



# Type 2

**Emilio José Bort Moreno**

Final Degree Work  
Bachelor's Degree  
in Video Game Design and Development

Tutor: Begoña Martínez Salvador  
Universitat Jaume I

## ACKNOWLEDGMENTS

First of all, I would like to thank Mónica Cubo, head of Paediatrics at the General Hospital of Castellón de la Plana for her collaboration that helped me understand all the necessary concepts about diabetes and the needs of the hospital and patients.

I would also like to thank Dra Mar del Mar Marcos Lopez, PhD from the Universitat Jaume I and professor of the subject of Programming I, together with Begoña Martínez Salvador, PhD in Engineering in Computer Science from the Universitat Jaume I, which has made it possible to accomplish the design and development of this application. Last but not least I would like to thank the work and dedication of all the teachers partaking in the Videogames Design and Development Degree.

## ABSTRACT

The project consists of an application for mobile devices where children should take care of a virtual pet keeping it happy thanks to the happiness bar used for tracking it.

These cares will be provided to the virtual pet by introducing in a simple and intuitive way the values to monitor the children illness through the day.

Once the child enters the necessary values and if they are within the correct ranges of glucose, the child will be rewarded with collectibles in the form of new creatures.

We find a food screen, where we have different dishes to know the amount of carbohydrates in each of them.

A simple endless runner has been developed to increase the happiness of our pet by playing with it.

This product has been designed to facilitate the daily life of both parents and children who suffer from this disease.

With this work, I have acquired new competences in terms of the development of applications and videogames with Unity, database management and improvement in the field of programming, since my profile both professionally and in the career is focused on design.

During the whole project I have been required to test the knowledge in each of the fields of work necessary for the application development and as a result, I have developed new skills related to diabetes and improved my knowledge of Unity and databases. Thanks to the previously stated, I have been able to create a project as completely as possible.

# INDICE

<b>1. INTRODUCTION</b> .....	6
1.1. Search for a need.....	6
1.2. Research on diabetes.....	6
1.2.1. What is the diabetes?.....	6
1.2.2. What is the ratio?.....	8
1.2.3. What is the carbohydrate ration?.....	8
1.2.4. Insulin.....	8
1.2.5. Correction of Hyperglycemia according to sensitivity index.....	9
1.3. Research on gamification methods.....	9
1.3.1. What is gamification?.....	9
1.3.1.1. Gamification process.....	10
1.3.1.2. Gamification strategies for data collection.....	10
1.3.2. Research or similar applications.....	12
1.3.2.1. MySugr.....	13
1.3.2.2. GlucoZor.....	14
1.4. Motivation.....	14
1.5. Objectives.....	15
1.6. Environment and initial state.....	16
2.1. Methodology.....	17
2.2. Planning.....	18
2.3. Resources evaluation.....	21
<b>3. SYSTEM ANALYSIS AND DESIGN</b> .....	22
3.1. Requirements analysis.....	22
3.1.1. Functionals requirements.....	22
3.1.2. Non-functionals requirements.....	24
3.2. Game mechanics.....	24
3.2.1. Playability.....	24
3.2.2. Game's flow.....	24
3.2.3. Endless Run Game.....	26
3.2.4. Activity diagram and case use diagram.....	27
3.2.5. Interaction and controls.....	29
3.3. System architecture.....	29
3.4. Interface design.....	31

3.4.1.	Characters.....	31
3.4.1.1.	Glu.....	31
3.4.1.2.	Collectables .....	32
3.4.1.3.	Main screen .....	33
3.4.1.4.	Screen introduction of values .....	34
3.4.1.5.	Open collectibles screen.....	34
3.4.1.6.	Collectables list screen .....	35
3.4.1.7.	Food screen .....	35
3.4.1.8.	Happiness Bar .....	36
3.4.1.9.	Endless Run Game .....	36
<b>4.</b>	<b>WORK DEVELOPMENT AND RESULTS.....</b>	<b>37</b>
4.1.	Saved the state of the game.....	37
4.2.	Use databases in Unity .....	37
4.3.	Scene handling .....	38
4.6.	Food screen .....	39
4.7.	States of the day .....	39
4.8.	Develop Endless Runner .....	40
<b>5.</b>	<b>RESULTS AND FUTURE WORK .....</b>	<b>42</b>
5.1.	Results .....	42
5.2.	Testing.....	42
5.3.	Evaluation .....	44
5.4.	Future work .....	44
<b>6.</b>	<b>CONCLUSIONS .....</b>	<b>46</b>
<b>7.</b>	<b>BIBLIOGRAPHY .....</b>	<b>47</b>
<b>8.</b>	<b>APPENDIX A – SOURCE CODE .....</b>	<b>49</b>
<b>9.</b>	<b>APPENDIX B – VIDEOGAME ART.....</b>	<b>90</b>

# 1. INTRODUCTION

In the following chapters we will proceed to describe and discuss the objectives that are intended to achieve with this work, we will also explain the process of how it has been carried out, the tools that have been used, the manner in which the idea was conceived and the key factors that have motivated us to carry out this specific project.

## 1.1. Search for a need

Before starting the project, on Wednesday, October 11, 2018, we met with Dra Mónica Cubo, (Head of Paediatrics at the General Hospital of Castellón), at the Burriana health centre, that a series of guidelines, tips and clues, about what the health centre needed, and what we can provide to children, so that the application is useful.

In the beginning, the application was going to be a combination of the MySugr and Glucozor applications, but after the meeting we changed the concept of the application and went to what is really useful for the health centre, which is our target, apart from the help for children. Also, we receive different books to extract part of the information necessary for the study of diabetes.

With the ideas already clear about the project and the changes made in the strategies for data collection, on April 12, 2019 we made a second interview, to teach the first images and the final idea and ask a series of questions about the diabetes.

## 1.2. Research on diabetes

In the first place, we had to carry out a study about [1] what diabetes is, for this we had to start asking what is this disease, which is the sensitivity factor, which is the ratio, carbohydrates, types of insulin and the corrections we can make in the data.

### 1.2.1. What is the diabetes?

Diabetes is a chronic disease that arises because the pancreas does not synthesize the amount of insulin the human body needs, makes it of inferior quality or is not able to use it effectively.

For the realization of the application we had to collect information about the different data we are going to use, among which we find the sensitivity factor, the ratio, the carbohydrate rations and the insulin.

### How is the sensitivity factor calculated?

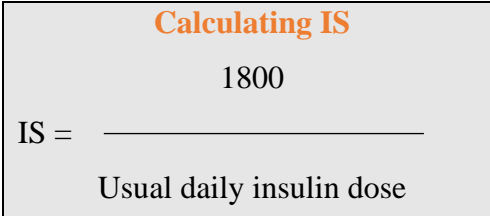
The insulin sensitivity factor (FSI), also known as the correction factor, is what decreases glucose, in each person (in milligrams / dl), each unit of rapid insulin that is put; that is, the amount of glucose that metabolizes 1 IU of insulin.

When we have high glucose levels, it is essential to know the amount of insulin each patient needs to cover this excess blood glucose, for this we need to know what the sensitivity factor is.

Although a formula can be used to calculate the sensitivity factor (see Figure 1), in our application we will only ask for that value when starting the application since it is a value that they provide us in the medical centre. We will have enabled in options the possibility of changing this data, since although for some patients it is a fixed value, for others it can vary. If we wanted to calculate this factor, it is essential to count the daily units of fixed insulin that we put in each day.

**For example:** 6 units of fast at lunch + 18 units of slow + 7 units of fast food + 5 units of fast snack, would give us a total of 36 total daily units of insulin.

After this step we would have to make the formula  $1800 /$  or  $1700 /$  daily dose (36 units) = sensitivity factor  $(1800/36) = 50$ . This means that for each unit of insulin that is injected, 50 milligrams / dl decreases glucose in our blood.


$$\text{IS} = \frac{1800}{\text{Usual daily insulin dose}}$$

(Figure 1) Calculating the sensitivity

### 1.2.2. What is the ratio?

The ratio or number of units of fast insulin needed for each 1 serving of HC (or 10 grams of HC), is a value that, although it can vary, is usually kept fixed for each patient, as is the sensitivity factor. In our application it is a value that we can modify.

### 1.2.3. What is the carbohydrate ration?

Counting carbohydrates (HC) is essential to keep diabetes under control. To measure carbohydrates the concept of ration is used.

A ration is the amount of food that contains certain grams of carbohydrates.

There are several systems to count carbohydrates: 1 serving = 10gr HC

On the food screen of our application we find the already measured carbohydrates that each food has

### 1.2.4. Insulin

At present, the types of insulin available for the treatment of diabetes are several, each of them equally valid and adapted according to the patient and their needs.

- Ultrafast action insulin
- Rapid-acting insulin
- Intermediate action insulin
- Insulins with slow or basal action
- Mixed Insulins

In this way insulins are classified according to the action of each of them, differentiating three times since injected:

The onset: is the process from the time insulin is injected until it reaches the bloodstream and begins to lower glucose levels. The insulin is not immediate, therefore, unless we are low in glucose, we must inject fast insulin before eating, after having calculated the amount that will be necessary.

The peak of action: it is the moment of maximum action, that is, when insulin makes a greater effect and quickly reduces blood sugar levels. When the patient is in periods of

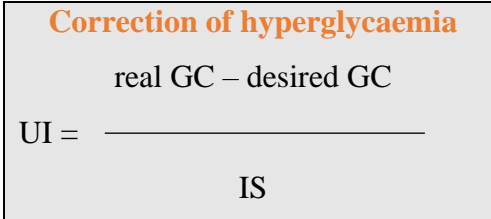


glycaemic decompensation, he has to wait for this period to pass so that the insulins do not overlap and we go from 200 to 40 glucose.

The total duration: corresponds to the time that insulin is taking effect in our body. Since we puncture until it stops eliminating glucose

### 1.2.5. Correction of Hyperglycaemia according to sensitivity index

We found another formula (see figure 2) to perform the correction, but since the ratio and sensitivity factor values are provided by the medical centre, we will not use it.


$$\text{UI} = \frac{\text{real GC} - \text{desired GC}}{\text{IS}}$$

(Figure 2) Calculating the correction of hyperglycaemia

## 1.3. Research on gamification methods

### 1.3.1. What is gamification?

When we speak of the term gamification, we refer to a series of dynamic learning techniques for the improvement of results, absorption of new knowledge or, in the case of our application, to be rewarded for carrying out a series of concrete actions.

The main idea of gamification is not to create a game, it is to use the reward-reward-objective systems that video games possess. The main objective that we find in our application is to motivate the child to keep his glucose levels within limits. We will do this through a reward system, in which the child will receive a new creature each time he meets these goals.

To meet these objectives the child will have to enter in the application: (1) the values of the glucose levels prior to meals, (2) the rations consumed, (3) the units of insulin administered taking into account those rations and (4) the values of glucose levels two

hours after food intake, trying to achieve the child's loyalty to the use of the application through the reward system.

Thanks to the playful nature we will be able to internalize knowledge and the necessary habits in a fun way, generating a positive experience for them.

#### 1.3.1.1. Gamification process

In the processes of gamification, it is important that the user can feel the application as personal and reactive, being rewarded for his efforts in a direct and recognizable personal way.

In the field of games this is known as a core loop or actionable action, being at this point necessary to make a list of key actions, objectives and rewards, as well as establishing the audience to which the action is directed. Therefore, we must offer certain limited resources so that users can feel progress, and their effort is rewarded. In this section we should not forget the users who start the application, incentivizing them with some of the rewards with less effort.

Digital goods are perfect for this task, in our application these rewards will be formed by a limited number of collectible creatures.

#### 1.3.1.2. Gamification strategies for data collection

Gamification is proving to be one of the most successful ways of collecting data. [2][3] An opportunity must be created for the collection of data for the monitoring of diabetes in patients for the hospital and for this reason I think that with this strategy it would be the best way to carry out this task.

In the currently technological moment, when we talk about processing large amounts of data, the term Big Data [4] comes to mind. We call Big Data the management and analysis of huge volumes of data that can not be treated in a conventional manner, since the collection of these data can be very costly, both in time and money, but the benefits of this data can be very valuable for companies and there have been many who have resorted to gamification techniques to obtain a greater number of data.

For this, companies are nourished by different reward systems, thus obtaining the data they need. Companies use reward programs through survey systems, gamification being

an almost essential element in the world of digital marketing for this type of commercial strategies.

Some examples in terms of elements for rewards are:

Achievements: Reward with a badge or level up to achieve some of the proposed achievements.

- Classification tables: Ranking systems by score
  - Countdown: Limited time to complete some task
- Progress: Shows the amount of progress of an activity.

We found these and many more examples in the deck of cards and app created by the company Zynga, SCVGR's Game Mechanics [5].

In our app the elements that we will find are, on the one hand, the achievements to meet the objectives (the child will be rewarded with a new creature when it meets the proposed objectives). On the other hand, we will find the elements of progress: this being in our application a list of all the creatures that we have been getting in the course of the game and those that we have yet to achieve. This is a technique used in some famous games such as the Pokémon franchise Game Freak and Nintendo, with its famous *Pokédex*.

Although in our application we store a small amount of daily values, this amount increases considerably as the months pass, that is why we have to take into account the way of storing them, so our app has a database to store them and be able to manipulate them so that the medical centre can have access to them in order to be able to carry out studies using these values.

At first the idea was that the player was given how to reward different objects to customize their character, as shown in the following image, but discarded as it was not at all satisfactory.

### 1.3.2. Research or similar applications

Before starting with the development of the application, a research was made both in the different stores of mobile devices and in desktop applications.

The types of application that we find are on the one hand to the purest daily style, as it is the case of "Diabetes - Diary of glucose", or "MySugr [6]" found in the Play Store and App Store of Google and Apple respectively.

They are applications for personal use, complex with many statistical data and graphics aimed at a more adult type of public and disconnected from medical centres. There are very different applications to the type of application that we want to develop since what we intend to achieve with our application is a direct connection between the health centre and the child, this being between 7 and 12 years old.

Other applications found for children are GlucoZor [7] and Monster Manor.

On the other hand, during the search for information about video games related to diabetes, specifically in the article Digital Games for Type 1 and Type 2 Diabetes Underpinning Theory With Three Illustrative Examples we find the Monster Manor application developed by the Ayogo Health study, where they tried to involving children in the management of diabetes while having fun, an application with a very elaborate graphic style, but unfortunately the application that was in a closed beta, remained that way permanently.

In the following subsections we will carry out a more in-depth study of these applications.

### 1.3.2.1. MySugr

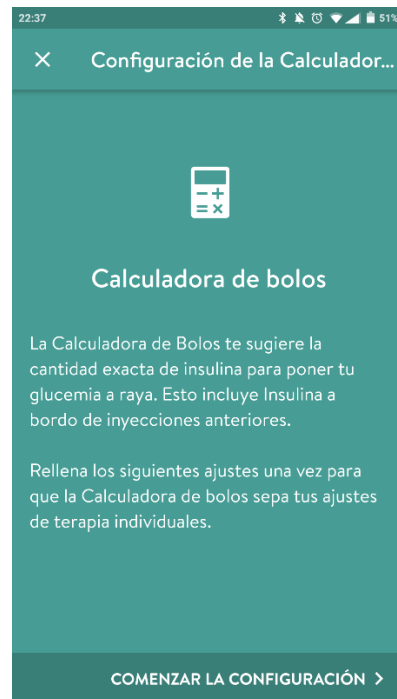
MySugr (Figures 6 and 7) is a journal of records to monitor and manage Type 1 and 2 diabetes.

Among its main characteristics we find all the necessary medical information easily controlled, have a control of our diet, be able to administer the insulin dosage, store the records to avoid hiccups / hypers and maintain diabetes control, being some of these characteristics exclusively of payment.

It is a very complete application in terms of options, as long as we choose the payment plan, but it can be a bit complex to use and its use is linked to more adult people, and who do not want to visit the centre so often doctor since they know their disease much better.



(Figure 6) MySugr App Start Screen



(Figure 7) Correction Bolus Calculator

### 1.3.2.2. GlucoZor

GlucoZor (Figures 8 and 9), is an application developed by Air Liquide's laboratory, an application that we find exclusively in French, in which the child must take care of a dinosaur so that it keeps its glucose levels within limits.



(Figure 8) GlucoZor app start screen



(Figure 9) Game screen GlucoZor app

## 1.4. Motivation

The development of this application born to cover a need that currently exists. This is the preservation of the data that diabetes patients (in this case children) must bring to the hospital to follow up on the disease. This control is currently based on a tedious and boring system for both parents and children, since it is done by means of a printed spreadsheet (see Figure 10), which must be filled in by hand with the values of the current day, the previous glucose, the consumed rations, and then to perform the necessary calculations for each of the meals of the day to know the total insulin needed to also fill in the table.

FECHA	DESAYUNO				ALMUERZO				COMIDA				MERIENDA				CENA			OBSERVACIONES	
	Gluc. Pre	Insul.	Racion H.C.	Gluc. 2H	Gluc. Pre	Insul.	Racion H.C.	Gluc. 2H	Gluc. Pre	Insul.	Racion H.C.	Gluc. 2H	Gluc. Pre	Insul.	Racion H.C.	Gluc. 2H	Gluc. Pre	Insul.	Racion H.C.		Gluc. 2H
17																					
18																					
19																					
20																					
21																					
22																					
23																					
24																					
25																					
26																					
27																					
28																					
29																					
30																					
31																					

(Figure 10) Table where the values are currently filled

In this table the values that we find are: the current day, the previous glucose, the insulin that has been delivered, the carbohydrate rations consumed, and the glucose after two hours. All these values for each of the moments of the day (breakfast, lunch, lunch, snack and dinner).

## 1.5. Objectives

The objectives we want to achieve with this application is to promote the lives of children suffering from diabetes, help the hospital to have the data in an orderly and digital format and finally improve the skills and knowledge that we have acquired throughout the grade. All these objectives converge in the creation of an application for Android mobile devices, through Unity.

To help the child, the application proposes three genders. On the one hand, the genre known as Virtual Pet on the other hand, that of Collectible Card Games (CCG, from the English Collectible Cards Games) and finally an Endless Runner.

**Virtual Pet:** The objective of these games is to care for a creature or animal, "since birth" and help it grow. In our case, we will have to introduce the values throughout the month and play with the game we have developed to keep our pet happy.

**CCG:** The CCGs are characterized by the individuality of each letter. Each card has a function and effects in the game, as well as some features. To motivate the user in the introduction of values in our case instead of letters you will be rewarded with unique creatures as collectibles.

**Endless Runner:** The goal of an endless runner is to get the maximum score by going through an infinite scenario, dodging objects, we have developed a simple endless runner to increase the happiness of our pet playing with it.

To help the hospital we can generate at the end of the month a CSV file with all the values entered by the child during the month. We achieve all these objectives with:

- A file: That stores the different states of the game, among them we find, the happiness indexes of our pet, the score of the game endless runner, the booleans to enable or disable the rewards received by the introduction of values.
- In a database: All values for the control of diabetes, in order to be able to export them to PDF or CSV to be delivered to the medical centre.
  - In a file: The different states of the game.
  - In a database: All values for the control of diabetes, in order to be able to export them to PDF or CSV to be delivered to the medical centre.

## 1.6. Environment and initial state

As mentioned in the previous section, due to the existing need for data preservation in most hospitals and for the families affected, the creation of a much more dynamic and interactive application arises, offering a much more efficient way to store data and to monitor this disease, that also includes a reward system that helps in motivating children into keep track of their our blood glucose values, insulin doses, etc...

