Master's Degree Final Project

Master's Degree in Neuroengineering and Rehabilitation

Cognitive research application for Huntington's disease

MEMORY

Autor:	Laura Daniela Pedreros Almeciga
Director:	Alejandro Bachiller Matarranz
Co- Director:	Sergio Romero Lafuente

Date: May 2022

ETSEIB

Barcelona School of Industrial Engineering (ETSEIB)

Abstract

Huntington's disease is an inherited neurodegenerative disorder caused by the gradual degeneration of parts of the basal ganglia called the caudate nucleus and putamen, which are responsible for the facilitation and coordination of movements. Huntington's disease is characterized by unequal and uncoordinated movements. In addition, people with Huntington's disease have impaired mental function, including self-control and memory. Most individuals with this disease have chorea that initially progresses but then, with the later onset of dystonia and rigidity, becomes less prominent. Although chorea is a useful marker in determining the diagnosis, it is a poor marker of disease severity, as patients with early-onset Huntington's disease may not develop chorea, or it may arise only transiently during their illness. Recent studies have determined that there are significant changes in the volume of the whole brain, long before symptoms are evident, causing different cognitive disorders, such as impairment in oculomotor tasks.

The implementation of software as a medical evaluation tool has been of great help to understand the role of some specific brain functions in different neurodegenerative disorders; however, for Huntington's disease existing cognitive software is not sensitive to the type of deficits that people with this disease present. Through this work we seek to provide a possible solution to this drawback, developing an application that can detect cognitive abnormalities associated with complex visual processing in people who are known to present the disease in the future but still have no symptoms, all this based on neurological studies such as neuroimaging and neurophysiology, which give us an idea of what are those cognitive processes that incipiently could begin to show damage many years before the disease manifests itself.

This application was developed with the help of a doctor from Hospital Sant Pau, specialist in Huntington's disease, with whom the tasks to be developed were defined. In order to carry out this work we used the Unity2D platform for the creation of computer applications, where all the programming for the environment was done.

In order to evaluate the functionality of the developed application, it was tested on 10 people who do not have the disease; thanks to the feedback from these tests and the data obtained, improvements were proposed for the developed application.

Keywords: Chorea, Huntington's disease, Neurodegenerative, Software, Unity.

Index

Glossary

HD	*Huntington Disease***.**
prHD	*Pre Huntington Disease*
MRI	*Magnetic Resonance Imaging*
fMRI	*Functional Magnetic Resonance Imaging*
sMRI	*Structural Resonance Imaging*
PET	*Positron emission tomography*
fPET	*Functional Positron emission tomography*
DTI	*Diffusion tensor imaging*
UI	*User Interface*
EEG	*Electroencephalography*

1. Preface

The software for the detection and diagnosis of neuronal diseases play a very important role in determining the stage of the disease in the patient and also dictate a treatment, as soon as possible. This Master thesis is focused on the creation of an application for the detection of cognitive abnormalities in people with Huntington's disease at an early stage, where there is no sign of symptoms yet.

1.1. Motivation

Huntington's disease shows a stable prevalence in most of the white populations of about 5 to 7 affected persons per 100,000. However, exceptions can be seen among different regions of the world that can vary from around 2.1 to 10 per 100,000 [1].

Because of its low prevalence rate it is considered a rare disease, and there is still no way to determine the early stage deterioration of the disease, that is why the main motivation of this work is to explore an unexplored terrain in this disease with the help of Dr. Saul Martinez, a specialist in Huntington's disease, together with him we decided to propose a study to determine whether patients with early Huntington's disease show deficits in predicting the trajectory of a movement pattern, all this based on literature reviews and the experience that the doctor has had with patients with this disease.

Another aspect that motivated the realization of this project is to be able to set a precedent about this topic, and in such a way that this project can be the basis of information for future research.

1.2 Previous requirements

In order to achieve the objectives proposed in this project it was necessary to acquire previous knowledge in programming languages such as C#, to use the Unity2D video game creation platform. In addition to this, to have a good knowledge about Huntington's disease, linked to basic knowledge in statistics to be able to perform the data analysis. On the other hand, to have access to a computer with Windows operating system and a Tablet or any touch device with Android operating system.

3. Introduction

3.1. Project objectives

3.1.1. General Objectives

- Development and implementation of a software used to detect cognitive abnormalities in people with Huntington's disease.

3.1.2. Specific Objectives

- Development and implementation of a cognitive analysis software specifically for early stage Huntington's disease..
- Develop different tasks focused on complex visual processing by means of trajectory prediction of a movement pattern.
- Develop an eye-catching, practical and effective interface that is easy to use for both the patient and the doctor.
- Extract data as predicted values of the circle position and the actual position of the circle at the time of the tap.
- Implementing the application in people who do not have the disease.
- Calculate the distances between the predicted trajectory position vectors and the actual position vectors at the time of the tap.
- Analyze the data obtained to determine the functionality, feasibility of the study and possible changes.

3.2. Project Scope

The scope of this project is to design a software focused on the detection of cognitive abnormalities based on complex visual processing in people with Huntington's disease and to test the first phase of this application in people without the disease to evaluate the functioning and viability of the disease.

4. State of the art

4.1. Huntington's disease

Huntington's disease is an autosomal dominant neurodegenerative disorder. This disease is clinically characterized by the development of involuntary movement disorders and cognitive impairment. Patients with this disease suffer a progressive deterioration, although the disorder can manifest itself at any time between childhood and adulthood, usually the onset of symptoms begins in mid-adulthood, until it develops into a chronic form [2].

4.1.1. Etiology

HD is one of the 10 autosomal dominant inherited diseases caused by excessive expansion of CAG (cytosine-adenine-guanine) triplets in their respective proteins (polyglutamine diseases)[3].

The Laboratory Committee of the Huntington's Disease Task Force, Bethesda, Maryland, proposed the following classification [4]:

1. Normal alleles: alleles with ≤26 CAG repeats are not pathological and segregate as stable polymorphic repeats in >99% of meiosis. The most common alleles are those with 17 and 19 CAG repeats.

2. Normal-mutated alleles: alleles with 27 to 35 CAG repeats, range referred to as meiotic instability range or intermediate alleles. They do not produce the HD phenotype but can be meiotically unstable in male germ cells.

3. HD alleles with reduced penetrance: alleles with 36 to 39 repeats, meiotically unstable and can produce HD.

4. HD alleles with full penetrance: have more than 40 CAG repeats and produce the HD phenotype.

5. Mosaicism: is due to mitotic and meiotic instability and has been described in brain and male germ cells and appears to be more pronounced in juvenile-onset cases associated with large expansions.

Trinucleotide segment expansion mutations are termed unstable, as they tend to increase from one generation to the next. There is a correlation between the size of the CAG sequence expansion with the age of onset and severity of the disease. Thus, the larger the size of the expansion, the earlier the age of onset and the faster the progression of the disease, and vice versa.

This anticipation is greater if the disease is transmitted by a male. The size of the expansions is particularly unstable in spermatozoa and meiosis probably has a major impact on their instability, causing an increase in the number of CAG repeats, which explains why paternal transmission is the most frequent cause of the phenomenon of anticipation [4].

4.1.2. Clinical Manifestations

In people with this disease the brain suffers a cortical atrophy in relation to the degree of evolution of the disease. Some of the clinical manifestations that occur are:

- Psychiatric disorders: in most cases, these disorders are part of the first symptoms presented by a patient with Huntington's disease. They include affective disorders such as depression and mania, in many cases suicidal tendencies, personality changes, attention deficits when continuing a conversation, personal distractions, decreased libido and sleep disorders [5, 6, 7].

- Motor disorders: consist of very slight abnormal movements at the beginning, but as the disease progresses the motor disorders become more evident and cause a functional disorder. As the disease progresses, voluntary movements and gait balance are affected. Other of the most common alterations that are presented are hyperreflexia [5, 6, 7].

- Cognitive disorders: mainly in the beginning these disorders are based on the alteration of short-term memory and loss of judgment, to the point of developing a dementia that incapacitates the development of activities of daily living. The dementia that these patients develop is of subcortical type, which causes a slowing of thinking accompanied by an attention deficit [5, 6, 7].

4.1.3. Diagnosis and Treatment

There is currently no curative treatment for Huntington's disease, nor are there any drugs to help stop or prevent physical and mental deterioration. In fact, treatments today are only focused on controlling symptoms such as involuntary movement disorders and psychological alterations. On the other hand, it is not possible to predict exactly when symptoms will appear, but thanks to molecular biology techniques, it is possible to detect carriers of the disease in pre-symptomatic stages [8].

As with other chronic diseases, Huntington's disease requires a proper appreciation of the medical limitations, because despite the advances in research over the past 20 years, the medical treatment of this disease has progressed very little. Survival of affected individuals where medical technology is largely unavailable is similar to that of populations with easy access to treatment[1, 9]. As mentioned above, there are antichoretic or neuroleptic drug treatments that offer patients with severe chorea a respite from their constant involuntary movements. However, to a large extent these drugs can lead to bradykinesia, rigidity and depression or sedation. Affective disorders in Huntington's disease are amenable to psychiatric treatment and immediate intervention is recommended [1].

4.1.4. Importance of Medical Imaging in Huntington's Disease

Currently the most feasible way to measure the progressive deterioration of the disease before symptoms occur is by medical imaging. Routine MRI and CT scans in moderate to severe Huntington's disease show loss of striatal volume and enlargement of the frontal horns of the lateral ventricles, but the scans are generally not useful for diagnosis of an early disorder [1, 10].

Data from PET and functional MRI studies have shown that changes occur in affected brains before the onset of symptoms, and some MRI techniques can accurately measure the cortex and striatum. In fact, with these techniques, caudate atrophy becomes evident as early as 11 years before the estimated onset of the disease and putaminal atrophy as early as 9 years [1].

4.2. Compilation Table of works referenced for the research study

REF	RESEARCH OBJECTIVE	METHOD USED	RESULTS
[11]	The present study sought to characterize cognitive domains underlying a large test battery and for the first time, evaluate their ability to predict time to diagnosis	They included participants with gene-negative and gene-positive prHD. They performed a factor analysis of 18 tests to identify sets of latent measures or factors that elucidated the core constructs of the tests.	Six factors were identified: speed/inhibition, verbal working memory, motor planning/speed, attention information integration, sensory perceptual processing and verbal learning/memory
[8]	Huntington's disease biomarker development research	Research on biomarker techniques and the advantages and disadvantages of these techniques. (Clinical (Cognitive measures, quantitative motor assessment), Neuroimaging (SMRI, fPET,fMRI,MRS), Electrophysiology(EEG)	The identification of readily available, reliable and robust biomarkers of Huntington's disease progression will be important for the development and evaluation of disease-modifying treatments.

[12]	Investigation about visuomotor integration deficits precede clinical onset in Huntington’s disease.	2239 subjects with the CAG expansion of HD, from more than a decade before clinical onset of HD to the HD clinical onset to the early stage of the disease, and 122 controls, completed a circle-tracing task, which included both direct and indirect visual direct and indirect visual feedback conditions. Measures included accuracy, speed, and speed of error detection and correction. Detection and correction speed. Correlating the results with 3T MRI.	Compared with controls, early HD was associated with lower accuracy and slower performance in both circle tracing conditions. While premanifest HD was associated with lower accuracy in both conditions and fewer rotations in the direct condition.
[13]	To identify sensitive and reliable biomarkers in carriers of HTT mutations and in individuals with early HD that could provide essential carriers of HTT mutations and in individuals with early HD that could provide an essential methodology for the evaluation of therapeutic interventions.	It utilizes a broad battery of novel assessments, including multisite 3T MRI, clinical, cognitive, quantitative motor, oculomotor, and neuropsychiatric measures. Cognitive, quantitative motor, oculomotor, and neuropsychiatric measures. Blinded analyses were performed on the cross-sectional data from 366 individuals: 123 controls, 120 premanifest (pre-HD) individuals, and 123 patients with early HD.	Identified significant changes in whole brain volume, regional differences in gray and white matter, impairment in a number of voluntary neurophysiological motor and oculomotor tasks, and cognitive and white matter, impairment in a range of voluntary neurophysiological motor and oculomotor tasks, and cognitive and neuropsychiatric dysfunction in pre-manifested HD gene carriers with normal to clinical stage neuropsychiatric dysfunction in premanifest HD gene carriers with normal motor scores up to clinical stage 2 disease.
[14]	review of recent advances in the development of biomarkers for HD, and the potential future roles of these biomarkers in clinical trials.	Research on biomarker techniques and the advantages and disadvantages of these techniques (MRI, fMRI, PET, DTI)	Biomarkers are likely to be more relevant in presymptomatic and prodromal presymptomatic and prodromal. Different biomarkers may be more useful at different times in the course of HD. Steady progression of atrophy of atrophy is observed in the striatum and in other brain regions, and has the potential regions of the brain, and has the potential to be useful over long long periods of time.

Table 1. Compilation of works referenced for the research study

5. Project planning

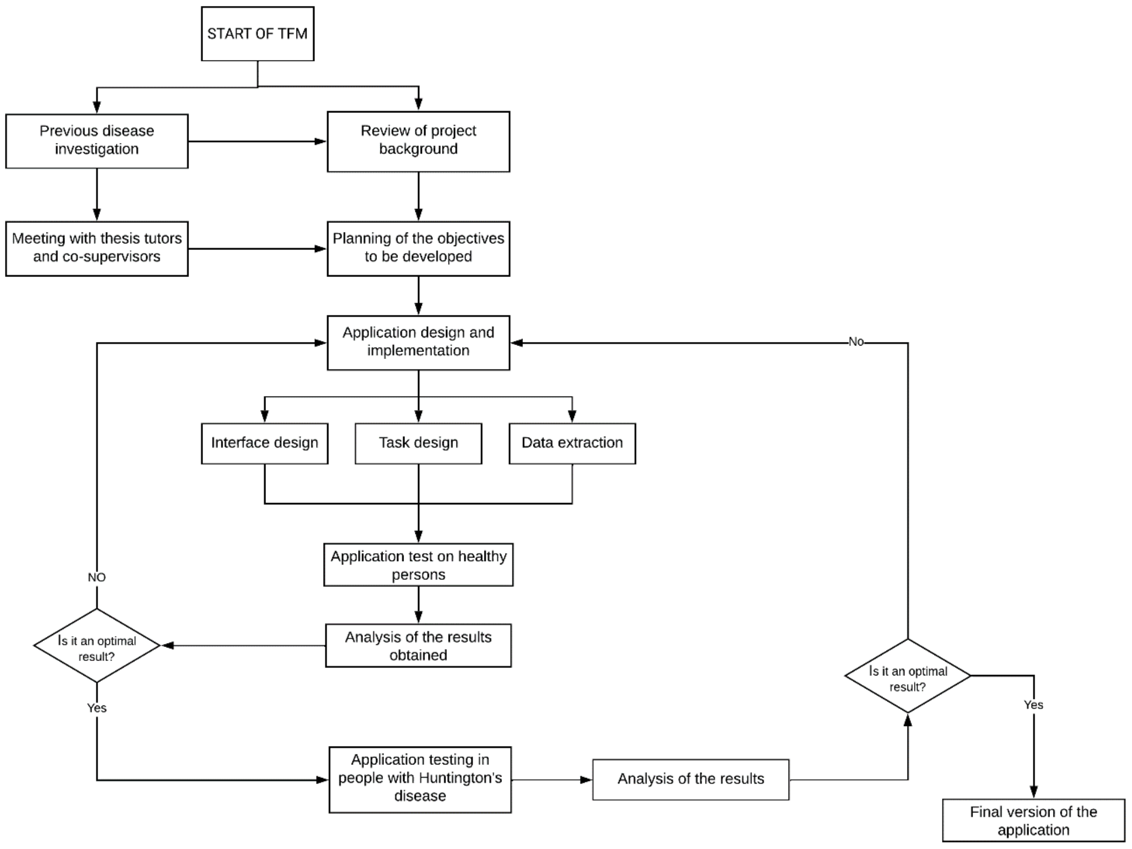

START OF TFM
Previous disease investigation
Review of project background
Meeting with thesis tutors and co-supervisors
Planning of the objectives to be developed
Application design and implementation
Interface design
Task design
Data extraction
Application test on healthy persons
Analysis of the results obtained
NO
Is it an optimal result?
Yes
Application testing in people with Huntington's disease
Analysis of the results
Is it an optimal result?
No
Yes
Final version of the application

Figure 1. Methodology diagram.

5.1. Methodology

In order to carry out this project, basic knowledge of the Unity platform and the C# programming language was acquired through the Udemy platform. Once the basic knowledge was acquired, meetings were held with the tutors to define the basis of the project, and a work plan was structured.

1. **Define objectives to be achieved:** First, a meeting was held with the university tutors and the hospital tutor, where the objectives to be achieved with the work and its scope were discussed.

2. **Research on the disease:** Once the objectives to be achieved were defined, research on the disease and background or possible similar work was carried out.

3. **Brainstorming:** Taking into account the research conducted and with the help of

Dr. Saul Martinez, a brainstorming session was carried out, where the different tasks to be performed by the user were proposed, the objective of the task was determined and the values of interest were determined.

4. **Determination of the design and content of the software:** In order to define the design and content of the software, an investigation was carried out on how medical software focused on the diagnosis or treatment of chronic neuronal diseases is. Based on this information and on what was previously discussed, the design and content of the software is proposed.

5. **Development phase:**

    a. **Development of the tasks:** Once the content of the software is clear, we proceed to its development. As already mentioned, the development is done through the Unity2D platform.

        i. **Performing tests in simulations:** Each time a scene is finalized, simulations are performed to test the operation of these. This section will be explained in detail later on.

        ii. **Restructuring of tasks:** Periodic meetings were held whenever a doubt arose or when a task was completed, where improvements were proposed and the software content was restructured.

    b. **Identification of fundamental data for the study:** Once the tasks had been developed, the data of interest were defined and the ways of obtaining them were proposed.

    c. **Data extraction methods:** Once the data of interest were obtained, they were extracted by means of Excel files.

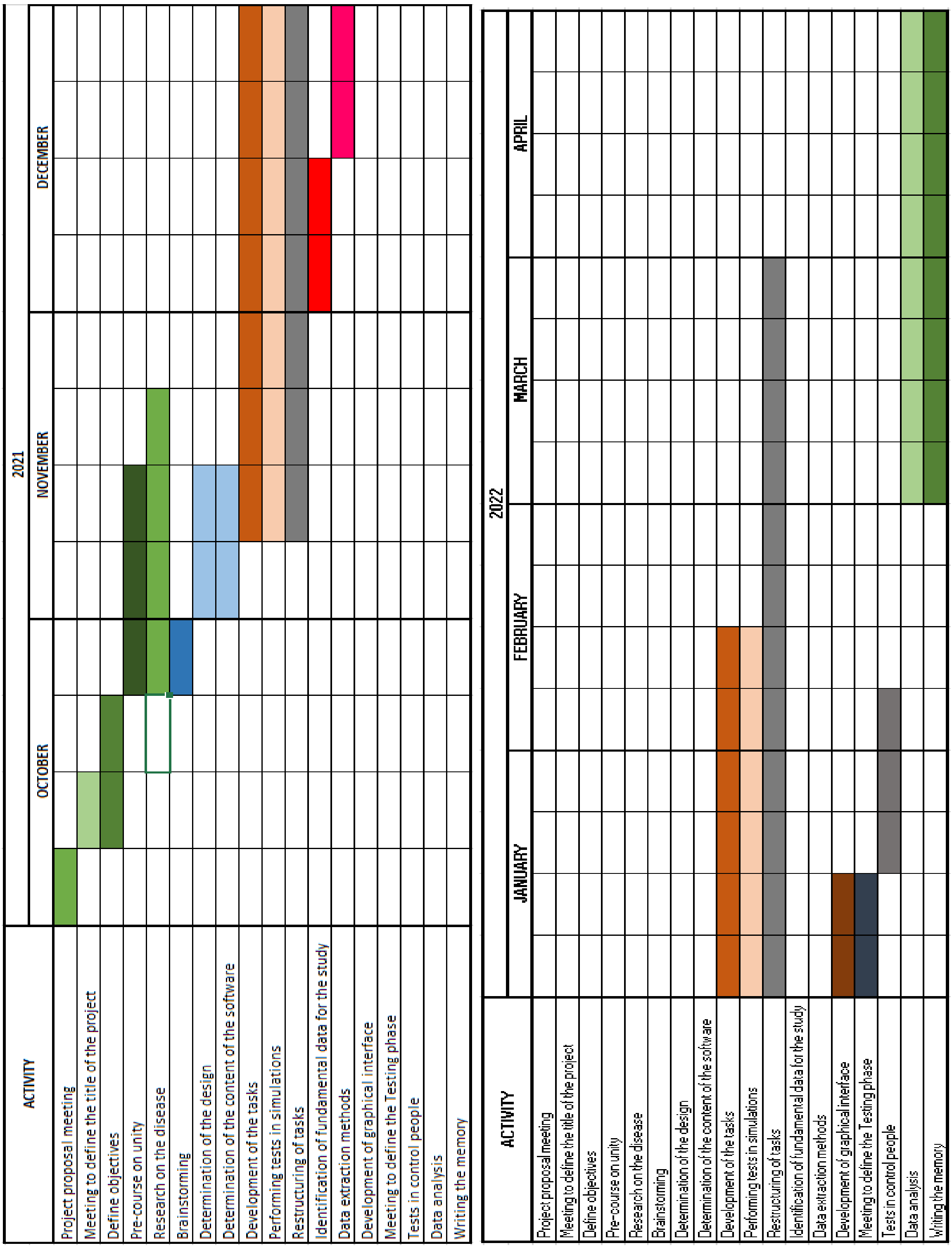
    d. **Development of graphical interface with registration and startup system:** Once the application content base is completed, the application startup interface is developed.

