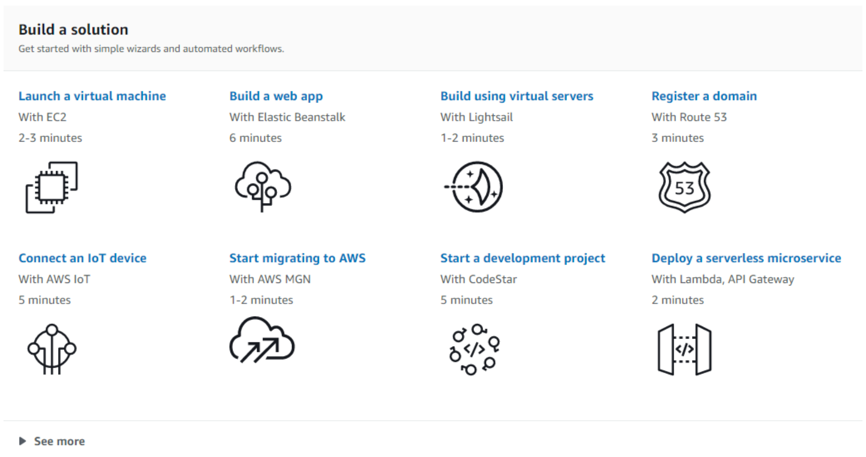


Figure 4.20: A picture of the timeline used to animate the loop movement of the elevator in the square. The green outline above the elevator is the trigger that sticks the player to it



Application Settings
System Language: English
Internet Reachability: Reachable Via Local Area Network

Figure 4.21: An example of the Device Simulator Window. Inside the Control panel (in the middle of the figure) it is possible to change: system languge, internet reachability and to trigger low memory situations

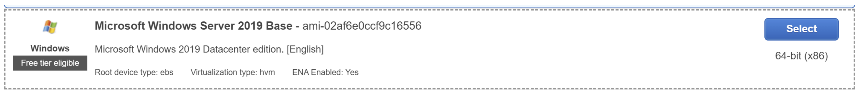


Build a solution
Get started with simple wizards and automated workflows.

Launch a virtual machine With EC2 2-3 minutes	Build a web app With Elastic Beanstalk 6 minutes	Build using virtual servers With Lightsail 1-2 minutes	Register a domain With Route 53 3 minutes
Connect an IoT device With AWS IoT 5 minutes	Start migrating to AWS With AWS MGN 1-2 minutes	Start a development project With CodeStar 5 minutes	Deploy a serverless microservice With Lambda, API Gateway 2 minutes

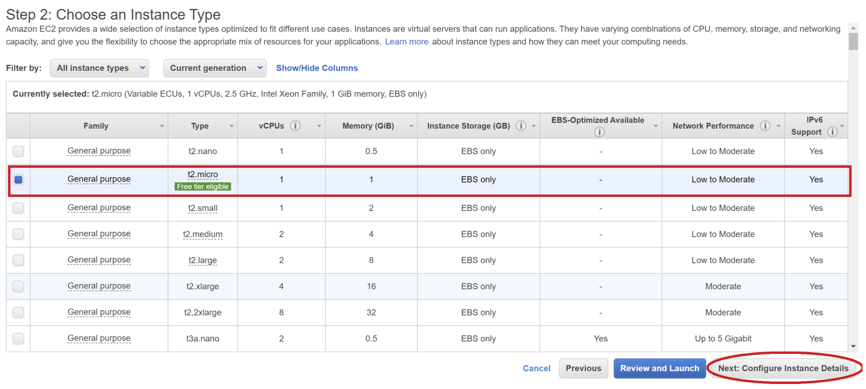
▶ See more

Figure 4.22: Amazon Web Services Dashboard, "Build a solution" section, showing the possible options, including the EC2 Virtual Machine



Microsoft Windows Server 2019 Base - ami-02af6e0ccf9c16556
Microsoft Windows 2019 Datacenter edition. [English]
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
64-bit (x86)

Figure 4.23: Microsoft Windows Server 2019 Base AMI solution (see Ref. [81])



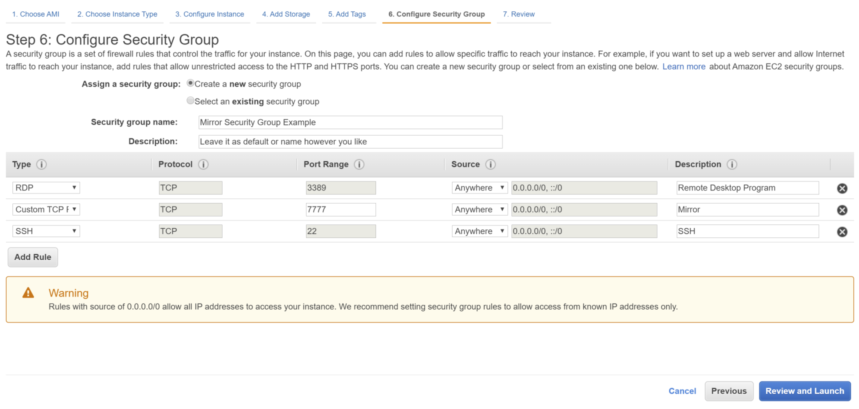
Step 2: Choose an Instance Type
Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Filter by: All instance types | Current generation | Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
☑	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
☐	General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
☐	General purpose	t3a.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes

Figure 4.24: Instance type choice options during EC2 Virtual Machine building (see Ref. [81])



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: [✓ Create a **new** security group] Select an **existing** security group

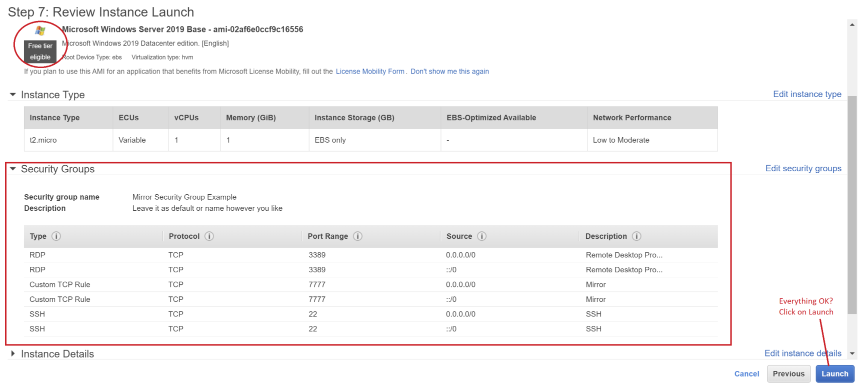
Security group name: Mirror Security Group Example

Description: Leave it as default or name however you like

Type	Protocol	Port Range	Source		Description
RDP	TCP	3389	Anywhere	0.0.0.0/0, ::/0	Remote Desktop Program
Custom TCP F	TCP	7777	Anywhere	0.0.0.0/0, ::/0	Mirror
SSH	TCP	22	Anywhere	0.0.0.0/0, ::/0	SSH

Warning
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Figure 4.25: Security Group configuration during EC2 Virtual Machine building (see Ref. [81])


Step 7: Review Instance Launch

Microsoft Windows Server 2019 Base - ami-02af6e0ccf9c16556
Microsoft Windows 2019 Datacenter edition. [English]
Root Device Type: ebs Virtualization type: hvm
If you plan to use this AMI for an application that benefits from Microsoft License Mobility, fill out the License Mobility Form. Don't show me this again

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

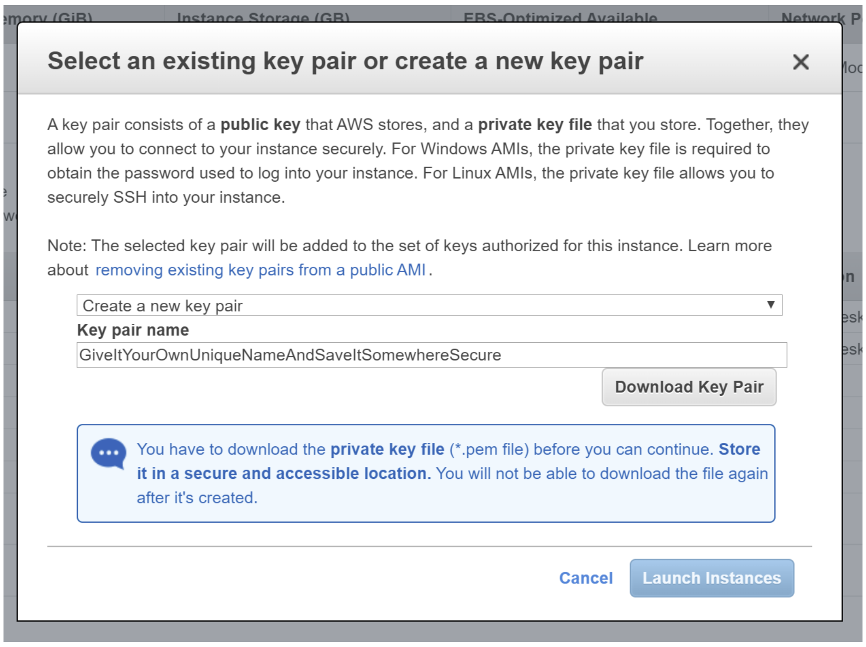
Security Groups

Security group name Mirror Security Group Example
Description Leave it as default or name however you like

Type	Protocol	Port Range	Source	Description
RDP	TCP	3389	0.0.0.0/0	Remote Desktop Pro...
RDP	TCP	3389	::/0	Remote Desktop Pro...
Custom TCP Rule	TCP	7777	0.0.0.0/0	Mirror
Custom TCP Rule	TCP	7777	::/0	Mirror
SSH	TCP	22	0.0.0.0/0	SSH
SSH	TCP	22	::/0	SSH

Instance Details

Figure 4.26: Review Instance Launch during EC2 Virtual Machine building (see Ref. [81])


Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

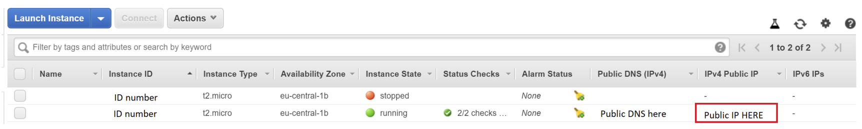
Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Create a new key pair

Key pair name
GiveItYourOwnUniqueNameAndSaveItSomewhereSecure

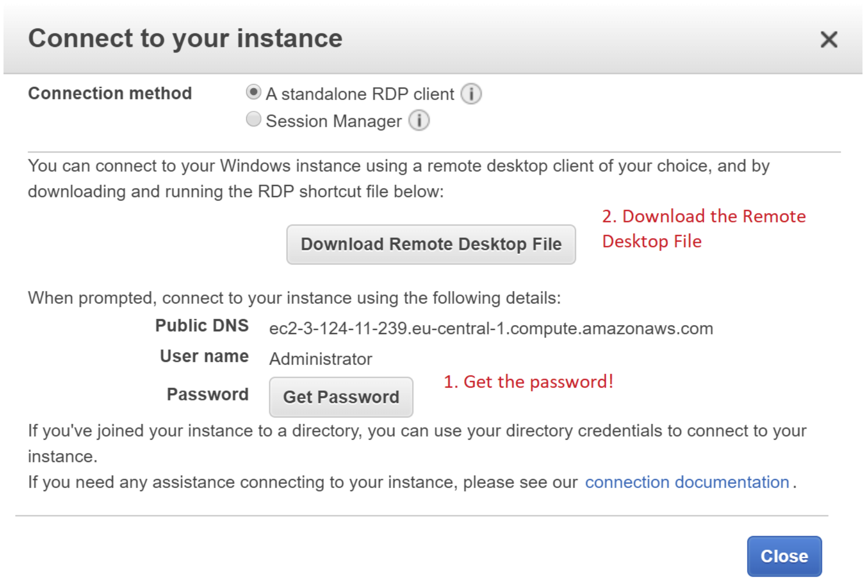
You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Figure 4.27: Keypair options popup window (see Ref. [81])



	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs
		ID number	t2.micro	eu-central-1b	🔴 stopped		*None*		-	-
		ID number	t2.micro	eu-central-1b	🟢 running	✓ 2/2 checks ...	*None*	Public DNS here	Public IP HERE	-

Figure 4.28: Running EC2 instances window (see Ref. [81])


Connect to your instance

Connection method [✓ A standalone RDP client] Session Manager

You can connect to your Windows instance using a remote desktop client of your choice, and by downloading and running the RDP shortcut file below:

Download Remote Desktop File
2. Download the Remote Desktop File

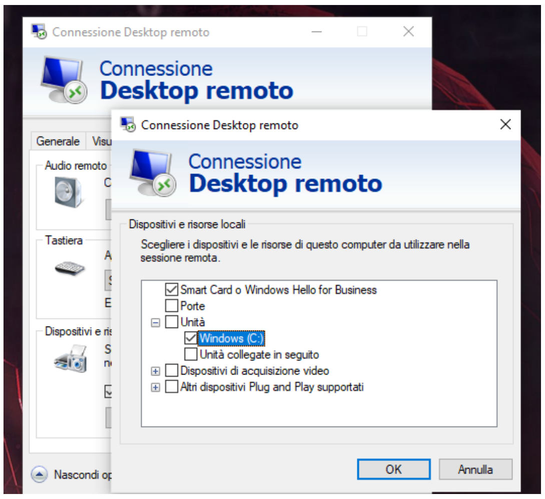
When prompted, connect to your instance using the following details:

Public DNS	ec2-3-124-11-239.eu-central-1.compute.amazonaws.com
User name	Administrator
Password	Get Password

If you've joined your instance to a directory, you can use your directory credentials to connect to your instance.

If you need any assistance connecting to your instance, please see our connection documentation.

Figure 4.29: Remote Desktop File (RDP) download window to connect to instance (see Ref. [81])

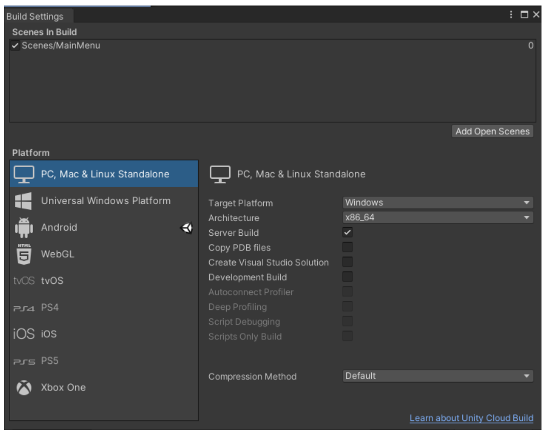


Dispositivi e risorse locali

Scegliere i dispositivi e le risorse di questo computer da utilizzare nella sessione remota.