For the adequate development of this project, all of the required steps established by both the tutor and the head of pediatrics at the Hospital General (Mónica Cubo), at all times



have been thoroughly contrasted after having conducted numerous interviews in order to obtain a better understanding of the disease that have enabled the development of the application, and to disperse other substantial queries.

## **PLANNING A RESOURCES EVALUATION**

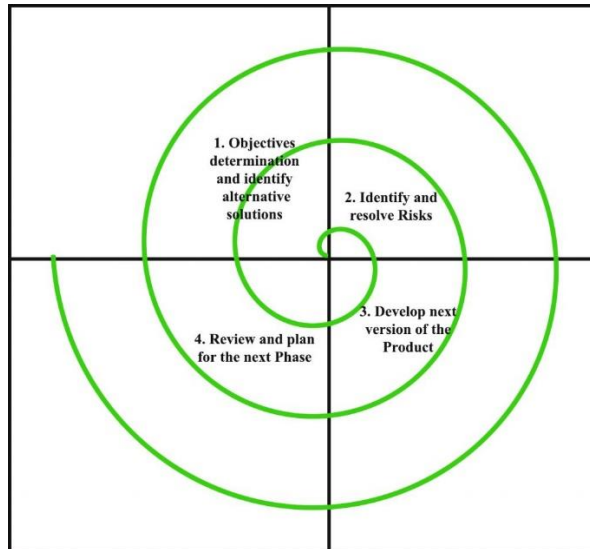
In this chapter, we will focus on the more technical aspects of the application, as well as detailing the steps taken and the material required to fulfill all of the aspects of the project, an analysis of the economical viability of the project and the equipment needed for the implementation of this application.

### **1.7. Methodology**

For the methodology, we have used the spiral model [8], each phase of Spiral Model is divided into four quadrants as shown in the above (figure 11)

The functions of these four quadrants

- Objectives determination and identify alternative solutions: Requirements are gathered from the customers and the objectives are identified, elaborated and analysed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
- Identify and resolve Risk: During the second quadrant all the possible solutions are evaluated to select the best possible solution. Then the risk associated with that solution is identified and the risks are resolved using the best possible strategy. At the end of this quadrant, Prototype is built for the best possible solution.
- Develop next version of the Product: During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available
- Review and plan for the next Phase: In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next version of the software is available.



(Figure 11) Spiral model quadrants

## 1.8. Planning

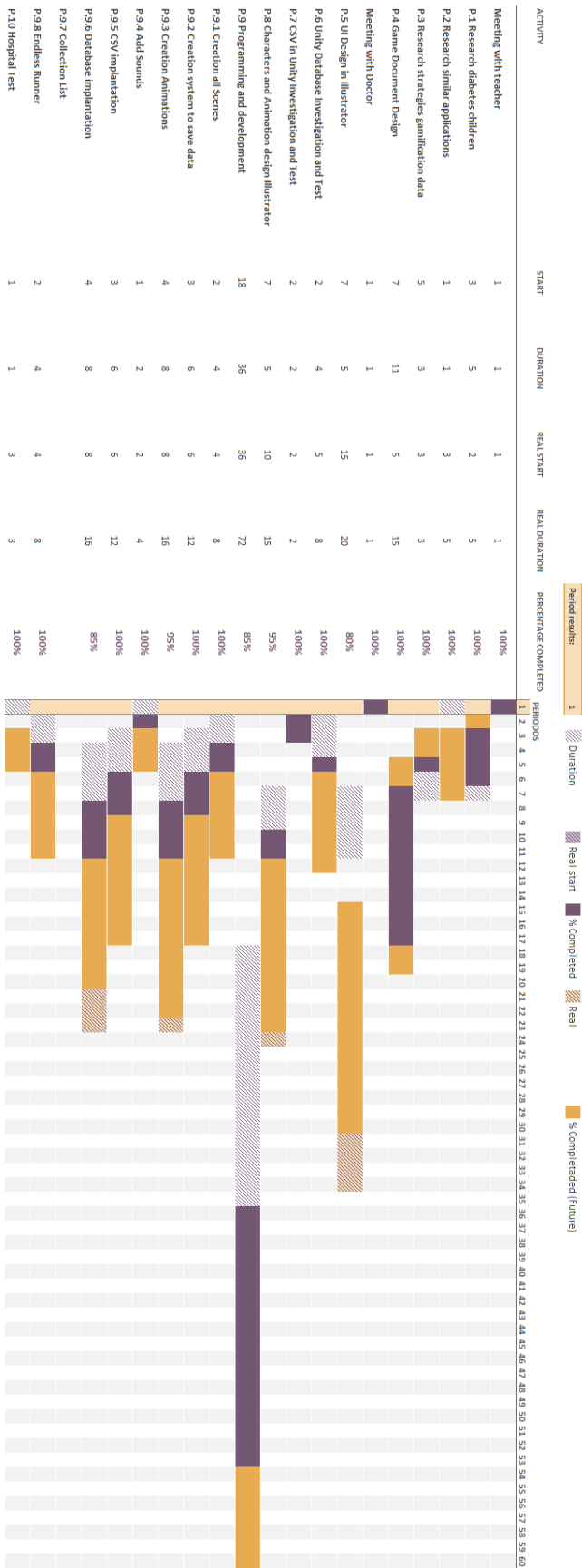
Next, we will detail the planning involved in the realization of this project, and to this effect, we will use (Figure 12)

Next, each step is described:

Activity	Description
Meeting with teacher	First meeting with tutor to talk about the different ideas for the application
P.1 Research diabetes children	Search of information about diabetes in children and adults, control of the disease
P.2 Research similar applications	Download and try similar applications and see what can be improved
P.3 Research strategies gamification data	Gamification information search to make our application fun
P.4 Game Document Design	Realization and modification of the design document
Meeting with Doctor	Interview with Mónica Cube for the search of the need of the hospital
P.5 UI Design	Design UI with Adobe Illustrator
P.6 Unity Database Investigation and Test	As we needed to save a lot of data, we used SQLite in Unity

P.7 CSV in Unity Investigation and Test	Search how to extract data from the database and export it to an easy-to-manage format
P.8 Characters and Animation	Design in Adobe illustrator
P.9 Programming and development	Program in Unity the application in this process is divided into different sections
P.9.1 Creation all Scenes	Create the scene manager
P.9.2 Creation system to save data	Create a save system
P.9.3 Creation Animations	Create animations using Unity
P.9.4 Add Sounds	Search for royalty-free sounds and enter into unity
P.9.5 CSV implantation	Search of tutorials and implementation for the creation of CSV files
P.9.6 Database implantation	Implement SQL and databases in unity
P.9.7 Collection List	Create the functionality to be able to store the creatures as if it were a <i>Pokédex</i>
P.9.8 Develop Endless Runner	For the lack of more playful part in the app a simple minigame is developed
P.10 Hospital Test	Test in the hospital to see the deficiencies of the application

# Gantt Diagram



(Figure 12) Gantt chart

## 1.9. Resources evaluation

In this section we will explain why we consider that our application is viable. The equipment needed to make it is mainly a PC / Mac, an image editing software such as Adobe Illustrator. Adobe Illustrator [9] is a vector graphics editor developed and marketed by Adobe, Originally designed for the Apple Macintosh, development of Adobe Illustrator began in 1985. Along with Creative Cloud (Adobe's shift to monthly or annual subscription service delivered over the Internet), Illustrator CC was released. The latest version, Illustrator CC 2019, was released in October 2018 and is the 23rd generation in the product line. Adobe Illustrator was reviewed as the best vector graphics editing program in 2018, and the software for the creation of the application Unity, Unity [10] is a cross-platform game engine developed by Unity Technologies, first announced and released in June 2005 at Apple Inc.'s Worldwide Developers Conference as a Mac OS X-exclusive game engine. As of 2018, the engine had been extended to support more than 25 platforms. The engine can be used to create three-dimensional, two-dimensional, virtual reality, and augmented reality games, as well as simulations and other experiences. The engine has been adopted by industries outside video gaming, such as film, automotive, architecture, engineering and construction, plus all the hours to invest, both research and development, therefore, the costs at the development level would be around 6,000 € or 8,000 € in the applications market.

It is an approximate calculation, counting the number of hours used for the project a total of 150 hours, the monthly price of adobe illustrator (creative cloud) that has a price of 24,15 euros, and the license of Unity plus that has a monthly price of 25 euros.

Needless to say that for merchandising purposes, an agreement must be established with the hospitals so that the application can be assigned to those patients that will benefit from it without having to upload it in any particular application market and this way preventing the use of any monetization system and the consequential issues concerning the requirements for these type of web stores.

## 2. SYSTEM ANALYSIS AND DESIGN

Hereunder we will discuss the design, workflow and the interface of our application, as well as the mechanics of the game.

### 2.1. Requirements analysis

In this section we talk about functional and non-functional aspects of our application.

#### 2.1.1. Functionals requirements

Requirement:	The player will have to enter the Ratio and Sensitivity Factor values the first time the application is started.
Input:	Ratio values and sensitivity factor.
Output:	Insulin value that the patient must inject.

Requirement:	The player will have to enter the values Previous Glucose and Carbon Hydrates in the scene of introduction of values.
Input:	Previous glucose and carbohydrate values.
Output:	Insulin value that the patient must inject.

Requirement:	If the previous glucose entered by the player is below 80 or above 250, 2 different alerts appear.
Input:	Previous glucose value.
Output:	Previous Glucose <80 - Alert (take a juice) Previous Glucose > 250 - Alert (use of acetate strip)

Requirement:	The player will have to enter the values for a week and if he has kept the glucose between 80 and 250.
Input:	Previous glucose and carbohydrate value.
Output:	The user gets a reward in the form of collectable and a happy pet

Requirement:	The player will have to fill the values of previous glucose and carbohydrates for a month.
Input:	Previous glucose and carbohydrate value.
Output:	Button to download the values from the table and take them to the medical centre.

Requirement:	The player needs to keep the pet happy
Input:	Play the endless runner
Output:	Increases the happiness bar

Requirement:	Playing endless runner keep Glu alive by dodging enemies
Input:	Arrows up and down
Output:	Character movement points increase

Requirement:	If Glu hits an enemy
Input:	Detect collision
Output:	If lives end, get out of the game

Requirement:	The player needs to see the carbohydrates of each food
Input:	Consult screen food
Output:	Screen with this data

### 3.1.2 Non-functionals requirements

We must point out that currently the application is only available for Android and therefore, for iOS we should wait. In addition, due to the lack of time dedicated to the implementation, the application does not adapt to resolutions for tablets, although its installation is possible in these devices.

Our application focus only on children between 6 and 12 years old with type 2 diabetes.

## 2.2. Game mechanics

In this section all of the mechanics and actions that the player can carry out with the application have been detailed, explaining the movements of our main character, the actions that can be carried out as well as the collectible elements that can be found in the application

### 2.2.1. Playability

The mechanics are quite simple, on the one hand, we have the data entry: when starting the application for the first time, we will be asked for the sensitivity factor and the ratio for each of the meals of the day. This will help us calculate the insulin that the player will have to inject to maintain glucose levels between 80 and 120 and keep Glu happy.

Once introduced the initial values, (values that will provide the player in the medical centre), a small tutorial with our pet will appear, which will guide us through the interface, and will indicate where we have to enter the different values, and the different ones sections of our application. As the weeks go by, we will unlock rewards in the form of new creatures, collectibles. To unlock them, you only have to press the egg-shaped button that will appear at the bottom of the main screen. By clicking on the egg, the app will take us to the screen of opening eggs, where we will unlock the new creatures.

To keep Glu happy, we can also play with it, by clicking on the play button

### 2.2.2. Game's flow

In this section of the document, the steps to be followed in the application are detailed from the start until the end of its use. When the child starts the application for the first



time, the start screen will appear where we will have the start button and the options button of the application.

Once the child clicks on the start button, we will have a small presentation, where we will see for the first time Glu, our main character and at the same time guide in the application, who will ask us to introduce the ratio values and the sensitivity factor that the medical centre will have provided us in order to obtain the insulin values.

After these steps, the child will be on the main screen, where we find the recipe button, the button where to see the unlocked rewards, the button to play at endless runner game and in the central area the button to enter values, this being a key button.

In the lower part of this screen we find a small stage where we will have Glu, the happiness bar and later the rest of the creatures that we unlock will appear

Once we click on the enter values button, the app will take us to the data entry screen. Thanks to the introduction of these values, the application will calculate the insulin dose that the child should inject.

When the child enters the glucose values, if they are below 80 or above 250, an alert will appear (see Figure 12 and 13) where Glu will notify us that we have to take a juice, if the value is below 80, and if it is above 250, you will have to use an acetate strip.



(Figure 12) Warning Scene

(Figure 13) Warning Scene

Once we finish the day, and if the child has kept the glucose values between 80 and 120, this having our pet happy, the player can see the happiness bar filled, an egg will appear at the bottom of the main screen

Clicking on it will take us to the egg opening screen, where the child should click on it, in order to discover our first collectible, it will appear with Glu at the bottom of the main screen.

Being on the main screen, we can observe the collectible button. If we click on it, we will see the creatures that we have unlocked already.

We can also click on the button to play the endless runner to fill the Glu happiness bar. In this main screen we also find the recipe button, when we click on it, it will take us to the screen where we can see different dishes and the amount of carbohydrates that each of them has.

### 2.2.3. Endless Run Game

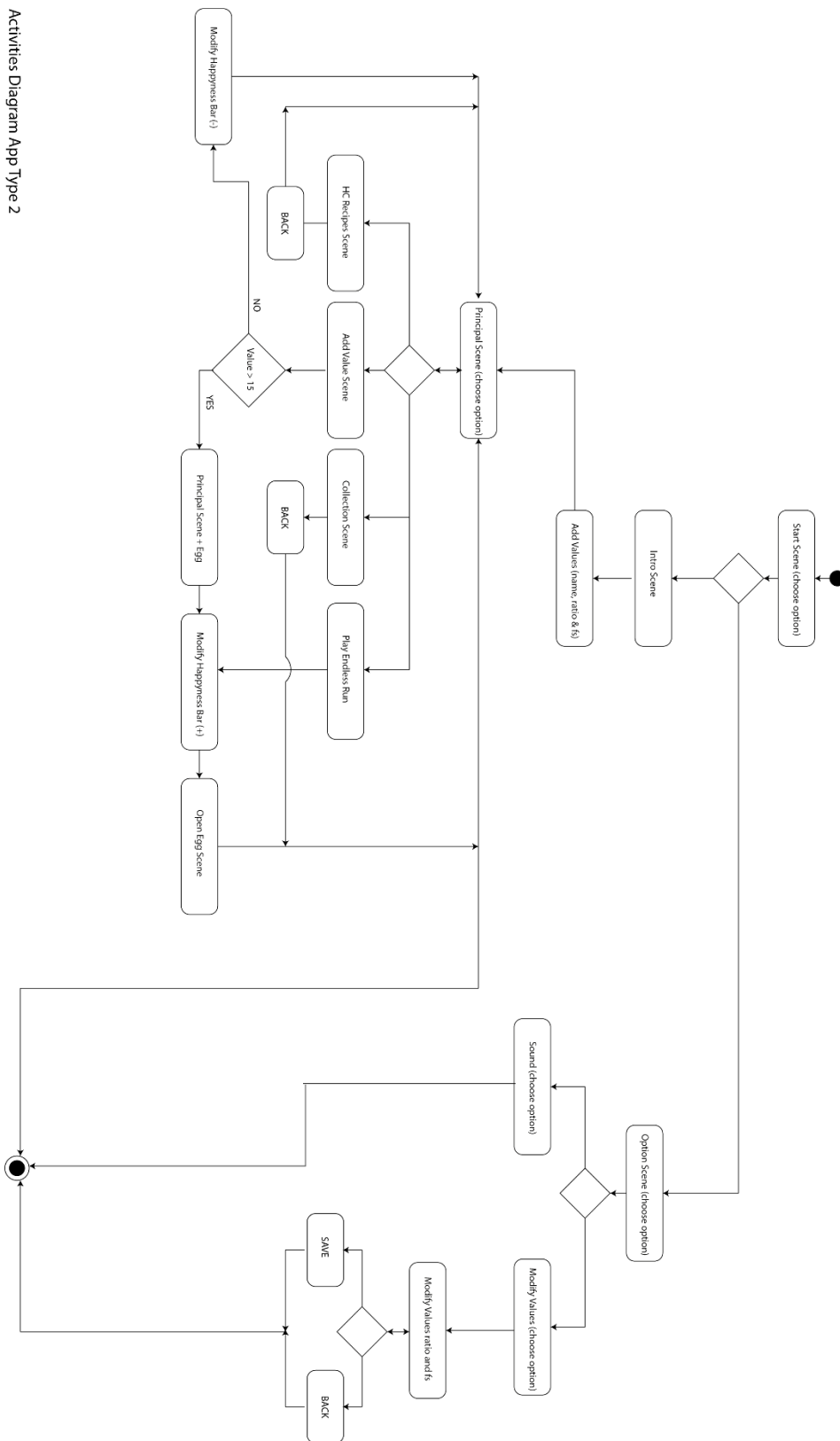
Endless runner games are defined by two things: The player-controller avatar cannot stop its forward momentum, and it has feet. Games that reward or celebrate speed such as ones in the Sonic franchise are spiritually similar, but subscribe to a different philosophy. It also excludes games where the player would ideally always be running such as *Mirror's Edge*, as stopping does not necessarily introduce a fail state. This also excludes games that use the mechanic of forced motion only during sections of gameplay, such as the Crash Bandicoot escape sequences, or the brain worm sequences in *Limbo*.

Generally presented as a 2D side-scroller, controls are generally simple, focusing on actions such as jumping over obstacles, although melee and gunplay are not out of the question. Success or progress is usually measured in distanced travelled without dying or crashing.