6. **Testing phase:** Finally, at the end of the development phase, tests are performed to verify in users who do not have the disease how accurate the application is and how reliable are the data that we are going to obtain from it.

    a. Tests in people who do not have the disease

    b. Data analysis

5.2. Gantt chart



ACTIVITY	2021											
	OCTOBER				NOVEMBER				DECEMBER			
Project proposal meeting	■											
Meeting to define the title of the project		■										
Define objectives		■	■									
Pre-course on unity				■	■	■						
Research on the disease				■	■	■	■					
Brainstorming				■								
Determination of the design					■	■						
Determination of the content of the software					■	■						
Development of the tasks						■	■	■	■	■	■	■
Performing tests in simulations						■	■	■	■	■	■	■
Restructuring of tasks						■	■	■	■	■	■	■
Identification of fundamental data for the study									■	■		
Data extraction methods											■	■
Development of graphical interface												
Meeting to define the Testing phase												
Tests in control people												
Data analysis												
Writing the memory												

ACTIVITY	2022															
	JANUARY				FEBRUARY				MARCH				APRIL			
Project proposal meeting																
Meeting to define the title of the project																
Define objectives																
Pre-course on unity																
Research on the disease																
Brainstorming																
Determination of the design																
Determination of the content of the software																
Development of the tasks	■	■	■	■	■	■										
Performing tests in simulations	■	■	■	■	■	■										
Restructuring of tasks	■	■	■	■	■	■	■	■	■	■	■	■				
Identification of fundamental data for the study																
Data extraction methods																
Development of graphical interface	■	■														
Meeting to define the Testing phase	■	■														
Tests in control people			■	■	■											
Data analysis									■	■	■	■	■	■	■	■
Writing the memory									■	■	■	■	■	■	■	■

Figure 3. Project timeline

5.3. Materials Used

5.3.1. Unity

Unity is a multiplatform video game engine, created by the company "Unity Technologies". This platform has a compilation support with other platforms, for example: WebGL, Linux, Windows, OS X, iOS, Android, Samsung TV, PlayStation, Xbox, Wii U, Oculus Rift, Google Cardboard, HTC Vive and many more[15]. This time we used the compilation support with Windows and Android for the development of this project [15]. Thanks to the simulation option provided by Unity, it was possible to carry out all the software design focused on the format and size of a Tablet.

Thanks to the simulation option provided by Unity, it was possible to make the entire software design focused on the format and size of a Tablet and a version for touch computers was also made.

5.3.1.1. Interface

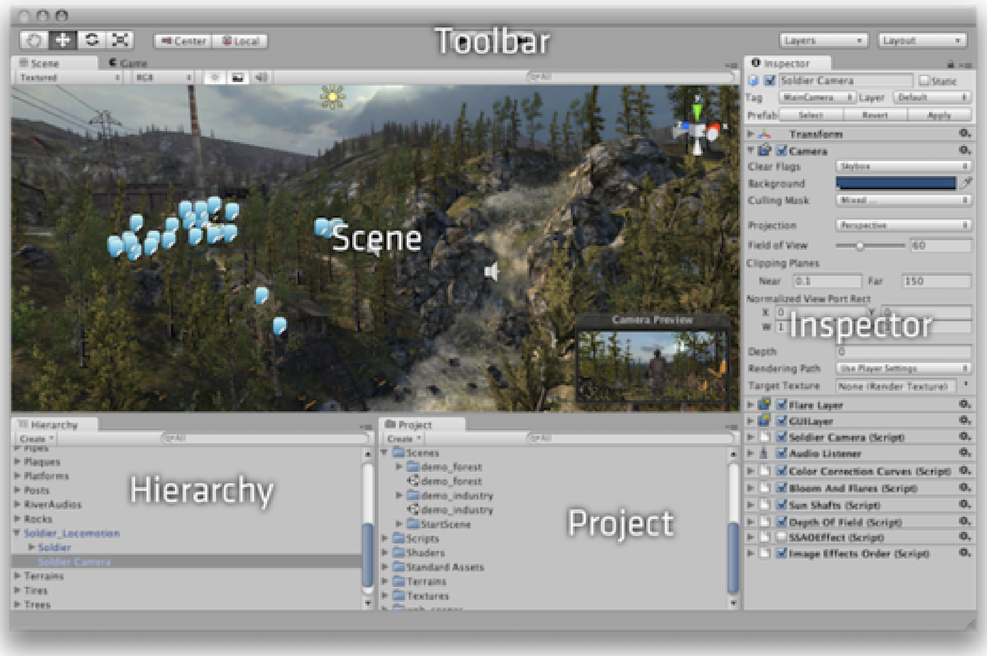


Figure 3. Unity interface. Source: http://deusexmachina.es/taller-empezando-con-unity-3d/

The Unity interface is divided into 5 tabs, the order of which can be modified according to the developer's preferences.

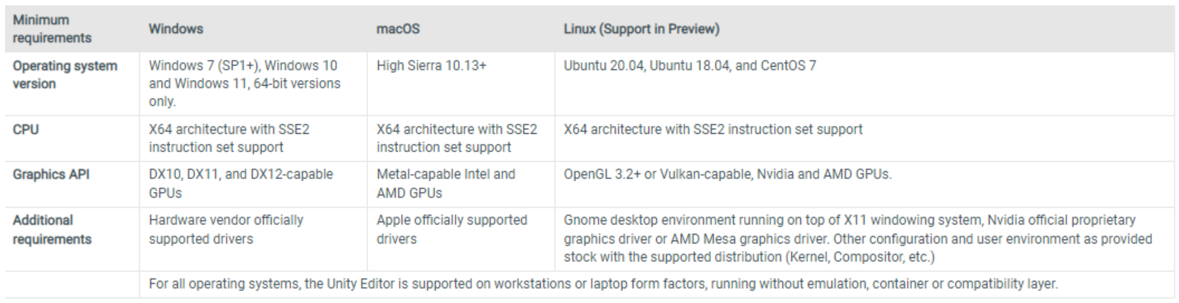
1. **ToolBar**: As can be seen in Figure 3, at the top is the "Toolbar" or toolbar. In the Toolbar you can create objects, folders and perform any type of deformation that you want to apply to an object represented in the "Scene" screen. Likewise, you

can also change the reference points of such an object from local to global depending on your needs. Finally, one of the main functions of the Toolbar is the simulation of the application being designed by means of the "Play" button located in the center of the upper part.

2. **Scene**: This is where the design of the application, game or user interface (UI) is carried out. This window is used to create and visualize the created objects, it is also here where the modifications in the space are made, by means of this tool it is determined what is seen in the final result of the application, however to be able to see the final result in color the Game window is used.
3. **Inspector**: This window is used to change the parameters of the created objects and add properties to them. You can change the size, position and scale, as well as introduce sounds, videos, images, textures, scripts, etc. It has many functionalities that allow you to create a whole virtual environment and customize it as much as you want.
4. **Hiderachy**: In this window we can see the created objects that we have in a scene. Also, this window is used to sort those objects and mark an order in their situation and importance with respect to the others..
5. **Project**: In this window we can sort all the created files such as scenes, textures, codes, among others.

5.3.1.2. Minimum usage requirements

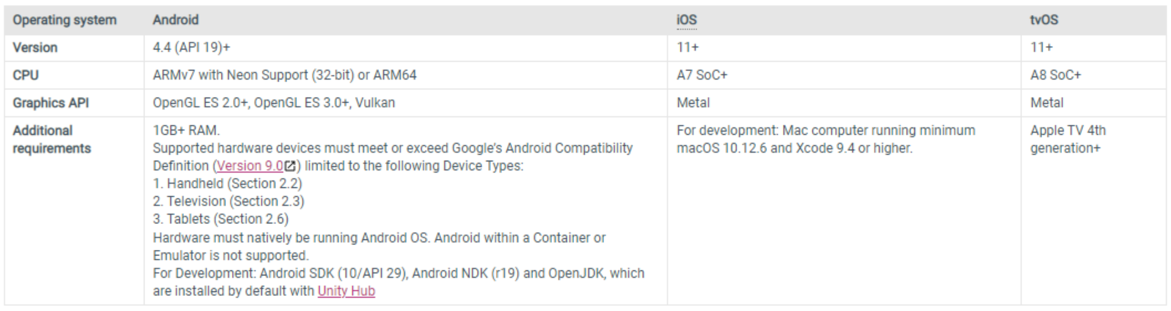
In order to use this platform as a developer it is necessary to obtain a computer that meets the following characteristics depending on the system used. (See Table 2)



Minimum requirements	Windows	macOS	Linux (Support in Preview)
Operating system version	Windows 7 (SP1+), Windows 10 and Windows 11, 64-bit versions only.	High Sierra 10.13+	Ubuntu 20.04, Ubuntu 18.04, and CentOS 7
CPU	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, and DX12-capable GPUs	Metal-capable Intel and AMD GPUs	OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs.
Additional requirements	Hardware vendor officially supported drivers	Apple officially supported drivers	Gnome desktop environment running on top of X11 windowing system, Nvidia official proprietary graphics driver or AMD Mesa graphics driver. Other configuration and user environment as provided stock with the supported distribution (Kernel, Compositor, etc.)
	For all operating systems, the Unity Editor is supported on workstations or laptop form factors, running without emulation, container or compatibility layer.		

Table 2. Minimum requirements for Unity functionality as a developer [16].

In addition to this, it is necessary to have some minimum requirements to be able to run the application on the device from which it is to be used (See Table 3).



Operating system	Android	iOS	tvOS
Version	4.4 (API 19)+	11+	11+
CPU	ARMv7 with Neon Support (32-bit) or ARM64	A7 SoC+	A8 SoC+
Graphics API	OpenGL ES 2.0+, OpenGL ES 3.0+, Vulkan	Metal	Metal
Additional requirements	1GB+ RAM. Supported hardware devices must meet or exceed Google's Android Compatibility Definition (Version 9.0) limited to the following Device Types: 1. Handheld (Section 2.2) 2. Television (Section 2.3) 3. Tablets (Section 2.6) Hardware must natively be running Android OS. Android within a Container or Emulator is not supported. For Development: Android SDK (10/API 29), Android NDK (r19) and OpenJDK, which are installed by default with Unity Hub	For development: Mac computer running minimum macOS 10.12.6 and Xcode 9.4 or higher.	Apple TV 4th generation+

Table 3. Minimum requirements for Unity functionality as an application [16].

5.4. Application Requirements

5.4.1. Functional Requirements

a. Correct registration of each user's session.
b. Correct login of the registered user.
c. Information on Task Instructions.
d. Extraction of data of interest

5.4.2. Non-Functional Requirements

a. High processing speed
b. Performance (Response time to user actions less than one second)
c. Hardware (Computer Lenovo ideapad 320 and Tablet Samsung A7)
d. Software (Unity)
e. C# programming language
f. Graphical interface
g. Windows 7 or higher version
h. Possibility of being used in several devices
i. Operation without internet
j. Local data collection

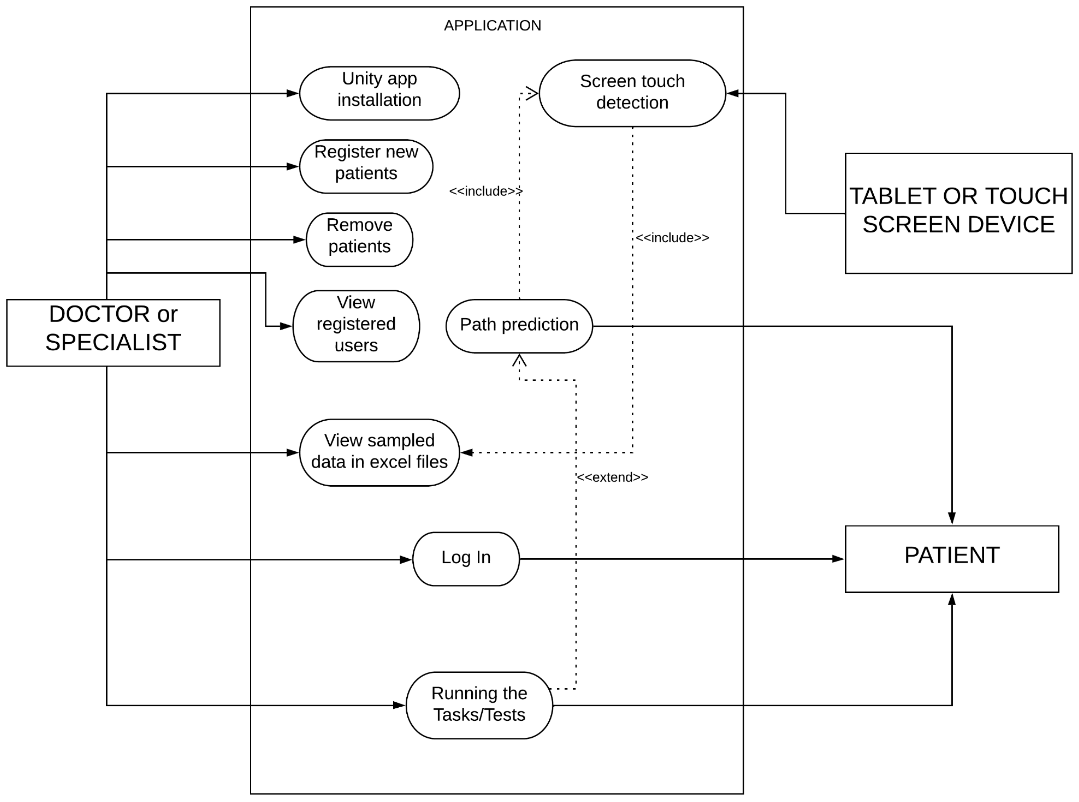


Figure 4. Use Case Diagram.

6. Software design and development

Taking into account the literature review, it was decided to focus the software tasks to complex visual processing tasks, where as already mentioned the main objective will be the trajectory prediction by the patient. In order to evaluate the trajectory prediction different tasks are proposed, in this section the step by step of the realization for the design and development of the software will be exposed, showing the different tasks created for the development of this, together with their respective description.

At first we will describe the steps to follow for the creation of the different necessary scenes, then, in the Figure 5 we find the window called "Project", as already explained, this window is located at the bottom of the Unity application, where you can see and sort all the files that have a scene or project. In this window as it is observed the created scenes, images, scripts and objects, necessary for the development of the application are saved.

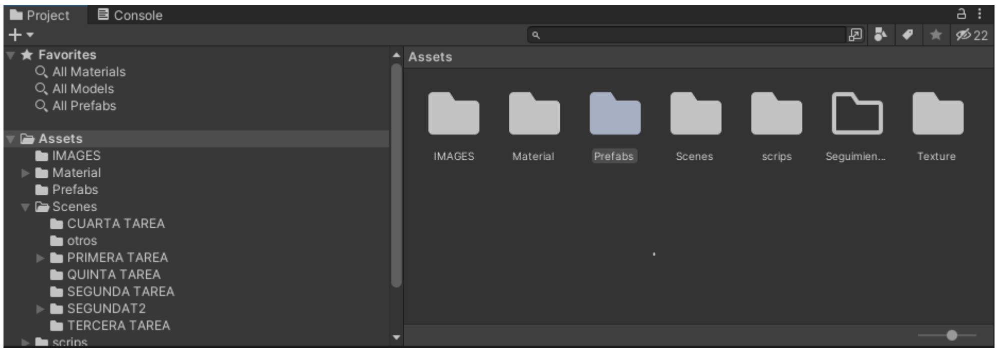


Figure 5. Project window view.

Once the scene is created, objects that are needed in the scene are created through the platform, thanks to the Assets tool as shown in Figure 6.

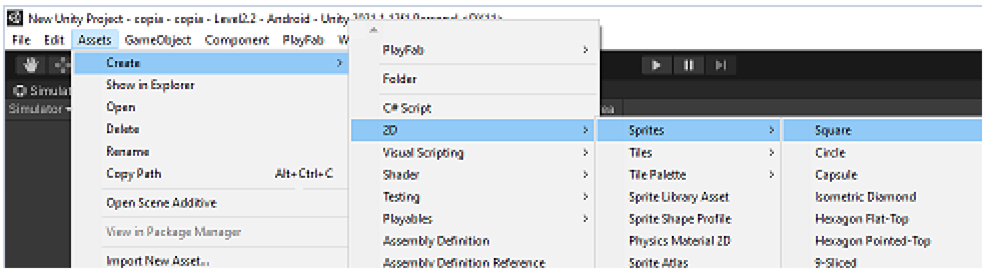
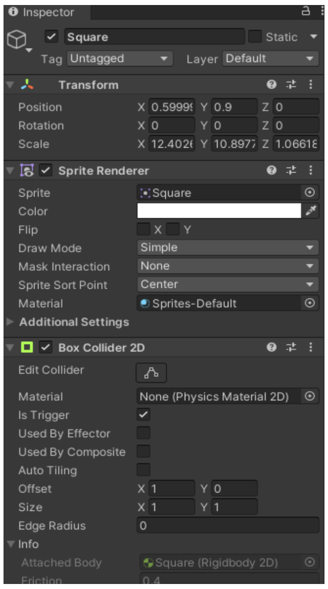


Figure 6. View of how an object would be created using the assets tool.

Once the objects have been created, the necessary properties are added to them from the inspector window, this window is located in the upper right part of the application, see Figure 7.



```
Inspector
Square                         Static
Tag Untagged        Layer Default
Transform
Position     X 0.5999  Y 0.9      Z 0
Rotation     X 0       Y 0        Z 0
Scale        X 12.402  Y 10.8977  Z 1.06618
Sprite Renderer
Sprite                 Square
Color
Flip                   X  Y
Draw Mode              Simple
Mask Interaction       None
Sprite Sort Point      Center
Material               Sprites-Default
Additional Settings
Box Collider 2D
Edit Collider
Material               None (Physics Material 2D)
Is Trigger             ✓
Used By Effector
Used By Composite
Auto Tiling
Offset                 X 1   Y 0
Size                   X 1   Y 1
Edge Radius            0
Info
Attached Body          Square (Rigidbody 2D)
Friction               0.4
```

Figure 7. Inspector window view.

6.1. Development of the graphical interface

Unity 2D allows the developer to design UI "User Interface" in order to create menus, letters or credits, buttons, controls, etc. For the development of this application different tools have been used to create all kinds of objects that interact with each other, these tools can be seen in Figure 8.

These tools will be used to create not only the start interface of the application, but also to create text interfaces to provide information to patients about the steps to follow throughout the test, as well as all kinds of objects needed for the different scenes.

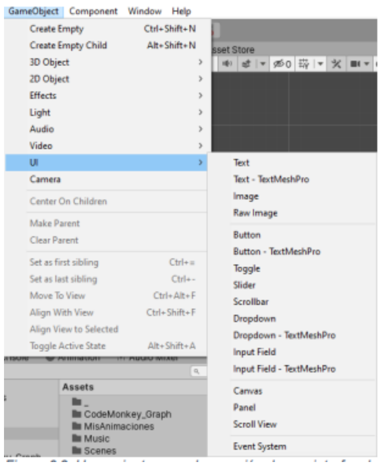


Figure 8. User Interface “UI”

For this project it is very important to have a graphical start interface with a registration and login system, in order to have a record of patients who make use of this application. For the creation of this start interface at first the option of using different software such as MySQL, PlayFab, among others, for the creation of a local database to store the data, but because it was sought that the software was an application for Android was not possible to save a record of the data, because of this it was decided to extract and save the data of each patient in text files (see Figure 9).

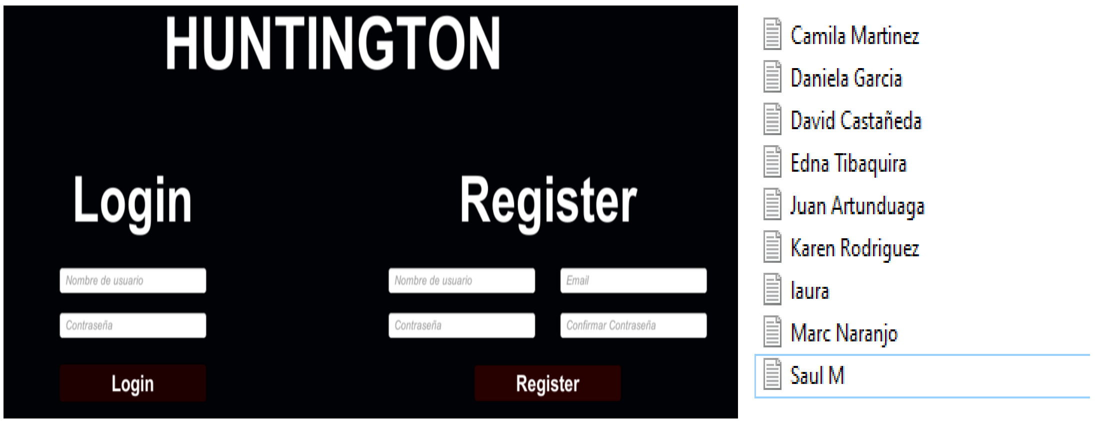


Figure 9. [Left] Graphical interface of application startup with Login and registration system. [Right] Example of text files exported from the registry.

To develop this login interface we first created a UI with the necessary text windows, after that to make it possible to enter data such as name, email and password we used the "login file" tool provided by Unity, but this tool by itself does not perform a registration or login, for this it is necessary to assign a code to control the information received, this will be explained in more detail below.

- **Registration System:** This system is composed of the user name, e-mail and password, for the development of the system it was necessary to use a code named "Register" which can be seen in its entirety in the Appendix IV. However, the most important aspects of the code will be explained as follows.

  First we declare the necessary variables. In this part we will find variables such as user name, email, password, password confirmation and a list of allowed characters. After the declaration of variables we will find 3 methods:

  - RegisterButton: As its name indicates, this method is entered by pressing the register button, for this the code called "register" is used, in this method a file path is provided and it is identified if the input variables already exist, if so, a message will be displayed indicating that the input variables already exist, otherwise, if these variables do not exist and are entered correctly, a text file will be created in the path indicated with the user name entered, Otherwise, if these variables do not exist and are entered correctly, a text file will be created in the path indicated with the user name entered, in this file the patient's personal information will be saved, because the password is also saved and a line of code is generated so that the password is encrypted when it is saved, in this way only the patient knows the password entered. In addition to this, some conditions are set such as the password must have a minimum of 6 digits and a password confirmation is added.

- Update: In this method, as its name says, the information of the application is updated frame by frame verifying what the user is typing in the fields, once the user completes the fields, it is verified if they are correct.
- EmailValidation: To verify that the email is true, a few lines of code are generated, stipulating conditions such as that the email must have an @ symbol and end in .com.