- [✓] Smart Card o Windows Hello for Business
- [ ] Porte
- [ ] Unità
  - [✓] Windows (C:)
  - [ ] Unità collegate in seguito
- [ ] Dispositivi di acquisizione video
- [ ] Altri dispositivi Plug and Play supportati

Figure 4.30: RDP file properties to set C: drive visible



Build Settings

Scenes In Build

✓ Scenes/MainMenu 0

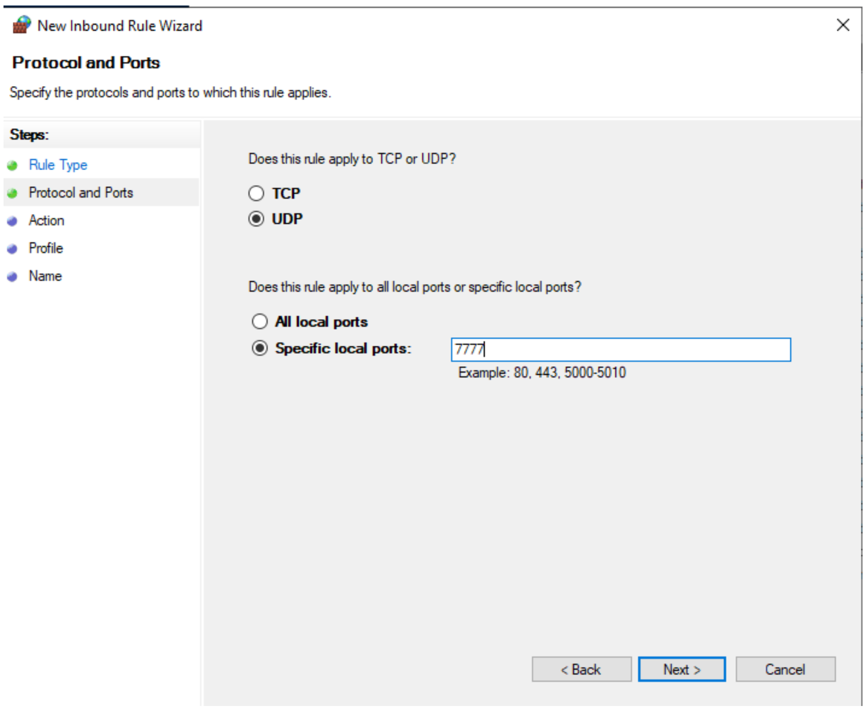
Platform

- PC, Mac & Linux Standalone
- Universal Windows Platform
- Android
- WebGL
- tvOS
- PS4
- iOS
- PS5
- Xbox One

PC, Mac & Linux Standalone

Target Platform	Windows
Architecture	x86_64
Server Build	✓
Copy PDB files	
Create Visual Studio Solution	
Development Build	
Autoconnect Profiler	
Deep Profiling	
Script Debugging	
Scripts Only Build	
Compression Method	Default

Figure 4.31: Unity Build Settings Window



New Inbound Rule Wizard

Protocol and Ports

Specify the protocols and ports to which this rule applies.

Steps:
- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

Does this rule apply to TCP or UDP?

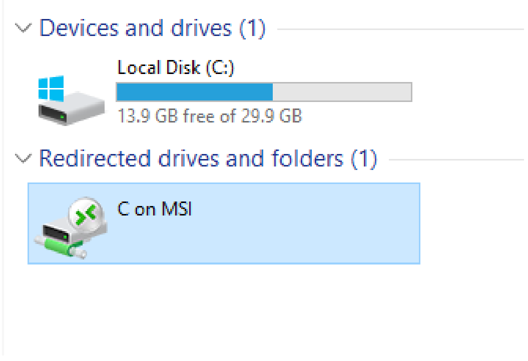
TCP [✓ UDP]

Does this rule apply to all local ports or specific local ports?

All local ports [✓ Specific local ports:] 7777

Example: 80, 443, 5000-5010

Figure 4.32: Inbound rule connection from port 7777 during configuration window



Devices and drives (1)

Local Disk (C:)

13.9 GB free of 29.9 GB

Redirected drives and folders (1)

C on MSI

Figure 4.33: Local drive seen from remote instance “This PC” folder, on Windows

Chapter 5

Results and Analysis

5.1 Performance analysis

Considering that the proposed work was developed on Unity, by using a social media API and a Networking asset, it is expected to deliver a satisfying real-time experience, both in terms of content download and multiplayer interaction. Indeed, a low refresh rate or too much RAM consumption, especially on mobile devices, would have a negative impact on the overall result and it would end to frustrate users. For a good experience, the game is expected to keep the frame-rate from 30fps and up, and also prevent lags as much as possible, so players can see fluid feedback on their actions and on the ones of their friends. To keep these aspects as better as possible, it was important to keep the level of polygons in the models as low as possible and to give them a nice look the same. Also it was wise to write C# scripts that did not consume too many CPU cycles and memory. To keep the level of polygons low it was important to find assets that respected this requirement. About the code, it was fundamental to keep it as simple as possible and to avoid Unity searching in the scene too many references of game objects. Every script in Unity can host an "Update" method. This method makes a refresh each frame, so if the project is made of 30 frames per second it means that the method is called 30 times a second. It was necessary so, to understand the limits of the operations to do every second and to move some others under a certain trigger activated by the player, like the start of a game, for example. Another solution was to avoid nested operations and loops by using coroutines (see Ref.[67]), or asynchronous methods that return a result after more than one frame, letting the application run more smoothly.

But there are some critical points about Facebook API. First of all, to work properly, it is fundamental to sign a deal between Facebook and the application company. Then, it is necessary to increase the number of permissions given to the app. Indeed, the app cannot let players see the tagged people in the pictures, look at the likes and reactions of people that are not friends of the player, or of the friends who haven't given any permission to see their info, but this feature may change when the app is published. Unfortunately, the PC version does not trigger the Facebook login screen, so it needs a built-in login path.

5.1.1 Profiling

To see the effective results, the application underwent some performance tests, under different devices. Thanks to Unity, it was possible to use one of its internal tools: the Profiler (see Ref.[82]) to measure the consumption of each hardware component, of the network, the frame-rate and the computation of the in-game physics. The profiler lets you get the performance information of the application also for external devices in two ways: by connecting physically the device to the machine or by locating it on the same network. Starting with the PC version, (see Fig. 5.1) it was noticeable the amount of frames needed to process the run-time. Depending on the activities, it kept almost every moment the rate constant at 30fps, except in some occasions where peaks reached 15fps. The RAM consumption instead, keeps staying stable, with small fluctuations. If we take a look at the Main Thread, in the Timeline, we see that the PlayerLoop takes most of the time to compute, followed by the Camera Render, with 2.88ms and the Drawing distance, with 2.26ms.

To try an effective physics stress test, it was profiled the activity of bowling (see Fig. 5.2). Since the player has to make a bowling ball roll down under a ramp to hit the pins, it required the activity of main physical factors like: the material of the ground, the friction and bounciness of the ball and the rigid bodies of all of the actors. This showed the maximum values that could be reached by the several physics forces in runtime. Consulting the official documentation (see Ref.[83]), we can see, the time the application spends in physics is based mostly on **active dynamics**, **contacts** values and constantly on **rigid bodies**, **static colliders** and **trigger overlaps**. The first value represents the number of active non-Kinematic Rigidbody components. These objects respond to gravity and are moved when pushed by a force over a certain value. The contacts count the collisions or triggers between a pair of colliders than can touch or even overlap. Unity creates a contact pair when the distance between two colliders is below a certain threshold configured by the user. The Rigidbody value measures the number of rigidbody components that are processed by the physics engine without distinguishing which one is in a sleeping state, thus it is not moving. The static colliders counts all of the colliders, with or without rigid bodies, that are connected to the game objects or their parent game objects. Last but not least, the trigger overlaps count, indeed, the number of the overlapping triggers during runtime.

Unity build settings, lets also the possibility to autoconnect a profiler to the device were the build is going to run (see Fig. 5.3). Indeed, it was possible, to test the android build performance and the results were different from the PC version (see Fig. 5.4). As we can see: the RAM here is stable too but the CPU computations in rendering the scene drop to 15fps, which compromises a lot the fluidity of the experience. The scripts and the physics, instead, can be computed in less than just 16ms. If we measure the profiler when the player does an important action, like downloading the posts of the feed (see Fig. 5.5) it recovers a lot of frames, reaching the rendering of 60fps, but the memory keeps staying stable.

It was made the same test for the physics, in Android (see Fig. 5.6) and the results are pretty much the same.

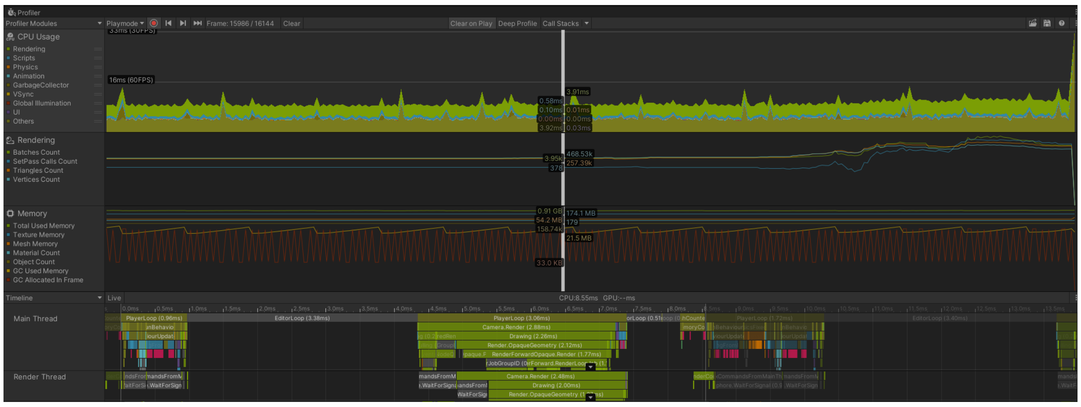
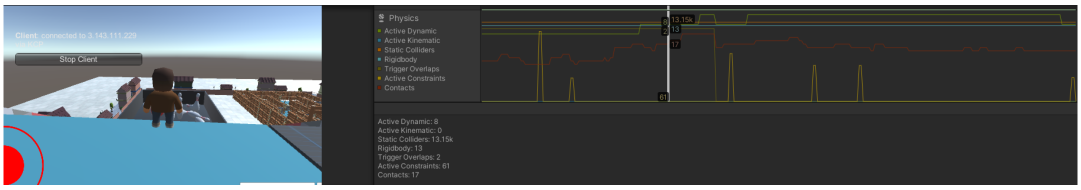
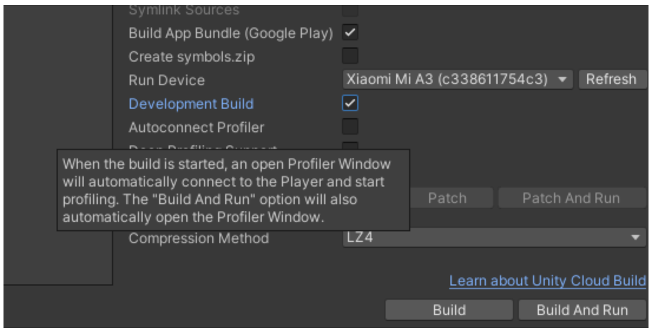


Figure 5.1: Profiler window focus on CPU and RAM usage



Active Dynamic: 8
Active Kinematic: 0
Static Colliders: 13.15k
Rigidbody: 13
Trigger Overlaps: 2
Active Constraints: 61
Contacts: 17

Figure 5.2: Profiler window focus on CPU and RAM usage during bowling activity



Symlink Sources
Build App Bundle (Google Play) ✓
Create symbols.zip
Run Device Xiaomi Mi A3 (c338611754c3)
Development Build ✓
Autoconnect Profiler

When the build is started, an open Profiler Window will automatically connect to the Player and start profiling. The "Build And Run" option will also automatically open the Profiler Window.

Compression Method LZ4

Figure 5.3: Android build with "autoconnect profiler" option to profile the device, by using a USB cable, from Unity Editor

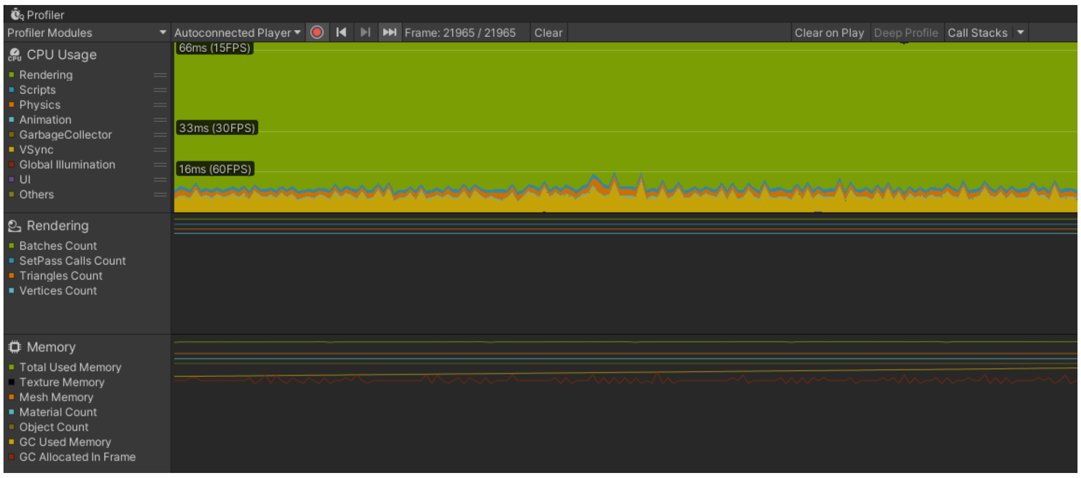


Figure 5.4: Android Profiler focus on CPU and RAM usage

5.2 Outcome of comparison between social elements on Facebook and on the Application

This work's main goal was to bring the social interactions into a 3D world and to give them a glimpse of gamification. Since the target in this case was Facebook, we will see in detail how it re-proposed the various features it offers, keeping the same social experience. It was important to make users navigate the game giving as much as possible the same feelings of the social platform, but also enhancing them with possibilities that a 3D counterpart may offer. First of all, studying the social medium, it comes out that the main elements to focus on are the friend connections and the content that a user hosts on his Facebook profile. The way these elements are arranged together is by means of a specific layout that gathers different sections: user's profile, user's friends, home (general Feed), pages, groups and so on (see Fig. 5.11a). So, in the game it was created a similar structure "translated" in a 3D alter ego (see Fig. 5.11b), that makes players navigate in each section by moving their avatar from

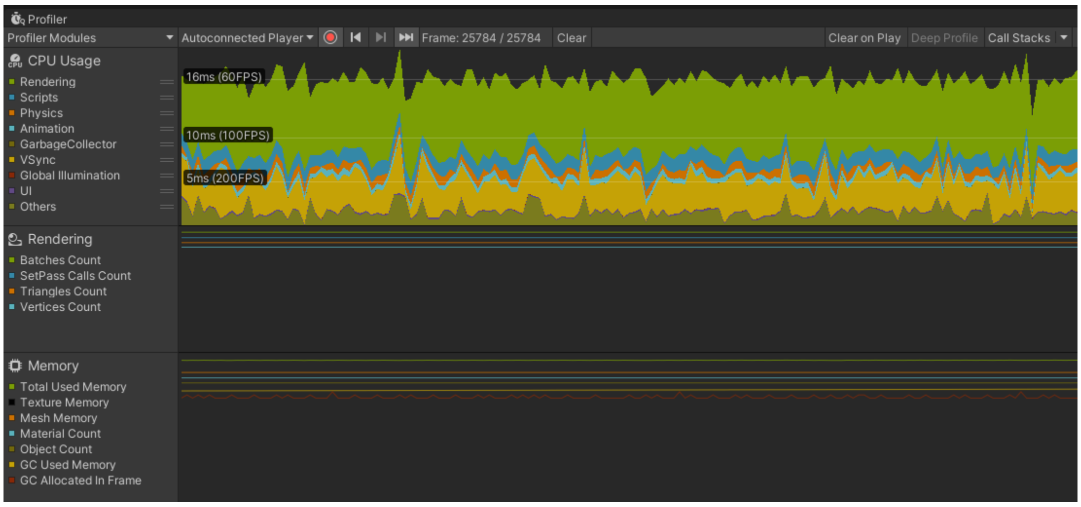


Figure 5.5: Android Profiler focus on CPU and RAM usage when downloading Facebook Feed

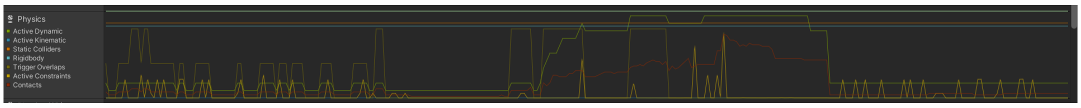


Figure 5.6: Android Profiler focus on CPU and RAM usage during bowling activity

one section to another. The square in the application, though, merges together two elements, Facebook's “Home” section and some elements of the “Facebook Gaming” section, since it offers the opportunity to try several mini-games with friends. When visiting a friend's profile on Facebook, the core part of his page is the Feed. It contains all of the posts the user has published or shared since he registered on the platform, but there are also posts where he was tagged. as mentioned before, it was put the effort to keep the feed in the application was as much as possible in the same way. Indeed, players can scroll the posts, see reactions and comments (see Fig. 5.9) and play videos. The Reactions have the following structure (see Fig. 5.10b): they show the name and the profile picture of the user and his reaction, of course. The comments, instead (see Fig. 5.9a) display also the date when the user commented and his comment.