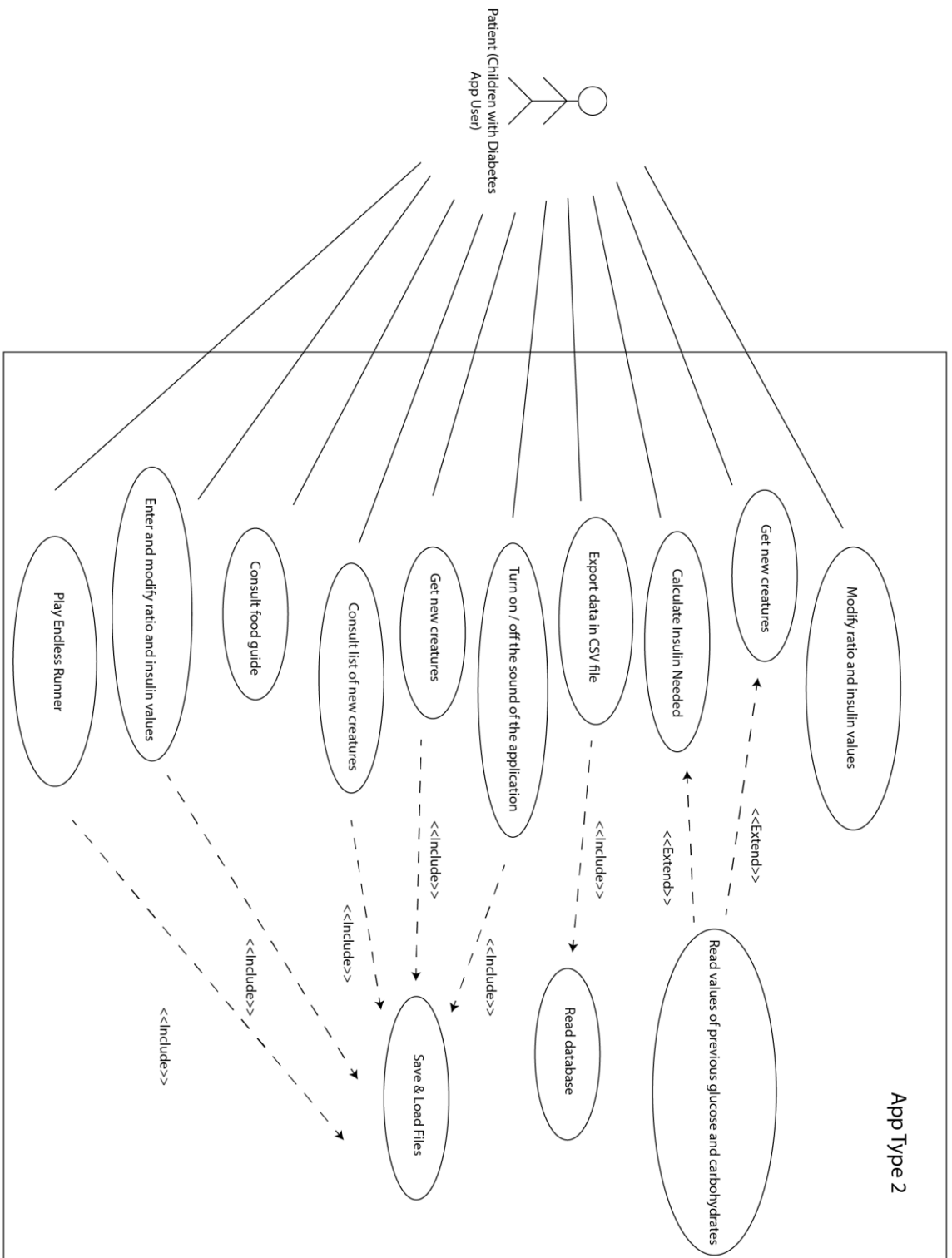
When we click on the play button, it will take us to the game screen, here we will have to click on the up and down arrows that we find to avoid the different enemies that appear on the screen, in order to increase our score and fill the bar Glu happiness for playing with him

## 2.2.4. Activity diagram and case use diagram

In this section we find the activity chart (Figure 14) and case use chart (Figure 15)



(Figure 14) – Activity chart.



(Figure 15) – Case use chart

### 2.2.5. Interaction and controls

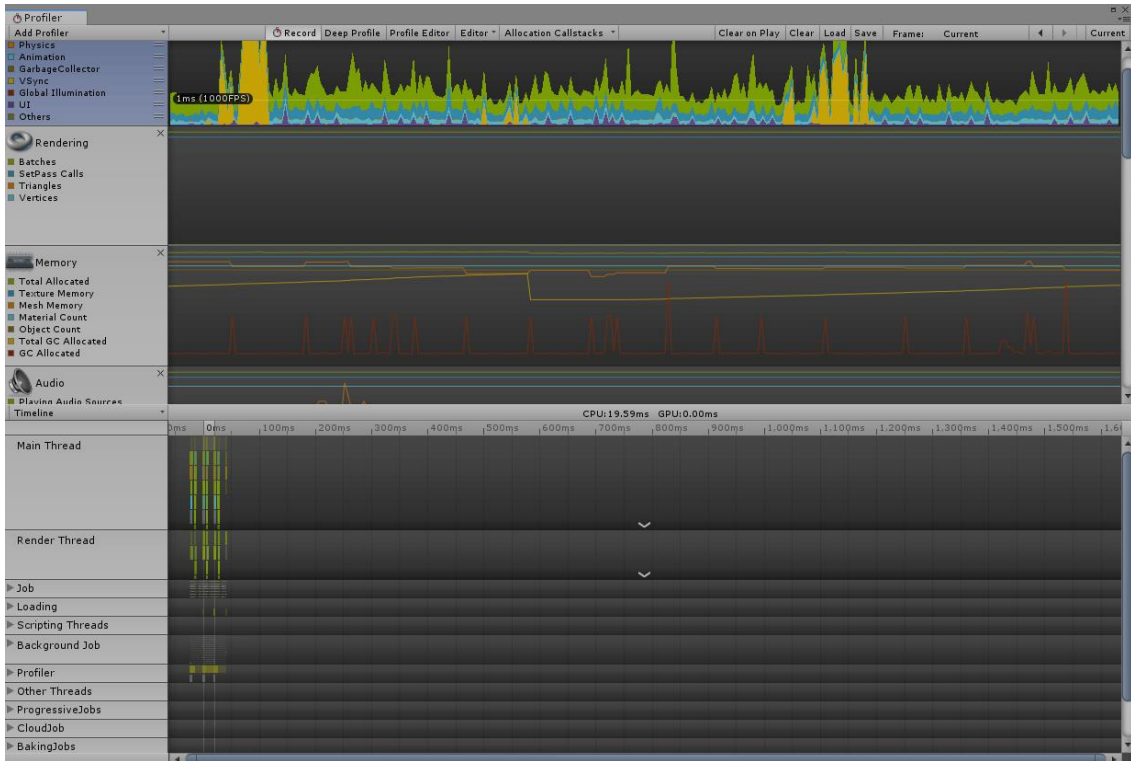
As we know, the main form of control for mobile devices is the touch screen, so video games designed specifically for them adapt to this form of input. Interact with different elements of the stage playing on them. The way we have to interact with the application will be through the different menus and buttons that we will find in it. On the other hand, we can interact with our main character by clicking on it.

The buttons with which we can interact are:

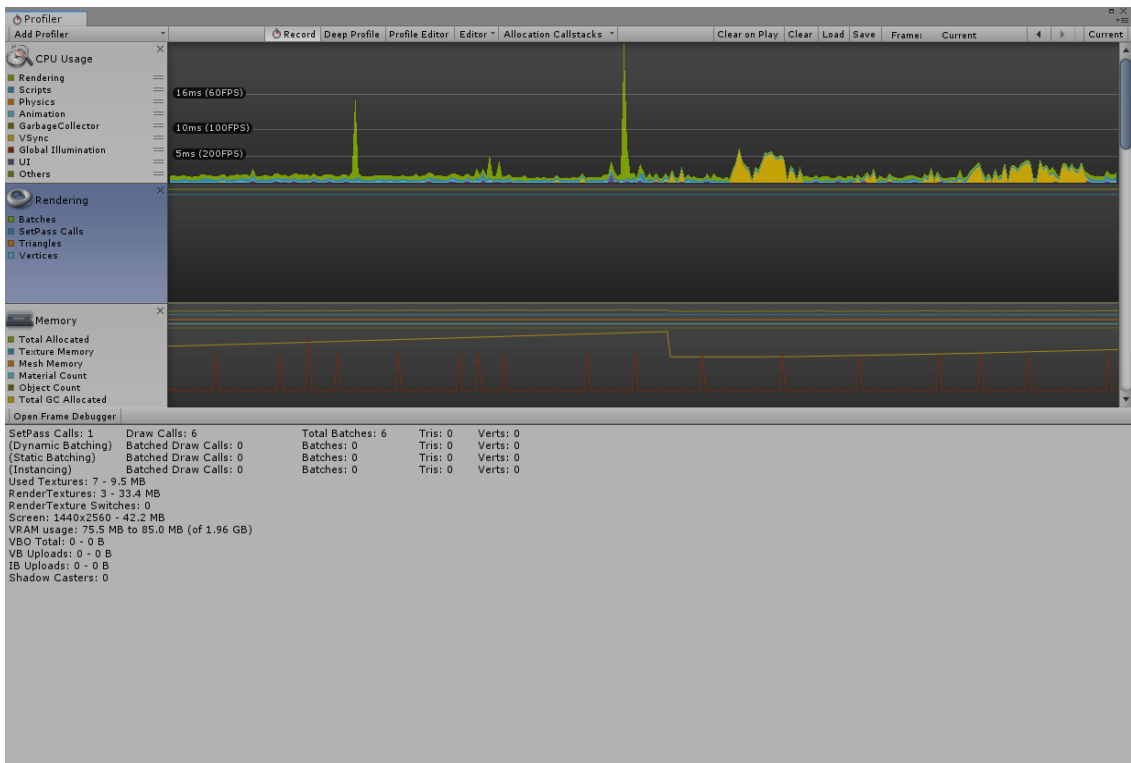
- Values entry button: Which takes us to the values introduction screen.
- Play button: Which will take us to the game Endless Runner
- HC Button: It takes us to the screen of the meals with the different carbohydrates of each of them
- Buttons of options: Activate and Deactivate sound of the app and Modify the introduced initial values
- Inside the game endless runner: Arrows up and down to go dodging the enemies and increase our score to increase our happiness bar
- Egg button: The one that will help us to open new collectibles to complete our collection
- Arrows return: To return and make the transition between scenes, it is worth mentioning here that between scene and scene a fade function has been programmed for each one of them.

## 2.3. System architecture

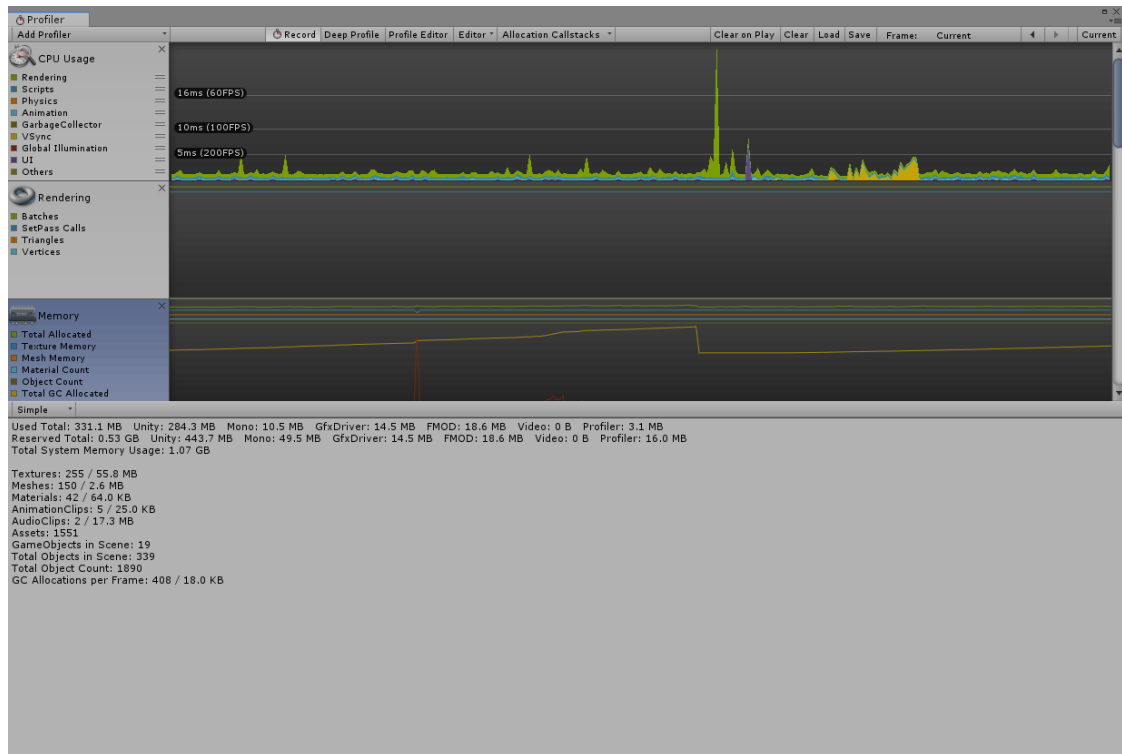
The minimum requirements for the correct operation of the application are: A smartphone with the Android OS version (Marshmallow 6.0), a free space in the phone memory exceeding 55 Megabytes and a minimum of 2Gb of RAM, although the application is developed in 2D , these requirements are those required by Unity for proper operation, below are the captures of the Unity profiler with the use of CPU (Figure 16), Memory (Figure 17) and the Rendering Engine (Figure 18) while we have executed our App



(Figure 16) Profiler CPU Unity



(Figure 17) Profiler Rendering Unity



(Figure 18) Profiler Memory Unity

## 2.4. Interface design

In this section we will proceed to talk about the sections that make up our interface. First, we would find the home screen. On this screen appears the title, the start buttons and options. When pressing options, we find the button to activate or deactivate the sound of the application, in addition to a button that allows us to modify the different ratios of the meals and the sensitivity factor.

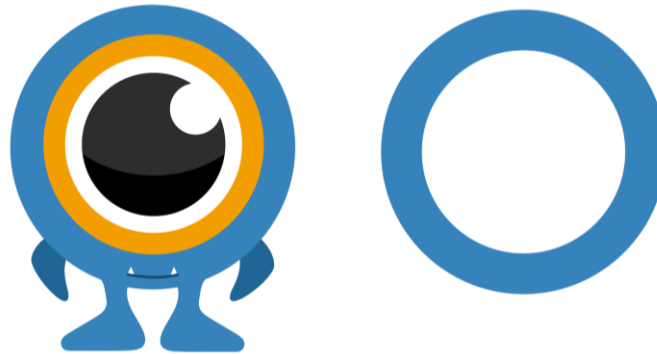
### 2.4.1. Characters

In the game on the one hand we will have the main character, who will guide us through the interface, and on the other hand we will find more creatures as collectables.

#### 2.4.1.1. Glu

Inspired by the international symbol for diabetes, the main character in our application is “Glu” (Figure 19). He will be in charge of guiding the user through the application as well as interacting with the user. The aim is to keep “Glu” happy with his glucose levels within the established limits.

The circle symbolizes life and health, its meaning is very positive, in many cultures. The colour blue represents the sky that unites all nations and is the colour of the United Nations flag. The blue circle embodies the unity of the international diabetes community.

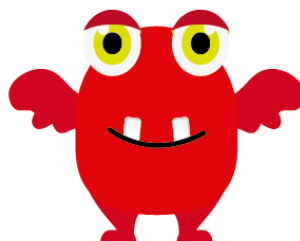


(Figure 19) Glu and the global symbol of diabetes.

#### 2.4.1.2. Collectables

The collectibles are another important part of the game as they represent the way children are rewarded and motivate them to continue entering the required data, week after week, in the application.

Once the child fulfills a series of requirements, these characters will appear in the shape of eggs on the lower part of the main screen. To open an egg and unlock the creature, the user must click on the egg and after doing so will then be taken to an "open egg" screen, where after clicking, a new creature will appear.



(Figure 20) First collectible

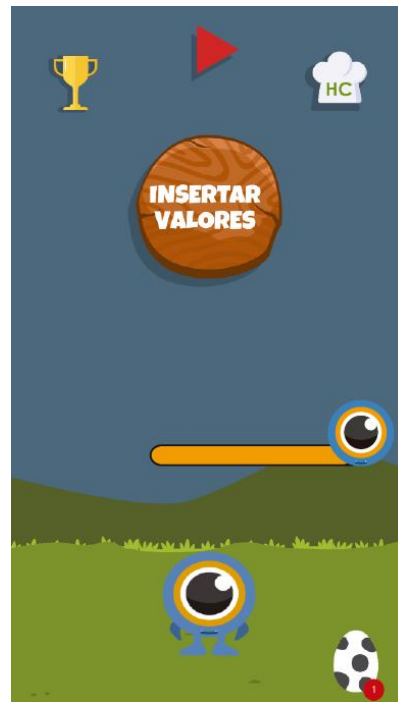


### 2.4.1.3. Main screen

The interface had different changes at the same time as progress was made in the game, at first, we found a coarser design (Figure 21) and finally it was modified, just like the main character stylized (Figure 22).



(Figure 21) Old Interface



(Figure 22) New Interface

As we can see the introduction of Values, in the first version was made directly on the main game screen.

Once the values are entered, we find the main screen

- In the upper part we will have four buttons, one that will take us to the screen of introduction of values of glucose levels prior to meals and rations consumed, with which we will obtain the insulin value that the child should inject. The button of the rewards, will take us to the screen with the list of collectibles, where we will have a list with all the creatures that we have unlocked, the play button for play at endless run game to keep Glu, and finally the carbohydrate button, that will take us to the food screen where they will be available. list of foods with the carbohydrates of each of them. If we fulfil a series of requirements in this screen will appear the button in the form of an egg, which will take us to the screen of opening collectibles.

- In the lower part we will find our main character with new creatures as we go forward

#### 2.4.1.4. Screen introduction of values

This is one of the most important screens in the game, it is divided into 3 parts: at the top we will have the current date, in the next division we will find the time of day in which we are, in the next we will have a field to enter the numerical values of previous glucose and carbohydrates. Thanks to the introduction of these values, the application will calculate the insulin dose that the child should inject. Finally, at the bottom we will find Glu, carrying out the actions of breakfast, lunch, lunch, afternoon snack or dinner. (Figures 23 to 27).



[Figure 23]  
Breakfast

(Figure 24)  
Lunch

(Figure 25)  
Lunch

(Figure 26)  
Afternoon snack

(Figure 27)  
Dinner

#### 2.4.1.5. Open collectibles screen

We find an egg (Figure 28) centred on the screen, on which we will have to press in order to unlock new creatures as collectables and in this way motivate the child every day to introduce values and maintain their glucose levels within limits



(Figure 28) Button that takes us to the screen of opening collectibles

#### 2.4.1.6. Collectables list screen

This screen shows the collectibles that we have already unlocked and those that we still need to unlock (Figure 29)



(Figure 29) Display with collectible list

#### 2.4.1.7. Food screen

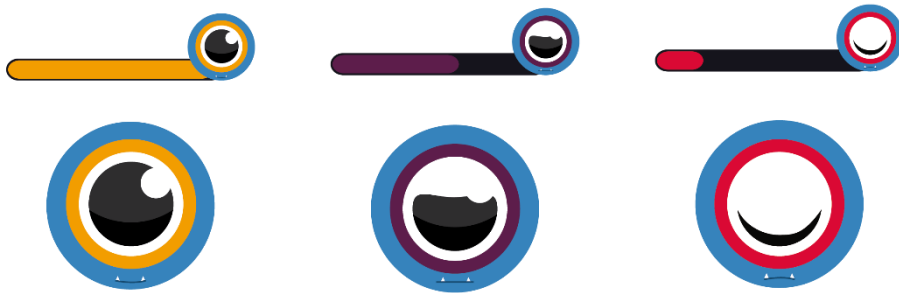
In this screen (Figure 30) we will find a list of foods, with the name of the food, the size of the ration, a drawing of the size of the approximate ration and the values of weight, carbohydrates, and the number of rations.