- **Login System:** This system is composed of the user name and password, for the development of the system it was necessary to use a code named "Login" which can be seen in full in the Appendix IV. However, the most important aspects of the code will be explained.

  First we declare the necessary variables in this part we will find variables such as user name and password. After the declaration of variables we will find 2 methods:

  - LoginButton: As its name indicates, this method is entered by pressing the Login button, the purpose of this button is to change to the next scene only if the username and password entered have already been registered, this is checked by means of the "register" code.
  - Update: Once again in this method, as its name says, the information of the application is updated frame by frame verifying what the user is typing in the fields, once the user completes the fields, it is verified if they are correct.

6.2. Development of Tasks

Based on the idea of creating tasks based on trajectory prediction, 3 tests are proposed at the beginning, each of them will have example scenes and after these there will be scenes where the patient will have to perform the trajectory prediction based on the previously seen example. On the other hand, it is important to clarify that both the background color and the color chosen for the objects meet a specific objective, what is sought by putting the dark background (Black) and light objects (White), is not to generate an excess of visual stimuli or distractions, so the patient can focus on the specific task of trajectory prediction.

All tasks follow the same principle. In the example scenes a circle will trace a movement and at the moment the circle crosses the box the circle will disappear for a few seconds and then reappear continuing its original trajectory, while in the trajectory prediction scenes once the circle passes through the box it does not reappear. However, each trajectory prediction scene is preceded by an example scene of such a task.

For each of them we must create 3 scenes that will represent the number of trials for each task, each scene will have 3 basic objects. These objects will be assigned properties, these properties for each object are described below:

a. **Square:** This object acts as an activator or trigger of the event, where the circle will disappear or reappear. The square will be static and located in the middle of the screen. To make this object work as a trigger, some properties are added to it through the "Inspector" window: RigidBody 2D which is important to add physical properties to the objects, in this case we set the Body Type to Kinematic, also we add a Box Collider 2D which is necessary to detect collisions between objects, to this collider we mark the option "Is Trigger" and we also change the value of OffSet to achieve that the reappearance takes place a few seconds after crossing the square, this we can see in Figure 10. These properties will be the same for both the example scene and the trajectory prediction scene.

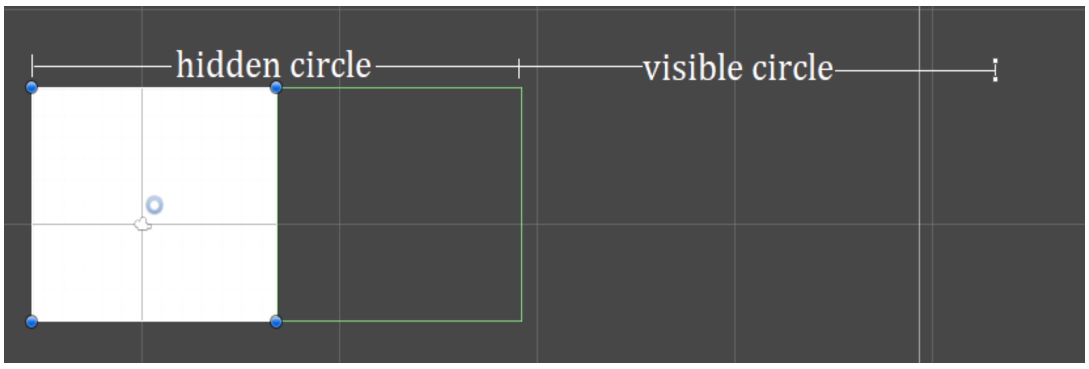
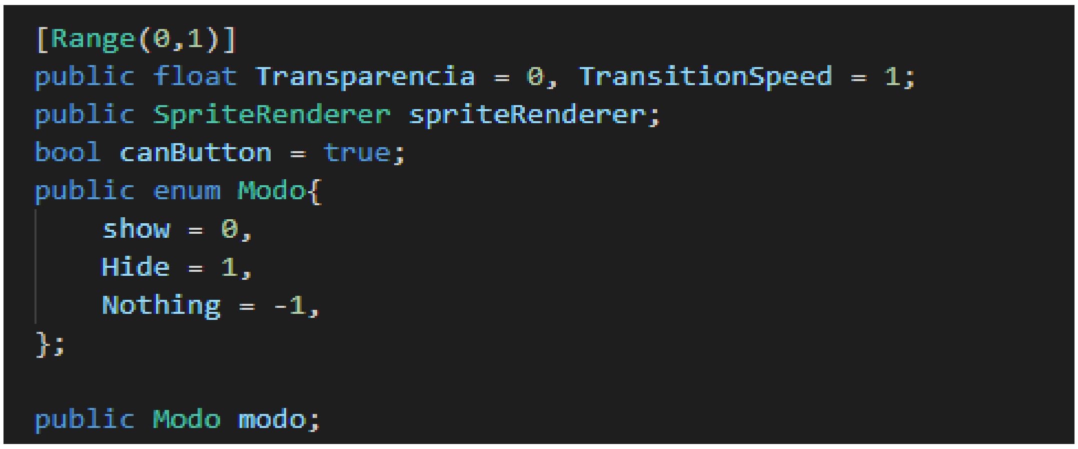


Figure 10. "Square", event triggering object.

b. **Circle:** This object acts as a moving visual stimulus that will describe different trajectories. To achieve this, first of all we define the initial position of the circle, besides this we add some important properties such as: a 2D RigidBody which is used to add physical properties to the objects, this must be set to Body Type Dynamic state, on the other hand a 2D Circle Collider is needed which is necessary to detect collisions between objects and finally code scripts are needed to control the speed, direction and disappearance of the circle and its subsequent appearance in cases where necessary, finally the Tag option is changed to Player, all this is done from the "Inspector" window.

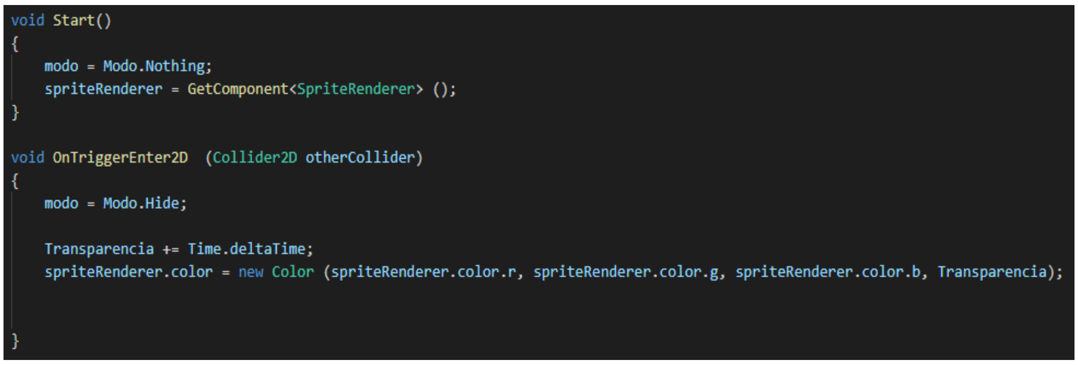
Example scene: In this scene the circle must disappear and reappear after an indefinite time, for this what is done is to use the Script called "transp2" as shown in Figure 11. Because we need the circle to "disappear" but still continue its trajectory, what is done is to use the transparency property of the circle. For this when declaring the variables and 3 modes are defined that consist of showing, hiding or not doing anything with the circle.



```
[Range(0,1)]
public float Transparencia = 0, TransitionSpeed = 1;
public SpriteRenderer spriteRenderer;
bool canButton = true;
public enum Modo{
    show = 0,
    Hide = 1,
    Nothing = -1,
};

public Modo modo;
```

Figure 11. Script "transp2", definition of variables.

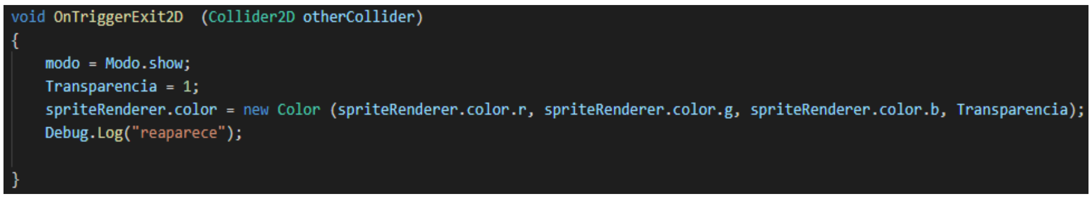
After this, by means of the Start method, the Mode variable is initialized in the "Nothing" state, we also name a spriteRender variable and we assign it the rendering of the circle object. Once the circle crosses the square what really happens is that the two objects collide, for this reason we create the method called "OnTriggerEnter2D" by means of which at the moment of the collision of the objects the variable Mode is changed to "Hide" and the color of the object is updated by means of the components (RGBA) making Alpha to be in transparency state. All this is evidenced in Figure 12.



```
void Start()
{
    modo = Modo.Nothing;
    spriteRenderer = GetComponent<SpriteRenderer> ();
}

void OnTriggerEnter2D  (Collider2D otherCollider)
{
    modo = Modo.Hide;

    Transparencia += Time.deltaTime;
    spriteRenderer.color = new Color (spriteRenderer.color.r, spriteRenderer.color.g, spriteRenderer.color.b, Transparencia);


}
```

Figure 12. Script "transp2", activation method “OnTriggerEnter2D”.

Finally, because we want the circle to appear again after some time, for this we create the OnTriggerExit2D method in which we will identify the moment in which the circle stops colliding with the square, once this happens we change the state of the Mode variable to "show" and remove the transparency of the object by setting it to 1, as shown in Figure 13.

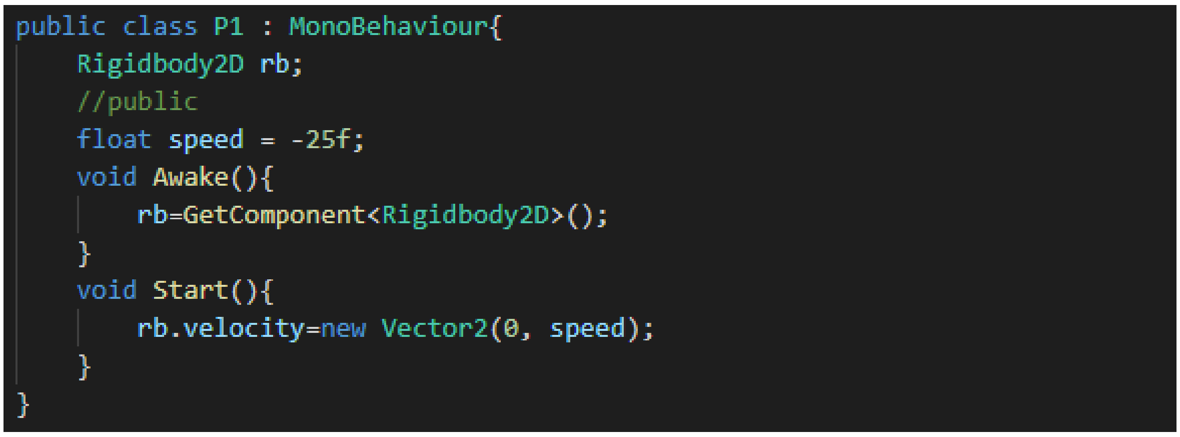


```
void OnTriggerExit2D  (Collider2D otherCollider)
{
    modo = Modo.show;
    Transparencia = 1;
    spriteRenderer.color = new Color (spriteRenderer.color.r, spriteRenderer.color.g, spriteRenderer.color.b, Transparencia);
    Debug.Log("reaparece");

}
```

Figura 13. Script "transp2", deactivation method “OnTriggerExit2D”.

Scene of trajectory prediction: In this scene the circle must disappear when crossing the square and not to appear again, nevertheless once it disappears it must continue with its trajectory for this what is done is to use the Script called "transp3", which is exactly the same as "transp2" the only difference is that in this code we will not include the method OnTriggerExit2D, due to this the circle never leaves the method OnTriggerEnter2D for this it continues its trajectory with the variable Mode in "Hide" thus the circle remains transparent during all the scene.

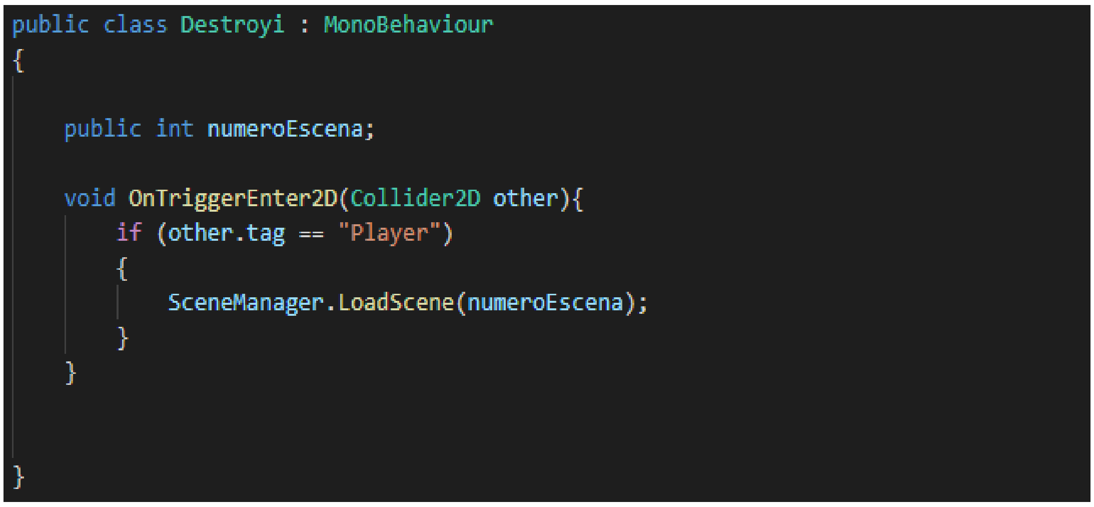
Finally, the circle object needs a code to generate the movement, dictate the trajectory and finally control the speed, an example of which can be seen in Figure 14. However, this code will change depending on the task since the direction and speed will be modified.



```
public class P1 : MonoBehaviour{
    Rigidbody2D rb;
    //public
    float speed = -25f;
    void Awake(){
        rb=GetComponent<Rigidbody2D>();
    }
    void Start(){
        rb.velocity=new Vector2(0, speed);
    }
}
```

Figure 14. Script "P1", code to determine circle object properties.

c. **Rectangle:** This object acts as an activator or trigger of the event where you want to change the scene without pressing a button. This object is not visible and will be located in different positions depending on the trajectory of the circle. To make the object not visible the Sprite Render property has been removed from the "Inspector" window that comes by default in each object, however, other properties are added such as a Box Collider 2D which is necessary to detect collisions between objects, this collider is marked with the option of "Is Trigger" to detect the moment in which the circle ends the trajectory, in addition to this a code script is needed to achieve the scene change at the time of collision, this can be seen in Figure 15.

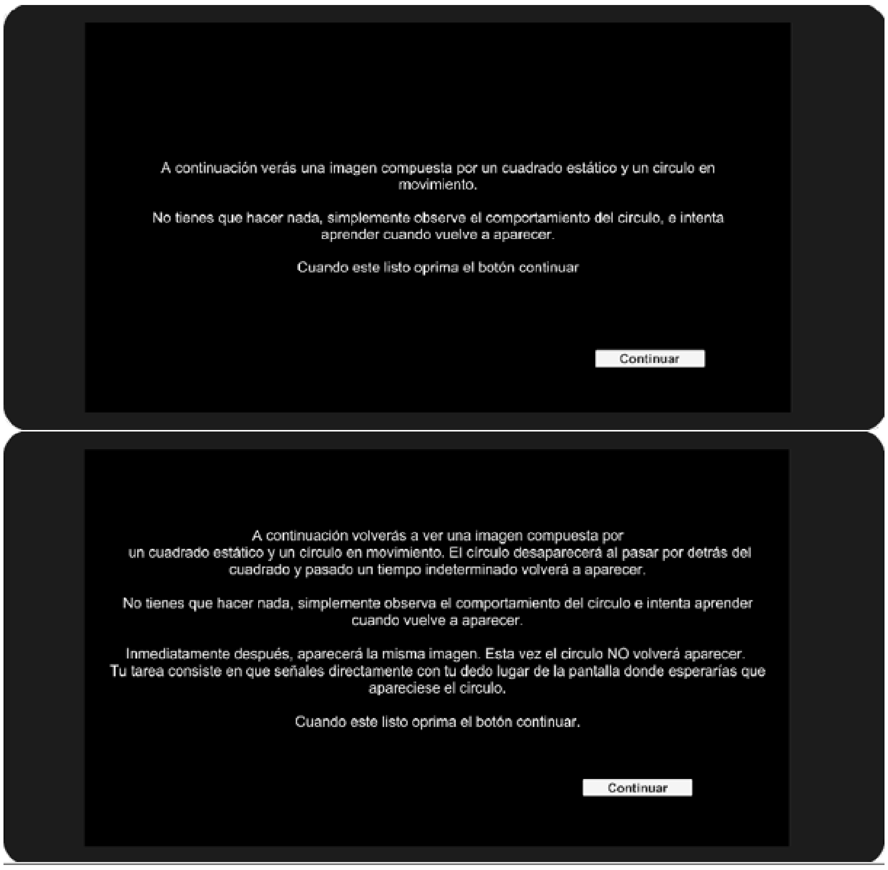


```
public class Destroyi : MonoBehaviour
{

    public int numeroEscena;

    void OnTriggerEnter2D(Collider2D other){
        if (other.tag == "Player")
        {
            SceneManager.LoadScene(numeroEscena);
        }
    }


}
```

Figure 15. Script "Destroyi", is in control of the change of scenes.

In this figure you can see the OnTriggerEnter2D method with it is possible to detect the collision between the rectangle and the circle. When entering the method it is verified that it has collided with the circle by means of the conditional "if", once corroborated this will be changed to the scene number indicated by means of the variable numberScene, which can be modified from the sale "Inspector".

Each task will start with an information scene that consists of a text interface where the patient will be explained what he/she will see and will be given indications on what to do (See Figure 16. A), once ready the patient will press the continue button, which has a code associated with which it is possible to make scene changes, this code is called "MainMenu" and can be seen in its entirety in the annexes, In this way, once the continue button is pressed, the patient will change to the next scene, which will be an example scene, the example scenes, as already mentioned, will show the patient the behavior of the circle so that he can learn it, once the example scenes are finished, he will automatically go back to an information scene (See Figure 16. B) where the patient will be explained what he will have to do once he goes to the trajectory prediction scenes.


A continuación verás una imagen compuesta por un cuadrado estático y un circulo en movimiento.

No tienes que hacer nada, simplemente observe el comportamiento del circulo, e intenta aprender cuando vuelve a aparecer.

Cuando este listo oprima el botón continuar

A continuación volverás a ver una imagen compuesta por un cuadrado estático y un circulo en movimiento. El circulo desaparecerá al pasar por detrás del cuadrado y pasado un tiempo indeterminado volverá a aparecer.

No tienes que hacer nada, simplemente observa el comportamiento del circulo e intenta aprender cuando vuelve a aparecer.

Inmediatamente después, aparecerá la misma imagen. Esta vez el circulo NO volverá aparecer. Tu tarea consiste en que señales directamente con tu dedo lugar de la pantalla donde esperarías que apareciese el circulo.

Cuando este listo oprima el botón continuar.

Figure 16. [Up]The "Instructions" scene, example of instructions for example scenes.
[Down] The "InfL1" scene, example of instructions for prediction scenes.

6.2.1. Task 1

This task consists of three trials coming from three different directions, an example of which can be seen in Figure 17. In the shown scenes you can see the trajectories that the circle will follow, exemplified by a dotted white line, and a red circle which indicates the place where the circle will be visible again.

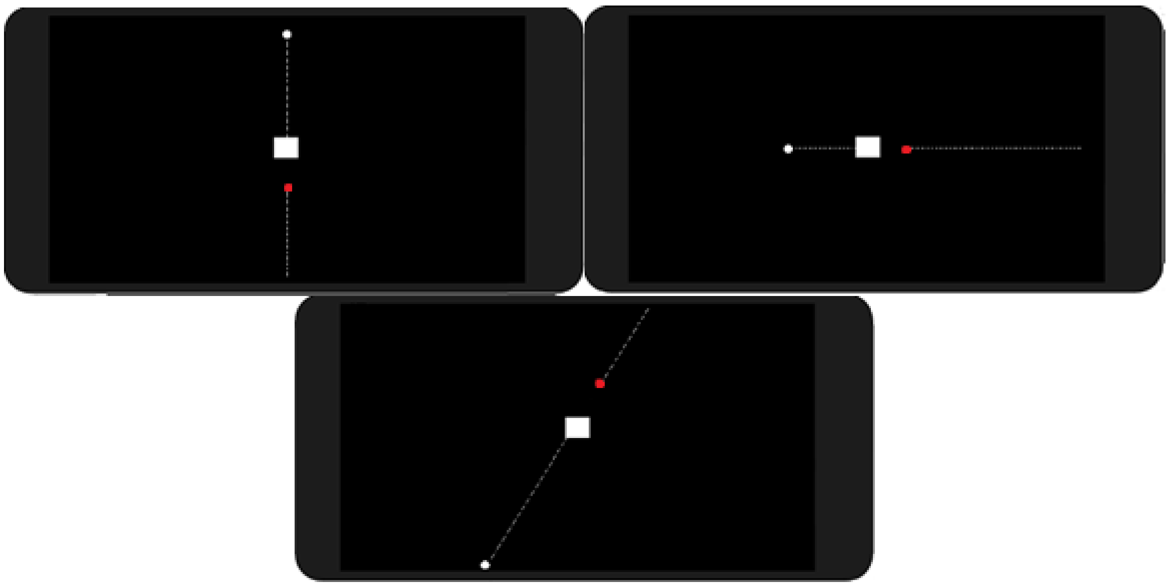


Figure 17. Example scenes of the different tests of task 1.