5.3 Requirements outcomes

Summing up, here there is a list of the remaining requirements developed during this work:

1. **semi-procedural generation of the town square and of the houses that belong to the user and to his friends:** the game opens with the generation of a spiral grid that spawns the roads and the houses of the friends and of the user. Each house has the same structure: it is composed by a a building with 2 rooms, and a garden. While outside there is a sign displaying “House of ” and the name of the owner, and there is also the profile picture of the player. All of it is meant for resembling the Facebook counterpart of the user profile (name + profile picture). If the house belongs to the player there is also a “(You)” noun after the name (see Fig. 5.7);

2. **possibility for the users to travel around the world and to visit friends' houses:** when the player spawns he can start to move wherever he wants: he can go the hub to play mini-games or meet other people, otherwise he can visit other player's homes by entering through the door by triggering an action (see Fig. 5.8). There is an issue with the UI joystick, it becomes stuck in the direction set with the input but it does not change while the user drags his finger to move it around but he needs to detouch the screen and touch again in the new direction;

3. **be able to chat with all the people present in the game session:** after logging in, the game displays a canvas containing the chat UI (see Fig. 4.4a). On the right down corner of the screen there is an input field where players can enter the text. In the mobile version there is also a button to send the message, while on PC it is enough pressing the "Enter" key. On the upper corner, both versions have the text sent from all the players, showing their name and the message. Players that connect afterwards cannot see the previous messages that were sent;

4. **mobile and PC build:** mobile build for Android can login through Facebook, by logging in and accepting the requested permissions. Unfortunately, the PC version does not trigger the Facebook login screen and needs a built-in login path;

5. **commands feedback:** about the feedback, player reacts instantly to commands. To move him around using the PC version, it is necessary to press the keys W,A,S, and D to move him forward, left, back and right. While the mobile version displays always an interactive joystick with the touchscreen (see Fig. 5.12). Some mini-games or actions display a special button to press (open the door, jump, ...) or enable a key to make those actions, in PC version. Instead, when moving the camera, the PC version only requires to move the mouse simply, while the mobile version lets the user swipe all over the area where there are no buttons, giving the same feedback;

6. **meet friends around the map:** when the user is in the hub or inside a house, he can meet the players wandering around that area. Indeed, the game shows them and enables their collision when coming in contact;

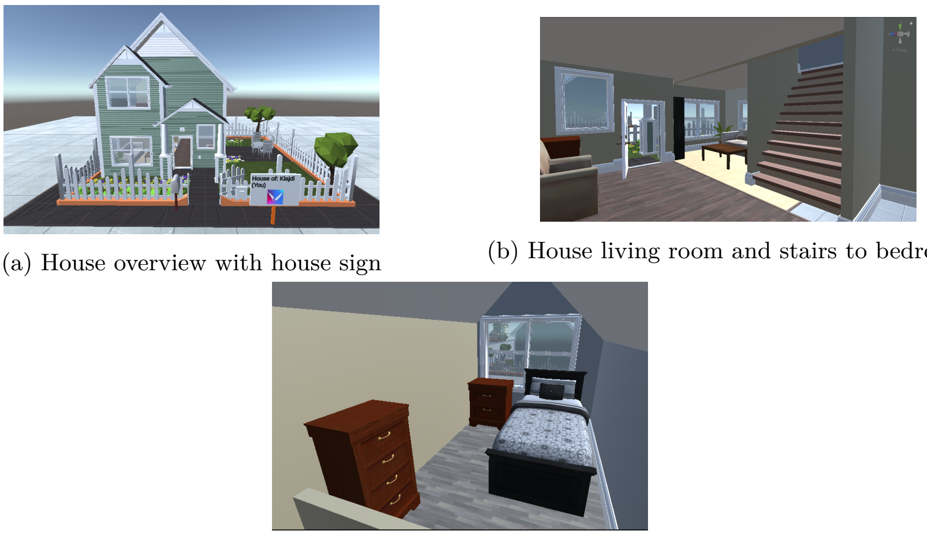
7. **names of the players:** every player met, displays his name above his head (see Fig. 4.4b), always, to help the user understand who he is interacting with;

8. **pet interaction:** the pet always follows the player, except when he is doing a mini-game;

9. **mini-games feedback:** there are two mini-games until now: obstacles run and bowling. Bowling can be played by a player at a time but not the obstacles run. If the player completes one of the games, there is a leader-board that shows player's name and his best result, putting the winner at first place. The board shows the best 5 players of the challenge. When new players connect though, they don't receive the actual situation on the leaderboard, so it is important to tell the server to update the new clients' sign;

10. **low-poly look:** as we can see from figure, both player and map have a very minimal look, since it is nice to see and it does not overload the memory usage and so it does not create damage to the performance of the game experience;

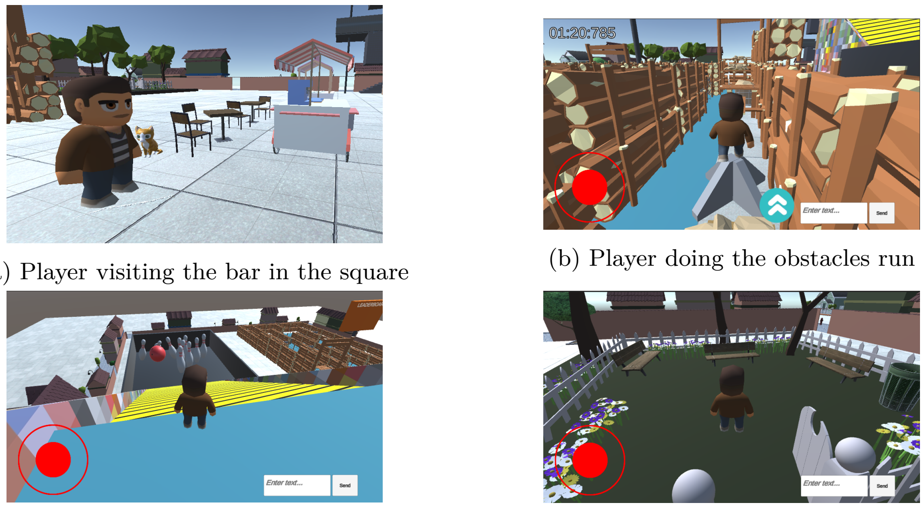
11. **camera feedback (player + feed):** when we rotate the camera, the player rotates his body too. When we enter a narrow area, the camera moves forward to let the user always look at the avatar. The Feed, instead has its own camera, that is triggered by the player when he decides to interact with it. Unfortunately, there were some optimization problems with narrow spaces were the camera distance seemed quite stressed and used to become buggy. Besides, when moving the UI joystick forward and beyond the player there is a building, the camera tends to jump in and out, making the experience annoying, and it can be controlled by moving the camera by touching the lookpad.


(a) House overview with house sign

(b) House living room and stairs to bedroom

(c) House bedroom view

Figure 5.7: A picture of any house present in the game. Any of them contains a garden and a sign 5.7a, a living room where players can display the feed 5.7b and a bedroom 5.7c



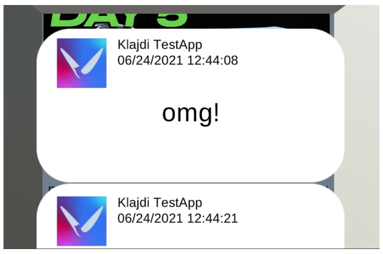
(a) Player visiting the bar in the square

(b) Player doing the obstacles run

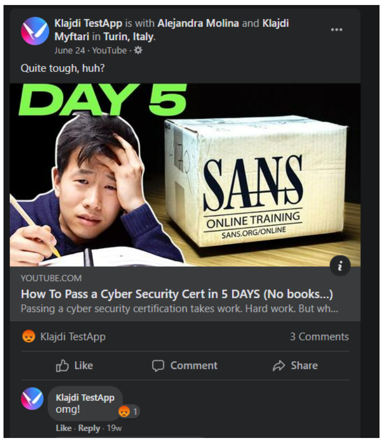
(c) Player playing bowling

(d) Player wandering in the park of the square

Figure 5.8: Some of the activities player can do in the map: go to the cafe of the square (picture 5.8a), do the obstacles race (picture 5.8b) or play bowling (picture 5.8c) or walk in the park 5.8d

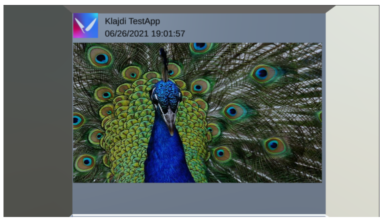


(a) Post with comments in Unity

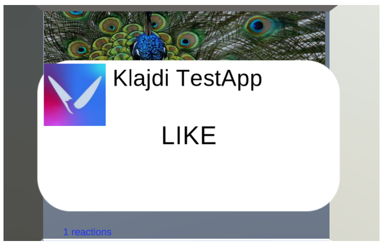

Klajdi TestApp is with Alejandra Molina and Klajdi Myftari in Turin, Italy.
June 24 · YouTube
Quite tough, huh?
DAY 5
SANS
ONLINE TRAINING
YOUTUBE.COM
How To Pass a Cyber Security Cert in 5 DAYS (No books...)
Passing a cyber security certification takes work. Hard work. But wh...
Klajdi TestApp
3 Comments
Like
Comment
Share
Klajdi TestApp
omg!
Like · Reply · 19w

(b) Same post with comments on Facebook

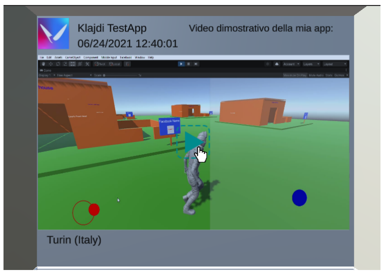
Figure 5.9: Here is a direct comparison of how comments are shown on Facebook 5.9b and on the application 5.9a



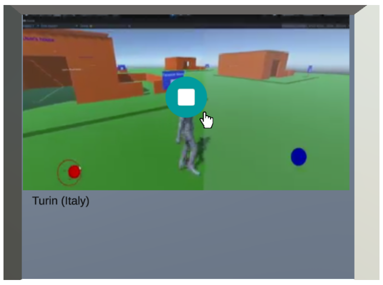
(a) Normal Feed post seen from the house Feed in Unity



(b) Same post, with the "Reactions" window open



(c) Post with comments in Unity



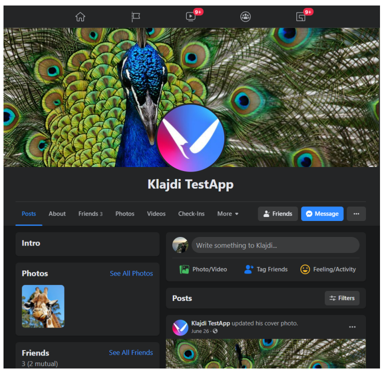
(d) Same post with comments on Facebook

Figure 5.10: This set of pictures shows the various parts of a post in the Facebook Feed developed in the application

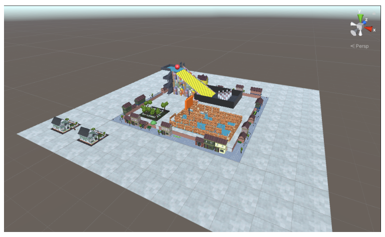
5.4 Subjective Evaluation tests

After the development of the prototype, in Unity 3D, and once the server was ready and running, an Android apk build was run on the users' mobile devices. The testers, this way had the possibility to try the app, exploring all of its features, like: multiplayer and Facebook. About Facebook, since the app was not yet in Live Mode, it was necessary to delegate them as testers, otherwise the app could not gain the permissions of their profile.