(Figure 30) Food screen

#### 2.4.1.8. Happiness Bar

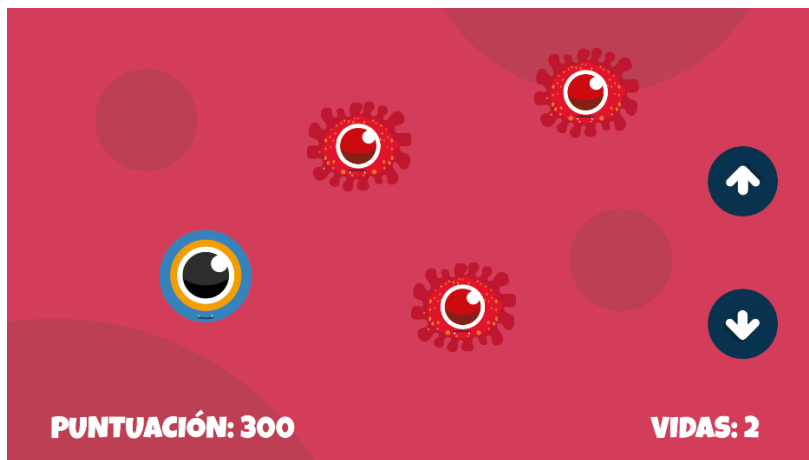
We can see on the main screen a happiness bar (Figure 31), this will change if our glucose values are very low or very high, we can also improve the happiness bar by playing with Glu to the included game.



(Figure 31) Happiness bar

#### 2.4.1.9. Endless Run Game

We can see (Figure 32) below the screen of the game, in which we find our player, the enemies and a score, when we play the game, we will also happy our pet to play with her.



(Figure 32) Endless Game

### 3. WORK DEVELOPMENT AND RESULTS

In the following breakdown of sections, we will proceed to describe each of the steps that have been carried out during the development of this application.

#### 3.1. Saved the state of the game

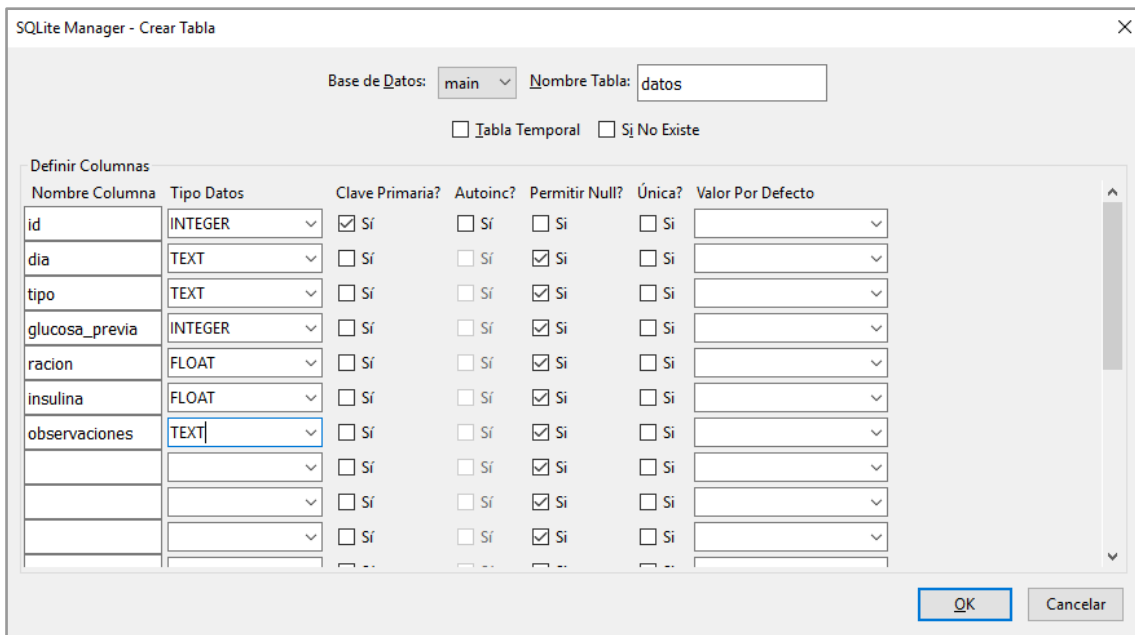
This was a complex task, since Unity has PlayerPrefs, which allow to save and load int, float and string no matter what platform we are working on but its great limitation is that are designed to store one variable at a time and save and load several variables can become annoying thing to do. For this reason, we finally use the Notification Center script, in order to perform the saving tasks, in this state of the game we store the values of Ratio and Sensitivity Factor, which can be modified in options of the application, the name of the patient, the amount of days that have passed to enable or disable the rewards screen and the saving of activation or deactivation of the different collectibles.

#### 3.2. Use databases in Unity

For the creation of databases in Unity we use the DB Browser for Windows program on the one hand. With it we can create the tables and fields necessary for our application.

In Unity we will have to install SQLite through the libraries for Android and creating the directory "StreamingAssets" for its correct functioning. The main table (Figure 33) of our database will store the necessary values to generate the table that we will provide to the hospital. The necessary fields of this database are:

- Id: as a primary key, to store the data in an orderly manner,
- Day: We store the current date
- Type: Where the value of the moment of the day is stored
- Previous Glucose: We store the value of the glucose that we have at that moment.
- Ration: The rations we have consumed
- Insulin: Calculation of insulin value thanks to all stored values
- Remarks: This field was created, but in the end it has not been used and is reserved for future work in the application



(Figure 33) Patient data table created in DB Browser

In the beginning we were going to store the data of Name, Ratio and Sensitivity Factor in another table, but finally, as those values are always the same, in order not to increase the size of the database and create repeated data we ignore the use of it.

There is not much information on how to use databases in Unity, and it was quite complicated to make SQL work correctly, apart from that, different SQL command tests were performed that can be found at the end of the memory.

### 3.3. Scene handling

A simple function (Figure 34), we quickly and efficiently switch between scenes

```

1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4. using UnityEngine.UI;
5. using UnityEngine.SceneManagement;
6.
7. public class SceneManagerScript : MonoBehaviour {
8.
9.     public Image fade;
10.
11.
12.     public void CambioEscena (string es)
13.     {
14.         fade.CrossFadeAlpha (1, 1, true);
15.         StartCoroutine (ActivoFade (es));
16.     }

```

```

17.
18.     IEnumerator ActivoFade(string e)
19.     {
20.         yield return new WaitForSeconds(1);
21.         SceneManager.LoadScene(e);
22.     }
23.
24. }

```

(Figure 34) Function scene handling

### 3.4. Sound

The sounds were searched in different sound banks with Creative Commons license in this case the application has to be commercialized, we do not have legal problems with the different authors and their works

### 3.5. Export of variables

We have tried to carry out this task using the iTextSharp [] plugin and it was not possible, after investigating a bit more I have found the plugin for Visual Studio SharpPDF, but there are problems with exporting on Android devices.

So, I opted for the exporting files in CSV, which is a very versatile format and can be easily edited by any spreadsheet program.

### 3.6. Food screen

For the food screen, we had to use a PDF provided by the General Hospital, currently we only have 3 foods introduced in the app, but they will be expanded in future work.

### 3.7. States of the day

For the days states we use the function (check Time) with DateTime.Now.Hour [Figure 34] and DateTime.Today to take the complete date, but then we have to format it to introduce it in Unity's .text (Figure 35) to thus change between all the states of the day.

```

1.     public Int32 hora = 0;
2.
3.     public void comprobarHora()
4.     {
5.         hora = DateTime.Now.Hour;

```

```
6. }
```

[Figure 34] Function get current system time

```
1. public DateTime today = DateTime.Today;
2. fecha.text = today.ToString("dd/MM/yyyy");
```

(Figure 35) Function get actual day

This section has a lot of work, both design, animation and sound, as there are different backgrounds, animations and sounds for each of the moments, we have to check with the warnings, when we have insulin below 80 or above 250 grams.

### 3.8. Develop Endless Runner

We have developed a small game of the genre Endless Runner, where Glu, must go dodging the enemies that come out in a random way in an infinite scenario, thanks to this game we will raise the happiness bar of Glu.

Among the most important functions we find the movement of the player (Figure 36) along with the control of the lives when it collides, the spawner (Figure 37) and the detection of the enemies (Figure 38).

```
1. private void Update ()
2. {
3.
4.     if (health <= 0) {
5.         spawner.SetActive(false);
6.         restartDisplay.SetActive(true);
7.         Destroy(gameObject);
8.     }
9.
10.    healthDisplay.text = health.ToString();
11.
12.    transform.position = Vector2.MoveTowards(transform.
13.    position, targetPos, speed * Time.deltaTime);
14.
15.    if (Input.GetKeyDown(KeyCode.UpArrow) && transform.
16.    position.y < maxY) {
17.        camAnim.SetTrigger("shake");
18.        Instantiate(moveEffect, transform.position,
19.        Quaternion.identity);
20.        targetPos = new Vector2(transform.position.x,
21.        transform.position.y + increment);
22.    } else if (Input.GetKeyDown(KeyCode.DownArrow) && t
23.    ransform.position.y > minY) {
24.        camAnim.SetTrigger("shake");
```



```

20.         Instantiate(moveEffect, transform.position,
Quaternion.identity);
21.         targetPos = new Vector2(transform.position.x,
transform.position.y - increment);
22.     }
23. }
24. }

```

(Figure 36) Player

```

1.     private void Start()
2.     {
3.         timeBtwSpawns = startTimeBtwSpawns;
4.     }
5.
6.     private void Update()
7.     {
8.         if (timeBtwSpawns <= 0)
9.         {
10.            int rand = Random.Range(0,
obstacleTemplate.Length);
11.            Instantiate(obstacleTemplate[rand],
transform.position, Quaternion.identity);
12.            timeBtwSpawns = startTimeBtwSpawns;
13.            if (startTimeBtwSpawns > minTime) {
14.                startTimeBtwSpawns -= timeDecrease;
15.            }
16.        }
17.        else {
18.            timeBtwSpawns -= Time.deltaTime;
19.        }
20.    }

```

(Figure 37) Spawner

```

1.     void Update () {
2.         transform.Translate(Vector2.left * speed * Time.deltaTime);
3.     }
4.
5.     private void OnTriggerEnter2D(Collider2D other)
6.     {
7.         if (other.CompareTag("Player")) {
8.             other.GetComponent<Player>().health--;
9.             other.GetComponent<Player>().camAnim.SetTrigger("shake");
10.            Instantiate(effect, transform.position,
Quaternion.identity);
11.            Destroy(gameObject);
12.        }
13.    }

```

(Figure 38) Enemy

## 4. RESULTS AND FUTURE WORK

In this section we will talk about the results, changes throughout the project and final result, we also find all the possibilities for expansion and modifications that should or may be made in the future.

### 4.1. Results

After many hours of work in questions of study, design and development for the application we have achieved satisfactory results, although throughout the project we have encountered different problems, such as the ones mentioned above export to PDF, problems to save the states with Unity's PlayerPrefs limitation and some other detail that could have been developed more efficiently. But in general, we have fulfilled all the objectives in my opinion, achieving a useful and functional application.

In this link you can find the apk file, for installation on an Android device, to test the application.

(<https://drive.google.com/file/d/1rCVSr0yybg4rACOm0eteAI3fVIKXI1J1/view?usp=sharing>)

### 4.2. Testing

We perform different tests in the application:

First, a usability test. To carry out this test, the application does not have to be completely finished, it is advisable to carry out this test in the initial stages of the design and development process, although in our case the application was almost finished and it served us for future work.

This first of usability was made on an Android phone, with a 7-year-old boy who has diabetes and usually fills in the table that is provided to him at the medical centre, we had to listen patiently to the suggestions and problems that the application presented in the state where he was, taking note of the presented problems of usability.

This test was divided into 3 stages:

In the first place, the preparation stage, where the participants were chosen, in our case, we only had a single patient who was shown the application and we had to see if he understood the application, what it was for and how it worked. A part of it was asked to enter the values that usually points in your Excel table.

You were also asked different questions to get useful information about the use of the application or improvements you might find.

The next phase of the usability test is the execution stage. For this, we started the application again and we were actively asking what he thought about each of the screens. Problems of slowness were found between some scenes that we improved in the final version.

In this phase the child was explained that there was no problem in saying what he thought, since it was useful to improve the application

Finally, we have the analysis test where the notes obtained during the test session were reviewed and the different solutions began to be considered.

The next test that was performed was the 5 second test, a test that provides a lot of information about the design of the application in the content hierarchy.

This test consisted of explaining a concrete situation, in our case, modifying the values of ratio and sensitivity factor in the options menu in 5 seconds, was asked everything he remembered from what he had seen, to see more things striking and see if it coincided with what was intended in that section.

To finish the tests and for lack of time, we gather different users, in our case 3 friends, and install the application on their phones, passing the .apk by Google Drive to perform the so-called guerrilla test [11] to see how the application behaves and correct all errors or modifications that could be made in the application in a fast manner.

### 4.3. Evaluation

We have made an appointment with Monica Cube for the evaluation of the application, but for the month of August, since she is currently absent. Although if we talked about WhatsApp and we sent a video of the application, explaining the operation of the application. She has been happy with the results, although at the time of presenting the data in the CSV she thought that some data is needed as a table of observations, and she saw incomplete the section of the meals, although it is thought to extend this part in a future work

For the appointment, he will bring 3 patients of his to directly test the application with them and see more things to improve.

### 4.4. Future work

The application as we discussed in the document can be expanded, and in future updates we will find new tools and aspects that could be developed in the future, we will describe everything below.

More Tools for the control of Diabetes.

Insulin sensitivity factor calculator: To make this calculation, the average of 3 days is necessary, for this we use the following formula:

$$\text{"INSULIN DAILY TOTAL DOSE"} = \text{Total Dose of BASAL Insulin Units} + \text{Rapid Insulin Total Dose in BOLUSES"}$$

Followed by:

$$\text{"SENSITIVITY FACTOR (or correction factor)} = 1700 / \text{TOTAL INSULIN DOSE"}$$

“Bolo corrector” calculator: To perform this calculation is necessary the sensitivity factor, which we have previously calculated and stored to use in the following formula

$$\text{"EXTRA INSULIN UNITS to put} = \text{REAL Glycemia} - \text{Glycemia OBJECTIVE} / \text{SENSITIVITY FACTOR}$$

Minigames

Different mini-games will be developed within the application so that the child gets used to the use of the app, for example the classic Arkanoid.

### Changes in the interface

The different animations of the main character as well as the different collectables must be improved

### Extension of rewards

Increase the number of creatures, since they are currently scarce. and that will be generated procedurally, adding to each of them a small description.

These collectibles may be repeated, special collectibles such as silver or gold may also appear.

### Screen food

Add more food on the food screen.

### PDF export

Export the PDF [12] database being a data visualization system more protected than the current system.

## 5. CONCLUSIONS

After more than 180 hours of work, and combining my work in the office as a graphic designer and my work as a freelancer, I have managed to make an application that seems satisfactory and quite complete.

The whole idea not to perform a typical work in a videogame degree, since mostly games are presented, and here although we have some sections of gamification, it is not the main functionality of the application.

I have achieved new skills in the development with Unity, working with databases and creating files to save the different states without the need for Unity PlayerPrefs, which, although it is a very efficient tool, has several limitations.

In the application and memory I have touched on concepts of different subjects, such as Programming II (C #), Videogame Engines (Unity and C #), Software Engineering (Diagrams of activities and use cases), 2D Design and Videogame Art (Design general of the application and finally the subject of Databases (Database of the application and queries to it).

For all this, I have been quite happy with the work, although if it is true, there are many that will be extended in the future, since the General Hospital is interested in the application, to be able to use it with their patients, once the corresponding tests.

## 6. BIBLIOGRAPHY

- [1] Fundación Diabetes [on line] Available in: <http://www.fundaciondiabetes.org>  
[Accessed : 18-05-2019]
- [1] Digital Games for Type 1 and Type 2 Diabetes: Underpinning Theory With Three Illustrative Examples [on line] Available in: <https://games.jmir.org/2015/1/e3/>  
[Accessed: 25-03-2019]
- [1] Videogames and Diabetes [on line] Available in:  
<http://www.diabetesincontrol.com/type-2-diabetes-patient-benefit-from-video-games/>  
[Accessed 30-03-2019]
- [2] Games and Simulations for Diabetes Education - WCER [online] Available in:  
[https://wcer.wisc.edu/docs/working-papers/Working\\_Paper\\_No\\_2011\\_01.pdf](https://wcer.wisc.edu/docs/working-papers/Working_Paper_No_2011_01.pdf)  
[Accessed: 28-03-2019]
- [2] Gamification [on line] Available in:  
<http://www.seanogle.com/entrepreneurship/gamification>  
[Accessed: 02-05-2019]
- [3] Gamification [on line] Available in: <https://www.locationrebel.com/gamification/>  
[Accessed: 02-05-2019]
- [4] Rewards and game mechanics [on line] Available in:  
<http://techcrunch.com/2010/08/25/scvngr-game-mechanics/>  
[Accessed: 02-05-2019]
- [5] Big data information [on line] Available  
<https://channels.theinnovationenterprise.com/articles/even-ski-resorts-are-benefiting-from-the-big-data-explosion>  
[Accessed: 03-05-2019]
- [5] Big data information [on line] Available  
<http://www.datasciencecentral.com/profiles/blogs/the-7-most-unusual-applications-of-big-data-you-ve-ever-seen>  
[Accessed: 03-05-2019]
- [6] MySugr information [on line] Available in: <http://www.waxnasbc.com/aplicacion-de-monitorizacion-de-la-diabetes-tipo-1-mysugr-publica-desafios-para-fomentar-el-registro/>  
[Accessed : 30-03-2019]
- [7] GlucoZor information [on line] Available in: <https://www.airliquide.com/glucozor>  
[Accessed: 30-03-2019]

[8] Spiral Model information [on line] Available in:

[https://en.wikipedia.org/wiki/Spiral\\_model](https://en.wikipedia.org/wiki/Spiral_model)

[9] Adobe Illustrator [on line] Available in:

[https://en.wikipedia.org/wiki/Adobe\\_Illustrator](https://en.wikipedia.org/wiki/Adobe_Illustrator)

[Accessed: 16-05-2019]

[10] Unity 3D [on line] Available in: [https://en.wikipedia.org/wiki/Unity\\_\(game\\_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine))

[Accessed: 16-05-2019]

[11] Guerrilla test [on line] <https://blog.maze.design/maze-guerilla-testing/>

[Last Accessed 18-06-2019]

[12] SharpPdf information in unity [online] Available in:

<http://www.devindia.biz/unity-pdf-generation-with-sharppdf-plugin/>

[Accessed: 20-05-2019]

[12] Creating Pdf in unity with C# [on line] Available in:

<http://www.rasteredge.com/how-to/csharp-imaging/pdf-creating/>

[Accessed: 23-05-2019]

[13] Documentation Unity [on line] Available in:

<https://docs.unity3d.com/es/current/Manual/index.html>

[Last Accessed 14-06-2019]