In the previous figure it is observed that the dotted line is not in a section of the trajectory, this represents the moment in which the circle will not be visible in the example scenes. On the other hand, for the trajectory prediction scenes, as we already know, once the circle crosses the square it will not be visible again and the patient must indicate where the point is located before the trajectory ends by tapping on the screen.

For this first task the circle will go at a medium speed and will be constant during the whole trajectory, at the moment that the circle passes behind the square the circle will disappear for a few seconds and will reappear continuing with its original trajectory. The scenes where the instructions for task one are given are shown in figure 16.

6.2.2. Task 2

This task consists of three trials coming from three different directions, very similar to the previous task, keeping the same speed as in the previous task with the exception that for this task an 8-second counter will be used, the counter will start at the same time the circle movement starts.

For the example scenes the circle will become visible once the counter reaches zero and will continue with its normal trajectory. On the other hand, for the trajectory prediction scenes, once the circle passes behind the square it will not appear again, the patient must indicate where the circle should be when the counter reaches zero.

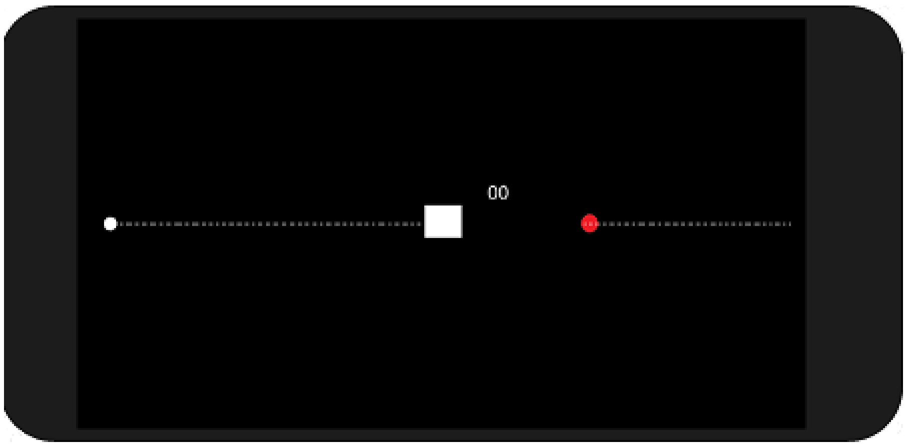
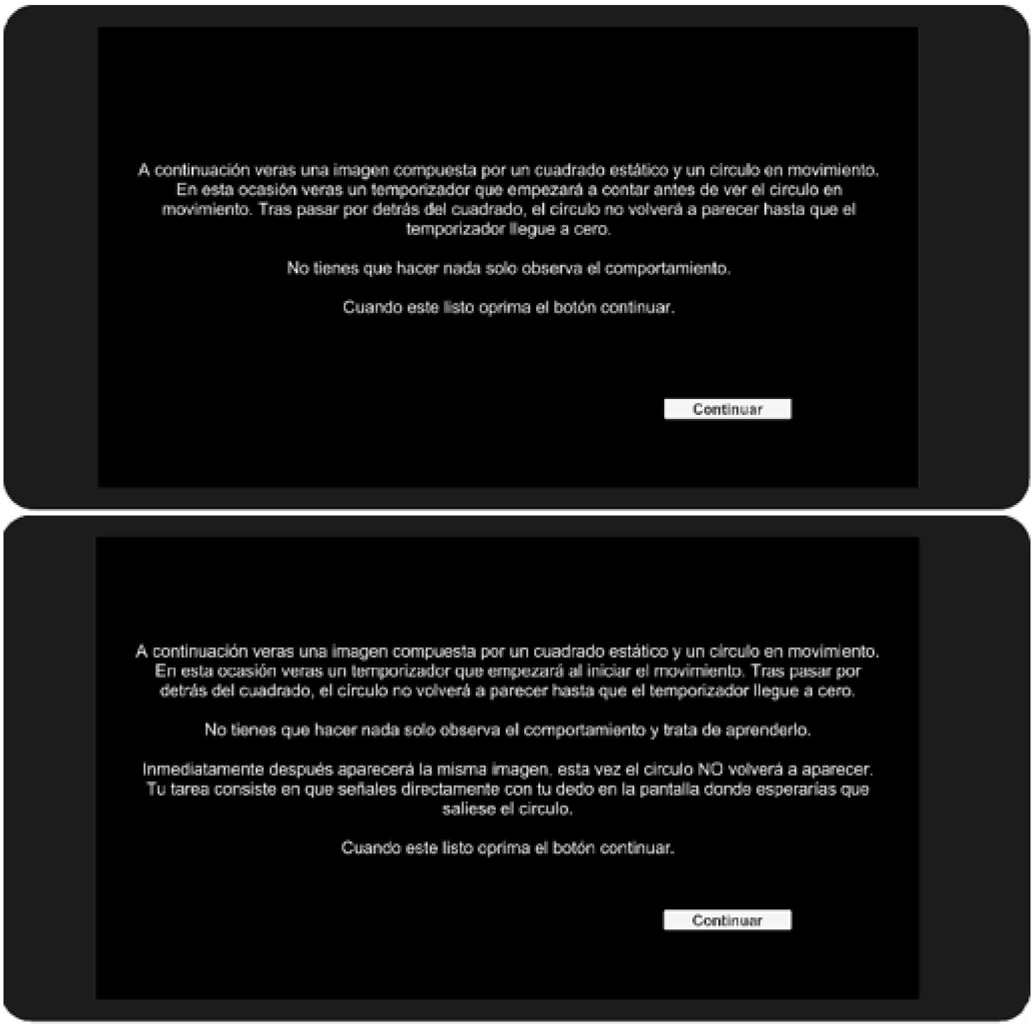


Figure 18. Example scene for task 2.

Figure 18 shows one of the trajectories that the circle will follow, exemplified by a dotted white line and a red circle that indicates the place where the circle becomes visible again. As it can be observed for this task the time in which the circle remains not visible in the example scenes increases with respect to the previous task. In addition, in the trajectory prediction task we are only interested in the patient identifying the moment when the circle should reappear, which is when the counter is equal to zero.

In addition, in the vertical and diagonal trajectories, the circle is not visible when the counter starts automatically, it appears one second later, so the patient is not predisposed to a trajectory.

The instructions given for task two are shown in Figure 19 below. As in the previous task, in this task the continue buttons also have the "MainMenu" method where the scene change is performed.


A continuación veras una imagen compuesta por un cuadrado estático y un circulo en movimiento. En esta ocasión veras un temporizador que empezará a contar antes de ver el circulo en movimiento. Tras pasar por detrás del cuadrado, el circulo no volverá a parecer hasta que el temporizador llegue a cero.

No tienes que hacer nada solo observa el comportamiento.

Cuando este listo oprima el botón continuar.

A continuación veras una imagen compuesta por un cuadrado estático y un circulo en movimiento. En esta ocasión veras un temporizador que empezará al iniciar el movimiento. Tras pasar por detrás del cuadrado, el circulo no volverá a parecer hasta que el temporizador llegue a cero.

No tienes que hacer nada solo observa el comportamiento y trata de aprenderlo.

Inmediatamente después aparecerá la misma imagen, esta vez el circulo NO volverá a aparecer. Tu tarea consiste en que señales directamente con tu dedo en la pantalla donde esperarías que saliese el circulo.

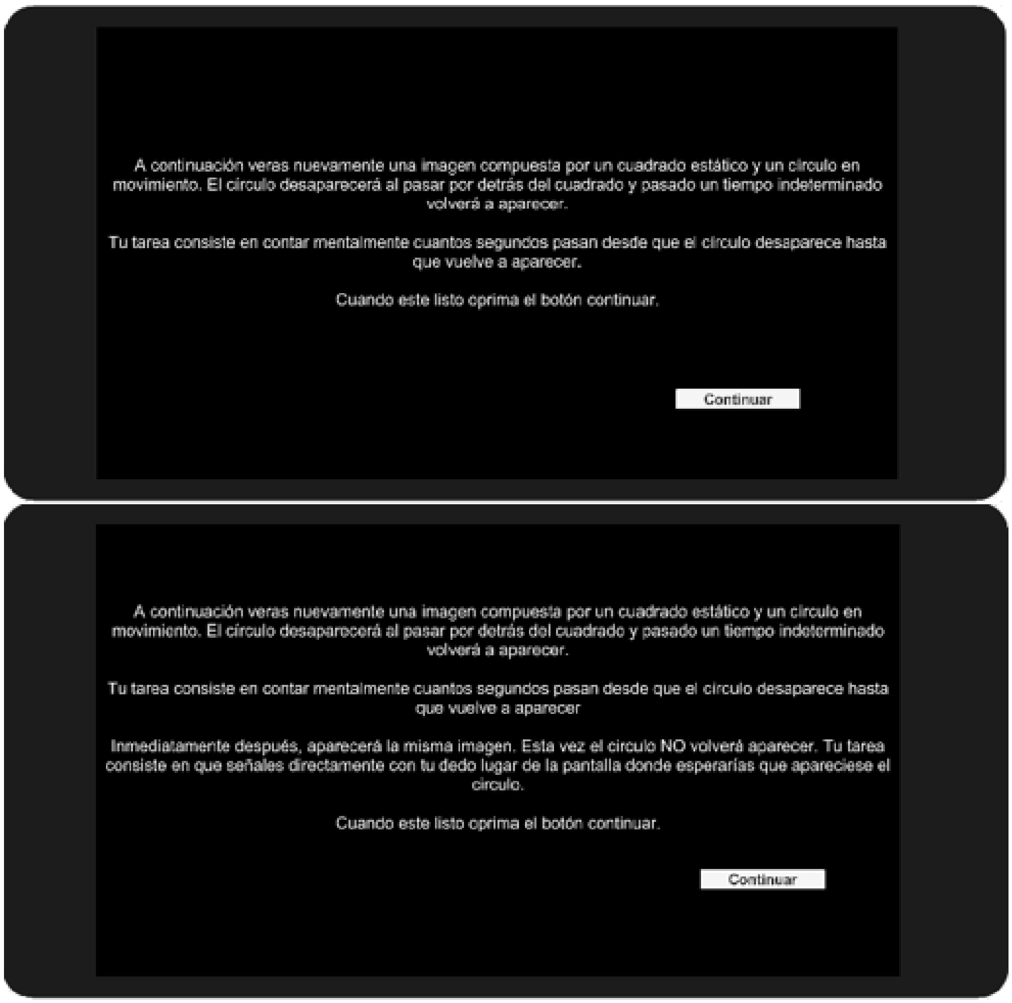
Cuando este listo oprima el botón continuar.

Figure 19. [Up] The "NL2" scene, example of instructions for example scenes. [Down] The "NLL3" scene, example of instructions for prediction scenes.

6.2.3. Task 3

This task is the same as the previous task, with the exception that no use is made of the counter, since for this task it is intended that the patient can mentally identify the time that elapses between the moment in which the circle passes behind the square until it becomes visible again in the example tasks and thus use this time determined by them to identify where the circle will reappear in the trajectory prediction tasks.

The instructions given to the user for performing this task are shown in Figure 20.


A continuación veras nuevamente una imagen compuesta por un cuadrado estático y un circulo en movimiento. El circulo desaparecerá al pasar por detrás del cuadrado y pasado un tiempo indeterminado volverá a aparecer.

Tu tarea consiste en contar mentalmente cuantos segundos pasan desde que el circulo desaparece hasta que vuelve a aparecer.

Cuando este listo oprima el botón continuar.

A continuación veras nuevamente una imagen compuesta por un cuadrado estático y un circulo en movimiento. El circulo desaparecerá al pasar por detrás del cuadrado y pasado un tiempo indeterminado volverá a aparecer.

Tu tarea consiste en contar mentalmente cuantos segundos pasan desde que el circulo desaparece hasta que vuelve a aparecer

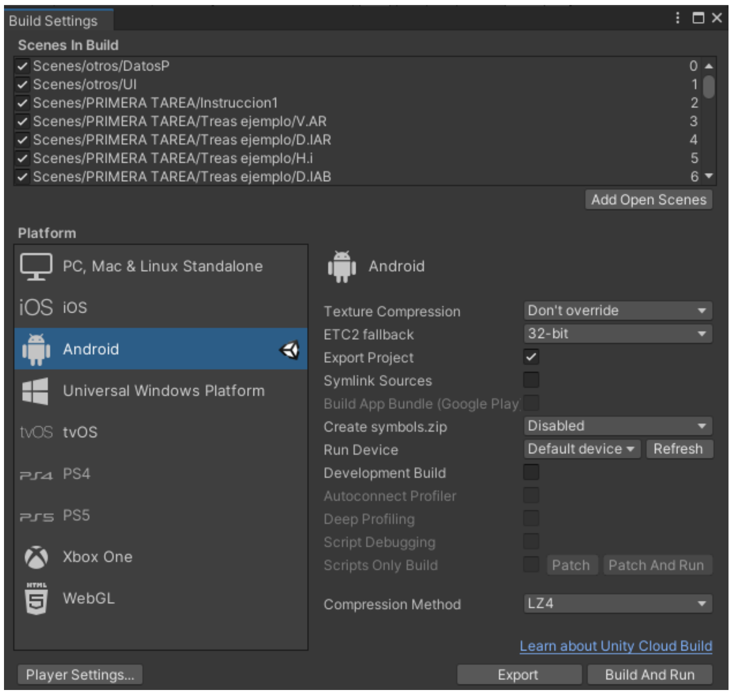
Inmediatamente después, aparecerá la misma imagen. Esta vez el circulo NO volverá aparecer. Tu tarea consiste en que señales directamente con tu dedo lugar de la pantalla donde esperarías que apareciese el circulo.

Cuando este listo oprima el botón continuar.

Figure 20. [Up] The "INFOR2" scene, example of instructions for example scenes. [Down] The "INFOR3" scene, example of instructions for prediction scenes.

6.2.4. Changing the format of the application to make Android work

Once the startup interface was ready and all the tasks were developed, the scenes were organized in folders in the unity project window. So far only the scenes had been executed through simulations, but to convert the different scenes into an Android application, go to the Toggle window and go to File and click on Build Settings, once this is done a box like the one shown in Figure 21 will appear.



Build Settings

Scenes In Build

✓	Scenes/otros/DatosP	0
✓	Scenes/otros/UI	1
✓	Scenes/PRIMERA TAREA/Instruccion1	2
✓	Scenes/PRIMERA TAREA/Treas ejemplo/V.AR	3
✓	Scenes/PRIMERA TAREA/Treas ejemplo/D.IAR	4
✓	Scenes/PRIMERA TAREA/Treas ejemplo/H.i	5
✓	Scenes/PRIMERA TAREA/Treas ejemplo/D.IAB	6

Platform

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

Android

Texture Compression	Don't override
ETC2 fallback	32-bit
Export Project	✓
Symlink Sources	
Build App Bundle (Google Play)	
Create symbols.zip	Disabled
Run Device	Default device
Development Build	
Autoconnect Profiler	
Deep Profiling	
Script Debugging	
Scripts Only Build	
Compression Method	LZ4

Figure 21. Ventana Build Stings

As can be seen in Figure 21, in this window we will find 2 divisions, one is at the top and it is Scenes In Build, in this section we will drag the scenes created in the order in which we want them to appear during the application, as can be seen each scene is assigned a number and this is very useful because knowing this numbers is much easier to make a change from one scene to another.

On the other hand, at the bottom of Figure 1 we see a window called Platform, in this section we can change the platform where you can play the application, because for this specific case we want to make an application for Android, we choose this platform and check the Export Project checkbox.

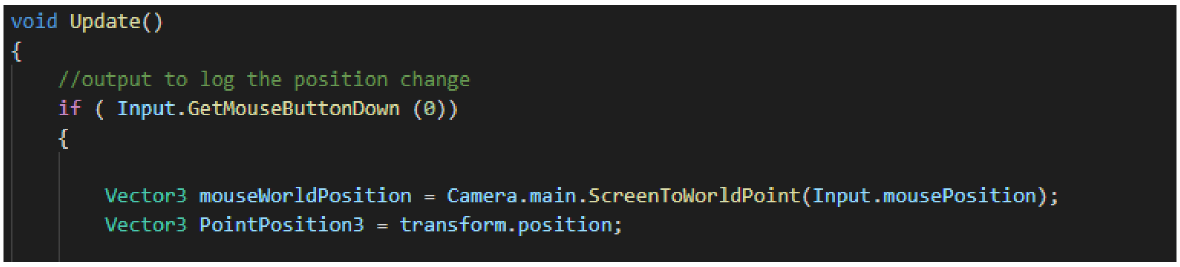
Once this process is finished, the application can be exported, however, so that the application could be played on the Tablet, first of all some settings had to be generated, such as activating the developer options and downloading Unity Remote 5 where it is possible to play the application and see if there are failures in it through the computer console.

Finally, once all the previous process is done, the application is as follows: Once the Tablet is connected to the computer through the Unity Remote 5 application, the startup graphical interface appears, where the user registers for subsequent login, once the user enters the game the information comes out where the example scene of the corresponding task is explained, after this the example scene is shown, followed by the information of the trajectory prediction scene corresponding to the example scene shown previously, like this for the three trials of each task and this is repeated for the 3 tasks. The video of the application can be found in the Appendix V.

6.3. Identifying and extracting data of interest

Once the tasks to be evaluated have been defined, the way in which the trajectory prediction can be measured is defined. For this, the position of the circle that the patient has indicated by means of the mouse or tap on the screen is extracted, and at the same time the real position of the circle at that moment is extracted, in order to compare how far the real position values are from the trajectory prediction made by the patient.

To obtain this data, the lines of code shown in Figure 22 are used. This code is implemented in the Update method of the code already created previously to generate the movement of the circle explained in section 6.2 in the explanation of the trajectory prediction scene.

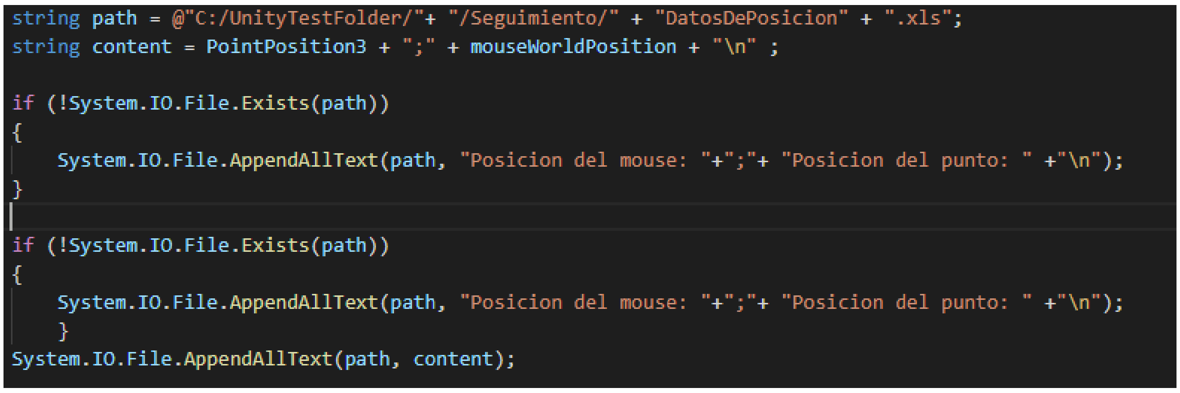


```
void Update()
{
    //output to log the position change
    if ( Input.GetMouseButtonDown (0))
    {

        Vector3 mouseWorldPosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
        Vector3 PointPosition3 = transform.position;
```

Figure 22. Code for obtaining position vectors.

As we can see in the previous image, a conditional is performed within the Update where, when detecting that the screen has been touched by means of the command GetMouseButtonDown, two vectors are created, one for the real position of the circle called Pointposition3 and another one for the position of the circle indicated by the patient called mouseWorldPosition. Once these vectors have been created, it is verified that they are really giving the data of interest by printing the data in the console by means of the DebugLog command.

Once the data has been checked, we proceed to the extraction of these data by means of the code that can be seen in Figure 23, this code is also added to the Update just after the creation of the vectors, this code allows the creation of directories and files in ".xls" format. By means of this format it is possible to create graphs and analyze the utility of the application and at the same time the patient's performance.

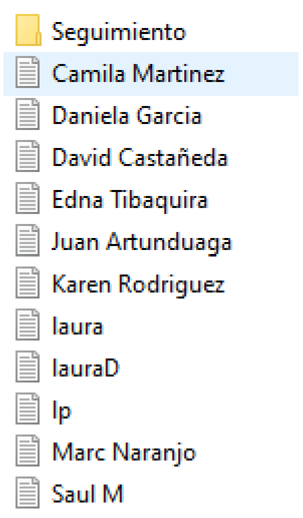


```
string path = @"C:/UnityTestFolder/"+ "/Seguimiento/" + "DatosDePosicion" + ".xls";
string content = PointPosition3 + ";" + mouseWorldPosition + "\n" ;

if (!System.IO.File.Exists(path))
{
    System.IO.File.AppendAllText(path, "Posicion del mouse: "+";"+ "Posicion del punto: " +"\n");
}

if (!System.IO.File.Exists(path))
{
    System.IO.File.AppendAllText(path, "Posicion del mouse: "+";"+ "Posicion del punto: " +"\n");
    }
System.IO.File.AppendAllText(path, content);
```

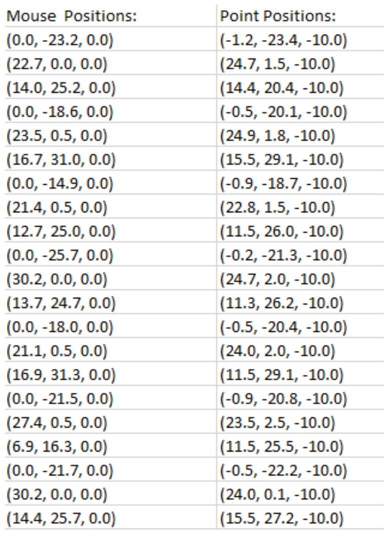
Figure 23. Code to extract data of interest from Excel files.

When analyzing the previous code it can be seen that, right at the beginning, the directory is created; where the position data of all the patients that use the application will be stored, these data are stored separated by semicolons. After indicating the directory a conditional is created that verifies that in that directory exists, if so it adds the position data and if it does not exist it creates it to later add the position data that were previously explained.