After the trial, they were asked to answer a custom survey of S.U.S. and targeted questions (see Fig. B.1) to collect feedback for further improvements. Once collecting enough answers, it was possible to extrapolate the following graphics. In total there were 12 answers. The testers were 9 females, 2 males and only one preferred not to specify the gender. The test was sent to users that have an age range similar to the worldwide chart of Facebook users (see Fig. 2.13). Indeed, as we can see from the



(a) Layout example of Facebook



(b) Layout of Facebook reinvented in Unity

Figure 5.11: In this picture there are the default Facebook layout of a page 5.11a and its counterpart re-imagined in Unity 5.11b

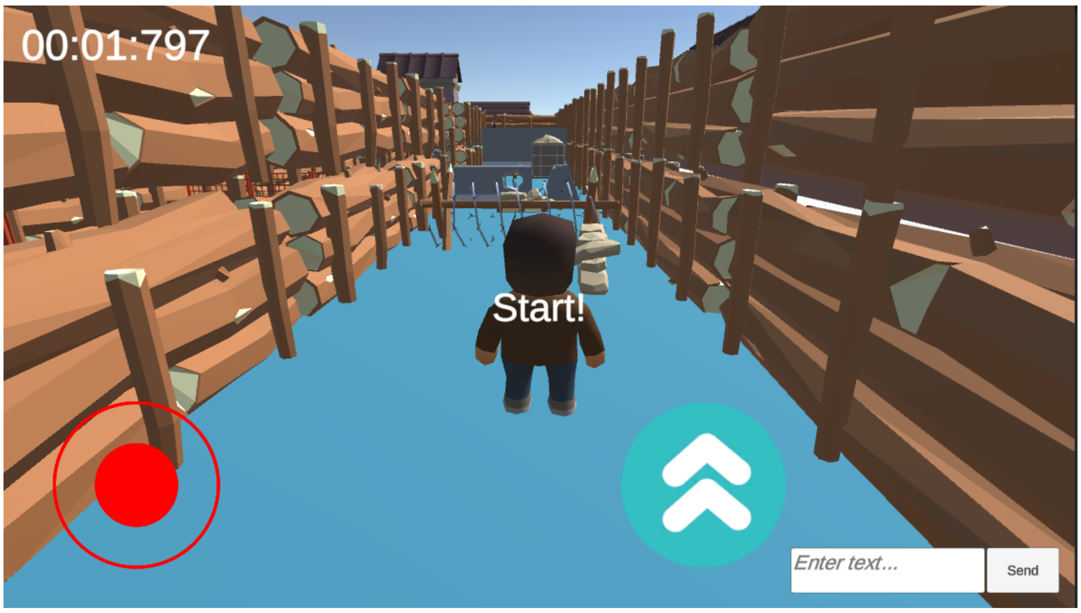
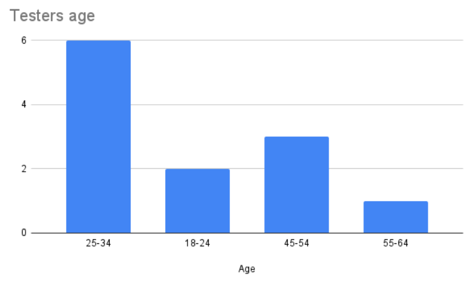


Figure 5.12: UI controls and info during the obstacles race

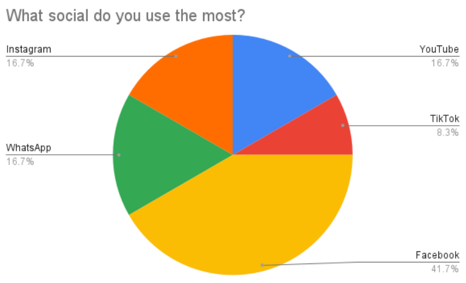
answers (see Fig. 5.13a), the majority of the testers are between 25 and 34 years old and almost half of them between 45 and 54 years old. Many of the users navigate on Facebook, indeed, 5 out of 12, while the others spread between Whatsapp, YouTube, TikTok and Instagram (see Fig. 5.13b). Almost everyone was fully convinced that Facebook is an ideal candidate for this game (see Fig. 5.14a) and so that the social component is important for it (see Fig. 5.14b). But, except Facebook, the majority of the testers think that Instagram is the candidate that fits the most with it (see Fig. 5.15a). About the activities proposed in the game, users would like to push the most on the interaction with players, then on mini-games and, at third place, on planning activities like parties, important events, team games and more (see Fig. 5.15b).

Now comes the System Usability Scale part. After playing the game, players could carry out this part. We can see that users enjoy playing the game and that they are keen on playing it frequently, except one rare case. Many answered that the complexity of the app is quite balanced, even if the half of them think that on some points it was complex without the need to be so, but in others it was made simple (see Fig. 5.16). But, many of them assert that the application was very easy to use, even though, at the same time, several needed help from someone who knew the app already (see Fig. 5.17). The average thinks that the implemented features were well integrated, even if half of them is half convinced. About the inconsistencies between the app features we find a center of mass pending more towards the lower values, so the users deny it, but one small fraction admits not to find it consistent in some small parts (see Fig. 5.18). The testers say that most of the people can learn how to use the app very easily. They state also that the commands were not hard to master but they were quite cumbersome to control the player (see Fig. 5.19). Lat but not least, many of the testers were very familiar with the app during the trial but almost half of them had to re-iterate on some processes to master at best the app itself (see Fig. 5.20).

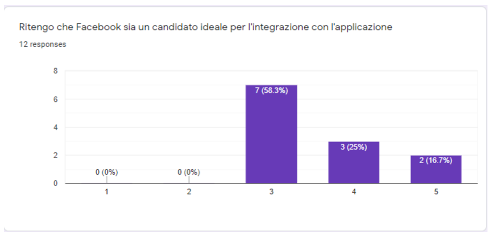
So it is evident that the application still needs loads of development and optimization, but the prototype helped to gain data to understand towards which direction head to.



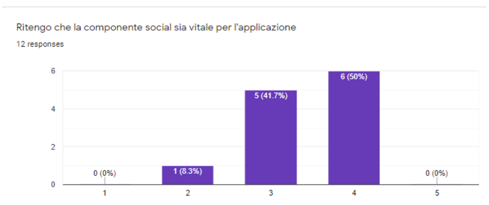
(a) Average users age



(b) Most used social by the users

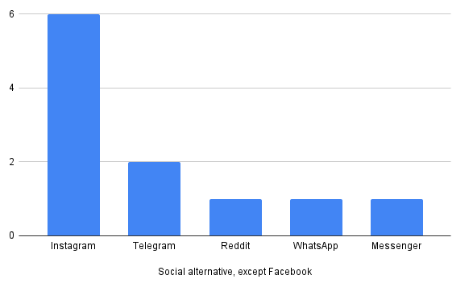


(a) Chart describing what users think about Facebook as an ideal candidate for the app

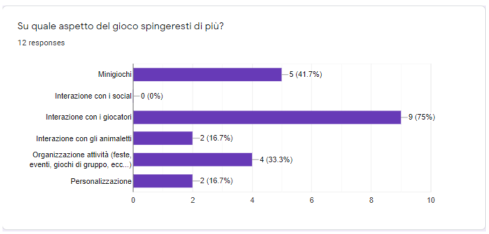


(b) Chart showing how much users think social component is important for the app

Figure 5.14: On the left we have the response about the presence of Facebook in the game 5.14a while on the right the answers about the importance of the presence of social components in the app 5.14b



(a) The chart shows that users would like to see Instagram, after Facebook, in the game



(b) The chart demonstrates that users would like to enhance the interaction with players first, then mini-games and organization of big events

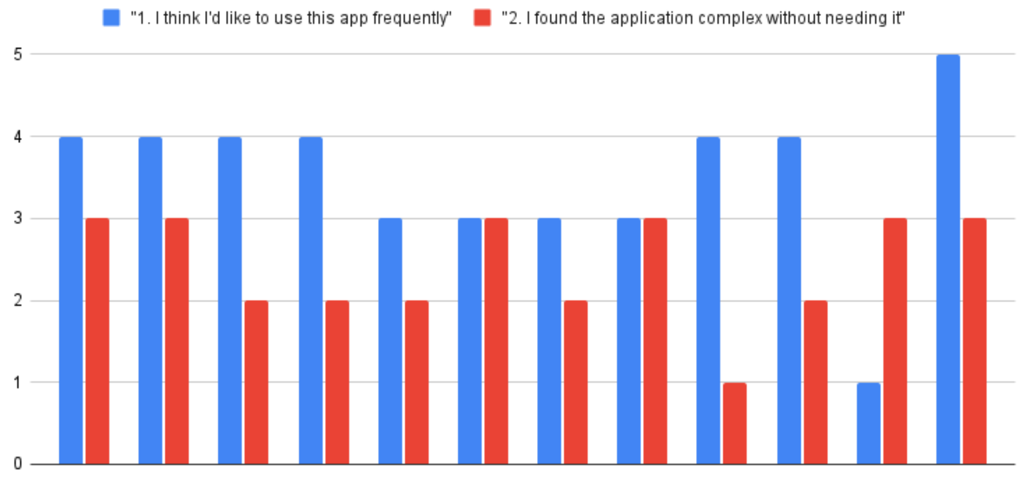


Figure 5.16: Average users response on app usability and complexity

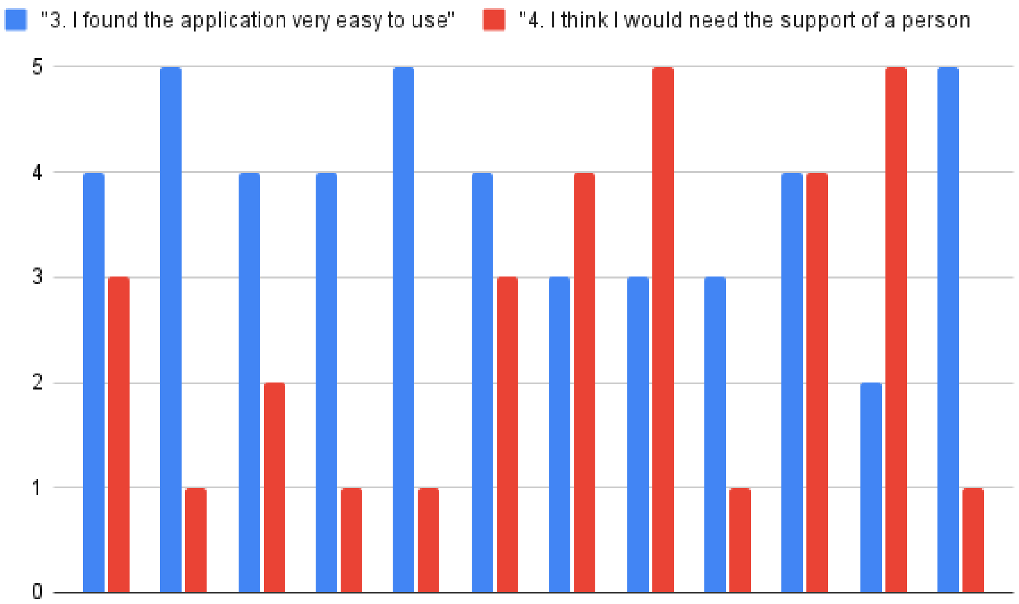


Figure 5.17: Average users response on app difficulty of use and need of qualified helper

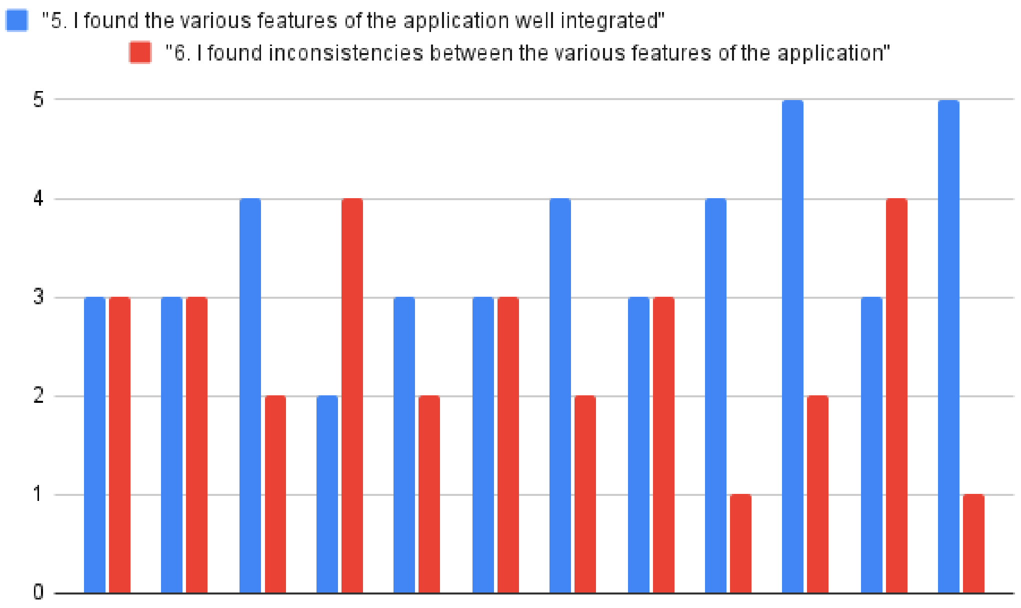


Figure 5.18: Average users response on app features integration

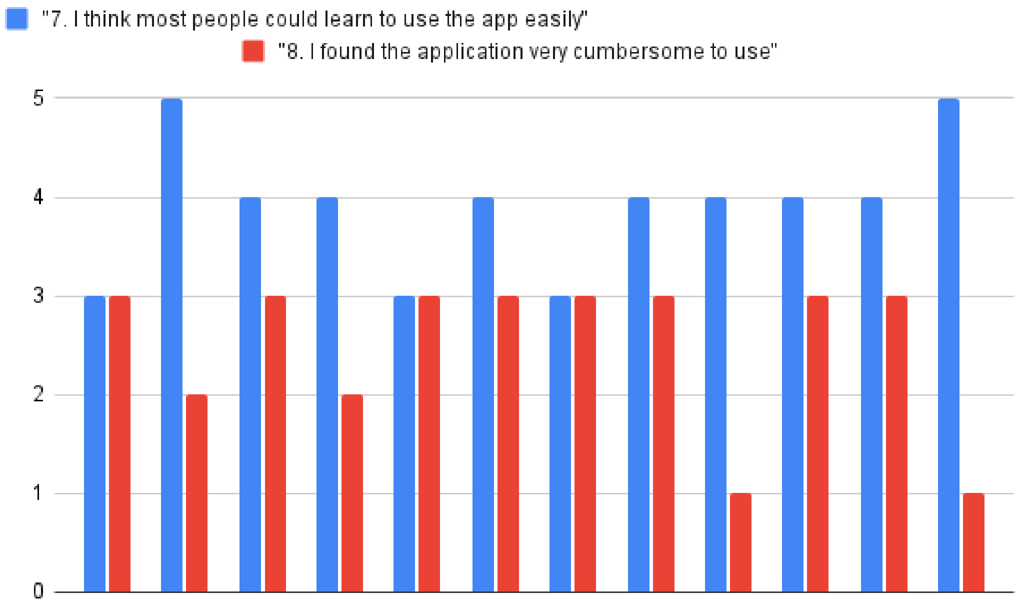


Figure 5.19: Average users response on other's difficulty to try the game and about the complexity of the interactions

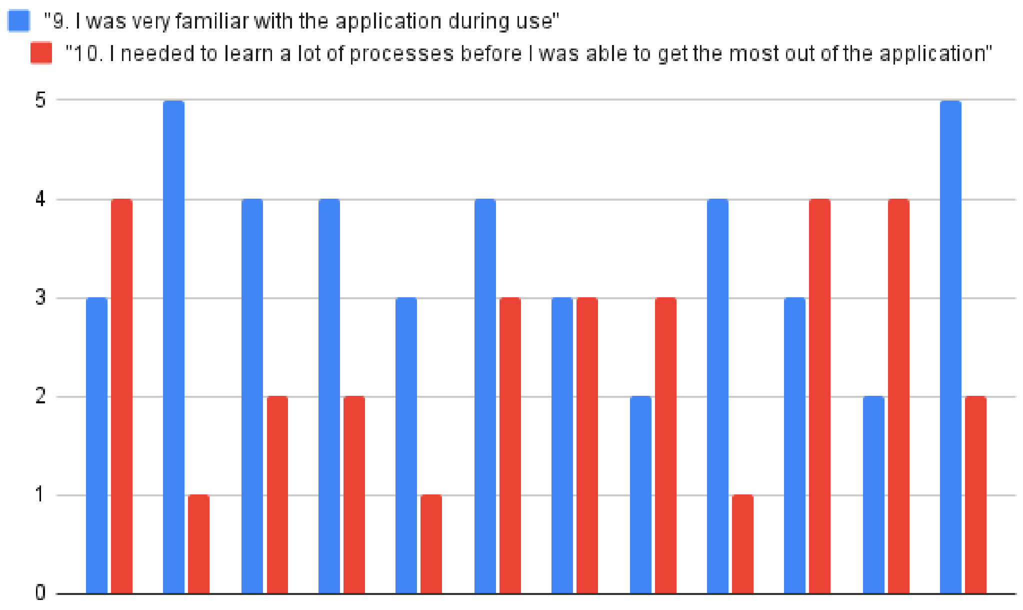


Figure 5.20: Average users response on how familiar they were with the use of the app and about the need to repeat some operations in order to learn how to use it

Chapter 6

Conclusions and Future Work

The proposed work, as mentioned above, described the development of the prototype of an MMO game strongly related to social media, in this case, Facebook. The purpose it wants to fulfill is to manage to bring the social media experience to the new and future metaverse culture. So, players can watch their social media activities, like before, while playing with an MMO game. So, when players enter the application, a 3D counterpart of their Facebook account is generated and they can navigate each section by moving their avatar with the controls. They can play mini-games with their friends too. Nowadays, this kind of work is becoming really viral. Think about Facebook Horizons that now is going to amplify the work introducing new retro games to increase the catchment area. If this work goes on, for the future, first of all it is important to establish a robust contract with Facebook and with the other social platforms that will join the application in the future. Facebook now has planned to change the name to Meta. The name itself shows Facebook full commission to push forward with metaverse development and to make users get used with this new reality, that is focused to be the new paradigm of social media.