## 7. APPENDIX A – SOURCE CODE

### Connection + Insert BD on both PC and Android.

```
1. using System;
2. using System.Collections;
3. using System.Collections.Generic;
4. using System.Data;
5. using System.IO;
6. using Mono.Data.Sqlite;
7. using UnityEngine;
8. using UnityEngine.UI;
9.
10. public class Insert : MonoBehaviour
11. {
12.     string rutaDB;
13.     string strConexion;
14.
15.
16.     string DBFileName = "datosPaciente.db";
17.
18.     IDbConnection dbConnection; // Crear conexión
19.     IDbCommand dbCommand; // Comandos
20.     IDataReader leerdatos; // Para poder leer
21.
22.
23.     void Start ()
24.     {
25.         AbrirDB ();
26.         CerrarDB ();
27.     }
28.
29.
30.     void InsertarDatosPacientes ()
31.     {
32.         ComandoINSERT ("");
33.     }
34.
35.     void AbrirDB ()
36.     {
37.         // Si es PC
38.         if (Application.platform == RuntimePlatform.Windows
39. Editor)
40.         {
41.             rutaDB = Application.dataPath + "/StreamingAssets/" + DBFileName; //Se divide la ruta para las diferentes
42. plataformas
43.         }
44.         // Si es Android
45.         else if (Application.platform == RuntimePlatform.Android)
46.         {
47.             rutaDB = Application.persistentDataPath + "/" +
48. DBFileName;
49.             // Comprobar si está en persistentData
50.             if (!File.Exists(rutaDB))
51.             {
```

```

51.         WWW
    loadDB = new WWW("jar:file://" + Application.dataPath + "!/asset
s/" + DBFileName); //Ruta en formato generico de android (Java)
52.         while (!loadDB.isDone)
53.             {
54.             }
55.             File.WriteAllBytes(rutaDB, loadDB.bytes);
56.         }
57.     }
58.
59.     // Crear y abrir conexión
60.
61.     strConexion = "URI=file:" + rutaDB;
62.     dbConnection = new SqliteConnection(strConexion);
63.     dbConnection.Open();
64.
65.     }
66.
67.     void CerrarDB()
68.     {
69.         // Cerrar conexiones
70.         leerdatos.Close();
71.         leerdatos = null;
72.         dbCommand.Dispose();
73.         dbCommand = null;
74.         dbConnection.Close();
75.         dbConnection = null;
76.
77.     }
78.
79.     void ComandoINSERT(string dato)
80.     {
81.         // Crear consulta
82.         dbCommand = dbConnection.CreateCommand();
83.         string sqlQuery = String.Format("insert into
datos_pacientes(glucosa_previa) values ({0})", dato);
84.         dbCommand.CommandText = sqlQuery; // Aquí pasamos
la consulta
85.         dbCommand.ExecuteScalar();
86.
87.         dbCommand.Dispose();
88.         dbCommand = null;
89.         dbConnection.Close();
90.         dbConnection = null;
91.
92.
93.     }

```

## Use Notification Center, with AddObserver and PostNotification

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4.
5. public class AbrirHuevo : MonoBehaviour
6. {
7.
8.     public bool nuevaCriatura;
9.
10.     // Start is called before the first frame update
11.     void Start()
12.     {
13.         NotificationCenter.DefaultCenter().AddObserver(this
14. , "NuevaCriatura");
15.         CargarGuardar.cargarGuardar.cuentaDias = 0;
16.         CargarGuardar.cargarGuardar.Guardar();
17.     }
18.
19.     // Update is called once per frame
20.     void Update()
21.     {
22.         NotificationCenter.DefaultCenter().PostNotification
23. (this, "NuevaCriatura");
24.     }
25.     public void NuevaCriatura(Notification notification)
26.     {
27.         CargarGuardar.cargarGuardar.criaturaUno = true;
28.         CargarGuardar.cargarGuardar.Guardar();
29.         //Debug.Log(nuevaCriatura);
30.     }
31. }
```

## Glu Animated text

```
1. using UnityEngine;
2. using UnityEngine.UI;
3. using System.Collections;
4.
5. public class AnimatedText : MonoBehaviour
6. {
7.     public float letterPaused = 0.025f;
8.     public string[] message;
9.     public Text textComp;
10.    public GameObject enter;
11.    public GameObject nombre;
12.    public GameObject continuar;
13.    public GameObject continuarNombre;
14.    public GameObject rat;
15.    public GameObject fac;
16.
17.
18.
19.    private bool checkNext = false;
20.    private int lineaActual = 0;
21.    private int contador = 0;
22.    private bool cambio = false;
23.
24.
25.    public void FuncionCambioBool ()
26.    {
27.        cambio = !cambio;
28.        //NotificationCenter.DefaultCenter().PostNotificati
on(this, "Incremento");
29.
30.    }
31.
32.    void Start ()
33.    {
34.        textComp.text = "";
35.        continuar = GameObject.Find("Boton");
36.        nombre = GameObject.Find("Nombre");
37.        rat = GameObject.Find("Ratio");
38.        fac = GameObject.Find("Factor");
39.
40.
41.
42.        StartCoroutine (TypeText (lineaActual));
43.        continuar.SetActive (false);
44.        nombre.SetActive (false);
45.        rat.SetActive (false);
46.        fac.SetActive (false);
47.    }
48.
49.
50.    public void GuardarDatos ()
51.    {
52.        NotificationCenter.DefaultCenter().PostNotification
(this, "GuardarNombre");
53.        NotificationCenter.DefaultCenter().PostNotification
(this, "GuardarRatio");
54.        NotificationCenter.DefaultCenter().PostNotification
(this, "GuardarFactor");
```

```

55.         NotificationCenter.DefaultCenter().PostNotification
        (this, "SaltaTutorial");
56.     }
57.
58.
59.
60.     private void Update ()
61.     {
62.
63.
64.         if (checkNext)
65.         {
66.             enter.SetActive (true);
67.
68.         }
69.         else
70.         {
71.             enter.SetActive (false);
72.         }
73.
74.         if (lineaActual < message.Length - 1 && checkNext &&
cambio == true && contador != 3 && contador != 5 && contador !=
7)
75.         {
76.             lineaActual++;
77.             contador++;
78.             checkNext = false;
79.             textComp.text = "";
80.             cambio = false;
81.
82.             StartCoroutine (TypeText (lineaActual));
83.         }
84.         if (contador == 3)
85.         {
86.
87.             if (cambio == true)
88.             {
89.                 nombre.SetActive (false);
90.                 lineaActual++;
91.                 contador++;
92.                 textComp.text = "";
93.                 cambio = false;
94.                 StartCoroutine (TypeText (lineaActual));
95.             }
96.             else
97.             {
98.                 nombre.SetActive (true);
99.
100.            }
101.        }
102.
103.        if (contador == 5)
104.        {
105.
106.            if (cambio == true)
107.            {
108.                rat.SetActive (false);
109.                lineaActual++;
110.                contador++;
111.                textComp.text = "";
112.                cambio = false;

```

```

113.         StartCoroutine (TypeText (lineaActual));
114.     }
115.     else
116.     {
117.         rat.SetActive (true);
118.     }
119. }
120.
121. if (contador == 7)
122. {
123.     if (cambio == true)
124.     {
125.         fac.SetActive (false);
126.         lineaActual++;
127.         contador++;
128.         textComp.text = "";
129.         cambio = false;
130.         StartCoroutine (TypeText (lineaActual));
131.     }
132.     else
133.     {
134.         fac.SetActive (true);
135.     }
136. }
137. if (contador == 9)
138. {
139.     continuar.SetActive (true);
140. }
141.
142. }
143.
144. IEnumerator TypeText (int line)
145. {
146.     foreach (char letter in message[line].ToCharArray ())
147.     {
148.         textComp.text += letter;
149.
150.         yield return 0;
151.         yield return new WaitForSeconds (letterPaused);
152.     }
153.
154.     checkNext = true;
155. }
156.
157. public int getLineaActual ()
158. {
159.     return lineaActual;
160. }
161.
162. public int getLastLine ()
163. {
164.     return message.Length - 1;
165. }
166.
167. public bool getCheckNext ()
168. {
169.     return checkNext;
170. }
171. }

```

## Load and Save System (Serializing and Deserializing the data)

```
1.
2. using System.Collections.Generic;
3. using UnityEngine;
4. using System.Collections;
5. using System.Runtime.Serialization.Formatters.Binary; // Para
   serializar
6. using System.IO; // Para trabajar con archivos
7. using System;
8. using UnityEngine.SceneManagement;
9.
10.    public class CargarGuardar : MonoBehaviour{
11.
12.
13.        // Son los valores del estado del juego que necesitamos
14.
15.        public bool tutorial = false; // Saltarse el tutorial
16.        public string factSensibilidad; //Factor de
   sensibilidad que se podrá cambiar en opciones
17.        public string ratio; // Ratio
18.        public string insulina;
19.        public string glucosa;
20.        public string hidratos;
21.        public string nombre;
22.        public string fecha;
23.        public string momento;
24.        public int cuentaDias = 0; //Contador de días correctos
   dentro de rango
25.        public bool criaturaUno = false; // Generar nueva
   criatura
26.        public bool criaturaDos = false;
27.        public bool criaturaTres = false;
28.        public bool DixyBlack = true;
29.        public bool JamesBlack = true;
30.        public bool BenBlack = true;
31.        public bool DixyCollect = false;
32.        public bool JamesCollect = false;
33.        public bool BenCollect = false;
34.        public int cuentaFelicidad = 3;
35.
36.        public static CargarGuardar cargarGuardar;
37.
38.        private String rutaArchivo;
39.
40.        public GameObject continuar;
41.
42.        void Awake ()
43.        {
44.            if(cargarGuardar == null)
45.            {
46.                cargarGuardar = this;
47.                DontDestroyOnLoad(gameObject);
48.                rutaArchivo = Application.persistentDataPath +
   "/estadoJuegoFinal.dat";
49.            } else if (cargarGuardar != this)
50.            {
51.                Destroy(gameObject);
52.            }
53.
```

```

54.
55.     }
56.
57.     // Para acceder CargarGuardar.cargarGuardar.tutorial
58.
59.     private void Start ()
60.     {
61.
62.         Cargar ();
63.     }
64.
65.     private void Update ()
66.     {
67.
68.     }
69.
70.     public void Guardar ()
71.     {
72.
73.         NotificationCenter.DefaultCenter ().AddObserver (this
74. , "GuardarNombre");
75.         NotificationCenter.DefaultCenter ().AddObserver (this
76. , "GuardarFactor");
77.         NotificationCenter.DefaultCenter ().AddObserver (this
78. , "GuardarRatio");
79.         NotificationCenter.DefaultCenter ().AddObserver (this
80. , "GuardarGlucosa");
81.         NotificationCenter.DefaultCenter ().AddObserver (this
82. , "GuardarFecha");
83.         NotificationCenter.DefaultCenter ().AddObserver (this
84. , "GuardarHidratos");
85.         NotificationCenter.DefaultCenter ().AddObserver (this
86. , "GuardarInsulina");
87.         NotificationCenter.DefaultCenter ().AddObserver (this
88. , "SaltaTutorial");
89.         NotificationCenter.DefaultCenter ().AddObserver (this
90. , "NuevaCriatura");
91.         NotificationCenter.DefaultCenter ().AddObserver (this
92. , "CuentaDias");
93.         NotificationCenter.DefaultCenter ().AddObserver (this
94. , "CuentaFelicidad");
95.         NotificationCenter.DefaultCenter ().AddObserver (this
96. , "GuardarMomento");
97.
98.         BinaryFormatter bf = new BinaryFormatter ();
99.
100.        FileStream file = File.Create (rutaArchivo);
101.
102.        DatosGuardar datos = new DatosGuardar ();
103.
104.        datos.tutorial = tutorial;
105.        datos.factSensibilidad = factSensibilidad;
106.        datos.ratio = ratio;
107.        datos.insulina = insulina;
108.        datos.hidratos = hidratos;
109.        datos.glucosa = glucosa;
110.        datos.fecha = fecha;
111.        datos.momento = momento;

```



```

103.         datos.criaturaUno = criaturaUno;
104.         datos.criaturaDos = criaturaDos;
105.         datos.criaturaTres = criaturaTres;
106.         datos.cuentaDias = cuentaDias;
107.         datos.cuentaFelicidad = cuentaFelicidad;
108.
109.         datos.nombre = nombre;
110.
111.         datos.DixyBlack = DixyBlack;
112.         datos.JamesBlack = JamesBlack;
113.         datos.BenBlack = BenBlack;
114.
115.         datos.DixyCollect = DixyCollect;
116.         datos.JamesCollect = JamesCollect;
117.         datos.BenCollect = BenCollect;
118.
119.
120.
121.         bf.Serialize(file, datos);
122.
123.         file.Close();
124.     }
125.
126.     public void Cargar()
127.     {
128.         if (File.Exists(rutaArchivo))
129.         {
130.             BinaryFormatter bf = new BinaryFormatter();
131.             FileStream file = File.Open(rutaArchivo,
132. FileMode.Open);
133.             DatosGuardar
134.             datos = (DatosGuardar)bf.Deserialize(file); //Hacer un casting
135.
136.             tutorial = datos.tutorial;
137.             factSensibilidad = datos.factSensibilidad;
138.             ratio = datos.ratio;
139.             insulina = datos.insulina;
140.             hidratos = datos.hidratos;
141.             glucosa = datos.glucosa;
142.             fecha = datos.fecha;
143.             momento = datos.momento;
144.
145.
146.
147.             criaturaUno = datos.criaturaUno;
148.             criaturaDos = datos.criaturaDos;
149.             criaturaTres = datos.criaturaTres;
150.             cuentaDias = datos.cuentaDias;
151.             nombre = datos.nombre;
152.
153.             DixyBlack = datos.DixyBlack;
154.             JamesBlack = datos.JamesBlack;
155.             BenBlack = datos.BenBlack;
156.
157.             DixyCollect = datos.DixyCollect;
158.             JamesCollect = datos.JamesCollect;
159.             BenCollect = datos.BenCollect;
160.
161.             cuentaFelicidad = datos.cuentaFelicidad;

```

```

162.
163.         file.Close();
164.     }
165.     else
166.     {
167.         tutorial = false;
168.         factSensibilidad = "";
169.         ratio = "";
170.         nombre = "";
171.         insulina = "";
172.         glucosa = "";
173.         hidratos = "";
174.         fecha = "";
175.         momento = "";
176.
177.         cuentaFelicidad = 3;
178.         cuentaDias = 0;
179.         criaturaUno = false;
180.         criaturaDos = false;
181.         criaturaTres = false;
182.         DixyBlack = true;
183.         JamesBlack = true;
184.         BenBlack = true;
185.         DixyCollect = false;
186.         JamesCollect = false;
187.         BenCollect = false;
188.     }
189.
190. }
191.
192.
193.     [Serializable]
194.     class DatosGuardar
195.     {
196.         public bool tutorial; //Si se salta el tutorial
197.         public string nombre;
198.         public string factSensibilidad; //Factor de
199.         sensibilidad (Se puede modificar en opciones)
200.         public string ratio; // Ratio (Se puede modificar
201.         en opciones)
202.         public string insulina;
203.         public string hidratos;
204.         public string glucosa;
205.         public string fecha;
206.         public string momento;
207.
208.         public int cuentaFelicidad;
209.         public int cuentaDias; // Dias con valores ok
210.         public bool criaturaUno; // Activar Criatura
211.         public bool criaturaDos;
212.         public bool criaturaTres;
213.         public bool DixyBlack = true;
214.         public bool JamesBlack = true;
215.         public bool BenBlack = true;
216.         public bool DixyCollect = false;
217.         public bool JamesCollect = false;
218.         public bool BenCollect = false;
219.     }

```

## Endless Runner Code

```
1. public class Player : MonoBehaviour {
2.
3.     public float speed;
4.     public float increment;
5.     public float maxY;
6.     public float minY;
7.
8.     private Vector2 targetPos;
9.
10.    public int health;
11.
12.    public GameObject moveEffect;
13.    public Animator camAnim;
14.    public Text healthDisplay;
15.
16.    public GameObject spawner;
17.    public GameObject restartDisplay;
18.
19.    private void Update ()
20.    {
21.
22.        if (health <= 0) {
23.            spawner.SetActive(false);
24.            restartDisplay.SetActive(true);
25.            Destroy(gameObject);
26.        }
27.
28.        healthDisplay.text = health.ToString();
29.
30.        transform.position = Vector2.MoveTowards(transform.
position, targetPos, speed * Time.deltaTime);
31.
32.        if (Input.GetKeyDown(KeyCode.UpArrow) && transform.
position.y < maxY) {
33.            camAnim.SetTrigger("shake");
34.            Instantiate(moveEffect, transform.position,
Quaternion.identity);
35.            targetPos = new Vector2(transform.position.x,
transform.position.y + increment);
36.        } else if (Input.GetKeyDown(KeyCode.DownArrow) && t
ransform.position.y > minY) {
37.            camAnim.SetTrigger("shake");
38.            Instantiate(moveEffect, transform.position,
Quaternion.identity);
39.            targetPos = new Vector2(transform.position.x,
transform.position.y - increment);
40.        }
41.    }
42. }
43. public class Spawner : MonoBehaviour {
44.
45.     private float timeBtwSpawns;
46.     public float startTimeBtwSpawns;
47.     public float timeDecrease;
48.     public float minTime;
49.
50.     public GameObject[] obstacleTemplate;
51.
52.     private void Start ()
```

```

53.     {
54.         timeBtwSpawns = startTimeBtwSpawns;
55.     }
56.
57.     private void Update ()
58.     {
59.         if (timeBtwSpawns <= 0)
60.         {
61.             int rand = Random.Range(0,
62. obstacleTemplate.Length);
63.             Instantiate(obstacleTemplate[rand],
64. transform.position, Quaternion.identity);
65.             timeBtwSpawns = startTimeBtwSpawns;
66.             if (startTimeBtwSpawns > minTime) {
67.                 startTimeBtwSpawns -= timeDecrease;
68.             }
69.             else {
70.                 timeBtwSpawns -= Time.deltaTime;
71.             }
72.         }
73.     }
74.     public float speed;
75.     public GameObject effect;
76.
77.     void Update () {
78.         transform.Translate(Vector2.left * speed * Time.del
79. taTime);
80.     }
81.     private void OnTriggerEnter2D(Collider2D other)
82.     {
83.         if (other.CompareTag("Player")) {
84.             other.GetComponent<Player>().health--;
85.             other.GetComponent<Player>().camAnim.SetTrigger
86. ("shake");
87.             Instantiate(effect, transform.position,
88. Quaternion.identity);
89.             Destroy(gameObject);
90.         }
91.     }
92.     public class Background : MonoBehaviour {
93.
94.         public float speed;
95.         public float Xend;
96.         public float Xstart;
97.
98.         private void Update ()
99.         {
100.            transform.Translate(Vector2.left * speed * Time.del
101. taTime);
102.            if (transform.position.x < Xend) {
103.                Vector2 pos = new Vector2(Xstart,
104. transform.position.y);
105.                transform.position = pos;
106.            }
107.        }
108.     }
109.     public class Score : MonoBehaviour {

```

```

107.
108.     public int score;
109.     public Text scoreDisplay;
110.
111.     private void Update ()
112.     {
113.         scoreDisplay.text = score.ToString();
114.     }
115.
116.     private void OnTriggerEnter2D(Collider2D other)
117.     {
118.         score++;
119.         Destroy(other.gameObject);
120.     }
121. }
122. using System.Collections;
123. using System.Collections.Generic;
124. using UnityEngine;
125.
126. public class Destroyer : MonoBehaviour {
127.
128.     public float lifetime;
129.
130.     private void Start ()
131.     {
132.         Destroy(gameObject, lifetime);
133.     }
134. }
135. using System.Collections;
136. using System.Collections.Generic;
137. using UnityEngine;
138.
139. public class Obstacle : MonoBehaviour {
140.
141.     public float speed;
142.     public GameObject effect;
143.
144.     void Update () {
145.         transform.Translate(Vector2.left * speed * Time.del
146. taTime);
147.     }
148.
149.     private void OnTriggerEnter2D(Collider2D other)
150.     {
151.         if (other.CompareTag("Player")) {
152.             other.GetComponent<Player>().health--;
153.             other.GetComponent<Player>().camAnim.SetTrigger
154. ("shake");
155.             Instantiate(effect, transform.position,
156. Quaternion.identity);
157.             Destroy(gameObject);
158.         }
159.     }

```

## Save CSV data