As can be seen the data is saved in the same path where the data of the registered patients is also saved, but in a separate folder as shown in Figure 24. [Left]. Also in Figure 24 [Right]. we can see how the data is saved in the Excel file which as already mentioned will be separated by comma, however later on we will explain how to convert this data to a table.



Seguimiento
Camila Martinez
Daniela Garcia
David Castañeda
Edna Tibaquira
Juan Artunduaga
Karen Rodriguez
laura
lauraD
lp
Marc Naranjo
Saul M



Mouse Positions:	Point Positions:
(0.0, -23.2, 0.0)	(-1.2, -23.4, -10.0)
(22.7, 0.0, 0.0)	(24.7, 1.5, -10.0)
(14.0, 25.2, 0.0)	(14.4, 20.4, -10.0)
(0.0, -18.6, 0.0)	(-0.5, -20.1, -10.0)
(23.5, 0.5, 0.0)	(24.9, 1.8, -10.0)
(16.7, 31.0, 0.0)	(15.5, 29.1, -10.0)
(0.0, -14.9, 0.0)	(-0.9, -18.7, -10.0)
(21.4, 0.5, 0.0)	(22.8, 1.5, -10.0)
(12.7, 25.0, 0.0)	(11.5, 26.0, -10.0)
(0.0, -25.7, 0.0)	(-0.2, -21.3, -10.0)
(30.2, 0.0, 0.0)	(24.7, 2.0, -10.0)
(13.7, 24.7, 0.0)	(11.3, 26.2, -10.0)
(0.0, -18.0, 0.0)	(-0.5, -20.4, -10.0)
(21.1, 0.5, 0.0)	(24.0, 2.0, -10.0)
(16.9, 31.3, 0.0)	(11.5, 29.1, -10.0)
(0.0, -21.5, 0.0)	(-0.9, -20.8, -10.0)
(27.4, 0.5, 0.0)	(23.5, 2.5, -10.0)
(6.9, 16.3, 0.0)	(11.5, 25.5, -10.0)
(0.0, -21.7, 0.0)	(-0.5, -22.2, -10.0)
(30.2, 0.0, 0.0)	(24.0, 0.1, -10.0)
(14.4, 25.7, 0.0)	(15.5, 27.2, -10.0)

Figure 24. [Left]. Tracking folder where data of interest are saved. Figure 24.[Right]. Data of interest obtained through Unity.

7. Results

In order to obtain the results, 10 healthy people with an age range between 24 and 35 years old were tested. These tests consisted of making a trajectory prediction by means of a tablet application.

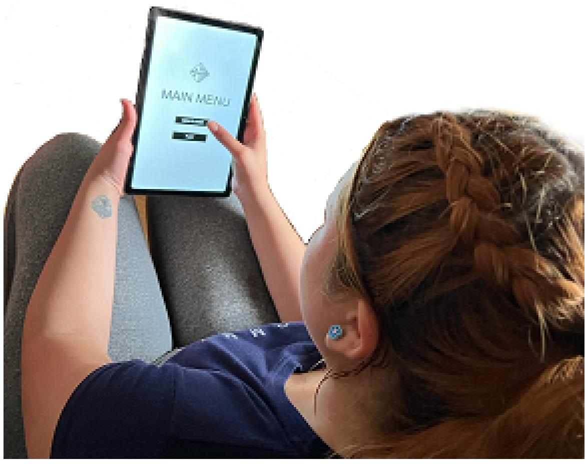
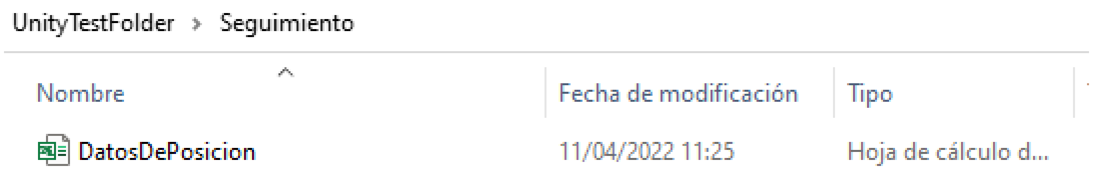


Figure 25. Photo of user testing the application.

As discussed throughout the project, 3 trajectory prediction tasks were designed with a different predetermined difficulty; all tasks were tested to evaluate their performance in healthy individuals.

Based on the fact that when testing the application the program automatically creates a document where the results of each user will be saved as explained in section 6.3, this is why we obtained an Excel file with 90 position data. As shown in Figure 26, we can see that the files are saved in the folder called tracking with the name *DatosDePosicion*.



UnityTestFolder › Seguimiento

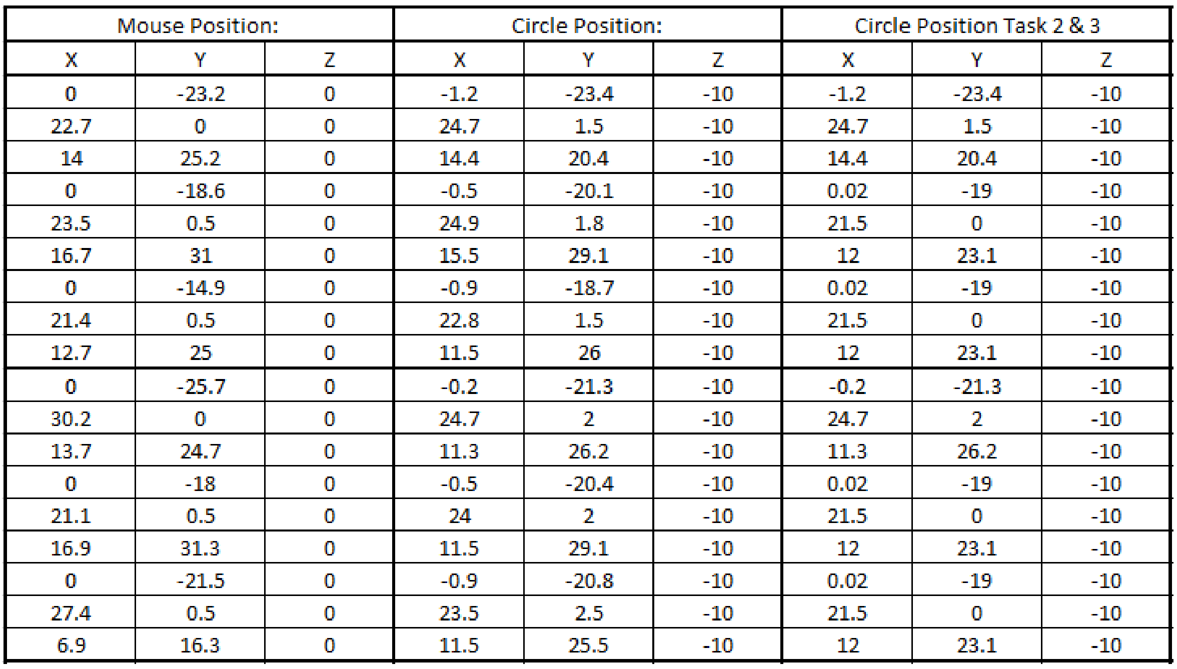
Nombre	Fecha de modificación	Tipo
DatosDePosicion	11/04/2022 11:25	Hoja de cálculo d...

Figure 26. User results tracking folder.

When analyzing the obtained results, we observe that not in all the graphs we find an ideal behavior, understanding as ideal that the distance between the real position of the circle and the trajectory prediction position is zero. However, it is possible to show that even though this distance is not strictly zero, it is very close to zero.

When opening the *DatosdePosicion* document, we can see that the data are stored separated by semicolons, however, to facilitate the analysis by means of the Excel Data tool, the text in columns option is used, thus achieving a better visualization of the data as we can see in Figure 27.

As can be seen in Figure 27, for the sake of practicality when explaining the analysis, the data from two users will be used, however, all the data analyzed are found in the Appendix I.



Mouse Position:			Circle Position:			Circle Position Task 2 & 3		
X	Y	Z	X	Y	Z	X	Y	Z
0	-23.2	0	-1.2	-23.4	-10	-1.2	-23.4	-10
22.7	0	0	24.7	1.5	-10	24.7	1.5	-10
14	25.2	0	14.4	20.4	-10	14.4	20.4	-10
0	-18.6	0	-0.5	-20.1	-10	0.02	-19	-10
23.5	0.5	0	24.9	1.8	-10	21.5	0	-10
16.7	31	0	15.5	29.1	-10	12	23.1	-10
0	-14.9	0	-0.9	-18.7	-10	0.02	-19	-10
21.4	0.5	0	22.8	1.5	-10	21.5	0	-10
12.7	25	0	11.5	26	-10	12	23.1	-10
0	-25.7	0	-0.2	-21.3	-10	-0.2	-21.3	-10
30.2	0	0	24.7	2	-10	24.7	2	-10
13.7	24.7	0	11.3	26.2	-10	11.3	26.2	-10
0	-18	0	-0.5	-20.4	-10	0.02	-19	-10
21.1	0.5	0	24	2	-10	21.5	0	-10
16.9	31.3	0	11.5	29.1	-10	12	23.1	-10
0	-21.5	0	-0.9	-20.8	-10	0.02	-19	-10
27.4	0.5	0	23.5	2.5	-10	21.5	0	-10
6.9	16.3	0	11.5	25.5	-10	12	23.1	-10

Figure 27. Table of data obtained from the Excel file extracted from Unity.

As can be seen in the previous figure the position of the circle is obtained in the X, Y, Z axes, but as the application is designed in 2D, then only the position values in the X and Y axis will be useful.

In addition to the tables, the position vector graphs were found for each user, in order to see in a more visual way the results obtained, in each graph a comparison is made between the actual position vector of the circle and the trajectory prediction vectors made by the user. For practicality reasons and taking into account that most of the results obtained by the users are very similar, the example of only two users is shown, however, all the graphs are in the Appendix I.

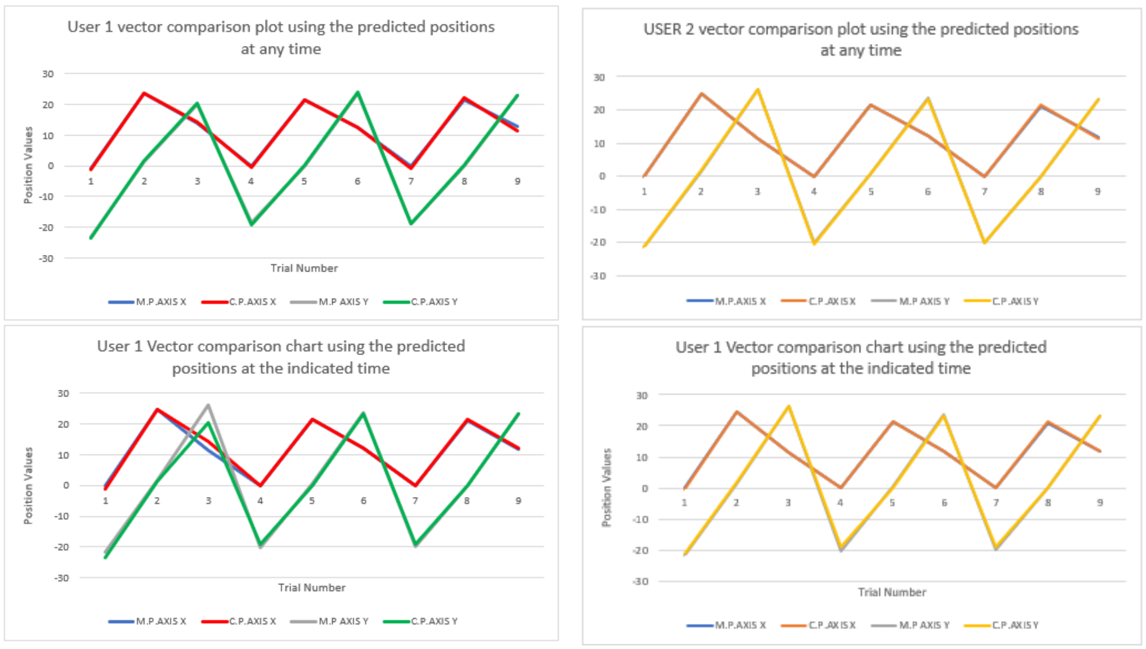
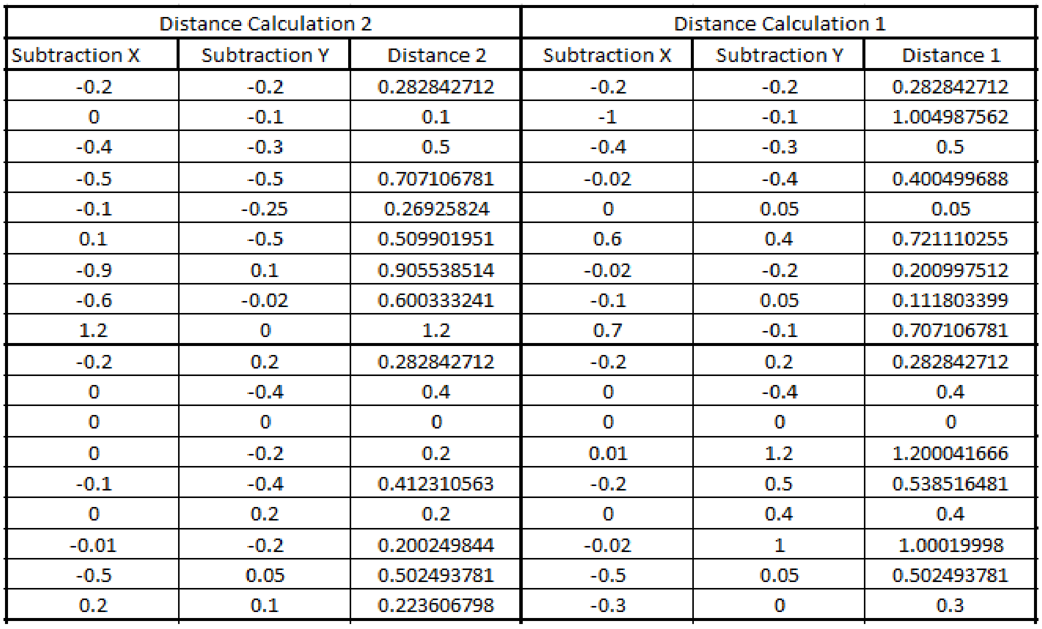


Figure 28. Position plots calculated with the data shown in Figure 27.

It is also important to note that the analysis for the second and third task was performed in two ways, the first evaluates that the person can predict the trajectory at the indicated time, for this part of the evaluation the exact position of the circle equivalent to the time that the user is instructed to perform the trajectory prediction is already known and the second is to perform the analysis as in task one which evaluates that the person can predict the trajectory at any time, This additional analysis is done to know if the person, despite not predicting the trajectory at the indicated time, can predict the trajectory at any time, stating that he/she would need more time to process the visual process. These data can be observed in Figure 27.

For the analysis of the data, it was necessary to find the distance between the vectors to compare how close or far they are from the ideal distance. On the other hand, to calculate the distance between the two vectors, first of all, a subtraction is performed between them, to later perform the square root of the sum of the squares, for this an Excel table was created, which can be seen in Figure 29.



Distance Calculation 2			Distance Calculation 1		
Subtraction X	Subtraction Y	Distance 2	Subtraction X	Subtraction Y	Distance 1
-0.2	-0.2	0.282842712	-0.2	-0.2	0.282842712
0	-0.1	0.1	-1	-0.1	1.004987562
-0.4	-0.3	0.5	-0.4	-0.3	0.5
-0.5	-0.5	0.707106781	-0.02	-0.4	0.400499688
-0.1	-0.25	0.26925824	0	0.05	0.05
0.1	-0.5	0.509901951	0.6	0.4	0.721110255
-0.9	0.1	0.905538514	-0.02	-0.2	0.200997512
-0.6	-0.02	0.600333241	-0.1	0.05	0.111803399
1.2	0	1.2	0.7	-0.1	0.707106781
-0.2	0.2	0.282842712	-0.2	0.2	0.282842712
0	-0.4	0.4	0	-0.4	0.4
0	0	0	0	0	0
0	-0.2	0.2	0.01	1.2	1.200041666
-0.1	-0.4	0.412310563	-0.2	0.5	0.538516481
0	0.2	0.2	0	0.4	0.4
-0.01	-0.2	0.200249844	-0.02	1	1.00019998
-0.5	0.05	0.502493781	-0.5	0.05	0.502493781
0.2	0.1	0.223606798	-0.3	0	0.3

Figure 29. Distance data table where Distance 2 corresponds to the distance of the prediction made at any time, while Distance 1 corresponds to the prediction made at the indicated time.

As can be seen in the table in Figure 29, we can observe the Distance 2, which indicates the distance between the vectors Mouse position and Circle position, where we can see if the user identified the correct trajectory at any time of the task, we also observe the Distance 1, which indicates the distance between the vectors Mouse Position and Circle Position Task 2 & 3, where we can see if the user could really predict the trajectory at the right time.

Based on the above, the results obtained were as follows:

First, with the distance values found, the average and standard deviation were calculated, and the following results were obtained:

- The average of the calculation of distance 1 is 0.47
- The average of the calculation of distance 2 is 0.41
- The standard deviation of the calculation of distance 1 is 0.42
- The standard deviation of the calculation of distance 2 is 0.41

Because we want to analyze how far away the position vectors of the trajectory prediction are from the actual position data of the circle, two scatter plots are made using the distances already found previously.

The following is a comparison of the scatter plots with the distance values found. To facilitate the analysis, the two graphs are on the same scale and the trend line is shown in red. As can be seen, the trend line for the Distance 1 graph is approximately 0.4 while the trend line for Distance 2 is approximately 0.3.

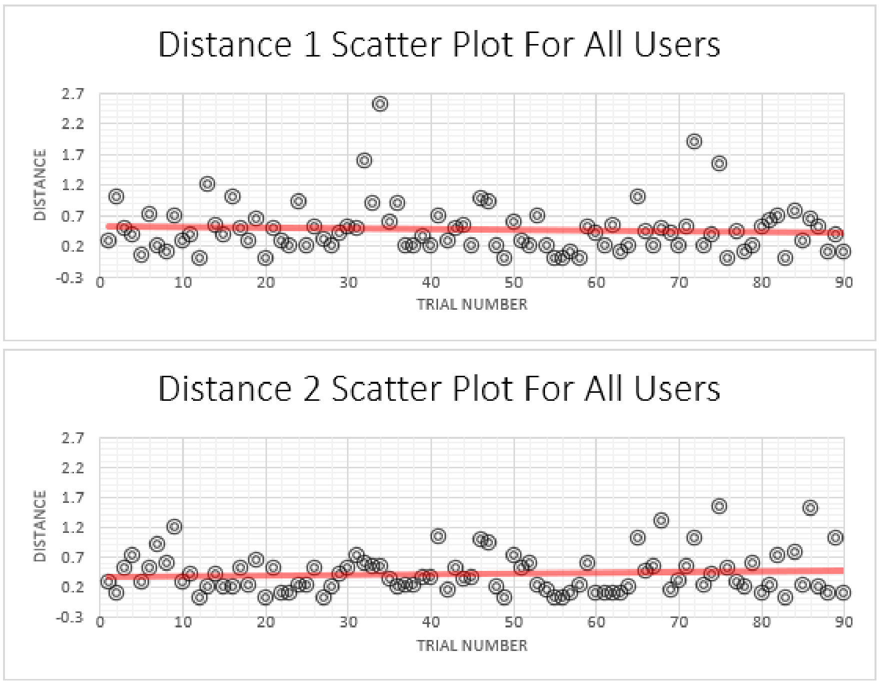


Figure 30. Scatterplot of distances for all users

On the other hand, these scatter plots were also made, but separated by tasks, in order to show if any task is more difficult for users to perform.

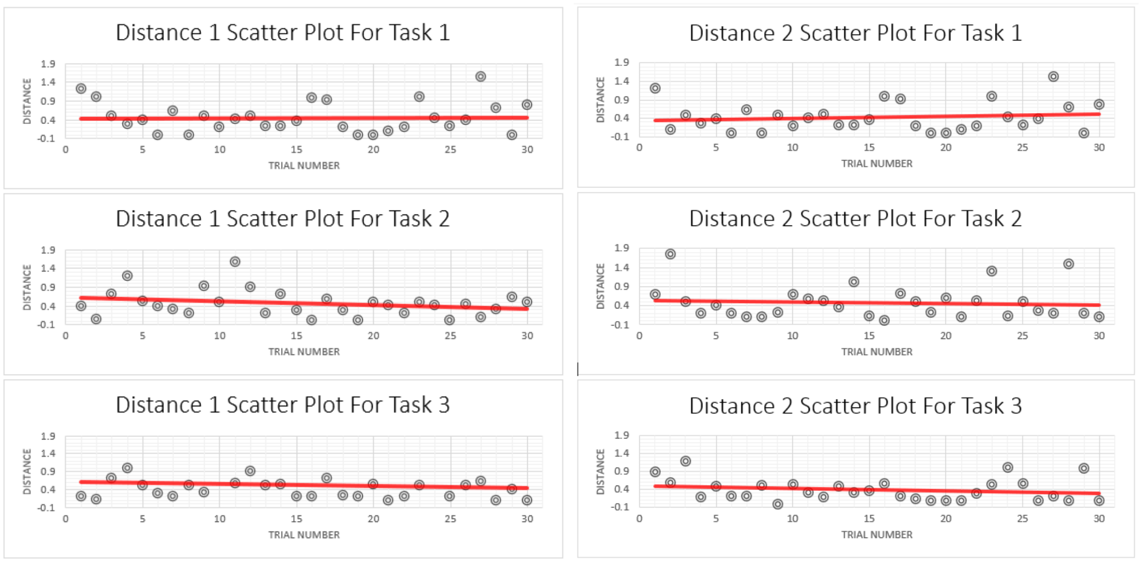


Figure 31. Scatterplot of distances for all users separated by tasks.

7.1. Restructuring of tasks

As explained in section 5.1 of the methodology, meetings were held to analyze the functioning of the tasks and to make the changes that were deemed appropriate based on the desired final result.