Even if the prototype is done, it does not gather all the features the application is going to host. In the future, players will be able to do the following activities too:

- send invitations to Facebook friends that haven't joined the game yet;
- since players can meet around new people, they can add them as Facebook friends, through the application;
- players will also have the opportunity to publish content to Facebook, from the game, and this content may be: shared posts or photos taken inside the application;
- the photos that they will take, can be saved into the gallery and the latter can be opened through the Main Menu or by pausing the game;
- as shown in pictures 4.6, there will be two menus: one when starting the game and another one when pausing it. Both will let the player do the following actions:
  - enter the options to adjust audio and video settings;
  - watch and/or publish the photos in the gallery;
  - invite friends to play the game;
  - logout and switch account;
  - resume or quit the game;
- players will be able to customize their house, their avatar, their pet and their town too, by adding new buildings hosting new activities to do;

  - the customization includes the purchase of new stuff, thanks to a fee;
  - but first of all, the game will propose a style based on player's Facebook preferences like: followed pages, movies and music liked and so on;

- the game will be open to other platforms in the future, if possible, e.g.: Virtual Reality. Indeed, players will be able to wear an immersive device and to follow their avatar around the map. Thanks to this device it will be possible to develop new interesting activities to do;
- avatars may display a floating window where players can show their content to others, everywhere around the map;
- there will be a dynamic weather that can be synced to the one of player's real location;
- users will set a way point through the map to orientate;
- the app will be open to host the creation of several activities from the players, like it happens with the most famous metaverse games (see Ref. [23]);
- the hub is still incomplete, indeed it misses the global Feed, so players can see the activities of all of their friends at once, and interact with it like they do inside their homes;
- players will be able to gather and party when one of their friend will have the birthday or in other important occasions, inside the garden of their house;
- players will also play several mini-games and take special photos;
- to interact each other, there will be, not only a global chat but also other social actions like: waving your hand in front of a player, dancing, inviting someone to follow you, etc;
- the game will save automatically every time the player does an important action;
- users will also be able to pet their fellow animal or to play with it;
- Another job to do is improve the connections to guarantee a better download of the content or a good connection on the server to support a larger and larger number of connections per session.
- lastly, since the Facebook SDK for Unity has a pre-built redirection for mobile devices, the Desktop application lacks of a login path.
- Another feature to decide is to place a unique building between the houses, by choice of the player, like a tennis club or basketball field, and unlock that activity.
- The game needs to optimize, especially in the mobile version, where a lower amount of CPU and RAM consumption is needed.

Appendix A

Project source codes

A.1 Grid generation

Source Code A.1: Spiral grid generetion code

```
public void SpawnSpiralGrid(Dictionary<string, string> friends)
{
    foreach (KeyValuePair<string, string> kvp in friends)
        Debug.Log(string.Format("ID: {0}, NAME: {1}", kvp.Key, kvp.Value));
    float cols = (float)friends.Count;
    // divide number of friends per all the friends that can stand in a squared perimeter around the square
    float maximum = 20;
    // understand how many rows of houses to spawn
    while (cols / maximum > 1) maximum += 8;
    float max_delta = Mathf.Ceil(cols / maximum);
    if (max_delta == 0) max_delta = 3;
    else max_delta += 2;
    int max_friends = friends.Count;
    int offset = -1, lato_foro = 7, delta = 0, friends_counter = 0, max, c, r, half_done, tmp;

    while (delta < max_delta)
    {
        max = lato_foro + delta;
        c = offset;
        half_done = 0;
        tmp = offset;

        while (half_done < 2)
        {
            for (r = tmp; r < max; r++)
            {
                if (offset % 2 != 0)
                    SpawnPlane(r, c);
                else
                {
                    if (r % 2 != 0)
                        SpawnPlane(r, c);
                    else // Decide Wether to Place a House or a Road according to friends remaining
                        friends_counter = CheckRemainingPlayers(r, c, friends_counter, max_friends, friends);
                }
            }
            // prepare for the other half of the spiral
            if (half_done > 0)
            {
                max--;
```

```
                    r = offset;
                    tmp = offset + 1;
                }
                for (c = tmp; c < max; c++)
                {
                    if (offset % 2 != 0)
                        SpawnPlane(r, c);
                    else
                    {
                        if (c % 2 != 0)
                            SpawnPlane(r, c);
                        else
                            friends_counter = CheckRemainingPlayers(r, c, friends_counter, max_friends, friends);
                    }
                }
                if (half_done == 0) max++;
                half_done++;
            }
            delta++;
            offset--;
        }
    }
    private int CheckRemainingPlayers(int r, int c, int friends_counter, int max_friends, Dictionary<string, string> friends)
    {
        if (friends_counter < max_friends)
        {
            SpawnHouse(r, c, friends_counter, friends);
            friends_counter++;
        }
        else
            SpawnPlane(r, c);
        return friends_counter;
    }

```

A.2 Facebook SDK API call methods

Source Code A.2: Facebook SDK initialization in C# in Unity

```
    void Awake ()
    {
        if (!FB.IsInitialized)
        {
            // Initialize the Facebook SDK
            FB.Init(SetInit, OnHideUnity);
        }
        else
        {
            // Already initialized, signal an app
            //activation App Event
            FB.ActivateApp();
        }
    }
    public void SetInit()
    {
        if (FB.IsInitialized)
        {
            // Signal an app activation App Event
            FB.ActivateApp();
            // Continue with Facebook SDK
            // ...
        }
        else
        {
            Debug.Log("Failed to Init Facebook SDK");
        }
    }
    public void OnHideUnity(bool isGameShown)
    {
        if(!isGameShown)
        {
            Time.timeScale = 0;     // pauses the game
        } else
        {
            Time.timeScale = 1;     // unpauses the game
        }
    }
```

Source Code A.3: Facebook Login method

```
    public void FBlogin()
    {
        List<string> permissions = new List<string> ();
        permissions.Add ("public_profile");
        permissions.Add("email");
        permissions.Add("user_friends");
        permissions.Add("user_posts");
        permissions.Add("user_photos");
        permissions.Add("user_videos");
        FB.LogInWithReadPermissions (permissions,
        AuthCallback);
    }

    public void FBlogout()
    {
        FB.LogOut();
        DealWithFBMenus(!FB.IsLoggedIn);
    }
```

Source Code A.4: Facebook Authentication success check

```
    void AuthCallback(ILoginResult result)
    {
        if (result.Error != null) {
            Debug.Log("FB is not logged in");
            Debug.Log (result.Error);
        } else {
            Debug.Log("FB is logged in");
            DealWithFBMenus (FB.IsLoggedIn);
        }
    }
```

Source Code A.5: Facebook method that sets up the profile if user is logged in or resets the screen if logged out

```
void DealWithFBMenus (bool IsLoggedIn)
{
    if(IsLoggedIn) {
        FB.API("/me?fields=first_name", HttpMethod.GET,
        DisplayUsername);
        FB.API("/me?fields=id", HttpMethod.GET, SetMyId);
        FB.API("/me?fields=first_name", HttpMethod.GET,
        SetMyName);
        FB.API("/me/picture?type=square&height=128\&width=128",
        HttpMethod.GET, DisplayProfilePic);

        // display logged-in screen only when profile
        //pic and username donwloaded
        StartCoroutine("WaitForLogin");
        StartCoroutine("DisableMenu");

        FB.API("/me/friends?fields=first_name",
        HttpMethod.GET, GetFriends);
    }
    // Logout
    else {
        DialogLoggedIn.SetActive (false);
        DialogLoggedOut.SetActive (true);
    }
}
```

Source Code A.6: Method that downloads and displays in-game, username and profile picture of user, downloaded from Facebook thorugh the API calls

```
void DisplayUsername(IResult result)
{
    Text UserName = DialogUsername.GetComponent<Text>();
    if (result.Error == null)
    {
        UserName.text = ‘‘Hi there, ’’ +
        result.ResultDictionary["first_name"];
        _myName = (string)result.ResultDictionary["first_name"];
        uname_ready = true;
    }
    else
    { Debug.Log(result.Error); }
}
void DisplayProfilePic (IGraphResult result)
{
    if (result.Texture != null)
    {
        Image ProfilePic=DialogProfilePic.GetComponent<Image>();

        ProfilePic.sprite = Sprite.Create(result.Texture,
        new Rect(0,0,128,128), new Vector2());
        pic_ready = true;   // set flag
    }
}
IEnumerator WaitForLogin()
{
    yield return new WaitUntil(() =>
    pic_ready && uname_ready);
    DialogLoggedIn.SetActive(true);
    DialogLoggedOut.SetActive(false);
}
```

Source Code A.7: Method that gathers friends list downloaded from Facebook, orders it with thier ids and commands the house spawner to generate their homes

```
void GetFriends(IResult result)
{
    if (result.Error == null)
    {
        var raw = result.RawResult;
        Debug.Log("Friendlist: \n\n" + raw);
        JObject jparse = JObject.Parse(raw);
        var data = jparse["data"];
        var summary = jparse["summary"];
        var total = summary["total_count"];
        var friends = new Dictionary<string, string>();
        // Saving friends from JToken into Dictionary list
        try {
            // to add my house first
            friends.Add(_myId, _myName + " (You)");
        } catch
        {
            friends.Clear();
            FBlogout();
            return;
        }
        if (data == null)
        {
            Debug.Log("No friends");
            return;
        }
        int i = 0;
        foreach (var root in data)
        {
            friends.Add(data[i]["id"].ToString(),
            data[i]["first_name"].ToString());
            i++;
        }
        _myFriendlist = friends;
        // Passing Friends to generate houses
        GridGen.GetComponent<GridGen>().UpdateFriendlist(
    _myFriendlist);
    }
    else
    {
        Debug.Log(result.Error);
    }
}
```

Source Code A.8: House sign creation of the house of each friend and of the user

```
public void CreateHouseSign(string sign_name, string id)
{
    userId = id;    // save for future needs
    FB.API("/" + id + "/picture", HttpMethod.GET,
    DisplayPic);
    FB.API("/" + id + "?fields=name", HttpMethod.GET,
    SetUserName);

    TMP_Text house_sign =
    gameObject.transform.Find("Full_Sign/Sign/Name").
    GetComponent<TMP_Text>();
    if (sign_name == null)
        house_sign.text = "Unknown user";
    else
        house_sign.text = "House of: " + sign_name;
}
void DisplayPic(IGraphResult result)
{
    if (result.Texture != null)
    {
        Texture2D img = result.Texture;
        Material material = new Material(Shader.Find("Diffuse"));
        material.mainTexture = img;
        // access to the child mesh hosting the profile pic
        gameObject.transform.Find("Full_Sign/Sign/ProfilePic").
        GetComponent<Renderer>().material = material;
    }
}
```

Source Code A.9: Feed generation inside each house in-game

```
FB.API("/{id}/posts?fields=from, message, source,
full_picture, description, created_time, place,
comments, reactions", HttpMethod.GET, getPosts);

void getPosts(IGraphResult result)
    {
        var raw = result.RawResult;
        JObject jparse = JObject.Parse(raw);
        var data = jparse["data"];
        if (data == null)
        {
            Debug.LogError("No posts available");
            return;
        }
        // defines distance between post prefabs
        Vector3 offset;
        float k = 0;
        var username = gameObject.transform.parent.
        GetComponent<UserHouse>().GetHouseName();
        foreach (var datum in data)
        {
            offset = new Vector3(0f, k, 0f);
            Vector3 post_position = _postsPosition.
            transform.position + offset;
            GameObject post = Instantiate(Post,
            post_position, _postsPosition.transform.rotation);
            post.GetComponent<PostGen>().print_feed(datum,
            id, username, _postScale);
            post.transform.localScale *= _postScale;
            // necessary to scroll the posts
            post.transform.parent = _parent.transform;
            _postsInstances.Add(post);
            k -= _postScale;
        }
    }
```

A.3 Leaderboard update

Source Code A.10: Update of the obstacles race leaderboard by contacting the server to make clients write the new results

```
public override void OnStartAuthority()
    {
    // subscribe to the event of leaderboard update
        ToJson += UpdateLeaderboard;
    }

    [Client]
    // called when a user completes the race
    public void EnterNewScore(string name, TimeSpan newTime)
    {
        Dictionary<string, TimeSpan> newScore = new Dictionary<string, TimeSpan>
        {
            {name, newTime}
        };
        // serialize
        var tmp = JsonConvert.SerializeObject(newScore, Formatting.Indented);
        CmdUpdateScore(tmp);
        // score updated locally by the client who called too
        UpdateLeaderboard(tmp);
    }

    [Command]
    public void CmdUpdateScore(string val)
    {
        RpcHandleScore(val);
    }

    [ClientRpc]
    private void RpcHandleScore(string score)
    {
    // pass the score to the update leaderboard method
        ToJson?.Invoke(score);
    }

    [ClientCallback]
    // when client disconnects unsubscribes from the event
    private void OnDestroy()
    {
        ToJson -= UpdateLeaderboard;
    }

    public void UpdateLeaderboard(string newScore)
    {
        int position = 1;
        // de-serialize
        var values = JsonConvert.DeserializeObject<Dictionary<string, TimeSpan>>(newScore);
        var name = values.ElementAt(0).Key;
        var newTime = values.ElementAt(0).Value;
        if (_winnersList.Count > 0)
        {
            foreach (KeyValuePair<string, TimeSpan> winner in _winnersList)
            {
            // check if player is new
                if (winner.Key == name)
                {
                    existing = true;
                    //check if player did a better timing whether to update the score
                    int result = TimeSpan.Compare(newTime, winner.Value);
                    // old time > new time
                    if (result == -1)
                    {
                        _winnersList[name] = newTime;
                        break;
                    }
                }
            }
        }
        if (!existing)
        {
            _winnersList.Add(name, newTime);
        }
        existing = false;
        // sort winners by the minimum time required to complete the
    race
        _list = "# - Player - Time(min:sec:millisec)";  // label
        foreach (var winner in _winnersList.OrderBy(Key => Key.Value)
    )
        {
            var time = winner.Value;
            _list += "\n" + position + " - " + winner.Key + " - " +
    time.Minutes + ":" + time.Seconds + ":" + time.Milliseconds;
            position++;
        }
        _leaderboard.text = _list;
    }
```