```
1. using System.Collections.Generic;
2. using UnityEngine;
3. using System.Collections;
4. using System.Runtime.Serialization.Formatters.Binary; // Para
   serializar
5. using System.IO; // Para trabajar con archivos
6. using System;
7. using UnityEngine.SceneManagement;
8.
9.
10.
11.     public class saveCsv : MonoBehaviour
12.     {
13.
14.
15.
16.         public string cargaRatio;
17.         public string cargarFactor;
18.         public string cargarHidratos;
19.         public string cargarGlucosa;
20.         public string cargarInsulina;
21.         public string cargarFecha;
22.         public string datos = "DatosPaciente";
23.         public int contador = 0;
24.
25.
26.
27.         public void cargaryGenerar ()
28.         {
29.
30.             cargaRatio = CargarGuardar.cargarGuardar.ratio;
31.             cargarFactor = CargarGuardar.cargarGuardar.factSens
   ibilidad;
32.             cargarGlucosa = CargarGuardar.cargarGuardar.glucosa
   ;
33.             cargarInsulina = CargarGuardar.cargarGuardar.insuli
   na;
34.             cargarHidratos = CargarGuardar.cargarGuardar.hidrat
   os;
35.             cargarFecha = CargarGuardar.cargarGuardar.fecha;
36.
37.             Debug.Log(cargaRatio + cargarFactor + cargarGlucosa
   + cargarInsulina + cargarHidratos + cargarFecha);
38.
39.             CrearArchivoCSV(datos);
40.
41.         }
42.
43.
44.
45.         public IEnumerator
   CrearArchivoCSV(string nombreArchivo)
46.         {
47.
48.
49.             string ruta = Application.persistentDataPath + "/"
   + nombreArchivo + ".csv";
50.
51.
```

```

52.         //Crear el archivo
53.         var sr = File.CreateText(ruta);
54.
55.         string datosCSV = "Fecha
Actual:" + cargarFecha + System.Environment.NewLine;
56.         datosCSV += "Ratio:" + cargaRatio + System.Environment.NewLine;
57.         datosCSV += "Factor de
sensibilidad:" + cargarFactor + System.Environment.NewLine;
58.         datosCSV += "Glucosa
Previa:" + cargarGlucosa + System.Environment.NewLine;
59.         datosCSV += "Hidratos
Carbono:" + cargarHidratos + System.Environment.NewLine;
60.         datosCSV += "Hidratos Carbono:" + cargarInsulina;
61.
62.         Debug.Log(datosCSV);
63.
64.         sr.WriteLine(datosCSV);
65.
66.         //Dejar como sólo de lectura
67.         FileInfo fInfo = new FileInfo(ruta);
68.         fInfo.IsReadOnly = true;
69.
70.         //Cerrar
71.         sr.Close();
72.
73.         yield return new WaitForSeconds(0.5f); //Esperamos
para estar seguros que escriba el archivo
74.
75.         //Abrimos archivo recién creado
76.         Application.OpenURL(ruta);
77.     }
78. }

```

## Test SQL sentence

```

1. using System.Collections;
2. using System.Collections.Generic;
3. using System.Data;
4. using System.IO;
5. using Mono.Data.Sqlite;
6. using UnityEngine;
7. using UnityEngine.Networking;
8. using UnityEngine.UI;
9.
10. public class ComandosSQL : MonoBehaviour {
11.     string rutaDB;
12.     string strConexion;
13.     //string DBFileName = "RopaDB.sqlite";
14.     // public Text myText; // Comprobar en pantalla android
15.
16.     string DBFileName = "chinook.db";
17.
18.     IDbConnection dbConnection; // Crear conexión
19.     IDbCommand dbCommand; // Comandos
20.     IDataReader leerdatos; // Para poder leer
21.
22.     // Use this for initialization
23.     void Start()

```

```

24.         {
25.             AbrirDB();
26.             //ComandoSelect("*", "albums");
27.             //ComandoWHERE("*", "albums", "AlbumId", "=",
"33");
28.             //ComandoWHERE_AND("CustomerId", "FirstName",
"LastName", "Country", "customers", "Country", "=", "Brazil",
"CustomerId", ">", "10");
29.             //ComandoWHERE_AND_ORDERBY("CustomerId",
"FirstName", "LastName", "Country", "customers", "Country", "=",
"Brazil", "CustomerId", ">", "10", "FirstName", "ASC");
30.             ComandoINSERT("Insertar Glucemia previa");
31.             CerrarDB();
32.         }
33.
34.     void AbrirDB()
35.     {
36.         // Si es PC
37.         if (Application.platform == RuntimePlatform.Windows
Editor)
38.         {
39.             rutaDB = Application.dataPath + "/StreamingAssets/" + DBFileName; //Se divide la ruta para las diferentes
plataformas
40.         }
41.
42.         // Si es Android
43.
44.         else if (Application.platform == RuntimePlatform.An
droid)
45.         {
46.             rutaDB = Application.persistentDataPath + "/" +
DBFileName;
47.             // Comprobar si está en persistentData
48.             if (!File.Exists(rutaDB))
49.             {
50.
51.
52.                 //UnityWebRequest loadDB =
UnityWebRequest.Get("jar:file://" + Application.dataPath +
"!/assets/" + DBFileName);
53.
54.                 WWW
loadDB = new WWW("jar:file://" + Application.dataPath + "!/asset
s/" + DBFileName); //Ruta en formato generico de android (Java)
55.                 while (!loadDB.isDone)
56.                 {
57.                 }
58.                 File.WriteAllBytes(rutaDB, loadDB.bytes);
59.             }
60.         }
61.
62.         // Crear y abrir conexión
63.
64.         strConexion = "URI=file:" + rutaDB;
65.         dbConnection = new SqliteConnection(strConexion);
66.         dbConnection.Open();
67.
68.     }
69.
70.     void CerrarDB()

```



```

71.     {
72.         // Cerrar conexiones
73.         leerdatos.Close();
74.         leerdatos = null;
75.         dbCommand.Dispose();
76.         dbCommand = null;
77.         dbConnection.Close();
78.         dbConnection = null;
79.     }
80.
81.
82.
83.     // select *
84.     // from albums
85.
86.     void ComandoSelect(string item, string tabla)
87.     {
88.         // Crear consulta
89.         dbCommand = dbConnection.CreateCommand();
90.         string sqlQuery = "select " + item + " from
    " + tabla;
91.         dbCommand.CommandText = sqlQuery; // Aquí pasamos
    la consulta
92.
93.         // Leer la base de datos
94.         leerdatos = dbCommand.ExecuteReader();
95.         while (leerdatos.Read())
96.         {
97.             // En pantalla
98.             //string testtoconsole =
    leerdatos.GetString(0); //marca
99.             //int testing = leerdatos.GetInt32(1); //id
100.            //myText.text = testtoconsole + " - " +
    testing.ToString();
101.
102.            //Control de errores
103.
104.            try
105.            {
106.                Debug.Log(leerdatos.GetInt32(0) + " -
    " + leerdatos.GetString(1) + " - " + leerdatos.GetInt32(2));
107.            }
108.
109.            catch (FormatException fe)
110.            {
111.                Debug.Log(fe.Message);
112.                continue;
113.            }
114.
115.            catch (Exception e)
116.            {
117.                Debug.Log(e.Message);
118.                continue;
119.            }
120.
121.
122.        }
123.    }
124.
125.
126.     // select AlbumId

```

```

127.         // from albums
128.         // where AlbumId = "33"
129.
130.
131.         void ComandoWHERE(string item, string tabla, string cam
po, string comparador, string dato)
132.         {
133.             // Crear consulta
134.             dbCommand = dbConnection.CreateCommand();
135.             string sqlQuery = "select " + item + " from
" + tabla + " where " + campo + " " + comparador + " " + dato;
136.             dbCommand.CommandText = sqlQuery; // Aquí pasamos
la consulta
137.
138.             // Leer la base de datos
139.             leerdatos = dbCommand.ExecuteReader();
140.             while (leerdatos.Read())
141.             {
142.
143.                 //Control de errores
144.
145.                 try
146.                 {
147.                     Debug.Log(leerdatos.GetInt32(0) + " -
" + leerdatos.GetString(1) + " - " + leerdatos.GetInt32(2));
148.                 }
149.
150.                 catch (FormatException fe)
151.                 {
152.                     Debug.Log(fe.Message);
153.                     continue;
154.                 }
155.
156.                 catch (Exception e)
157.                 {
158.                     Debug.Log(e.Message);
159.                     continue;
160.                 }
161.
162.             }
163.         }
164.
165.
166.
167.         // select CustomerID, FirstName, LastName, Country
168.         // from customers
169.         // where Country = "Brazil"
170.         // and CustomerId > 10
171.
172.
173.         void ComandoWHERE_AND(string item1, string item2, strin
g item3, string item4,
174.                                string tabla,
175.                                string campo1, string comparado
r1, string dato1,
176.                                string campo2, string comparado
r2, string dato2)
177.         {
178.             // Crear consulta
179.             dbCommand = dbConnection.CreateCommand();
180.

```

```

181.         string sqlQuery = "select
    " + item1 + "," + item2 + "," + item3 + "," + item4 + //Ojo
    separar las columnas por ","
182.             " from " + tabla +
183.             " where " + campo1 + "
    " + comparador1 + " " + "'" + dato1 + "'" +
184.             " and " + campo2 + "
    " + comparador2 + dato2;
185.
186.
187.         dbCommand.CommandText = sqlQuery; // Aquí pasamos
    la consulta
188.
189.         // Leer la base de datos
190.         leerdatos = dbCommand.ExecuteReader();
191.         while (leerdatos.Read())
192.         {
193.
194.
195.             //Control de errores
196.
197.             try
198.             {
199.                 Debug.Log(leerdatos.GetInt32(0) + "
    " + leerdatos.GetString(1) + " " + leerdatos.GetString(2) + "-
    " + leerdatos.GetString(3));
200.             }
201.
202.             catch (FormatException fe)
203.             {
204.                 Debug.Log(fe.Message);
205.                 continue;
206.             }
207.
208.             catch (Exception e)
209.             {
210.                 Debug.Log(e.Message);
211.                 continue;
212.             }
213.
214.
215.         }
216.     }
217.
218.     // select CustomerID, FirstName, LastName, Country
219.     // from customers
220.     // where Country = "Brazil"
221.     // and CustomerId > 10
222.     // order by FirstName ASC
223.
224.
225.     void ComandoWHERE_AND_ORDERBY(string item1, string item
    2, string item3, string item4,
226.         string tabla,
227.         string campo1, string comparado
    r1, string dato1,
228.         string campo2, string comparado
    r2, string dato2,
229.         string campo3, string orden)
230.     {
231.         // Crear consulta

```

```

232.         dbCommand = dbConnection.CreateCommand();
233.
234.         string sqlQuery = "select
    + item1 + "," + item2 + "," + item3 + "," + item4 + //Ojo
separar las columnas por ","
235.             " from " + tabla +
236.             " where " + campol + "
    + comparador1 + " " + "'" + dato1 + "'" +
237.             " and " + campo2 + "
    + comparador2 + dato2 +
238.             " order by " + campo3 + "
    + orden;
239.
240.
241.         dbCommand.CommandText = sqlQuery; // Aquí pasamos
la consulta
242.
243.         // Leer la base de datos
244.         leerdatos = dbCommand.ExecuteReader();
245.         while (leerdatos.Read())
246.         {
247.
248.
249.             //Control de errores
250.
251.             try
252.             {
253.                 Debug.Log(leerdatos.GetInt32(0) + "
    + leerdatos.GetString(1) + " " + leerdatos.GetString(2) + "-
    + leerdatos.GetString(3));
254.             }
255.
256.             catch (FormatException fe)
257.             {
258.                 Debug.Log(fe.Message);
259.                 continue;
260.             }
261.
262.             catch (Exception e)
263.             {
264.                 Debug.Log(e.Message);
265.                 continue;
266.             }
267.
268.
269.         }
270.     }
271.
272.     // select CustomerID, FirstName, LastName, Country
273.     // from customers
274.     // where Country IN ("Brazil", "USA");
275.
276.
277.
278.     void ComandoIN(string item1, string item2, string item3
, string item4,
279.                   string tabla,
280.                   string campol, string dato1, st
ring dato2)
281.     {
282.         // Crear consulta

```

```

283.         dbCommand = dbConnection.CreateCommand();
284.
285.         string sqlQuery = "select
    + item1 + "," + item2 + "," + item3 + "," + item4 + //Ojo
separar las columnas por ","
286.             " from " + tabla +
287.             " where " + campol + "
    + "IN" + " " + "(" + dato1 + "" + "," + " + campo2 " + ")";
288.
289.
290.
291.         dbCommand.CommandText = sqlQuery; // Aquí pasamos
la consulta
292.
293.         // Leer la base de datos
294.         leerdatos = dbCommand.ExecuteReader();
295.         while (leerdatos.Read())
296.         {
297.
298.
299.             //Control de errores
300.
301.             try
302.             {
303.                 Debug.Log(leerdatos.GetInt32(0) + "
    + leerdatos.GetString(1) + " " + leerdatos.GetString(2) + "-
    + leerdatos.GetString(3));
304.             }
305.
306.             catch (FormatException fe)
307.             {
308.                 Debug.Log(fe.Message);
309.                 continue;
310.             }
311.
312.             catch (Exception e)
313.             {
314.                 Debug.Log(e.Message);
315.                 continue;
316.             }
317.
318.
319.         }
320.     }
321.
322.     // insert into media_types(name) values ("image")
323.
324.
325.
326.     void ComandoINSERT(string dato)
327.     {
328.         // Crear consulta
329.         dbCommand = dbConnection.CreateCommand();
330.         string sqlQuery = String.Format("insert into
media_types(Name) values (\">{0}\>", dato);
331.         dbCommand.CommandText = sqlQuery; // Aquí pasamos
la consulta
332.         dbCommand.ExecuteScalar();
333.
334.         dbCommand.Dispose();
335.         dbCommand = null;

```

```
336.         dbConnection.Close();
337.         dbConnection = null;
338.
339.
340.     }
341.
342. }
```

## Change Moment of Day

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4.
5. public class CambioMomentoPro : MonoBehaviour
6. {
7.
8.
9.     public GameObject volver;
10.
11.     public GameObject fondoDes;
12.     public GameObject fondoAlm;
13.     public GameObject fondoCom;
14.     public GameObject fondoMer;
15.     public GameObject fondoCen;
16.
17.     public GameObject textoDes;
18.     public GameObject textoAlm;
19.     public GameObject textoCom;
20.     public GameObject textoMer;
21.     public GameObject textoCen;
22.
23.     public GameObject aniDes;
24.     public GameObject aniAlm;
25.     public GameObject aniCom;
26.     public GameObject aniMer;
27.     public GameObject aniCen;
28.
29.
30.
31.
32.     public Int32 hora = 0;
33.
34.     public void comprobarHora ()
35.     {
36.         hora = DateTime.Now.Hour;
37.     }
38.
39.
40.     public int contador = 0;
41.
42.
43.
44.     public void FuncionContadorMas ()
45.     {
46.
47.
48.         contador++;
49.
50.     }
51.
52.
53.
54.     public void FuncionContadorMenos ()
55.     {
56.         contador--;
57.
58.     }
59.
```

```

60.         // Start is called before the first frame update
61.         void Start()
62.         {
63.
64.             NotificationCenter.DefaultCenter().AddObserver(this
, "GuardarMomento");
65.
66.             comprobarHora();
67.             volver = GameObject.Find("Back");
68.
69.             fondoDes = GameObject.Find("BGDes"); ;
70.             fondoAlm = GameObject.Find("BGAlm"); ;
71.             fondoCom = GameObject.Find("BGCom"); ;
72.             fondoMer = GameObject.Find("BGMer"); ;
73.             fondoCen = GameObject.Find("BGCen"); ;
74.
75.             textoDes = GameObject.Find("DesayunoM"); ;
76.             textoAlm = GameObject.Find("AlmuerzoM"); ;
77.             textoCom = GameObject.Find("ComidaM"); ;
78.             textoMer = GameObject.Find("MeriendaM"); ;
79.             textoCen = GameObject.Find("CenaM"); ;
80.
81.             aniDes = GameObject.Find("Desayuno"); ;
82.             aniAlm = GameObject.Find("Almuerzo"); ;
83.             aniCom = GameObject.Find("Comida"); ;
84.             aniMer = GameObject.Find("Merienda"); ;
85.             aniCen = GameObject.Find("Cena"); ;
86.
87.             volver.SetActive(false);
88.
89.             fondoDes.SetActive(false);
90.             textoDes.SetActive(false);
91.             aniDes.SetActive(false);
92.
93.             fondoAlm.SetActive(false);
94.             textoAlm.SetActive(false);
95.             aniAlm.SetActive(false);
96.
97.             fondoCom.SetActive(false);
98.             textoCom.SetActive(false);
99.             aniCom.SetActive(false);
100.
101.             fondoMer.SetActive(false);
102.             textoMer.SetActive(false);
103.             aniMer.SetActive(false);
104.
105.             fondoCen.SetActive(false);
106.             textoCen.SetActive(false);
107.             aniCen.SetActive(false);
108.
109.
110.
111.         }
112.         /*
113.         public void GuardarMomento(Notification notification)
114.         {
115.             {
116.                 if (hora >= 7 && hora <= 10)
117.                 {
118.                     CargarGuardar.cargarGuardar.momento =
"Desayuno";

```



```

119.         CargarGuardar.cargarGuardar.Guardar();
120.     }
121.     else if (hora >= 10 && hora <= 12)
122.     {
123.         CargarGuardar.cargarGuardar.momento =
124.         "Almuerzo";
125.         CargarGuardar.cargarGuardar.Guardar();
126.     }
127.     else if (hora >= 12 && hora <= 15)
128.     {
129.         CargarGuardar.cargarGuardar.momento = "Comida";
130.         CargarGuardar.cargarGuardar.Guardar();
131.     }
132.     else if (hora >= 17 && hora <= 19)
133.     {
134.         CargarGuardar.cargarGuardar.momento =
135.         "Merienda";
136.         CargarGuardar.cargarGuardar.Guardar();
137.     }
138.     else if (hora >= 17 && hora <= 19)
139.     {
140.         CargarGuardar.cargarGuardar.momento = "Cena";
141.         CargarGuardar.cargarGuardar.Guardar();
142.     }
143.     }
144.     */
145.
146.     public void GuardarMomento(Notification notification)
147.     {
148.         if (contador == 0)
149.         {
150.             CargarGuardar.cargarGuardar.momento = "Desayuno
151.             ";
152.             CargarGuardar.cargarGuardar.Guardar();
153.         }
154.         else if (contador == 1)
155.         {
156.             CargarGuardar.cargarGuardar.momento = "Almuerzo
157.             ";
158.             CargarGuardar.cargarGuardar.Guardar();
159.         }
160.         else if (contador == 2)
161.         {
162.             CargarGuardar.cargarGuardar.momento = "Comida";
163.             CargarGuardar.cargarGuardar.Guardar();
164.         }
165.         else if (contador == 3)
166.         {
167.             CargarGuardar.cargarGuardar.momento = "Merienda
168.             ";
169.             CargarGuardar.cargarGuardar.Guardar();
170.         }
171.         else if (contador == 4)
172.         {
173.             CargarGuardar.cargarGuardar.momento = "Cena";
174.             CargarGuardar.cargarGuardar.Guardar();