Once the previous tests of this version of the game had been carried out on people who did not have the disease, changes in the application were proposed, and these changes will be described below:

1. Determined that there was no need to repeat the instructions before each example scene and before each trajectory prediction scene, since it generated a distraction and its realization became longer and more tedious, that is why general instructions were implemented for the example and prediction scenes.
2. Complementing the previous point, it is decided to show all the example scenes one after the other so that the patient has time to process the given instruction, and the corresponding example scene is also shown before each trajectory prediction scene.
3. At first the idea was thought to be an Android application to perform the tests through a Tablet, but we had a great difficulty when saving the data on the Tablet, in addition to this we realized that when testing on the Tablet the screen space for both the examples and the trajectory prediction was very small so it was decided to move to an application for touch computer.
4. We came to the conclusion that 3 trials for each task was too few, that's why now for each task there are 8 trials coming out of different positions each.
5. We discuss the importance of the patient having the opportunity to perform the prediction task more than once, which is why each prediction scene is performed twice instead of once.
6. We were interested in analyzing a nonlinear trajectory, for this reason we implemented one more task, this time the trajectory is parabolic (see Figure 32). In the figure we can see 4 of the 8 trials of the task.

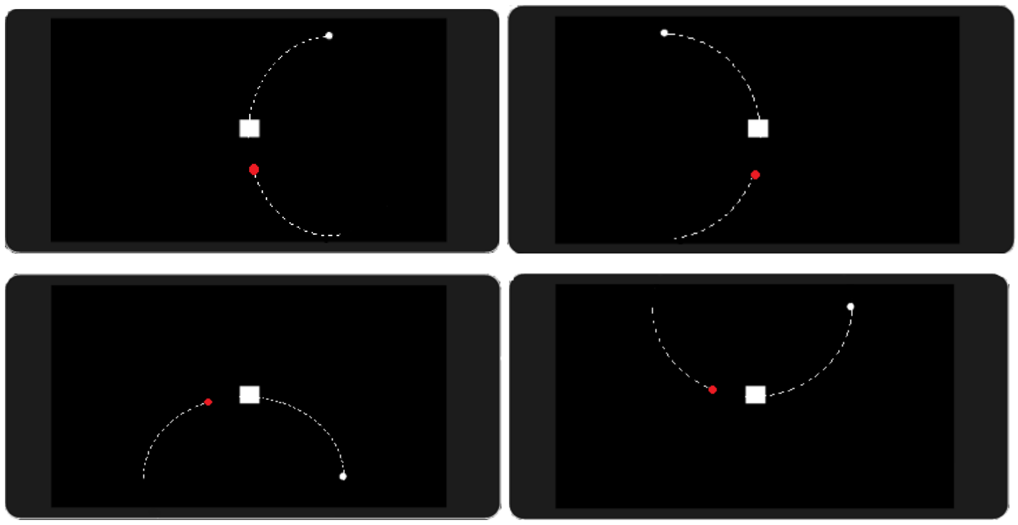
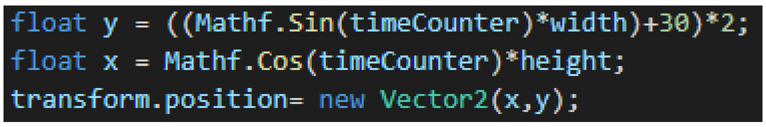


Figure 32. Example scenes of the different tests of task 4.

To perform the parabolic trajectories shown in the previous figure, two lines of code are added to the update of the script that controls the movement of the circle, these are shown in Figure 33, however, the complete code can be found in the Appendix IV.



```
float y = ((Mathf.Sin(timeCounter)*width)+30)*2;
float x = Mathf.Cos(timeCounter)*height;
transform.position= new Vector2(x,y);
```

Figure 33. Code for parabolic trajectory.

To summarize the latest version of the application is as follows: The graphical interface remains the same, when entering the application the user registers for subsequent entry, once the user enters the game the information comes out where the example scenes of the corresponding task are explained, after this the 8 example scenes are shown, followed by the information of the trajectory prediction scenes and after this the example scene corresponding to the trajectory prediction scene to perform is shown, this is preceded by the two trajectory prediction scenes, this for each of the trials and the same is repeated for each of the four tasks.

8. Data analysis

When analyzing the data obtained from the control patients, Figure 27 shows that the predicted position values of the users are very close to the real values of the circle both at the time of the tap and at the time indicated by the counter. This result was expected since when correctly predicting the trajectory the 3 position values of the vectors will be the same or very similar.

Firstly, if we observe Figure 28 with the vector position graphs, we see that for the control population it was easy to predict the proposed trajectories, this is corroborated at the moment of finding the distances between these vectors shown in Figure 29, giving distances very close to zero, which would be the ideal distance.

However, when comparing both the averages obtained and the standard deviation between Distance 1 (actual position values of the circle Vs predicted values at the time indicated by the counter) and Distance 2 (actual position values of the circle Vs predicted values of the trajectory at any time), we see that Distance 1 is very slightly greater, moving away a little from the ideal distance, this tells us that in a way it is more difficult to predict the trajectory taking into account the extra factor of time, however in these specific cases with control patients the difference has not been much noticed.

Continuing with the above, if we analyze the total dispersion graphs of the users we see that some distances are a little far from the ideal distance, this could be due to some factors such as those explained in section 7.1.1 of task restructuring, such as the space on the screen to predict the trajectory and the fact that only one opportunity was given to make the prediction. This also shows that it is necessary to establish a margin of error when performing the prediction tasks since a tap on the screen is a little imprecise and the trajectory prediction may be correct but the thickness of the finger or the relativity of the position of the center of the circle will alter the results.

On the other hand, as already mentioned, it was decided to analyze the scatter plots separating them by tasks in order to find if any of the tasks are more difficult and when buying the data we see that task 2 is the one that presents a greater dispersion of data, which once again corroborates what was previously stated and is that the addition of an extra factor such as time makes predicting the trajectory accurately more difficult, This is also why in the distance graph 2 we can see that the values are a little closer to zero, also affirming the fact that predicting the value a little later than the indicated time becomes easier, however we do not rule out the idea that it is due to an error not because of poor trajectory prediction but as stated above not having defined a margin of error allowed since the prediction may be correct, but the tap was not made exactly in the center of the circle.

After analyzing all the results obtained from all the users who tested the application, we can say that the application provides precise and accurate results since all the values obtained show a low dispersion and the values obtained are very close to the expected values.

Finally, it is important to clarify that this analysis will be very useful when testing people with the disease; however, due to factors such as the current situation with COVID19 and the availability schedule of patients, it was difficult to test patients. In addition, we sought to postpone the performance of these tests until we had a definitive version of the application, more than anything else to avoid wasting time and to avoid wasting patients' time, since when they go to the hospital they have to undergo many tests. On the other hand, to present a definitive version of the application at this moment is very complicated because being a new project and without many references it is exposed to constant changes and modifications, however it is a good start to determine a precedent about this topic, and in such a way that this project can be the basis of information for future research.

Conclusions

In short, great progress was made with the development of the application, taking into account that it is starting from scratch. As can be seen throughout the work, there is still a long way to go, but thanks to the support of Dr. Saul Martines of the Hospital Sant Pau, we were able to propose and develop a solution to a problem that had not yet been addressed. Currently the application has already been approved by the Doctor to perform the first tests on HD patients. Other findings are:

- Functional software was developed for both Android and Windows operating system devices.
- The first tests of the application focused on cognitive analysis of Huntington's disease were implemented.
- Four tasks were developed with different difficulty levels focused on linear and nonlinear trajectory prediction.
- Currently, through the designed application it is possible to extract data of interest by means of Excel files for further analysis.
- With the data extracted from the tests performed on healthy people it has been possible to evaluate the functionality of the application.
- From the results obtained it can be concluded that the application provides precise and accurate results.
- The development of this application can be of great help in the future to help us understand the behavior of the brain of people with Huntington's disease.
- The project contributes to the development and evolution of health sciences.

To conclude, the development of this work has been an enriching experience at an academic and professional level, since I have learned new things related to an area of my career that interests me and in addition to this, I have been able to consolidate previously acquired knowledge. On the other hand, I will continue working with the Doctor in the development of the application, performing tests on HD patients and the subsequent publication of a scientific article.

Future lines

As described above, this project is still in its "beta" phase and there are many possible ways to improve this application:

- To be able to test the software by linking it to an EEG study, where in addition to seeing through the application that there is a failure at the level of trajectory prediction, to be able to see what implication this has at the level of the electrical signals produced by the brain.
- To develop more tasks where more complex trajectories are traced.
- To be able to combine this application with an ocular reader such as Tobi, since, as stated in some articles, patients tend to present slow saccadic movements[].
- Improve the developed tares by adding trajectories with changes in speed during the same task.
- Add feedback to users, in real time by creating scenes where trajectory prediction can be tested with feedback before performing the actual prediction.
- Develop a robust database where patient information such as age, symptoms, medications, etc. can be stored.
- Change the server (currently "offline") to an "online" one. With this change it would be possible to make an application that can be used by the health professional on any device.
- Finally, test the study on Huntington's patients.

Environmental impact

In order to analyze the environmental impact of this project, different factors must be taken into account, such as the life cycle of the elements that make it up.

On the one hand, for the computer hardware, between 5040 and 5600 million Joules of energy are required for the production process and 115 watts of power for daily use consumption in addition to an emission of between 52 and 234 grams of CO2 [17, 18]. Although these numbers are high compared to other devices or products, the devices used for the project have a long lifetime and can exceed 8-10 years of operation. The above mentioned makes the highest environmental impact of the study to be the energetic cost of its operation. It can emit between 880.6 and 1,872 kg of CO2 during its useful life [19, 20]. As mentioned above, the highest environmental impact of the study is the energy cost of its operation. It can emit between 880.6 and 1,872 kg of CO2 over its lifetime [19,20]. Another impact that electronics have on the environment is the discarded waste. The electronic devices used in the project have heavy metals such as mercury, lead, nickel and many more [21].

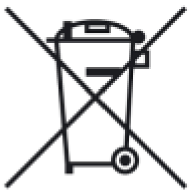


Figure 34. Symbol indicating the selective collection of electrical/electronic equipment

So that the disposal of these materials and electronic devices is controlled and does not end up polluting the subsoil or nearby waters such as rivers, seas or lakes, the European Union obliged the management and collection of electrical and electronic equipment. The new and revised European Directive 2012/19/EU on waste electrical and electronic equipment (transposed in Spain through Royal Decree 110/2015) obliges the management and collection of appliances, electrical and electronic, highlighting household appliances and consumer electronics once they cease to be used and become waste [21]. These products had to be marked with a symbol indicating a crossed-out bin and the need to create a waste collection and recycling system [21]. The symbol can be seen in Figure 34. All hardware products used for the development of the project have the symbol in Figure 34 and thus comply with the collection and recycling regulations set by the European Union (EU).

Acknowledgments

During the development of this Master thesis, I met people who were helping and supporting me to make this project a reality. That is why I would like to thank the ETSEIB University for giving me the opportunity to develop this project, in addition to Dr. Saul Martinez and the Hospital Sant Pau for their dedication, time and collaboration. On the other hand to my thesis director Alejandro Bachiller and co-directors Sergio Romero and Joan Frances Alonso, for their accompaniment during this process and for giving me the necessary tools to overcome the obstacles presented during the realization of the project. In addition, I would like to thank all the professors and students of this institution who directly or indirectly participated in my process, since their contribution, big or small, is reflected in the culmination of this project.

I would also like to thank my parents Gloria Amparo Almeciga and Alfonso Pedreros and my brothers Andrés Felipe Pedreros Almeciga and Sebastián Pedreros Almeciga for their unconditional support, for trusting in me, in my aptitudes, thanks to them for teaching me to be constant and to work to achieve my dreams, for each of their words and advice that have helped me to be who I am at this moment.

Finally, I want to thank Edna Tibaquira and Juan Diego Tibaquira for their unconditional support, for always being there encouraging me to move forward when difficulties arose in the development of this work, also thanks for the help and contributions not only for the development of the thesis, but also for my life.

Economic Analysis

In this section the economic analysis for the development of the project will be presented; the cost of personnel will be included in the budget, as well as the cost of the material used.

Personal. In the following tables the costs of personnel will be taken into account, these are subjective values considering that in the development of the project tutoring meetings, research, methodology, among others, were carried out.

For this section it is taken into account that the final degree project has 18 ECTS and 25 hours of work are done for each ECTS, that is why 450h/engineer were dedicated but an additional extension was requested for the completion of the project working another 270h/engineer.

This project started on September 18 and ended on May 29, working approximately 7.5 hours per day working on it 3 times per week.

Personal	Time [h]	Cost per hour €	Total Cost [€/h]
1	720	30	21.600

Table 2. Personnel Costs.

Materiales. The following table shows the cost breakdown of the materials used:

Hardware Costs: For the calculation related to Hardware costs, the useful life of each material used was taken into account. The formula used to find the actual cost was as follows:

$$\frac{Months\ of\ Use}{Lifetime} * 100 = Depreciation * Cost$$

MATERIAL	COST [€/ud]	QUANTITY [ud]	LIFETIME [Month]	DEPRECIATION [%]	REAL COST [€]
Lenovo Computer	600.00	1	48	16.67	10000.00
Tablet Samsung Tap A7	200.00	1	40	20	4000.00
Total costs of Hardware	-	-	-	-	14000.00

Table 3. Hardware Costs.

Software Costs: The cost of the software was based on the Unity course taken through the Udemy platform.

MATERIAL	COST [€/ud]	QUANTITY [ud]	TOTAL COST [€]
Unity Couse	9.00	1	9
Unity version 2020	0.00	1	0
Microsoft Visual Studio Code	0	1	0
Total Cost			9

Table 4. Software Costs.

- **Total Costs**

	COST [€]
Personal Cost	21.60
Hardware Cost	14000.00
Software Cost	9
Total Cost	14030.60

Table 5. Total Costs.

Bibliography

Bibliographic references

[1] F. O Walker, "Huntington’s disease", *Seminar*, vol. 369, n.º 218-28, p. 11, febrero de 2007.

[2] P. C. Nopoulos *et al.*, "Cerebral cortex structure in prodromal Huntington disease", *Elsevier*, vol. 40, n.º 544-554, p. 11, 2010.

[3] Rosales Reynoso MA, Barros Núñez P. Biología molecular y Medicina. Diagnóstico molecular de la enfermedad de Huntington. *Gac Méd Méx*. 2008; 144(3):271-3.

[4] Velázquez Pérez L, Rodríguez Labrada R. General characteristics of polyglutamine diseases. In: Early manifestations of Spinocerebellar Ataxia type 2. Holguín: Ediciones Holguín; 2012.p.11-32.

[5] Azzarelli A. Huntington's disease and degenerations of some subcortical nuclei. In: Neuropathology, diagnosis and clinic. Barcelona: Editorial *Edisma*; 2000.p. 603-20.

[6] Potter NT, Spector EB, Prior TW. Technical standards and guidelines for Huntington disease testing. Genet Med. 2004; 6:61-65.

[7] Ropper AH, Samuels A. Degenerative Diseases of the Nervous System En: Adams & Victors' Principles of Neurology. 9 ed. Philadelphia: Ed. McGraw-Hill; 2009.p.895-954.

[8] David W Weir, Aaron Sturrock y Blair R Leavit, "Development of biomarkers for Huntington’s disease", *Lancet Neurol*, vol. 10, n.º 573-90, p. 18, junio de 2011.

[9] Craufurd D, Snowden J. Neuropsychological and neuropsychiatric aspects of Huntington's disease. In: Bates G, Harper P, Jones L, eds. Huntington's disease. Nueva York: Oxford University Press, 2002: 62–94.

[10] Kaytor MD, Wilkinson KD, Warren ST. Modulation of huntingtin half-life alters polyglutamine-dependent aggregate formation and cellular latoxicity.J neurochemistry2004;89:961-73.

[11] D. L. Harrington, 2. M. M. Smith, Y. Zhang, N. E. Carlozzi, J. S. Paulsen y the PREDICT-HD Investigators of the Huntington Study Group, "Cognitive domains that predict time to diagnosis in prodromal Huntington disease", *Dr J S Paulsen, University of Iowa Carver College of Medicine, Psychiatry Research, 1-305 Medical Education Building*, vol. 83, n.º 612-619, marzo de 2012. Accedido el 15 de octubre de 2021. [En línea]. Disponible: http://jnnp.bmj.com

[12] "Visuomotor integration deficits precede clinical onset in Huntington’s disease", *Neuropsychologia*, vol. 49, n.º 264-270, p. 7, noviembre de 2010. Accedido el 20 de octubre de 2021. [En línea]. Disponible: http://www.elsevier.com/locate/neuropsychologia

[13] S. J. Tabrizi *et al.*, "Biological and clinical manifestations of Huntington’s disease in the longitudinal TRACK-HD study: Cross-sectional analysis of baseline data", julio de 2009, art. n.º 4422(09)70170-X.

[14] C. A. Ross *et al.*, "Huntington disease: Natural history, biomarkers and prospects for

therapeutics", *Nature Reviws, Neurology*, p. 13, marzo de 2014C. A. Ross *et al.*, "Huntington disease: Natural history, biomarkers and prospects for therapeutics", *Nature Reviws, Neurology*, p. 13, marzo de 2014.

[15] "Unity - manual: System requirements for unity 2021 LTS". Unity - Manual: Unity User Manual 2021.3 (LTS). https://docs.unity3d.com/Manual/system-requirements.html (accedido el 7 de enero de 2022).

[16] "Unity - manual: System requirements for unity 2021 LTS". Unity - Manual: Unity User Manual 2021.3 (LTS). https://docs.unity3d.com/Manual/system-requirements.html (accedido el 7 de enero de 2022).

[17] GREFA - Inicio. https://www.grefa.org/grefa/containfor.pdf (accedido el 1 de abril de 2022).

[18] "¿Cuánta energía consume tu PC? Así puedes medirlo de manera fiable". HardZone. https://hardzone.es/tutoriales/mantenimiento/medir-consumo-pc/ (accedido el 2 de abril de 2022).

[19] "Los ordenadores también emiten CO2 - Ecoembes | The Circular Lab". Ecoembes | The Circular Lab. https://www.thecircularlab.com/los-ordenadores-tambien-emiten-co2/ (accedido el 1 de abril de 2022).

[20] "RAEE II: Revisión de la Directiva de residuos de aparatos eléctricos y electrónicos". Bienvenido al Blog de Schneider Electric. https://blogespanol.se.com/normativa-energetica/2018/01/15/revision-de-la-directiva-de-residuos-de-aparatos-electricos-y-electronicos/ (accedido el 1 de abril de 2022).

Appendix I: Data obtained by means of the Unity platform

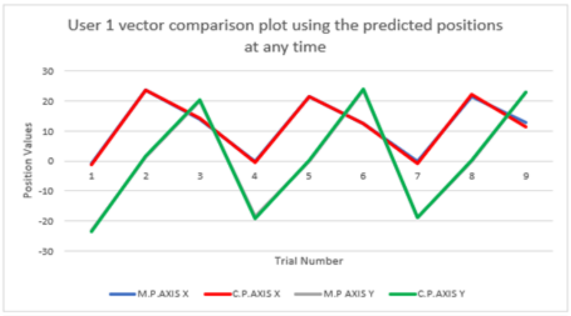
The following tables show the data obtained from the 10 control users. As there are 3 tasks each with 3 tests, there are a total of 90 position data in the trajectory prediction tasks.