A.4 Elevator script

Source Code A.11: Elevator's behaviour when a player climbs on it and when he offs it

```
public class ElevatorScript : MonoBehaviour
{
    private void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.transform.tag == "Player")
        {
        // the platform becomes the parent of the player, so it
    constraints his move
            other.transform.parent = transform;
        }
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.gameObject.transform.tag == "Player")
        {
            other.transform.parent = null;
        }
    }
}
```

A.5 Gamertags display

Source Code A.12: Player Names displaying on top of head and the facing of those names always in front of the camera

```
    public class ThirdPersonCharacterController : MonoBehaviour
    {
    [SyncVar(hook = "DisplayPlayerName")]
        public string PlayerDisplayName;
        public TMP_Text gamertag;

        private void Awake()
        {
            if (!isLocalPlayer)
            {
                SetPlayerName();
            }
            else
            // turn off floating name of my player
                gamertag.enabled = false;
        }

        void SetPlayerName()
        {
            FB.API("/" + _myHouseId + "?fields=name", HttpMethod.GET,
 SetPlayerTagName);
        }

        void SetPlayerTagName(IResult result)
        {
            if (result.Error == null)
            {
                var gotName = (string)result.ResultDictionary["name"
];
                if (gotName == null)
                    gamertag.enabled = false;
                else
                    if (isLocalPlayer) { CmdSendName(gotName); }
            }
            else
                Debug.Log(result.Error);
        }

        [Command]
        void CmdSendName(string pName)
        {
            PlayerDisplayName = pName;
            // no rpc needed because the magic of SyncVar.
            // a syncvar listen to data that is on the server
            // thus, we change data on the server and ALL clients
update the name with the use of the hook.
        }
        public void DisplayPlayerName(string oldName, string newName)
        {
            Debug.Log("Player changed name from " + oldName + " to "
+ newName);
            gamertag.text = newName;
        }
    }

    public class ThirdPersonCameraController : MonoBehaviour
    {
    // make player names always look at camera
        var gamertag = GameObject.FindGameObjectsWithTag("Gamertag");

        foreach (var g in gamertag)
        {
            g.transform.LookAt(transform);
        }
    }
```

A.6 Interactive camera distance from player

Source Code A.13: Camera a.i. behaviour to look always at player even in narrow spaces

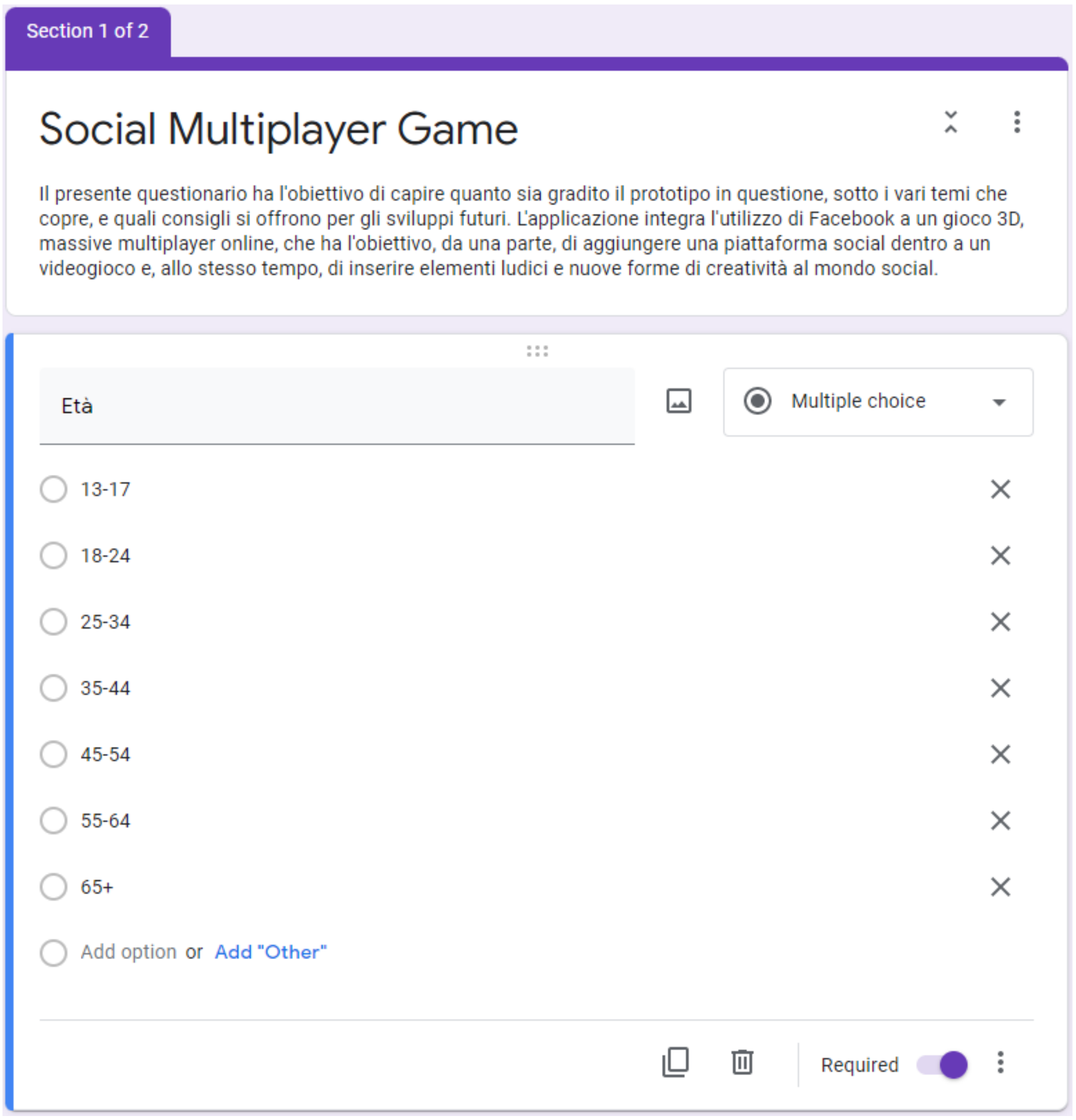
```

```

```
    public class ThirdPersonCameraController : MonoBehaviour
    {
        private void Awake()
        {
            _dollyDir = transform.localPosition.normalized;
            Distance = transform.localPosition.magnitude;
        }
        private void Update()
        {
            Vector3 desiredCameraPos = Player.TransformPoint
            (_dollyDir * MaxDistance);
            RaycastHit hit;
            if (Physics.Linecast(Player.position, desiredCameraPos,
    out hit))
                Distance = Mathf.Clamp((hit.distance * 0.8f),
                MinDistance, MaxDistance);
            else
                Distance = MaxDistance;
            transform.localPosition = Vector3.Lerp
            (transform.localPosition, _dollyDir * Distance, Time.
    deltaTime * Smooth);
        }
    }
```

Appendix B

Storyboard

B.1 Survey



Section 1 of 2

Social Multiplayer Game

Il presente questionario ha l'obiettivo di capire quanto sia gradito il prototipo in questione, sotto i vari temi che copre, e quali consigli si offrono per gli sviluppi futuri. L'applicazione integra l'utilizzo di Facebook a un gioco 3D, massive multiplayer online, che ha l'obiettivo, da una parte, di aggiungere una piattaforma social dentro a un videogioco e, allo stesso tempo, di inserire elementi ludici e nuove forme di creatività al mondo social.

Età

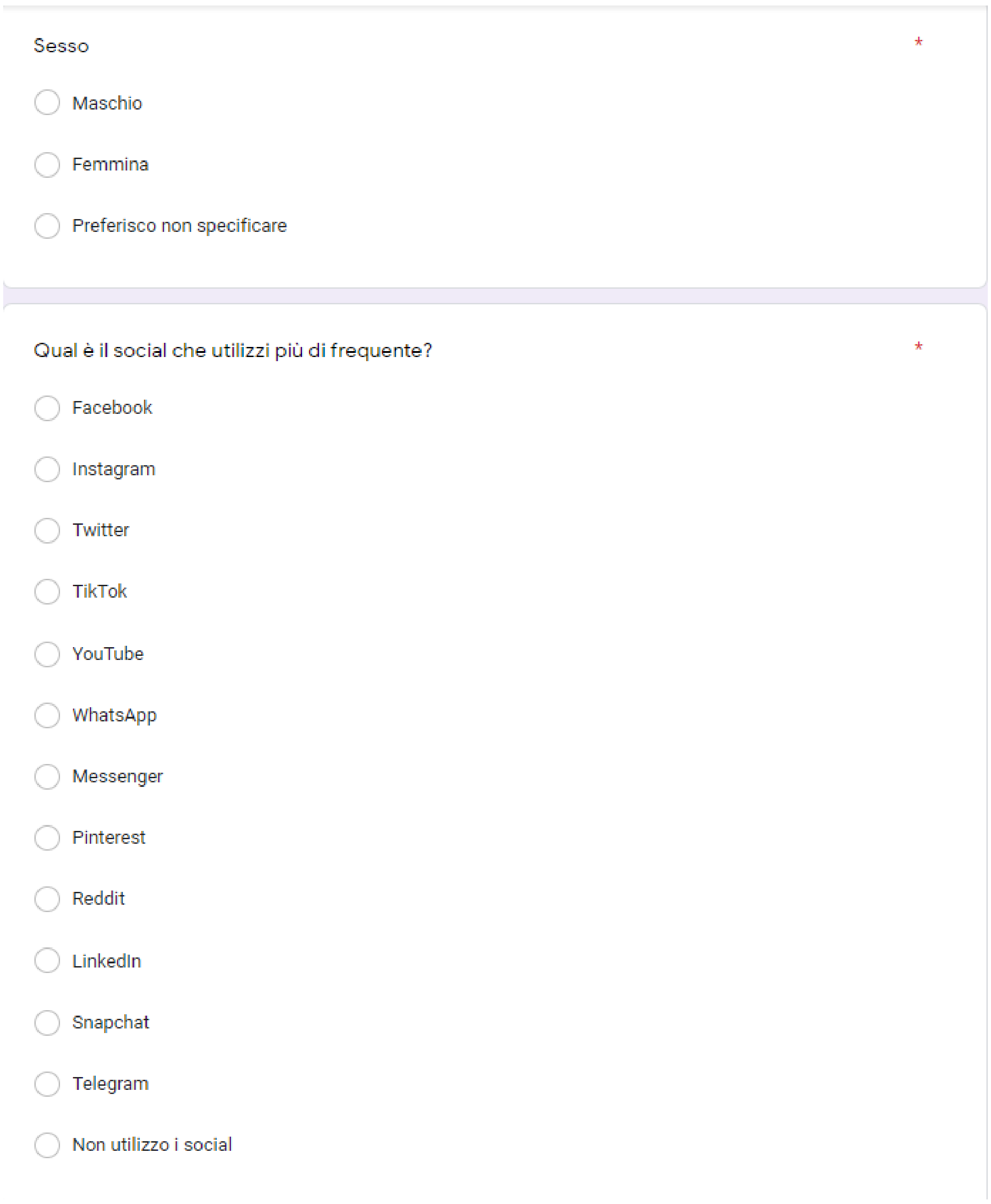
Multiple choice

- 13-17
- 18-24
- 25-34
- 35-44
- 45-54
- 55-64
- 65+

Add option or Add "Other"

Required

Figure B.1: Survey questions provided to users to get statistics on customer satisfaction

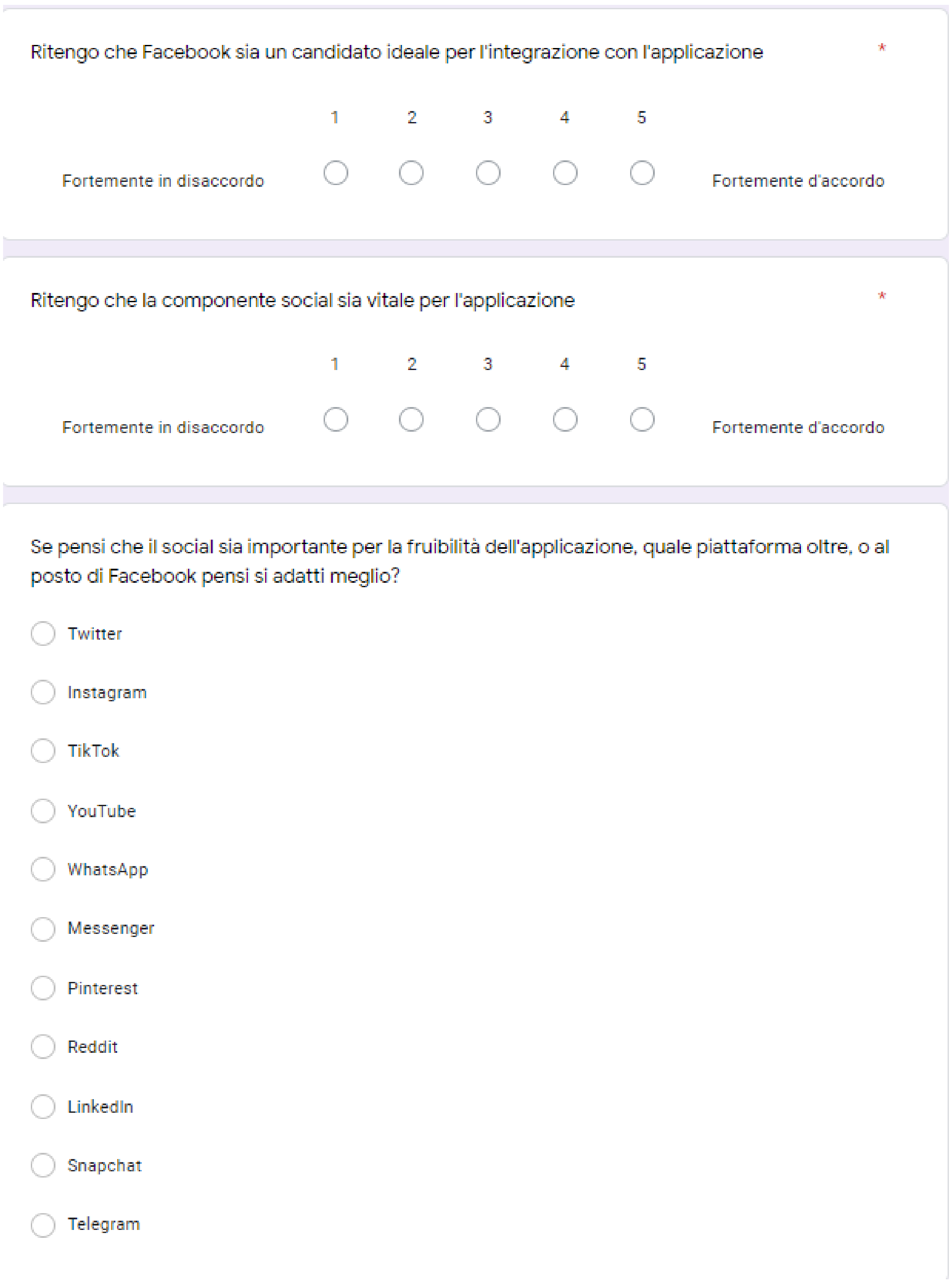


Sesso *

- Maschio
- Femmina
- Preferisco non specificare

Qual è il social che utilizzi più di frequente? *

- Facebook
- Instagram
- Twitter
- TikTok
- YouTube
- WhatsApp
- Messenger
- Pinterest
- Reddit
- LinkedIn
- Snapchat
- Telegram
- Non utilizzo i social