```

```

175.     }
176.
177.
178.
179.     // Update is called once per frame
180.     void Update ()
181.     {
182.         //CAMBIAR MOMENTO SEGÚN HORA DEL SISTEMA
183.         /*if (hora >= 7 && hora <= 10)
184.         {
185.             volver.SetActive(false);
186.
187.             fondoAlm.SetActive(false);
188.             textoAlm.SetActive(false);
189.             aniAlm.SetActive(false);
190.
191.             fondoDes.SetActive(true);
192.             textoDes.SetActive(true);
193.             aniDes.SetActive(true);
194.             NotificationCenter.DefaultCenter().PostNotifica
tion(this, "GuardarMomento");
195.
196.
197.         }
198.         else if (hora >= 10 && hora <= 12)
199.         {
200.
201.             fondoDes.SetActive(false);
202.             textoDes.SetActive(false);
203.             aniDes.SetActive(false);
204.
205.             fondoCom.SetActive(false);
206.             textoCom.SetActive(false);
207.             aniCom.SetActive(false);
208.
209.             volver.SetActive(true);
210.             fondoAlm.SetActive(true);
211.             textoAlm.SetActive(true);
212.             aniAlm.SetActive(true);
213.             NotificationCenter.DefaultCenter().PostNotifica
tion(this, "GuardarMomento");
214.
215.         }
216.         else if (hora >= 12 && hora <= 15)
217.         {
218.             fondoAlm.SetActive(false);
219.             textoAlm.SetActive(false);
220.             aniAlm.SetActive(false);
221.
222.             fondoMer.SetActive(false);
223.             textoMer.SetActive(false);
224.             aniMer.SetActive(false);
225.
226.             fondoCom.SetActive(true);
227.             textoCom.SetActive(true);
228.             aniCom.SetActive(true);
229.             NotificationCenter.DefaultCenter().PostNotifica
tion(this, "GuardarMomento");
230.
231.         }
232.         else if (hora >= 17 && hora <= 19)
233.         {
234.             fondoCom.SetActive(false);

```

```

233.         textoCom.SetActive (false);
234.         aniCom.SetActive (false);
235.
236.         fondoCen.SetActive (false);
237.         textoCen.SetActive (false);
238.         aniCen.SetActive (false);
239.
240.         fondoMer.SetActive (true);
241.         textoMer.SetActive (true);
242.         aniMer.SetActive (true);
243.         NotificationCenter.DefaultCenter().PostNotifica
tion(this, "GuardarMomento");
244.     }
245.     else if (hora >= 21 && hora <= 23)
246.     {
247.         fondoMer.SetActive (false);
248.         textoMer.SetActive (false);
249.         aniMer.SetActive (false);
250.
251.         fondoDes.SetActive (false);
252.         textoDes.SetActive (false);
253.         aniDes.SetActive (false);
254.
255.         fondoCen.SetActive (true);
256.         textoCen.SetActive (true);
257.         aniCen.SetActive (true);
258.         NotificationCenter.DefaultCenter().PostNotifica
tion(this, "GuardarMomento");
259.     }
260.     else
261.     {
262.         volver.SetActive (false);
263.
264.         fondoAlm.SetActive (false);
265.         textoAlm.SetActive (false);
266.         aniAlm.SetActive (false);
267.
268.         fondoDes.SetActive (true);
269.         textoDes.SetActive (true);
270.         aniDes.SetActive (true);
271.     }
272.     }*/
273.
274.
275.
276.     if (contador == 0)
277.     {
278.         volver.SetActive (false);
279.
280.         fondoAlm.SetActive (false);
281.         textoAlm.SetActive (false);
282.         aniAlm.SetActive (false);
283.
284.         fondoDes.SetActive (true);
285.         textoDes.SetActive (true);
286.         aniDes.SetActive (true);
287.         NotificationCenter.DefaultCenter().PostNotifica
tion(this, "GuardarMomento");
288.     }
289.     else if (contador == 1)
290.     {

```

```

291.
292.         fondoDes.SetActive (false);
293.         textoDes.SetActive (false);
294.         aniDes.SetActive (false);
295.
296.         fondoCom.SetActive (false);
297.         textoCom.SetActive (false);
298.         aniCom.SetActive (false);
299.
300.         volver.SetActive (true);
301.         fondoAlm.SetActive (true);
302.         textoAlm.SetActive (true);
303.         aniAlm.SetActive (true);
304.         NotificationCenter.DefaultCenter().PostNotifica
tion(this, "GuardarMomento");
305.     }
306.     else if (contador == 2)
307.     {
308.         fondoAlm.SetActive (false);
309.         textoAlm.SetActive (false);
310.         aniAlm.SetActive (false);
311.
312.         fondoMer.SetActive (false);
313.         textoMer.SetActive (false);
314.         aniMer.SetActive (false);
315.
316.         fondoCom.SetActive (true);
317.         textoCom.SetActive (true);
318.         aniCom.SetActive (true);
319.         NotificationCenter.DefaultCenter().PostNotifica
tion(this, "GuardarMomento");
320.     }
321.     else if (contador == 3)
322.     {
323.         fondoCom.SetActive (false);
324.         textoCom.SetActive (false);
325.         aniCom.SetActive (false);
326.
327.         fondoCen.SetActive (false);
328.         textoCen.SetActive (false);
329.         aniCen.SetActive (false);
330.
331.         fondoMer.SetActive (true);
332.         textoMer.SetActive (true);
333.         aniMer.SetActive (true);
334.         NotificationCenter.DefaultCenter().PostNotifica
tion(this, "GuardarMomento");
335.     }
336.     else if (contador == 4)
337.     {
338.         fondoMer.SetActive (false);
339.         textoMer.SetActive (false);
340.         aniMer.SetActive (false);
341.
342.         fondoDes.SetActive (false);
343.         textoDes.SetActive (false);
344.         aniDes.SetActive (false);
345.
346.         fondoCen.SetActive (true);
347.         textoCen.SetActive (true);
348.         aniCen.SetActive (true);

```

```

349.         NotificationCenter.DefaultCenter().PostNotifica
tion(this, "GuardarMomento");
350.     }
351.     else
352.     {
353.         fondoCen.SetActive(false);
354.         textoCen.SetActive(false);
355.         aniCen.SetActive(false);
356.         volver.SetActive(false);
357.         contador = 0;
358.     }
359.
360.
361.     }
362. }

```

## New Collectible

```

1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4.
5. public class NuevoColeccionable : MonoBehaviour
6. {
7.
8.     public GameObject DixyDark;
9.     public GameObject JamesDark;
10.    public GameObject BenDark;
11.
12.    public GameObject Dixy;
13.    public GameObject James;
14.    public GameObject Ben;
15.
16.    public int contador = 0;
17.
18.    public int dias;
19.
20.    public bool DixyBlack;
21.    public bool JamesBlack;
22.    public bool BenBlack;
23.
24.    public bool DixyCollect;
25.    public bool JamesCollect;
26.    public bool BenCollect;
27.
28.    // Start is called before the first frame update
29.    void Start()
30.    {
31.
32.        dias = CargarGuardar.cargarGuardar.cuentaDias;
33.
34.        DixyBlack = CargarGuardar.cargarGuardar.DixyBlack;
35.        JamesBlack = CargarGuardar.cargarGuardar.JamesBlack
;
36.        BenBlack = CargarGuardar.cargarGuardar.BenBlack;
37.
38.        DixyCollect = CargarGuardar.cargarGuardar.DixyColle
ct;
39.        JamesCollect = CargarGuardar.cargarGuardar.JamesCol
lect;

```

```

40.         BenCollect = CargarGuardar.cargarGuardar.BenCollect
41.     ;
42.         DixyDark= GameObject.Find("DixyDark");
43.         JamesDark = GameObject.Find("JamesDark");
44.         BenDark= GameObject.Find("BenDark");
45.
46.         Dixy = GameObject.Find("Dixy");
47.         James = GameObject.Find("James");
48.         Ben = GameObject.Find("Ben");
49.
50.         DixyDark.SetActive(true);
51.         JamesDark.SetActive(true);
52.         BenDark.SetActive(true);
53.
54.         Dixy.SetActive(false);
55.         James.SetActive(false);
56.         Ben.SetActive(false);
57.
58.     }
59.
60.     // Update is called once per frame
61.     void Update()
62.     {
63.         if (dias == 2)
64.         {
65.             DixyDark.SetActive(false);
66.             Dixy.SetActive(true);
67.
68.         }
69.         if (dias == 3)
70.         {
71.             DixyDark.SetActive(false);
72.             Dixy.SetActive(true);
73.             JamesDark.SetActive(false);
74.             James.SetActive(true);
75.
76.         }
77.         if (dias >= 4)
78.         {
79.             DixyDark.SetActive(false);
80.             Dixy.SetActive(true);
81.             JamesDark.SetActive(false);
82.             James.SetActive(true);
83.             BenDark.SetActive(false);
84.             Ben.SetActive(true);
85.
86.         }
87.
88.     }
89. }
90.

```

## Open Egg

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4.
5. public class AbrirHuevo : MonoBehaviour
6. {
7.
8.     public bool nuevaCriatura;
9.
10.     // Start is called before the first frame update
11.     void Start()
12.     {
13.         NotificationCenter.DefaultCenter().AddObserver(this
14. , "NuevaCriatura");
15.
16.         //CargarGuardar.cargarGuardar.cuentaDias = 0;
17.         //CargarGuardar.cargarGuardar.Guardar();
18.
19.     }
20.
21.     // Update is called once per frame
22.     void Update()
23.     {
24.         NotificationCenter.DefaultCenter().PostNotification
25. (this, "NuevaCriatura");
26.
27.     public void NuevaCriatura(Notification notification)
28.     {
29.         if (CargarGuardar.cargarGuardar.cuentaDias == 2)
30.         {
31.             CargarGuardar.cargarGuardar.criaturaUno = true;
32.             CargarGuardar.cargarGuardar.Guardar();
33.         }
34.         else if(CargarGuardar.cargarGuardar.cuentaDias == 3
35. )
36.         {
37.             CargarGuardar.cargarGuardar.criaturaDos = true;
38.             CargarGuardar.cargarGuardar.Guardar();
39.         }
40.         else if(CargarGuardar.cargarGuardar.cuentaDias == 4
41. )
42.         {
43.             CargarGuardar.cargarGuardar.criaturaTres = true
44. ;
45.             CargarGuardar.cargarGuardar.Guardar();
46.         }
47.
48.         Debug.Log(nuevaCriatura);
49.     }
50. }
```

## Food Change

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4.
5. public class CambiarComida : MonoBehaviour
6. {
7.
8.     public int contadorComida = 0;
9.     public int contadorTamaño = 0;
10.
11.     public GameObject macaPeq;
12.     public GameObject macaMed;
13.     public GameObject macaGrd;
14.
15.     public GameObject arrozPeq;
16.     public GameObject arrozMed;
17.     public GameObject arrozGrd;
18.
19.     public GameObject guiPeq;
20.     public GameObject guiMed;
21.     public GameObject guiGrd;
22.
23.     public bool mp = false;
24.     public bool mm = false;
25.     public bool mg = false;
26.
27.     public bool ap = false;
28.     public bool am = false;
29.     public bool ag = false;
30.
31.     public bool gp = false;
32.     public bool gm = false;
33.     public bool gg = false;
34.
35.     public void FuncionContadorMasComida ()
36.     {
37.         contadorComida++;
38.     }
39.
40.     public void FuncionContadorMasTamaño ()
41.     {
42.         contadorTamaño++;
43.     }
44.
45.     public void FuncionContadorMenosComida ()
46.     {
47.         contadorComida--;
48.     }
49.
50.     public void FuncionContadorMenosTamaño ()
51.     {
52.         contadorTamaño--;
53.     }
54.
55.     // Start is called before the first frame update
56.     void Start ()
57.     {
58.
59.         macaPeq = GameObject.Find("MacarronesPeq");
```



```

60.         macaMed = GameObject.Find("MacarronesMed");
61.         macaGrd = GameObject.Find("MacarronesGrd");
62.
63.         arrozPeq = GameObject.Find("ArrozPeq");
64.         arrozMed = GameObject.Find("ArrozMed");
65.         arrozGrd = GameObject.Find("ArrozGrd");
66.
67.         guiPeq = GameObject.Find("GuisantesPeq");
68.         guiMed = GameObject.Find("GuisantesMed");
69.         guiGrd = GameObject.Find("GuisantesGrd");
70.
71.         macaPeq.SetActive(false);
72.         macaMed.SetActive(false);
73.         macaGrd.SetActive(false);
74.
75.         arrozPeq.SetActive(false);
76.         arrozMed.SetActive(false);
77.         arrozGrd.SetActive(false);
78.
79.         guiPeq.SetActive(false);
80.         guiMed.SetActive(false);
81.         guiGrd.SetActive(false);
82.
83.     }
84.
85.     // Update is called once per frame
86.     void Update()
87.     {
88.         if(contadorComida == 0)
89.         {
90.             if (contadorTamaño == 0)
91.             {
92.                 macaPeq.SetActive(true);
93.                 macaMed.SetActive(false);
94.                 macaGrd.SetActive(false);
95.                 arrozPeq.SetActive(false);
96.                 arrozMed.SetActive(false);
97.                 arrozGrd.SetActive(false);
98.                 guiPeq.SetActive(false);
99.                 guiMed.SetActive(false);
100.                guiGrd.SetActive(false);
101.
102.
103.            }
104.            else if(contadorTamaño == 1)
105.            {
106.                macaPeq.SetActive(false);
107.                macaMed.SetActive(true);
108.                macaGrd.SetActive(false);
109.                arrozPeq.SetActive(false);
110.                arrozMed.SetActive(false);
111.                arrozGrd.SetActive(false);
112.                guiPeq.SetActive(false);
113.                guiMed.SetActive(false);
114.                guiGrd.SetActive(false);
115.            }
116.            else if(contadorTamaño == 2)
117.            {
118.                macaPeq.SetActive(false);
119.                macaMed.SetActive(false);
120.                macaGrd.SetActive(true);

```

```

121.         arrozPeq.SetActive (false);
122.         arrozMed.SetActive (false);
123.         arrozGrd.SetActive (false);
124.         guiPeq.SetActive (false);
125.         guiMed.SetActive (false);
126.         guiGrd.SetActive (false);
127.     }
128.
129. }
130. else if (contadorComida == 1)
131. {
132.     if (contadorTamaño == 0)
133.     {
134.         macaPeq.SetActive (false);
135.         macaMed.SetActive (false);
136.         macaGrd.SetActive (false);
137.         arrozPeq.SetActive (true);
138.         arrozMed.SetActive (false);
139.         arrozGrd.SetActive (false);
140.         guiPeq.SetActive (false);
141.         guiMed.SetActive (false);
142.         guiGrd.SetActive (false);
143.
144.     }
145.     else if (contadorTamaño == 1)
146.     {
147.         macaPeq.SetActive (false);
148.         macaMed.SetActive (false);
149.         macaGrd.SetActive (false);
150.         arrozPeq.SetActive (false);
151.         arrozMed.SetActive (true);
152.         arrozGrd.SetActive (false);
153.         guiPeq.SetActive (false);
154.         guiMed.SetActive (false);
155.         guiGrd.SetActive (false);
156.     }
157.     else if (contadorTamaño == 2)
158.     {
159.         macaPeq.SetActive (false);
160.         macaMed.SetActive (false);
161.         macaGrd.SetActive (false);
162.         arrozPeq.SetActive (false);
163.         arrozMed.SetActive (false);
164.         arrozGrd.SetActive (true);
165.         guiPeq.SetActive (false);
166.         guiMed.SetActive (false);
167.         guiGrd.SetActive (false);
168.     }
169. }
170. else if (contadorComida == 2)
171. {
172.     if (contadorTamaño == 0)
173.     {
174.         macaPeq.SetActive (false);
175.         macaMed.SetActive (false);
176.         macaGrd.SetActive (false);
177.         arrozPeq.SetActive (false);
178.         arrozMed.SetActive (false);
179.         arrozGrd.SetActive (false);
180.         guiPeq.SetActive (true);
181.         guiMed.SetActive (false);

```

```

182.         guiGrd.SetActive(false);
183.     }
184.     else if (contadorTamaño == 1)
185.     {
186.         macaPeq.SetActive(false);
187.         macaMed.SetActive(false);
188.         macaGrd.SetActive(false);
189.         arrozPeq.SetActive(false);
190.         arrozMed.SetActive(false);
191.         arrozGrd.SetActive(false);
192.         guiPeq.SetActive(false);
193.         guiMed.SetActive(true);
194.         guiGrd.SetActive(false);
195.     }
196.     else if (contadorTamaño == 2)
197.     {
198.         macaPeq.SetActive(false);
199.         macaMed.SetActive(false);
200.         macaGrd.SetActive(false);
201.         arrozPeq.SetActive(false);
202.         arrozMed.SetActive(false);
203.         arrozGrd.SetActive(false);
204.         guiPeq.SetActive(false);
205.         guiMed.SetActive(false);
206.         guiGrd.SetActive(true);
207.     }
208. }
209. if (contadorTamaño > 2)
210. {
211.     contadorTamaño = 0;
212. }
213. if (contadorTamaño < 0)
214. {
215.     contadorTamaño = 2;
216. }
217. if (contadorComida > 2)
218. {
219.     contadorComida = 0;
220. }
221. if (contadorComida < 0)
222. {
223.     contadorComida = 2;
224. }
225. }
226.
227. }

```

## Insert data in Load and Save System

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4. using UnityEngine.UI;
5. using UnityEngine.SceneManagement;
6. using System;
7.
8. public class Insertar : MonoBehaviour
9. {
10.
11.
12.     public GameObject warning80;
13.     public GameObject warning250;
14.
15.     public GameObject hidratos;
16.     public GameObject glucosa;
17.     public GameObject insulina;
18.
19.     public GameObject TextHidratos;
20.     public GameObject TextGlucosa;
21.     public GameObject TextInsulina;
22.
23.     public GameObject botonContinuar;
24.
25.
26.
27.     public float nhidratos;
28.     public float nprevia;
29.     public float ninsulina;
30.     public float nratio;
31.     public float nfactor;
32.
33.     public DateTime today = DateTime.Today;
34.
35.
36.
37.
38.
39.     public int contador = 0;
40.     public string cargarNombre;
41.     public string cargaRatio;
42.     public string cargarFactor;
43.     public bool cargarSaltar;
44.     public float calculo;
45.
46.     public InputField iglu;
47.     public InputField ihc;
48.     public InputField ins;
49.     public InputField fecha;
50.
51.
52.
53.     public int cuentaDias = 0;
54.
55.     public int contadordias;
56.     public int contadorfelicidad;
57.
58.     public bool activo = false;
59.     public int cuenta = 3;
```