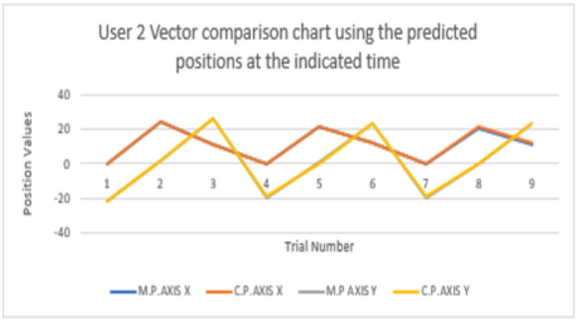
Mouse Position:			Circle Position:			Circle Position Task 2 & 3		
X	Y	Z	X	Y	Z	X	Y	Z
-1	-23.2	0	-1.2	-23.4	-10	-1.2	-23.4	-10
23.7	1.4	0	23.7	1.5	-10	24.7	1.5	-10
14	20.1	0	14.4	20.4	-10	14.4	20.4	-10
0	-18.6	0	-0.5	-19.1	-10	0.02	-19	-10
21.5	0.05	0	21.6	0.3	-10	21.5	0	-10
12.6	23.5	0	12.5	24	-10	12	23.1	-10
0	-18.8	0	-0.9	-18.7	-10	0.02	-19	-10
21.4	0.05	0	22	0.07	-10	21.5	0	-10
12.7	23	0	11.5	23	-10	12	23.1	-10
0	-21.5	0	-0.2	-21.3	-10	-0.2	-21.3	-10
24.7	1.6	0	24.7	2	-10	24.7	2	-10
11.3	26.2	0	11.3	26.2	-10	11.3	26.2	-10
-0.03	-20.2	0	-0.03	-20.4	-10	0.02	-19	-10
21.3	0.5	0	21.4	0.9	-10	21.5	0	-10
12	23.5	0	12	23.3	-10	12	23.1	-10
0	-20	0	-0.01	-20.2	-10	0.02	-19	-10
21	0.05	0	21.5	0	-10	21.5	0	-10
11.7	23.1	0	11.5	23	-10	12	23.1	-10
0	-21.8	0	-0.5	-22.2	-10	-0.5	-22.2	-10
24	0.1	0	24	0.1	-10	24	0.1	-10
15.2	26.8	0	15.5	27.2	-10	15.5	27.2	-10
0	-19.3	0	-0.02	-19.4	-10	0.02	-19	-10
21.3	0.05	0	21.2	0.05	-10	21.5	0	-10
12.2	24	0	12.3	24.2	-10	12	23.1	-10
0	-19.2	0	-0.07	-19	-10	0.02	-19	-10
21.4	0.5	0	21.5	1	-10	21.5	0	-10
12.3	23	0	12.3	23	-10	12	23.1	-10

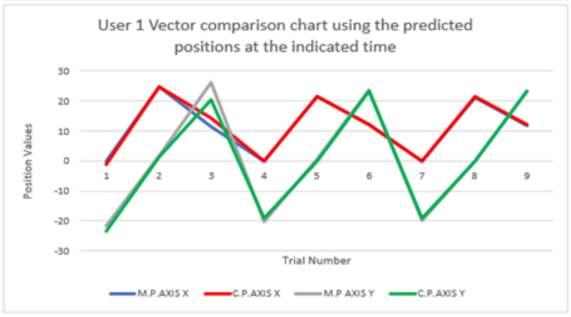
ETSEIB

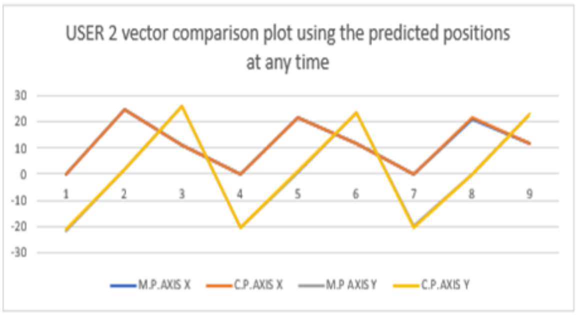
1.4	-25.1	0	1.4	-25.1	-10	1.4	-25.1	-10
23.4	0.1	0	23.4	0.1	-10	23.4	0.1	-10
13.5	30.7	0	13.4	30.7	-10	13.4	30.7	-10
0	-19	0	0.2	-19.1	-10	0.02	-19	-10
21.4	0.5	0	21.4	1.1	-10	21.5	0	-10
12.1	23.5	0	12	23.5	-10	12	23.1	-10
0	-18.8	0	0	-18.9	-10	0.02	-19	-10
21.3	0.5	0	21.2	0.5	-10	21.5	0	-10
12	23	0	12	23.1	-10	12	23.1	-10
-1.2	-23.2	0	-1.2	-23.4	-10	-1.2	-23.4	-10
23.7	1.5	0	24.7	1.5	-10	24.7	1.5	-10
14	20.2	0	14.4	20.4	-10	14.4	20.4	-10
0	-18.8	0	-0.5	-19	-10	0.02	-19	-10
21.5	0.5	0	21.5	1.8	-10	21.5	0	-10
12.4	23	0	12.5	23.1	-10	12	23.1	-10
0	-18.8	0	-0.3	-18.8	-10	0.02	-19	-10
21.4	0.5	0	21.6	1	-10	21.5	0	-10
12.2	25	0	12	26	-10	12	23.1	-10
0	-21.2	0	-0.2	-21.3	-10	-0.2	-21.3	-10
24.3	2	0	24.7	2	-10	24.7	2	-10
11	24.7	0	11.3	26.2	-10	11.3	26.2	-10
0	-19	0	-0.5	-19	-10	0.02	-19	-10
21.1	0.2	0	21.3	0	-10	21.5	0	-10
12	23	0	12	23.2	-10	12	23.1	-10
0	-19.2	0	-0.5	-19.5	-10	0.02	-19	-10
21.4	0.5	0	21.5	0.5	-10	21.5	0	-10
12.6	23.3	0	12.5	23.5	-10	12	23.1	-10
0	-22.7	0	-0.5	-22.2	-10	-0.5	-22.2	-10
24	0.1	0	24	0.1	-10	24	0.1	-10
14.9	27.7	0	15.5	27.2	-10	15.5	27.2	-10
0	-19.3	0	0.2	-19.4	-10	0.02	-19	-10
21.1	0.5	0	21.2	2	-10	21.5	0	-10
12.5	23	0	12.7	23	-10	12	23.1	-10
0	-18.9	0	-0.02	-18.8	-10	0.02	-19	-10
21.1	0.05	0	22.1	0.02	-10	21.5	0	-10
12	23.2	0	12	23.3	-10	12	23.1	-10

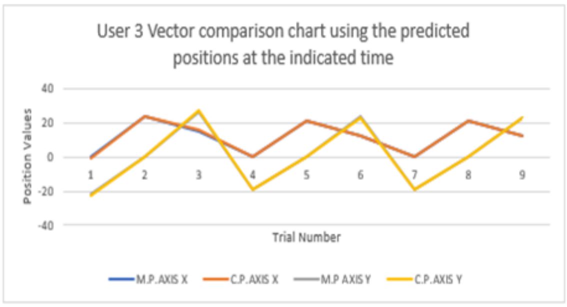
To better observe the results obtained, plots were made of the position vectors found. Two graphs were made for each user comparing their prediction with the real position of the circle at the time of the tap. These graphs are shown below:

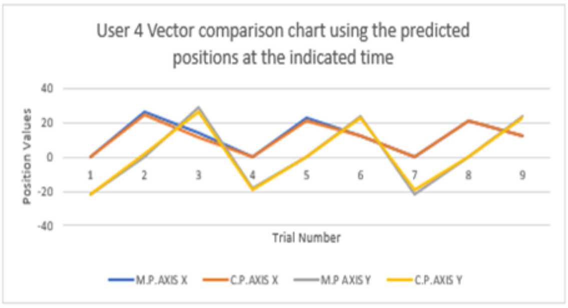


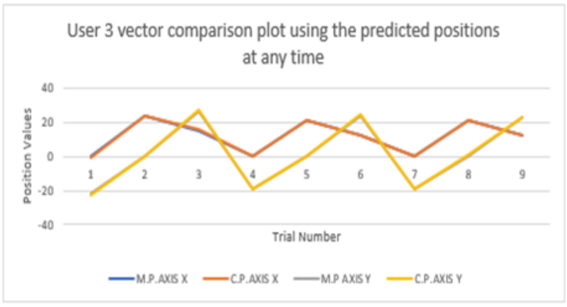


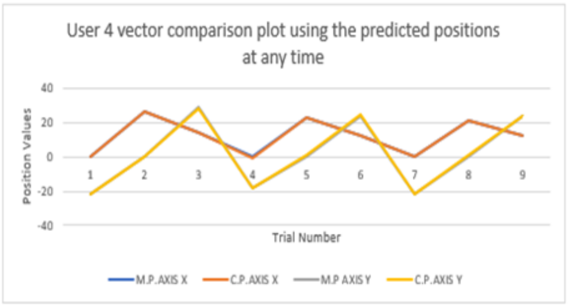


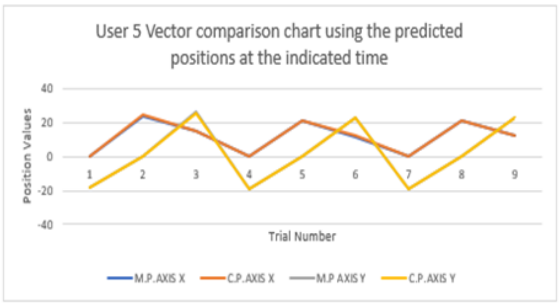


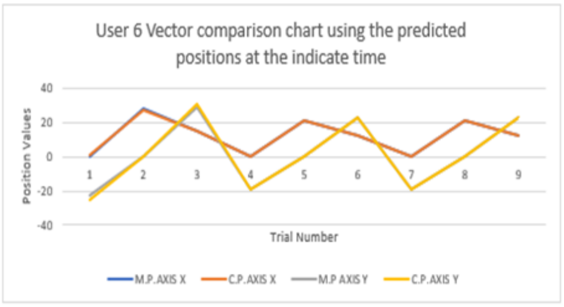

User 3 Vector comparison chart using the predicted positions at the indicated time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y

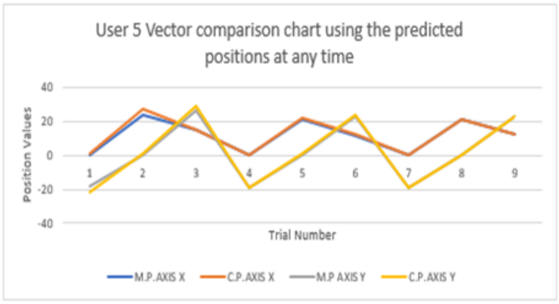

User 4 Vector comparison chart using the predicted positions at the indicated time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y

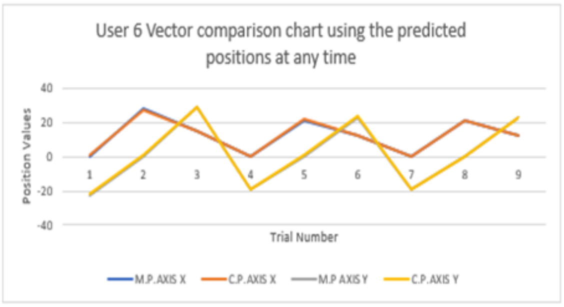

User 3 vector comparison plot using the predicted positions at any time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y

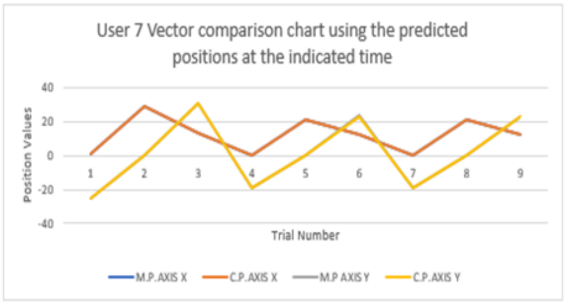

User 4 vector comparison plot using the predicted positions at any time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y

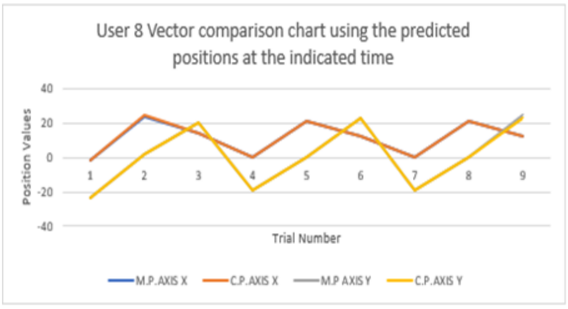

User 5 Vector comparison chart using the predicted positions at the indicated time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y

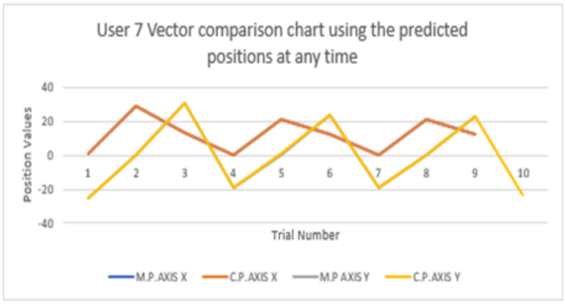

User 6 Vector comparison chart using the predicted positions at the indicate time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y

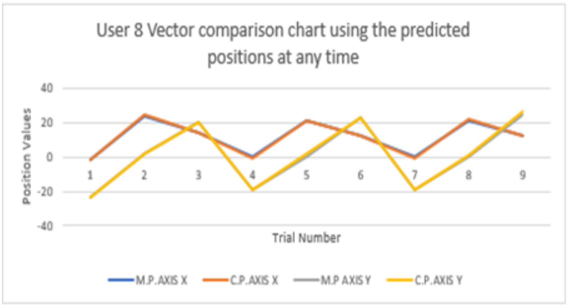

User 5 Vector comparison chart using the predicted positions at any time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y

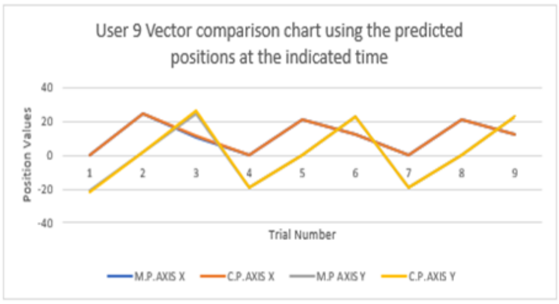

User 6 Vector comparison chart using the predicted positions at any time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y

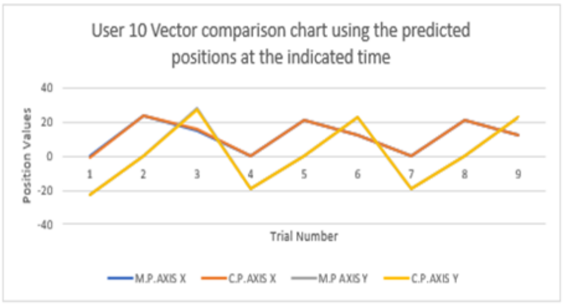

User 7 Vector comparison chart using the predicted positions at the indicated time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y

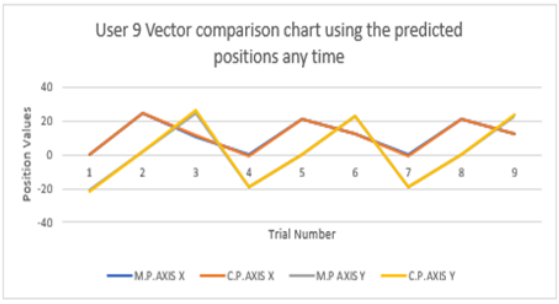

User 8 Vector comparison chart using the predicted positions at the indicated time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y

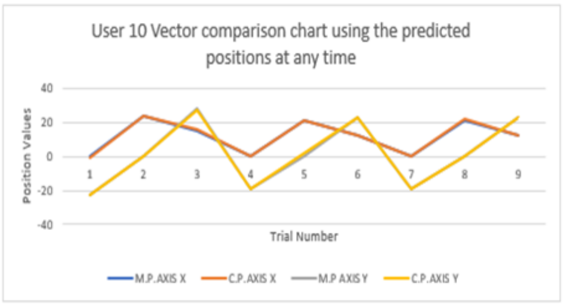

User 7 Vector comparison chart using the predicted positions at any time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y


User 8 Vector comparison chart using the predicted positions at any time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y

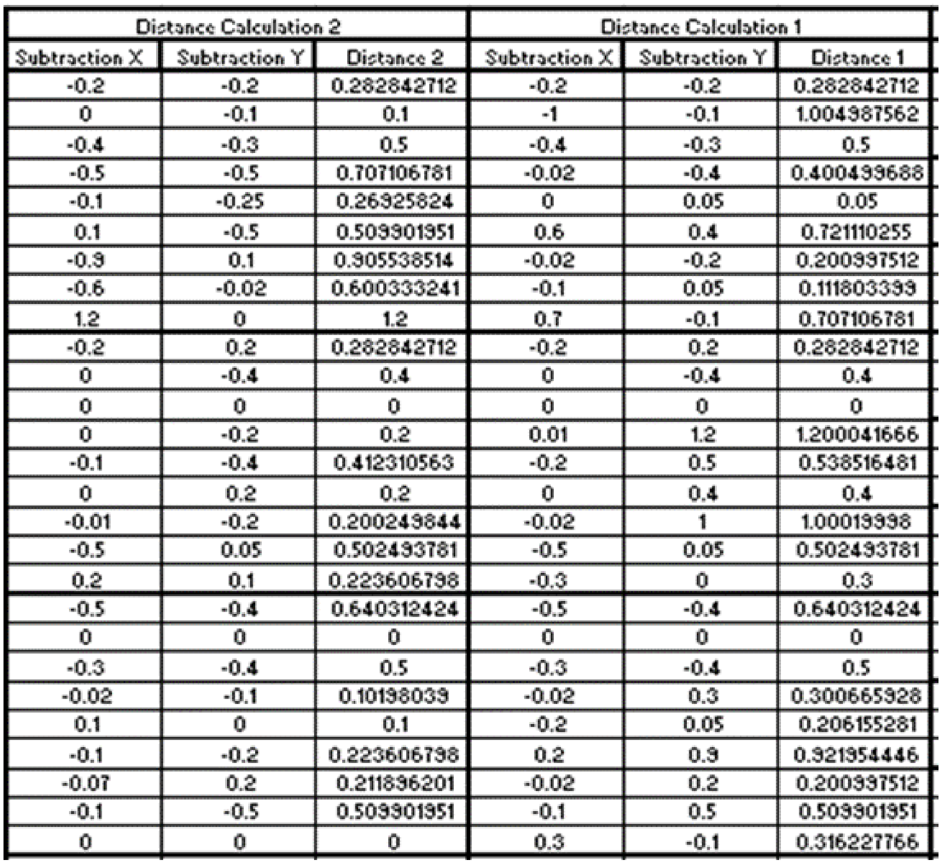

User 9 Vector comparison chart using the predicted positions at the indicated time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y


User 10 Vector comparison chart using the predicted positions at the indicated time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y


User 9 Vector comparison chart using the predicted positions any time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y


User 10 Vector comparison chart using the predicted positions at any time
Position Values
Trial Number
M.P. AXIS X
C.P. AXIS X
M.P AXIS Y
C.P. AXIS Y

Appendix II: Table of distance data calculated with the information in Appendix I



Distance Calculation 2			Distance Calculation 1		
Subtraction X	Subtraction Y	Distance 2	Subtraction X	Subtraction Y	Distance 1
-0.2	-0.2	0.282842712	-0.2	-0.2	0.282842712
0	-0.1	0.1	-1	-0.1	1.004987562
-0.4	-0.3	0.5	-0.4	-0.3	0.5
-0.5	-0.5	0.707106781	-0.02	-0.4	0.400499688
-0.1	-0.25	0.26925824	0	0.05	0.05
0.1	-0.5	0.509901951	0.6	0.4	0.721110255
-0.9	0.1	0.905538514	-0.02	-0.2	0.200997512
-0.6	-0.02	0.600333241	-0.1	0.05	0.111803399
1.2	0	1.2	0.7	-0.1	0.707106781
-0.2	0.2	0.282842712	-0.2	0.2	0.282842712
0	-0.4	0.4	0	-0.4	0.4
0	0	0	0	0	0
0	-0.2	0.2	0.01	1.2	1.200041666
-0.1	-0.4	0.412310563	-0.2	0.5	0.538516481
0	0.2	0.2	0	0.4	0.4
-0.01	-0.2	0.200249844	-0.02	1	1.00019998
-0.5	0.05	0.502493781	-0.5	0.05	0.502493781
0.2	0.1	0.223606798	-0.3	0	0.3
-0.5	-0.4	0.640312424	-0.5	-0.4	0.640312424
0	0	0	0	0	0
-0.3	-0.4	0.5	-0.3	-0.4	0.5
-0.02	-0.1	0.10198039	-0.02	0.3	0.300665928
0.1	0	0.1	-0.2	0.05	0.206155281
-0.1	-0.2	0.223606798	0.2	0.9	0.921954446
-0.07	0.2	0.211896201	-0.02	0.2	0.200997512
-0.1	-0.5	0.509901951	-0.1	0.5	0.509901951
0	0	0	0.3	-0.1	0.316227766

-0.2	0	0.2	-0.2	0	0.2
0.1	-0.4	0.412310563	0.1	-0.4	0.412310563
0.1	0.5	0.509901951	0.1	0.5	0.509901951
-0.5	0.5	0.707106781	-0.02	-0.5	0.50039984
-0.3	-0.5	0.583095189	1.5	0.5	1.58113883
-0.5	-0.2	0.538516481	0	0.9	0.9
-0.2	-0.5	0.538516481	-0.02	2.5	2.500079999
-0.3	-0.1	0.316227766	-0.3	0.5	0.583095189
-0.2	0	0.2	0	0.9	0.9
-0.2	0.1	0.223606798	-0.2	0.1	0.223606798
-0.2	-0.1	0.223606798	-0.2	-0.1	0.223606798
-0.3	0.2	0.360555128	-0.3	0.2	0.360555128
-0.2	0.3	0.360555128	-0.02	0.2	0.200997512
-0.9	-0.5	1.029563014	-0.5	0.5	0.707106781
-0.1	-0.1	0.141421356	-0.2	0.2	0.282842712
-0.5	0	0.5	-0.02	0.5	0.50039984
-0.3	-0.1	0.316227766	-0.2	0.5	0.538516481
-0.3	-0.2	0.360555128	0	-0.2	0.2
-0.7	0.7	0.989949494	-0.7	0.7	0.989949494
0.7	-0.6	0.921954446	0.7	-0.6	0.921954446
0.2	0	0.2	0.2	0	0.2
-0.01	0	0.01	-0.02	0	0.02
-0.4	-0.6	0.721110255	-0.3	0.5	0.583095189
-0.1	-0.5	0.509901951	0.2	-0.2	0.282842712
-0.5	-0.3	0.583095189	-0.02	0.2	0.200997512
-0.2	0.1	0.223606798	-0.5	0.5	0.707106781
0.1	-0.1	0.141421356	0.2	0.1	0.223606798

0	0	0	0	0	0
0	0	0	0	0	0
0.1	0	0.1	0.1	0	0.1
-0.2	-0.1	0.223606798	-0.02	0	0.02
0	-0.6	0.6	-0.1	0.5	0.509901951
0.1	0	0.1	0.1	0.4	0.412310563
0	-0.1	0.1	-0.02	-0.2	0.200997512
0.1	0	0.1	-0.2	0.5	0.538516481
0	-0.1	0.1	0	-0.1	0.1
0	-0.2	0.2	0	-0.2	0.2
-1	0	1	-1	0	1
-0.4	-0.2	0.447213595	-0.4	-0.2	0.447213595
-0.5	-0.2	0.538516481	-0.02	-0.2	0.200997512
0	-1.3	1.3	0	0.5	0.5
-0.1	-0.1	0.141421356	0.4	-0.1	0.412310563
-0.3	0	0.3	-0.02	-0.2	0.200997512
-0.2	-0.5	0.538516481	-0.1	0.5	0.509901951
0.2	-1	1.019803903	0.2	1.9	1.910497317
-0.2	-0.1	0.223606798	-0.2	-0.1	0.223606798
-0.4	0	0.4	-0.4	0	0.4
-0.3	-1.5	1.529705854	-0.3	-1.5	1.529705854
-0.5	0	0.5	-0.02	0	0.02
-0.2	0.2	0.282842712	-0.4	0.2	0.447213595
0	-0.2	0.2	0	-0.1	0.1
-0.5	-0.3	0.583095189	-0.02	0.2	0.200997512
-0.1	0	0.1	-0.1	0.5	0.509901951
0.1	-0.2	0.223606798	0.6	0.2	0.632455532
-0.5	0.5	0.707106781	-0.5	0.5	0.707106781
0	0	0	0	0	0
-0.6	0.5	0.781024968	-0.6	0.5	0.781024968
-0.2	-0.1	0.223606798	-0.02	0.3	0.300665928
-0.1	-1.5	1.503329638	-0.4	0.5	0.640312424
-0.2	0	0.2	0.5	-0.1	0.509901951
-0.02	0.1	0.10198039	-0.02	-0.1	0.10198039
-1	0.03	1.000449899	-0.4	0.05	0.403112887
0	-0.1	0.1	0	0.1	0.1

Appendix III:Table of distance data calculated with the information in Appendix I, divided by task.