Ritengo che Facebook sia un candidato ideale per l'integrazione con l'applicazione *

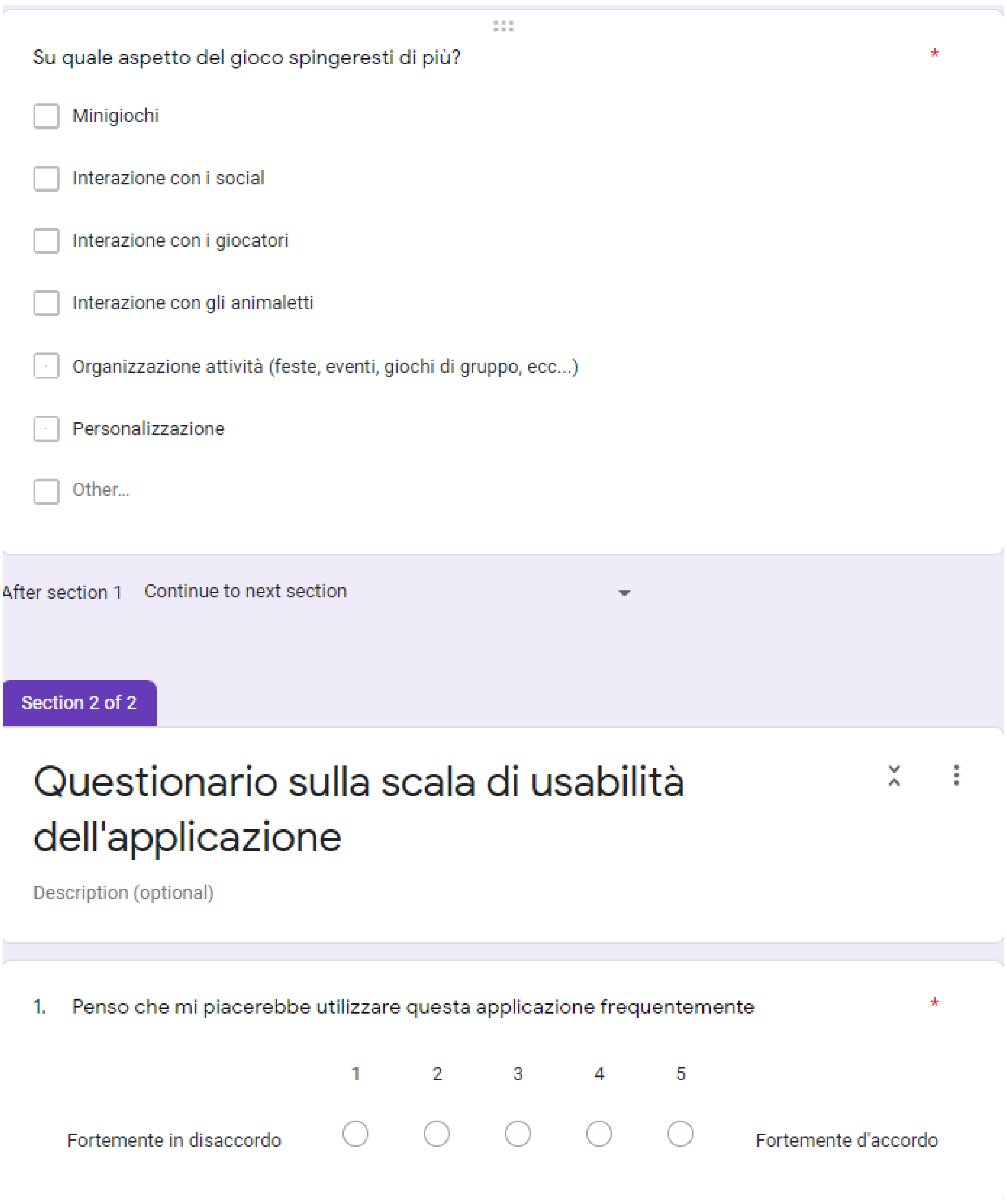
	1	2	3	4	5	
Fortemente in disaccordo	○	○	○	○	○	Fortemente d'accordo

Ritengo che la componente social sia vitale per l'applicazione *

	1	2	3	4	5	
Fortemente in disaccordo	○	○	○	○	○	Fortemente d'accordo

Se pensi che il social sia importante per la fruibilità dell'applicazione, quale piattaforma oltre, o al posto di Facebook pensi si adatti meglio?

- ○ Twitter
- ○ Instagram
- ○ TikTok
- ○ YouTube
- ○ WhatsApp
- ○ Messenger
- ○ Pinterest
- ○ Reddit
- ○ LinkedIn
- ○ Snapchat
- ○ Telegram



Su quale aspetto del gioco spingeresti di più? *

☐ Minigiochi

☐ Interazione con i social

☐ Interazione con i giocatori

☐ Interazione con gli animaletti

☐ Organizzazione attività (feste, eventi, giochi di gruppo, ecc...)

☐ Personalizzazione

☐ Other...

After section 1 Continue to next section

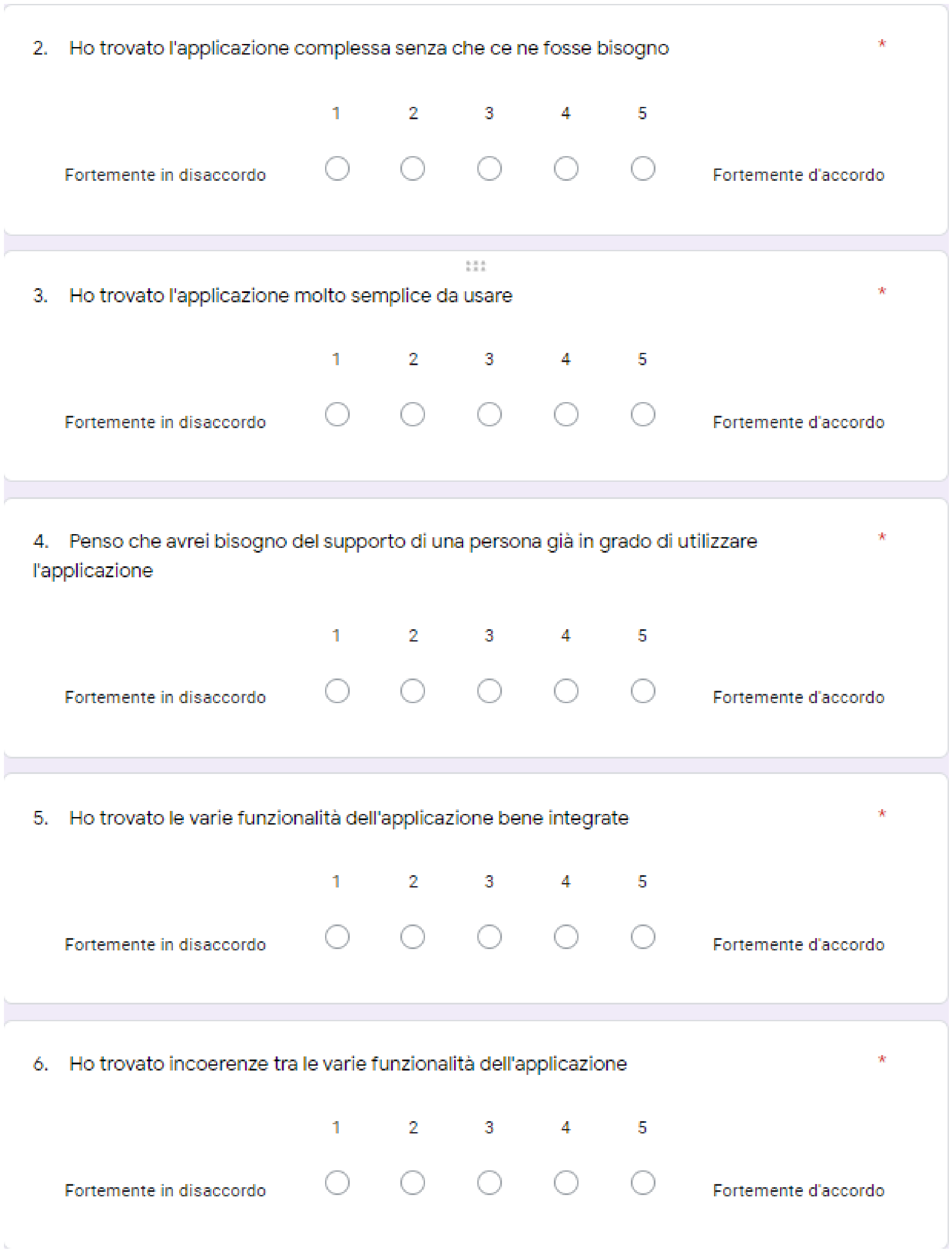
Section 2 of 2

Questionario sulla scala di usabilità dell'applicazione

Description (optional)

1. Penso che mi piacerebbe utilizzare questa applicazione frequentemente *

	1	2	3	4	5	
Fortemente in disaccordo	○	○	○	○	○	Fortemente d'accordo



2. Ho trovato l'applicazione complessa senza che ce ne fosse bisogno *

	1	2	3	4	5	
Fortemente in disaccordo	○	○	○	○	○	Fortemente d'accordo

3. Ho trovato l'applicazione molto semplice da usare *

	1	2	3	4	5	
Fortemente in disaccordo	○	○	○	○	○	Fortemente d'accordo

4. Penso che avrei bisogno del supporto di una persona già in grado di utilizzare l'applicazione *

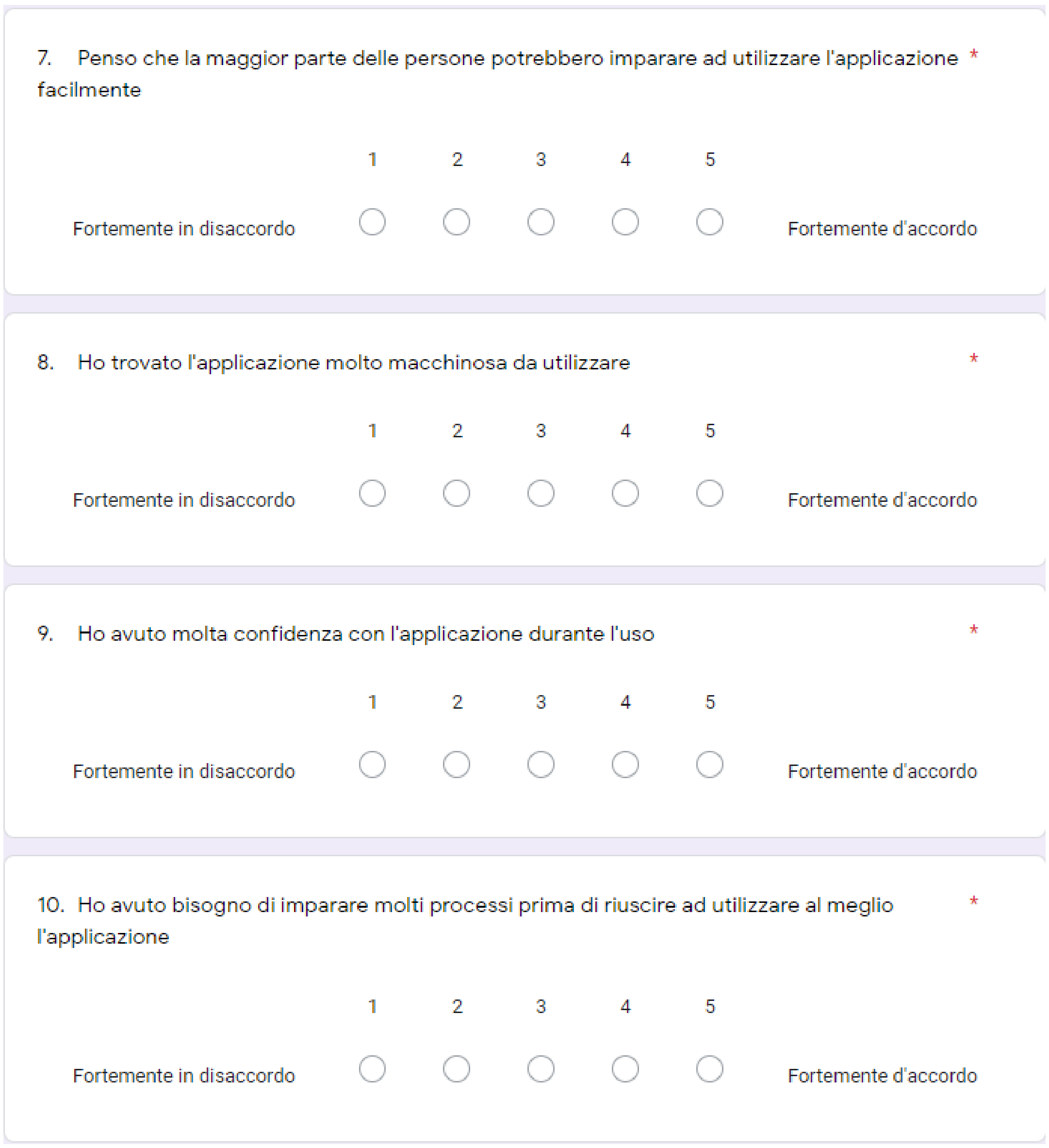
	1	2	3	4	5	
Fortemente in disaccordo	○	○	○	○	○	Fortemente d'accordo

5. Ho trovato le varie funzionalità dell'applicazione bene integrate *

	1	2	3	4	5	
Fortemente in disaccordo	○	○	○	○	○	Fortemente d'accordo

6. Ho trovato incoerenze tra le varie funzionalità dell'applicazione *

	1	2	3	4	5	
Fortemente in disaccordo	○	○	○	○	○	Fortemente d'accordo



7. Penso che la maggior parte delle persone potrebbero imparare ad utilizzare l'applicazione facilmente *

	1	2	3	4	5	
Fortemente in disaccordo	○	○	○	○	○	Fortemente d'accordo

8. Ho trovato l'applicazione molto macchinosa da utilizzare *

	1	2	3	4	5	
Fortemente in disaccordo	○	○	○	○	○	Fortemente d'accordo

9. Ho avuto molta confidenza con l'applicazione durante l'uso *

	1	2	3	4	5	
Fortemente in disaccordo	○	○	○	○	○	Fortemente d'accordo

10. Ho avuto bisogno di imparare molti processi prima di riuscire ad utilizzare al meglio l'applicazione *

	1	2	3	4	5	
Fortemente in disaccordo	○	○	○	○	○	Fortemente d'accordo

Bibliography

[1] Activate Consulting Report. *Activate Technology & media outlook 2021*. URL: https://activate.com/outlook/2021/.

[2] Activate Consulting Report. *GameHunters.Club article*. URL: https://gamehunters.club/top-games/on-facebook.

[3] The Sims FreePlay website. *The Sims FreePlay*. URL: https://www.ea.com/it-it/games/the-sims/the-sims-freeplay.

[4] Next Reality article. *The Sims Freeplay AR integration*. URL: https://next.reality.news/news/sims-freeplay-adds-multiplayer-augmented-reality-mode-via-arkit-2-0-0188738/.

[5] App Annie. *App Annie The Sims Free Play app stats (Apple)*. URL: https://www.appannie.com/apps/ios/app/the-simstm-freeplay/details?date=!(\%272021-04-18\%27,\%272021-07-10\%27)\&granularity=weekly\&country_code=US.

[6] App Annie. *App Annie The Sims Free Play app stats (Android)*. URL: https://www.appannie.com/apps/google-play/app/com.ea.games.simsfreeplay_na/details?date=!(\%272021-05-02\%27,\%272021-07-10\%27)\&granularity=weekly\&country_code=WW.

[7] Facebook. *Facebook Games Top Charts*. URL: https://www.facebook.com/games/browse/top.

[8] Business Wire article. *Worlds FRVR*. URL: https://www.businesswire.com/news/home/20180718005072/en/FRVR-Raises-3-Million-Led-by-Accel-to-Bring-Gaming-Directly-to-Users-across-All-Channels.

[9] Wordls FRVR games statistics. *Worlds FRVR - Statistics*. URL: https://instantintel.co/game/649638655530792/worlds-frvr.

[10] 2020 Influener Marketing Hub article of Dec 30. *Facebook Gaming and E-Sports*. URL: https://influencermarketinghub.com/facebook-gaming-stats/.

[11] 2020 10:07am EDT Forbes article Jun 23. *Microsoft Shuts Down Mixer After Announcing Partnership With Facebook Gaming*. URL: https://www.forbes.com/sites/qai/2020/06/23/microsoft-shuts-down-mixer-after-announcing-partnership-with-facebook-gaming/.

[12] Oct 12 2019 20:54 IST Business Insider India article of Kat Tenbarge. *Facebook Gaming Revenue*. URL: https://www.businessinsider.in/international/news/gamers-say-theyre-earning-more-money-on-facebooks-streaming-platform-than-on-twitch-and-youtube/articleshow/71557977.cms.

[13] 2020 | Updated Nov. 18 2020 The New York Times article by Seth Schiesel Published Aug. 7. *Facebook Gaming Finally Clears Apple Hurdle, Arriving in App Store*. URL: https://www.nytimes.com/2020/08/07/technology/facebook-apple-gaming-app-store.html?action=click\&module=RelatedLinks\&pgtype=Article.

[14] subtitle ’Let’s Play! 2020 The European esports market’report Page 4. *Facebook Gaming Finally Clears Apple Hurdle, Arriving in App Store*. URL: https://www2.deloitte.com/it/it/pages/technology-media-and-telecommunications/articles/let-s-play--2020---deloitte-italy---tmt.html.

[15] Figure 10 ’Let’s Play! 2020 The European esports market’report Page 21. *Facebook Gaming Finally Clears Apple Hurdle, Arriving in App Store*. URL: https://www2.deloitte.com/it/it/pages/technology-media-and-telecommunications/articles/let-s-play--2020---deloitte-italy---tmt.html.

[16] Figure 11 ’Let’s Play! 2020 The European esports market’report Page 21. *Facebook Gaming Finally Clears Apple Hurdle, Arriving in App Store*. URL: https://www2.deloitte.com/it/it/pages/technology-media-and-telecommunications/articles/let-s-play--2020---deloitte-italy---tmt.html.

[17] 2021 7:50 AM Games Beat article by Jan 20. *The future of creation and play in the Metaverse*. URL: https://venturebeat.com/2021/01/20/the-future-of-creation-and-play-in-the-metaverse/.

[18] 2021 7:49 PM GMT+2 Tech Crunch article by Lucas Matney @lucasmtny Jun 4. *Facebook buys studio behind Roblox-like Crayta gaming platform*. URL: https://techcrunch.com/2021/06/04/facebook-buys-studio-behind-roblox-like-crayta-gaming-platform/?guccounter=1\&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8\&guce_referrer_sig=AQAAAGxZZXR0kX0ZI7ocdatFw-ttXfQMICsXUZfp65SgCCiK50_cCJDb87vTkCZ-IWD888rbVghiK0iMYgB476GCAPXJebbaLUaKy_kmuDRALiu_fqNxyPCMdnHrW-dRRQwKNilMDmeVtQkkzBxp0WzxG0JtQF0vHGXv-0LF-Jt1zHLC.

[19] 2021 Routenote article by Madeleine Amos Jun 10. *Facebook buys studio behind Roblox-style games platform Crayta*. URL: https://routenote.com/blog/facebook-buys-games-platform-crayta/.

[20] 2020 Evening Standard article by Mark Blunden June 4. *London firm creates Fortnite-like simulated 3D offices for those working from home*. URL: https://www.standard.co.uk/tech/fortnitelike-simulated-offices-for-working-from-home-a4458891.html.

[21] 2020 9:00 AM Venture Beat article by Dean Takahashi @deantak Apr 3. *The Machine GamesBeat Jobs Special Issue Account Settings Log Out Sine Wave Entertainment launches Breakroom 3D social hub for remote teams in VR, PC, or mobile devices*. URL: https://venturebeat.com/2020/04/03/sine-wave-entertainment-launches-breakroom-3d-social-hub-for-remote-teams-in-vr-pc-or-mobile-devices/.

[22] 2020 8:30 AM Games Beat article by Dean Takahashi @deantak Dec 28. *IMVU: Making the coin of the realm for the metaverse*. URL: https://venturebeat.com/2020/12/28/imvu-making-the-coin-of-the-realm-for-the-metaverse/.

[23] 2021 8:00 AM Games Beat article by Dean Takahashi @deantak Mar 5. *The DeanBeat: Roblox public offering is a vote about the metaverse*. URL: https://venturebeat.com/2021/03/05/the-deanbeat-roblox-public-offering-is-a-vote-about-the-metaverse/.

[24] 2021 9:56 PM Games Beat article by Dean Takahashi @deantak Mar 23. *Social gaming platform Rec Room raises $100M at $1.25B valuation.* URL: https://venturebeat.com/2021/03/23/social-gaming-platform-rec-room-raises-100m-at-1-25b-valuation/.

[25] 2021 06:30am EDT Forbes article by Dean Kate O'Flaherty Jul 4. *Facebook Just Gave 1 Million Oculus Users A Reason To Quit.* URL: https://www.forbes.com/sites/kateoflahertyuk/2021/07/04/facebook-just-gave-1-million-oculus-users-a-reason-to-leave/?sh=329a368776f5.

[26] 2021 4:10 AM Venture Beat article by VB Staff Feb 1. *'We're only just scratching the surface': The potential of 5G in creating the metaverse.* URL: https://venturebeat.com/2021/02/01/were-only-just-scratching-the-surface-the-potential-of-5g-in-creating-the-metaverse/.

[27] 2021 Sprout Social article by Jenn Chen Feb 17. *20 Facebook stats to guide your 2021 Facebook strategy.* URL: https://sproutsocial.com/insights/facebook-stats-for-marketers/.

[28] 2021 10:00 AM GamesBeat article by Dean Takahashi Jun 15. *Facebook launches gaming fan groups to grow player communities.* URL: https://venturebeat.com/2021/06/15/facebook-launches-gaming-fan-groups-to-grow-player-communities/.

[29] Microsoft Corporation. *Visual Studio 2019.* URL: https://visualstudio.microsoft.com/it/vs/.

[30] Google. *Google Cloud Platform.* URL: https://cloud.google.com/gcp?utm_source=google&utm_medium=cpc&utm_campaign=emea-it-all-en-bkws-all-all-trial-e-gcp-1010042&utm_content=text-ad-none-any-DEV_c-CRE_500236788711-ADGP_Hybrid\%20\%7C\%20BKWS\%20-\%20EXA\%20\%7C\%20Txt\%20~\%20GCP\%20~\%20General\%23v3-KWID_43700060384861756-aud-606988877894\%3Akwd-26415313501-userloc_1008857&utm_term=KW_google\%20cloud\%20platform-NET_g-PLAC_&gclid=Cj0KCQjwwY-LBhD6ARIsACvT72OIjMXTh7yYx5GQKEcUh8mcFOYx30lEGh3NpNPa7tSSg8v6M9mFInUaAg0eEALw_wcB&gclsrc=aw.ds.

[31] Facebook. *Facebook for Developers.* URL: https://developers.facebook.com/?no_redirect=1.

[32] SalesForce. *Heroku.* URL: https://www.heroku.com.

[33] Unity Technologies. *Unity.* URL: https://unity.com.

[34] Facebook. *Facebook SDK for Unity.* URL: https://developers.facebook.com/docs/unity.

[35] Fenerax Studios. *Joystick Pack.* URL: https://assetstore.unity.com/packages/tools/input-management/joystick-pack-107631.

[36] LLC parentElement. *JSON .NET For Unity.* URL: https://assetstore.unity.com/packages/tools/input-management/json-net-for-unity-11347.

[37] Unity Technologies. *Standard Assets (for Unity 2018.4).* URL: https://assetstore.unity.com/packages/essentials/asset-packs/standard-assets-for-unity-2018-4-32351.

[38] Unity Technologies. *TextMeshPro.* URL: https://docs.unity3d.com/Manual/com.unity.textmeshpro.html.

[39] vis2k. *Mirror*. URL: https://assetstore.unity.com/packages/tools/network/mirror-129321.

[40] Evgeny Nikolsky. *Bowling: Kegel Ball*. URL: https://assetstore.unity.com/packages/3d/props/bowling-kegel-ball-67371.

[41] JustCreate. *Cartoon Low Poly City Pack Lite*. URL: https://assetstore.unity.com/packages/3d/environments/urban/cartoon-low-poly-city-pack-lite-166617#content.

[42] DigitalKonstrukt. *Prototyping Pack (Free)*. URL: https://assetstore.unity.com/packages/3d/prototyping-pack-free-94277.

[43] Ladymito. *Free chibi cat*. URL: https://assetstore.unity.com/packages/3d/characters/animals/mammals/free-chibi-cat-165490.

[44] StreakByte. *Low Poly Ultimate Home Pack*. URL: https://assetstore.unity.com/packages/3d/environments/low-poly-ultimate-home-pack-164671.

[45] EVPO Games. *Modular Lowpoly Streets (Free)*. URL: https://assetstore.unity.com/packages/3d/environments/urban/modular-lowpoly-streets-free-192094#content.

[46] 314pies. *ParrelSync*. URL: https://github.com/VeriorPies/ParrelSync.

[47] AnimPic Studio. *Polygon - FastFood*. URL: https://assetstore.unity.com/packages/3d/props/polygon-fastfood-196151.

[48] Polygon Blacksmith. *Toony Tiny People Demo*. URL: https://assetstore.unity.com/packages/3d/characters/toony-tiny-people-demo-113188.

[49] Elcanetay. *Toon Furniture*. URL: https://assetstore.unity.com/packages/3d/props/furniture/toon-furniture-88740.

[50] Ferocious Industries. *FREE Suburban Structure Kit*. URL: https://assetstore.unity.com/packages/3d/props/free-suburban-structure-kit-142401#content.

[51] vis2k. *Tests*. URL: https://mirror-networking.gitbook.io/docs/general/tests.

[52] noobtuts. *uMMORPG 480 CCU *Worst Case* Stress Test - 2019-01-23*. URL: https://www.youtube.com/watch?v=mDCNff1S9ZU.

[53] Microsoft. *Azure PlayFab documentation*. URL: https://docs.microsoft.com/en-us/gaming/playfab/.

[54] Photon. *Photon Documentation*. URL: https://doc.photonengine.com/en-us/realtime/current/getting-started/realtime-intro.

[55] Unity. *MLAPI Documentation*. URL: https://docs-multiplayer.unity3d.com/docs/getting-started/about/index.html.

[56] vis2k. *Mirror Documentation*. URL: https://mirror-networking.gitbook.io/docs/.

[57] Nintendo. *Animal Crossing*. URL: https://animal-crossing.com.

[58] Microsoft. *An introduction to NuGet*. URL: https://docs.microsoft.com/en-us/nuget/what-is-nuget.

[59] Facebook. *App Dashboard*. URL: https://developers.facebook.com/docs/development/create-an-app/app-dashboard/.

[60] Facebook. *All apps must be set to Live Mode for production use.* URL: https://developers.facebook.com/blog/post/2019/09/23/live-mode-for-production-use/.

[61] Facebook. *App Roles.* URL: https://developers.facebook.com/docs/development/build-and-test/app-roles/.

[62] Facebook. *Test Users.* URL: https://developers.facebook.com/docs/development/build-and-test/test-users/.

[63] Brackeys. *MESH GENERATION in Unity - Basics.* URL: https://www.youtube.com/watch?v=eJEpeUH1EMg.

[64] Facebook. *Facebook SDK for Unity: Getting Started.* URL: https://developers.facebook.com/docs/unity/gettingstarted.

[65] Facebook. *Facebook Login for Android - Quickstart.* URL: https://developers.facebook.com/docs/facebook-login/android.

[66] Facebook. *Facebook SDK for Unity - Examples.* URL: https://developers.facebook.com/docs/unity/examples.

[67] Unity. *Coroutines.* URL: https://docs.unity3d.com/Manual/Coroutines.html.

[68] vis2k. *Components.* URL: https://mirror-networking.gitbook.io/docs/components.

[69] vis2k. *kcp2k - KCP Transport.* URL: https://mirror-networking.gitbook.io/docs/transports/kcp-transport.

[70] skywind3000. *KCP - A Fast and Reliable ARQ Protocol.* URL: https://github.com/skywind3000/kcp/blob/master/README.en.md.

[71] vis2k. *Network Identity.* URL: https://docs.microsoft.com/en-us/nuget/what-is-nuget.

[72] Unity. *Physics.Raycast.* URL: https://docs.unity3d.com/ScriptReference/Physics.Raycast.html.

[73] Unity. *LayerMask.* URL: https://docs.unity3d.com/ScriptReference/LayerMask.html.

[74] vis2k. *Interest Management.* URL: https://mirror-networking.gitbook.io/docs/guides/interest-management.

[75] vis2k. *Remote Actions.* URL: https://mirror-networking.gitbook.io/docs/guides/communications/remote-actions.

[76] vis2k. *SyncVar Hooks.* URL: https://mirror-networking.gitbook.io/docs/guides/synchronization/syncvar-hooks.

[77] vis2k. *NetworkBehaviour.* URL: https://mirror-networking.gitbook.io/docs/guides/networkbehaviour.

[78] Unity. *Physics.Linecast.* URL: https://docs.unity3d.com/ScriptReference/Physics.Linecast.html.

[79] Unity. *Animator Controller.* URL: https://docs.unity3d.com/Manual/class-AnimatorController.html.

[80] Unity. *Platform define directives.* URL: https://docs.unity3d.com/Manual/PlatformDependentCompilation.html.

[81] vis2k. *AWS.* URL: https://mirror-networking.gitbook.io/docs/guides/server-hosting/aws.

[82] Unity. *Profiler overview.* URL: https://docs.unity3d.com/Manual/Profiler.html.

[83] Unity. *Physics Profiler module.* URL: https://docs.unity3d.com/Manual/ProfilerPhysics.html.