DISTANCE 2 TASK 1	DISTANCE 2 TASK 2	DISTANCE 2TASK 3	DISTANCE 1 TASK 1	DISTANCE 1 TASK 2	DISTANCE 1 TASK 3
0.283	0.707	0.906	0.283	0.400	0.201
0.100	0.269	0.600	1.005	0.050	0.112
0.500	0.510	1.200	0.500	0.721	0.707
0.283	0.200	0.200	0.283	1.200	1.000
0.400	0.412	0.502	0.400	0.539	0.502
0.000	0.200	0.224	0.000	0.400	0.300
0.640	0.102	0.212	0.640	0.301	0.201
0.000	0.100	0.510	0.000	0.206	0.510
0.500	0.224	0.000	0.500	0.922	0.316
0.200	0.707	0.539	0.200	0.500	2.500
0.412	0.583	0.316	0.412	1.581	0.583
0.510	0.539	0.200	0.510	0.900	0.900
0.224	0.361	0.500	0.224	0.201	0.500
0.224	1.030	0.316	0.224	0.707	0.539
0.361	0.141	0.361	0.361	0.283	0.200
0.990	0.010	0.583	0.990	0.020	0.201
0.922	0.721	0.224	0.922	0.583	0.707
0.200	0.510	0.141	0.200	0.283	0.224
0.000	0.224	0.100	0.000	0.020	0.201
0.000	0.600	0.100	0.000	0.510	0.539
0.100	0.100	0.100	0.100	0.412	0.100
0.200	0.539	0.300	0.200	0.201	0.201
1.000	1.300	0.539	1.000	0.500	0.510
0.447	0.141	1.020	0.447	0.412	1.910
0.224	0.500	0.583	0.224	0.020	0.201
0.400	0.283	0.100	0.400	0.447	0.510
1.530	0.200	0.224	1.530	0.100	0.632
0.707	1.503	0.102	0.707	0.301	0.102
0.000	0.200	1.000	0.000	0.640	0.403
0.781	0.102	0.100	0.781	0.510	0.100

Appendix IV: Scripts C#

LogIn

```
using UnityEngine;

using UnityEngine.UI;

using System.Collections;

using System;

using System.Text.RegularExpressions;

public class Login : MonoBehaviour {

    public GameObject username;

    public GameObject password;

    public static string Username;

    private string Password;

    private String[] Lines;
    private string DecryptedPass;
    string Nombre;
    public void LoginButton(){
        bool UN = false;
        bool PW = false;
        if (Username != ""){

if(System.IO.File.Exists(@"C:/UnityTestFolder/"+Username+".txt")){
                UN = true;
                Lines = 
System.IO.File.ReadAllLines(@"C:/UnityTestFolder/"+Username+".txt");
            } else {
                Debug.LogWarning("Username Invaild");
            }
        } else {
            Debug.LogWarning("Username Field Empty");
        }
        if (Password != ""){
            if
(System.IO.File.Exists(@"C:/UnityTestFolder/"+Username+".txt")){
                int i = 1;
                foreach(char c in Lines[2]){
                    i++;
                    char Decrypted = (char)(c / i);
                    DecryptedPass += Decrypted.ToString();

                }
                if (Password == DecryptedPass){
                    PW = true;
                } else {
                    Debug.LogWarning("Password Is invalid");
                    Debug.LogWarning(DecryptedPass);
                }
            } else {
                Debug.LogWarning("Password Is invalid");
                Debug.LogWarning(DecryptedPass);
            }
        } else {
            Debug.LogWarning("Password Field Empty");
        }
        if (UN == true&&PW == true){
            username.GetComponent<InputField>().text = "";
            password.GetComponent<InputField>().text = "";
            print ("Login Sucessful");

            Application.LoadLevel("UI");
        }
    }
    // Update is called once per frame
    void Update () {
        if (Input.GetKeyDown(KeyCode.Tab)){
            if (username.GetComponent<InputField>().isFocused){
                password.GetComponent<InputField>().Select();
            }
        }
        if (Input.GetKeyDown(KeyCode.Return)){
            if (Password != ""&&Password != ""){
                LoginButton();

            }

        }


        Username = username.GetComponent<InputField>().text;

        Nombre = Username;

        Password = password.GetComponent<InputField>().text;

    }

}
```

Register

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using System;

using System.Text.RegularExpressions;


public class Register : MonoBehaviour

{

    public GameObject username;

    public GameObject email;

    public GameObject password;

    public GameObject Confpassword;

    private string Username;

    private string Email;

    private string Password;
    private string ConfPassword;

    private string form;

    private bool EmailValid = false;

    private string[] Characters = { "a", "b", "c", "d","e","f", "g",
"h", "i", "j", "k", "l", "m", "o","p", "q", "r", "s", "t", "u",
"v","w", "x", "y", "z",
"A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","
R","S","T","U","V","W","X","Y","Z","1","2","3","4","5","6","7","8","9
","0","_","-"};


    public void RegisterButton(){


        bool UN = false;

        bool EM = false;

        bool PW = false;

        bool CPW = false;


        if(Username != ""){

if(!System.IO.File.Exists(@"C:/UnityTestFolder/"+Username+".txt")){

                UN = true;

            }

            else{

            Debug.LogWarning("Ese nombre ya existe");

            }

        }

        else {

            Debug.LogWarning("Escriba el nombre");

        }

        if(Email != ""){

            EmailValidation();
            if(EmailValid){

                if(Email.Contains("@")){

                    if (Email.Contains(".")){

                        EM = true;


                    } else {

                        Debug.LogWarning("el Email es incorrecto");

                        }

                } else {

                    Debug.LogWarning(" el Email es Incorrecto");

                    }

            } else {

                Debug.LogWarning("el Email es Incorrecto");

                }

        } else {

            Debug.LogWarning("Escriba el Email");

        }

        if (Password != ""){

            if(Password.Length > 5){

                PW = true;

            } else {

                Debug.LogWarning("La Contraseña debe ser de 6 digitos
como minimo");

            }

        } else {

            Debug.LogWarning(" escriba la contraseña");

        }

        if (ConfPassword != ""){
            if (ConfPassword == Password){

                CPW = true;

            } else {

                Debug.LogWarning("La contraseña no coincide");

            }

        } else {

            Debug.LogWarning("Confirme la contraseña");

        }

        if (UN == true&&EM == true&&PW == true&&CPW == true){

            bool Clear = true;

            int i = 1;

            foreach(char c in Password){

                if(Clear){

                    Password = "";

                    Clear = false;

                }

                i++;

                char Encrypted = (char) (c * i);

                Password += Encrypted.ToString();

            }

            form = (Username+"\n"+Email+"\n"+Password);

System.IO.File.WriteAllText(@"C:/UnityTestFolder/"+Username+".txt",fo
rm);

            username.GetComponent<InputField>().text = "";

            email.GetComponent<InputField>().text = "";

            password.GetComponent<InputField>().text = "";

            Confpassword.GetComponent<InputField>().text = "";

            print ("Registro completo");
```

ETSEIB

```
        }

    }

    // Update is called once per frame

    void Update()

    {

        if(Input.GetKeyDown(KeyCode.Tab)){

            if(username.GetComponent<InputField>().isFocused){

                email.GetComponent<InputField>().Select();

            }

            if(email.GetComponent<InputField>().isFocused){

                password.GetComponent<InputField>().Select();

            }

            if(password.GetComponent<InputField>().isFocused){

                Confpassword.GetComponent<InputField>().Select();

            }

        }


        if (Input.GetMouseButtonDown(0)){

            if(Password != ""&&Email != ""&&Password !=
""&&ConfPassword != ""){

                RegisterButton();

            }

        }


        Username = username.GetComponent<InputField>().text;

        Email = email.GetComponent<InputField>().text;

        Password = password.GetComponent<InputField>().text;

        ConfPassword = Confpassword.GetComponent<InputField>().text;
    }

    void EmailValidation(){

        bool SW = false;

        bool EW = false;

        for(int i = 0;i<Characters.Length;i++){

            if (Email.StartsWith(Characters[i])){

                SW = true;

            }

        }

        for(int i = 0;i<Characters.Length;i++){

            if (Email.EndsWith(Characters[i])){

                EW = true;

            }

        }

        if(SW == true&&EW == true){

            EmailValid = true;

        }else {

            EmailValid = false;

        }

    }

}
```

A continuación se encontrarán los scripts de movimiento del círculo debido a que son muchos pero entre ellos solo cambia la dirección se mostrará como ejemplo uno de los 8 movimientos por cada tarea.

Movement First task example scene

```
using System.Collections;

using System.Collections.Generic;
using UnityEngine;

public class P1 : MonoBehaviour{

    Rigidbody2D rb;

    //public

    float speed = -25f;

    void Awake(){

        rb=GetComponent<Rigidbody2D>();

    }

    // Start is called before the first frame update

    void Start(){

        rb.velocity=new Vector2(0, speed);

    }

}
```

Movement first task trajectory prediction scene

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System;

using System.Text.RegularExpressions;

using UnityEngine.UI;


public class PP1 : MonoBehaviour

{

    //movement speed in units per second

    private float movementSpeed = -25f;

    Rigidbody2D rb;
    void Awake(){

        rb=GetComponent<Rigidbody2D>();

    }

    // Start is called before the first frame update

    void Start(){


        rb.velocity=new Vector2(0, movementSpeed);

    }

    void Update()

    {

        //get the Input from Horizontal axis

        float horizontalInput = Input.GetAxis("Horizontal");

        //get the Input from Vertical axis

        float verticalInput = Input.GetAxis("Vertical");


        //output to log the position change

        if ( Input.GetMouseButtonDown (0))

        {


            Vector3 mouseWorldPosition =
Camera.main.ScreenToWorldPoint(Input.mousePosition);

            Vector3 PointPosition3 = transform.position;


            string path = @"C:/UnityTestFolder/"+ "/Seguimiento/" +
"DatosDePosicion" + ".xls";

            string content = PointPosition3 + ";" +
mouseWorldPosition + "\n" ;


            if (!System.IO.File.Exists(path))
            {

                System.IO.File.AppendAllText(path, "Posicion del
mouse: "+";"+ "Posicion del punto: " +"\n");

            }


            if (!System.IO.File.Exists(path))

            {

                System.IO.File.AppendAllText(path, "Posicion del
mouse: "+";"+ "Posicion del punto: " +"\n");

                }

            System.IO.File.AppendAllText(path, content);


            Debug.Log(" Point position : "+ PointPosition3);

            Debug.Log ("New mouse position: " + mouseWorldPosition);


        }

    }

}
```

Movement second task example scene

```
using System.Collections;

using UnityEngine;


public class MOV2 : MonoBehaviour

{

    float timeCounter=0;

    float speed;

    float width;

    float height;
    // Start is called before the first frame update
    void Start()
    {
        speed = -0.5f;
        width = 30;
        height = 60;
    }
    void Update()
    {
        timeCounter += Time.deltaTime*speed;
        float x = ((Mathf.Sin(timeCounter)*width)+30)*2;
        float y = Mathf.Cos(timeCounter)*height;
        transform.position= new Vector2(x,y);
    }
}
```

Movement second task trajectory prediction scene

```
using System.Collections;
using UnityEngine;
public class MOVT2 : MonoBehaviour
{
    float timeCounter=0;
    float speed;
    float width;
    float height;
    // Start is called before the first frame update
    void Start()
    {
        speed = -0.5f;

        width = 30;

        height = 60;

    }

    void Update()

    {

        timeCounter += Time.deltaTime*speed;

        float x = ((Mathf.Sin(timeCounter)*width)+30)*2;

        float y = Mathf.Cos(timeCounter)*height;

        transform.position= new Vector2(x,y);

        //get the Input from Horizontal axis

        float horizontalInput = Input.GetAxis("Horizontal");

        //get the Input from Vertical axis

        float verticalInput = Input.GetAxis("Vertical");

        //output to log the position change

        if ( Input.GetMouseButtonDown (0))

        {

            Vector3 mouseWorldPosition =
Camera.main.ScreenToWorldPoint(Input.mousePosition);

            Vector3 PointPosition3 = transform.position;

            string path = @"C:/UnityTestFolder/"+ "/Seguimiento/" +
"DatosDePosicion" + ".xls";

            string content = PointPosition3 + ";" +
mouseWorldPosition + "\n" ;

            if (!System.IO.File.Exists(path))

            {

                System.IO.File.AppendAllText(path, "Posicion del
mouse: "+";"+ "Posicion del punto: " +"\n");

            }

            if (!System.IO.File.Exists(path))
            {

                System.IO.File.AppendAllText(path, "Posicion del
mouse: "+";"+ "Posicion del punto: " +"\n");

                }

            System.IO.File.AppendAllText(path, content);

            Debug.Log(" Point position : "+ PointPosition3);

            Debug.Log ("New mouse position: " + mouseWorldPosition);

        }

    }

}

using System.Collections;

using UnityEngine;

public class MOVT2 : MonoBehaviour

{

    float timeCounter=0;

    float speed;

    float width;

    float height;

    // Start is called before the first frame update

    void Start()

    {

        speed = -0.5f;

        width = 30;

        height = 60;

    }

    void Update()

    {

        timeCounter += Time.deltaTime*speed;
        float x = ((Mathf.Sin(timeCounter)*width)+30)*2;

        float y = Mathf.Cos(timeCounter)*height;

        transform.position= new Vector2(x,y);

        //get the Input from Horizontal axis

        float horizontalInput = Input.GetAxis("Horizontal");

        //get the Input from Vertical axis

        float verticalInput = Input.GetAxis("Vertical");

        //output to log the position change

        if ( Input.GetMouseButtonDown (0))

        {

            Vector3 mouseWorldPosition =
Camera.main.ScreenToWorldPoint(Input.mousePosition);

            Vector3 PointPosition3 = transform.position;

            string path = @"C:/UnityTestFolder/"+ "/Seguimiento/" +
"DatosDePosicion" + ".xls";

            string content = PointPosition3 + ";" +
mouseWorldPosition + "\n" ;

            if (!System.IO.File.Exists(path))

            {

                System.IO.File.AppendAllText(path, "Posicion del
mouse: "+";"+ "Posicion del punto: " +"\n");

            }

            if (!System.IO.File.Exists(path))

            {

                System.IO.File.AppendAllText(path, "Posicion del
mouse: "+";"+ "Posicion del punto: " +"\n");

                }

            System.IO.File.AppendAllText(path, content);

            Debug.Log(" Point position : "+ PointPosition3);

            Debug.Log ("New mouse position: " + mouseWorldPosition);
        }

    }

}
```

Movement third task example scene

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class M1  : MonoBehaviour{

    Rigidbody2D rb;

    //public

    float speed = -15f;

    void Awake(){

        rb=GetComponent<Rigidbody2D>();

    }

    // Start is called before the first frame update

    void Start(){

        rb.velocity=new Vector2(0, speed);

    }

}
```

Movement third task trajectory prediction scene

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class MV1T3  : MonoBehaviour{
    Rigidbody2D rb;

    float speed = -15f;

    void Awake(){

        rb=GetComponent<Rigidbody2D>();

    }

    // Start is called before the first frame update

    void Start(){

        rb.velocity=new Vector2(0, speed);

    }

    // Update is called once per frame

    void Update(){

        //get the Input from Horizontal axis

        float horizontalInput = Input.GetAxis("Horizontal");

        //get the Input from Vertical axis

        float verticalInput = Input.GetAxis("Vertical");

        //output to log the position change

         if ( Input.GetMouseButtonDown (0))

        {

            Vector3 mouseWorldPosition =
Camera.main.ScreenToWorldPoint(Input.mousePosition);// este si

            Vector3 PointPosition3 = transform.position; // este si

            string path = @"C:/UnityTestFolder/"+ "/Seguimiento/" +
"DatosDePosicion" + ".xls";

            string content = PointPosition3 + ";" +
mouseWorldPosition + "\n" ;


            if (!System.IO.File.Exists(path))

            {

                System.IO.File.AppendAllText(path, "Posicion del
mouse: "+";"+ "Posicion del punto: " +"\n");
```

```
            }

            if (!System.IO.File.Exists(path))

            {

                System.IO.File.AppendAllText(path, "Posicion del
mouse: "+";"+ "Posicion del punto: " +"\n");

                }

            System.IO.File.AppendAllText(path, content);

            Debug.Log(" Point position : "+ PointPosition3);

            Debug.Log ("New mouse position: " + mouseWorldPosition);

        }

    }

}
```

“Transp2” : Script para hacer que el círculo no sea visible por un tiempo y se haga visible de nuevo.

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Sprites;

public class transp2 : MonoBehaviour

{

    [Range(0,1)]

    public float Transparencia = 0, TransitionSpeed = 1;

    public SpriteRenderer spriteRenderer;

    bool canButton = true;

    public enum Modo{

        show = 0,

        Hide = 1,
        Nothing = -1,

    };

    public Modo modo;

    void Start()

    {

        modo = Modo.Nothing;

        spriteRenderer = GetComponent<SpriteRenderer> ();

    }

    void OnTriggerEnter2D  (Collider2D otherCollider)

    {

        modo = Modo.Hide;

        Transparencia += Time.deltaTime;

        spriteRenderer.color = new Color (spriteRenderer.color.r,
spriteRenderer.color.g, spriteRenderer.color.b, Transparencia);

    }

    void OnTriggerExit2D  (Collider2D otherCollider)

    {

        modo = Modo.show;

        Transparencia = 1;

        spriteRenderer.color = new Color (spriteRenderer.color.r,
spriteRenderer.color.g, spriteRenderer.color.b, Transparencia);

        Debug.Log("reaparece");

    }

    public void Activaate ()

    {

        canButton = true;

    }

}
```

“Transp3“ Script para hacer que el círculo no sea visible después de atravesar el cuadrado

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Transp3 : MonoBehaviour

{

    [Range(0,1)]

    public float Transparencia = 0, TransitionSpeed = 1;

    public SpriteRenderer spriteRenderer;

    bool canButton = true;

    public enum Modo{

        show = 0,

        Hide = 1,

        Nothing = -1,

    };

    public Modo modo;

    void Start()

    {

        modo = Modo.Nothing;

        spriteRenderer = GetComponent<SpriteRenderer> ();

    }

    void OnTriggerEnter2D  (Collider2D otherCollider)

    {

        modo = Modo.Hide;

        Transparencia += Time.deltaTime;

        spriteRenderer.color = new Color (spriteRenderer.color.r,
spriteRenderer.color.g, spriteRenderer.color.b, Transparencia);

    }
    public void Activaate ()

    {

        canButton = true;

    }

}
```

“Destroi“ Script Cambiar de escena por medio de colisiones

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Destroyi : MonoBehaviour

{

    public int numeroEscena;

    void OnTriggerEnter2D(Collider2D other){

        if (other.tag == "Player")

        {

            SceneManager.LoadScene(numeroEscena);

        }

    }

}
```

“MainMenu”: Script para cambiar de escena por medio de Botones

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;
```

```
public class MainMenu : MonoBehaviour
{
    public void ChooseScene()
    {
        SceneManager.LoadScene("LEVES");
    }
    public void ChooseScene1()
    {
        SceneManager.LoadScene("INF");
    }
    public void ChooseScene2()
    {
        SceneManager.LoadScene("V.AR");
    }
    public void ChooseScene3()
    {
        SceneManager.LoadScene("Instruccion1");
    }
    public void ChooseScene4()
    {
        SceneManager.LoadScene("Sample1");
    }
    public void ChooseScene5()
    {
        SceneManager.LoadScene("Sample2");
    }
     public void ChooseScene6()
    {
        SceneManager.LoadScene("Level2");
    }
     public void ChooseScene7()
    {
        SceneManager.LoadScene("Sample3");
    }
     public void ChooseScene8()
    {
        SceneManager.LoadScene("Level3");
    }
    public void ChooseScene9()
    {
        SceneManager.LoadScene("UI");
    }
    public void ChooseScene10()
    {
        SceneManager.LoadScene("Sample1T2");
    }
    public void ChooseScene11()
    {
        SceneManager.LoadScene("E1");
    }
    public void ChooseScene12()
    {
        SceneManager.LoadScene("ejemplo1");
    }
    public void ChooseScene13()
    {
        SceneManager.LoadScene("EX1");
    }
    public void ChooseScene14()
    {
        SceneManager.LoadScene("EJ1");
    }
    public void ChooseScene15()
    {
        SceneManager.LoadScene("EXA1");
    }
    public void ChooseScene17()
    {
        SceneManager.LoadScene("NV1");
    }
    public void ChooseScene18()
    {
        SceneManager.LoadScene("EXA2");
    }
    public void ChooseScene19()
    {
        SceneManager.LoadScene("NV2");
    }
    public void ChooseScene20()
    {
        SceneManager.LoadScene("EXA3");
    }
    public void ChooseScene21()
    {

        SceneManager.LoadScene("NV3");

    }

}
```

“Timer”: Script temporizador

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class Timer : MonoBehaviour

{

    public Text contador;

    private float tiempo = 8f;

    void Start()

    {

        contador.text = " " + tiempo;

    }

    void Update()

    {

       tiempo -= Time.deltaTime;

       contador.text = " " + tiempo.ToString("f0");

       if(tiempo <= 0)

       {

           contador.text ="";

       }

    }

}
```

Appendix V: Videos application

The video of the first phase of the application can be found as Appendix V.A. and the video of the application with the restructuring done can be found as Appendix V.B. The videos of the application can be viewed at the following link :
https://drive.google.com/drive/u/2/folders/13b53sX0gIucIlyHffeS5t1ASqnxNomo