```

60.
61.
62.
63.
64.
65.
66.     public void FuncionContador()
67.     {
68.         contador++;
69.         activo = false;
70.     }
71.
72.     public void cargarRatioyFactor()
73.     {
74.
75.
76.
77.
78.         //cargarNombre =
    CargarGuardar.cargarGuardar.nombre;
79.         cargaRatio = CargarGuardar.cargarGuardar.ratio;
80.         cargarFactor = CargarGuardar.cargarGuardar.factSens
ibilidad;
81.         //calculo = calculo + cargaRatio;
82.
83.         nratio = float.Parse(cargaRatio); // Convertimos el
string en float
84.         nfactor = float.Parse(cargarFactor);
85.
86.
87.
88.
89.         /*Debug.Log(cargarNombre);
90.         Debug.Log(cargarFactor);
91.         Debug.Log(cargaRatio);
92.         Debug.Log(nhidratos);
93.         Debug.Log(nprevia);
94.         Debug.Log(calculo);
95.         Debug.Log(today);*/
96.     }
97.
98.     public void cambioActivo()
99.     {
100.         activo = true;
101.     }
102.
103.
104.
105.     // Start is called before the first frame update
106.     void Start()
107.     {
108.         Debug.Log(contador);
109.
110.         NotificationCenter.DefaultCenter().AddObserver(this
, "GuardarHidratos");
111.         NotificationCenter.DefaultCenter().AddObserver(this
, "GuardarGlucosa");
112.         NotificationCenter.DefaultCenter().AddObserver(this
, "GuardarInsulina");
113.         NotificationCenter.DefaultCenter().AddObserver(this
, "GuardarFecha");

```

```

114.         NotificationCenter.DefaultCenter().AddObserver(this
115.         , "CuentaDias");
116.         NotificationCenter.DefaultCenter().AddObserver(this
117.         , "CuentaFelicidad");
118.
119.         fecha.text = today.ToString("dd/MM/yyyy");
120.
121.         TextHidratos = GameObject.Find("hc");
122.         hidratos = GameObject.Find("introduceHc");
123.
124.         TextGlucosa = GameObject.Find("glucosa");
125.         glucosa = GameObject.Find("introduceGlucosa");
126.
127.         TextInsulina = GameObject.Find("insulina");
128.         insulina = GameObject.Find("insulinaNecesaria");
129.
130.         warning80 = GameObject.Find("w80"); ;
131.         warning250 = GameObject.Find("w250"); ;
132.
133.         botonContinuar = GameObject.Find("Continuar");
134.
135.         botonContinuar.SetActive(false);
136.
137.         TextHidratos.SetActive(false);
138.         hidratos.SetActive(false);
139.
140.         TextGlucosa.SetActive(false);
141.         glucosa.SetActive(false);
142.
143.         TextInsulina.SetActive(false);
144.         insulina.SetActive(false);
145.
146.         warning80.SetActive(false);
147.         warning250.SetActive(false);
148.
149.     }
150.
151.     public void GuardarFecha(Notification notification)
152.     {
153.         CargarGuardar.cargarGuardar.fecha = today.ToString(
154.         "dd/MM/yyyy");
155.         CargarGuardar.cargarGuardar.Guardar();
156.     }
157.
158.     public void GuardarHidratos(Notification notification)
159.     {
160.         CargarGuardar.cargarGuardar.glucosa = iglu.text;
161.         CargarGuardar.cargarGuardar.Guardar();
162.     }
163.
164.     public void GuardarGlucosa(Notification notification)
165.     {
166.         CargarGuardar.cargarGuardar.hidratos = ihc.text;
167.         CargarGuardar.cargarGuardar.Guardar();
168.     }
169.
170.
171.     public void GuardarInsulina(Notification notification)

```

```

172.     {
173.         CargarGuardar.cargarGuardar.insulina = ins.text;
174.         CargarGuardar.cargarGuardar.Guardar ();
175.     }
176.     }
177.
178.     public void CuentaDias(Notification notification)
179.     {
180.         CargarGuardar.cargarGuardar.cuentaDias = CargarGuard
181.         dar.cargarGuardar.cuentaDias + contadordias;
182.         CargarGuardar.cargarGuardar.Guardar ();
183.         Debug.Log(cuentaDias);
184.     }
185.     public void CuentaFelicidad(Notification notification)
186.     {
187.         CargarGuardar.cargarGuardar.cuentaFelicidad = Carga
188.         rGuardar.cargarGuardar.cuentaFelicidad - contadorfelicidad;
189.         CargarGuardar.cargarGuardar.Guardar ();
190.         Debug.Log(contadorfelicidad);
191.     }
192.     // Update is called once per frame
193.     void Update ()
194.     {
195.
196.
197.
198.         if (contador == 0)
199.         {
200.             TextHidratos.SetActive (true);
201.             hidratos.SetActive (true);
202.
203.             if (activo == false)
204.             {
205.                 botonContinuar.SetActive (false);
206.             }
207.             if (activo == true)
208.             {
209.                 botonContinuar.SetActive (true);
210.             }
211.         }
212.     }
213.     else if (contador == 1)
214.     {
215.         TextHidratos.SetActive (false);
216.         hidratos.SetActive (false);
217.         TextGlucosa.SetActive (true);
218.         glucosa.SetActive (true);
219.
220.
221.
222.
223.         if (activo == false)
224.         {
225.             botonContinuar.SetActive (false);
226.         }
227.         if (activo == true)
228.         {
229.             botonContinuar.SetActive (true);
230.         }

```

```

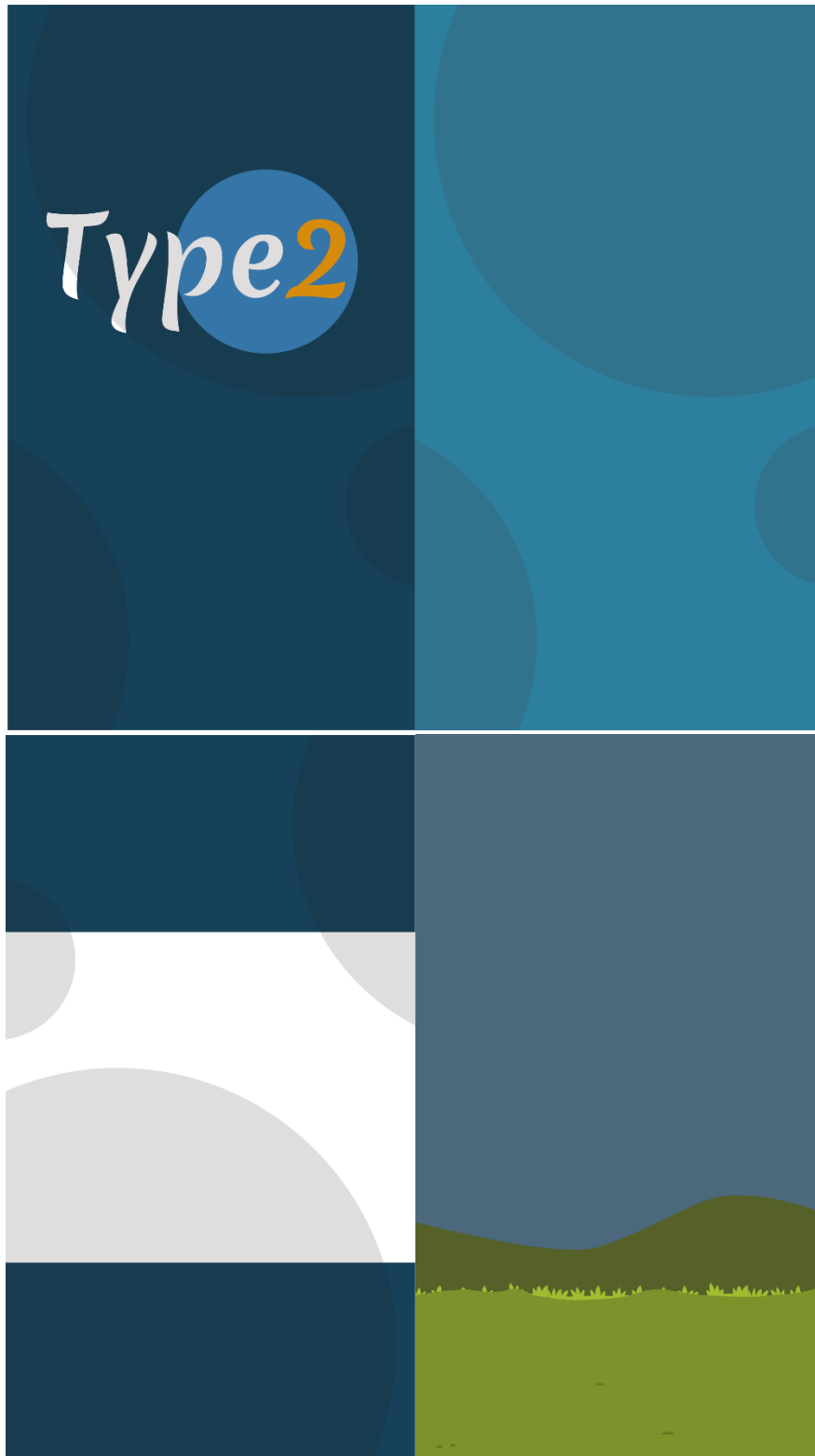
231.
232.
233.
234.
235.
236.
237.     }
238.     else if (contador == 2)
239.     {
240.         nprevia = float.Parse(iglu.text);
241.         if (nprevia > 250)
242.         {
243.             warning250.SetActive(true);
244.             contadorfelicidad = 1;
245.         }
246.         else if (nprevia < 80)
247.         {
248.             warning80.SetActive(true);
249.             contadorfelicidad = 1;
250.         }
251.         else
252.         {
253.             contador = contador + 1;
254.         }
255.     }
256.     }
257.     else if (contador == 3)
258.     {
259.         warning80.SetActive(false);
260.         warning250.SetActive(false);
261.         TextGlucosa.SetActive(false);
262.         glucosa.SetActive(false);
263.         TextInsulina.SetActive(true);
264.         insulina.SetActive(true);
265.
266.         nhidratos = float.Parse(ihc.text);
267.         //nprevia = float.Parse(iglu.text);
268.
269.         // (ratio * hc) +- (glucosa previa - valor
deseado) / factor de sensibilidad) - Insulina activa
270.
271.         calculo = ((nratio * nhidratos) + (nprevia - 10
0) / nfactor);
272.         ins.text = calculo.ToString();
273.
274.
275.
276.     }
277.     else if(contador == 4)
278.     {
279.
280.         contadordias++;
281.
282.         NotificationCenter.DefaultCenter().PostNotifica
tion(this, "GuardarHidratos");
283.         NotificationCenter.DefaultCenter().PostNotifica
tion(this, "GuardarGlucosa");
284.         NotificationCenter.DefaultCenter().PostNotifica
tion(this, "GuardarInsulina");
285.         NotificationCenter.DefaultCenter().PostNotifica
tion(this, "GuardarFecha");

```

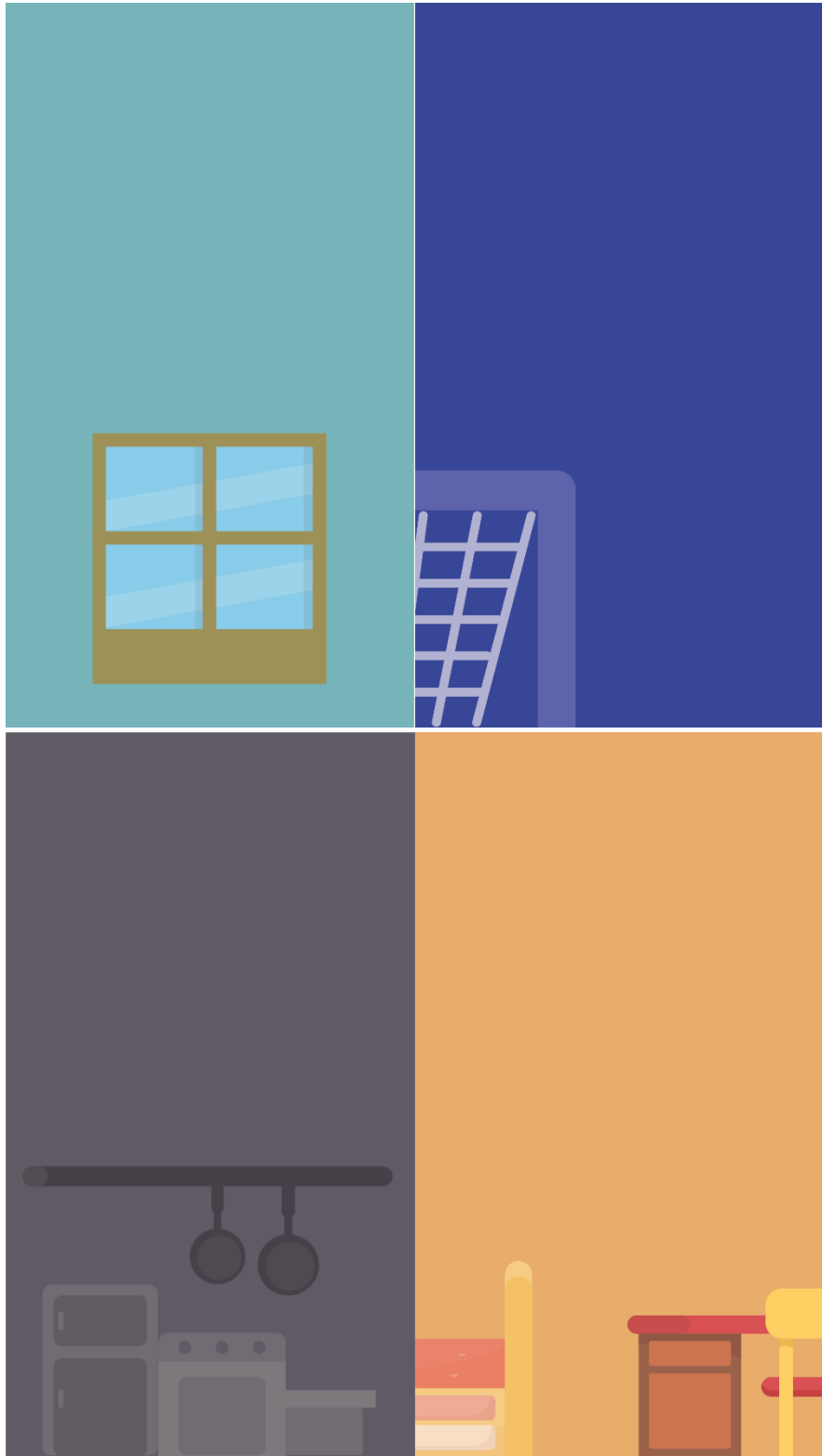


```
286.         NotificationCenter.DefaultCenter().PostNotifica
tion(this, "CuentaDias");
287.         NotificationCenter.DefaultCenter().PostNotifica
tion(this, "CuentaFelicidad");
288.         SceneManager.LoadScene("Principal");
289.
290.     }
291. }
292.
293.
294.
295.
296. }
```

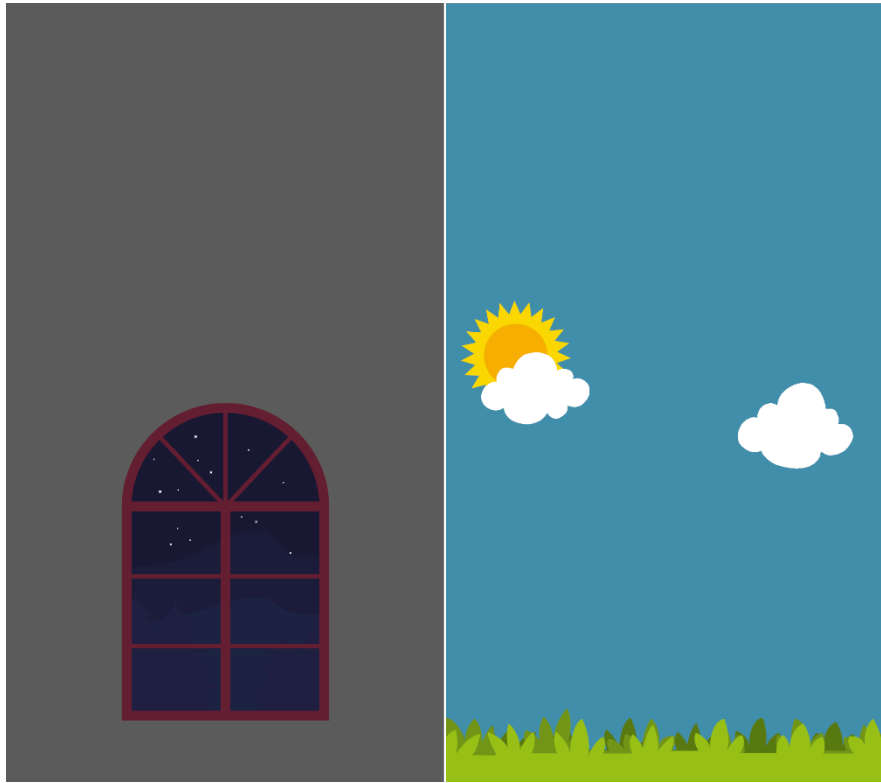
## 8. APPENDIX B – VIDEOGAME ART



Start / Option / Tutorial / Start Backgrounds



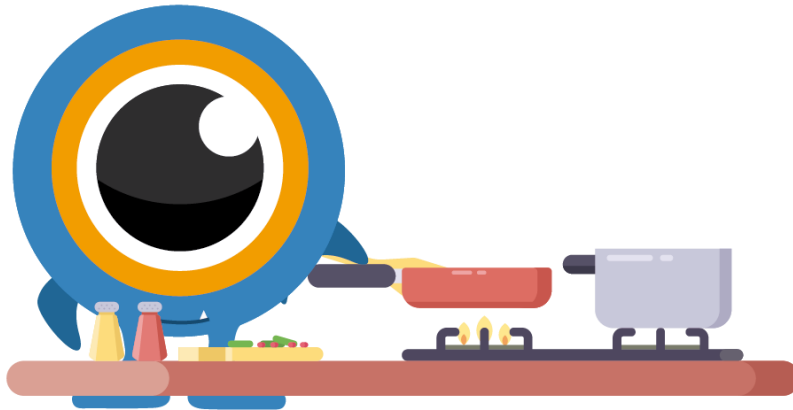
Values Background



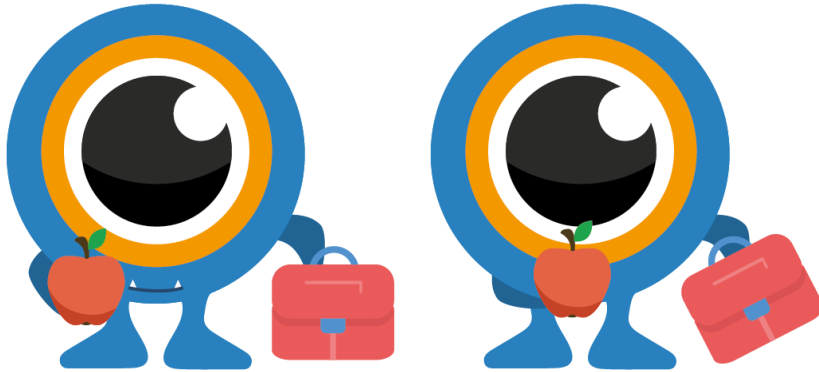
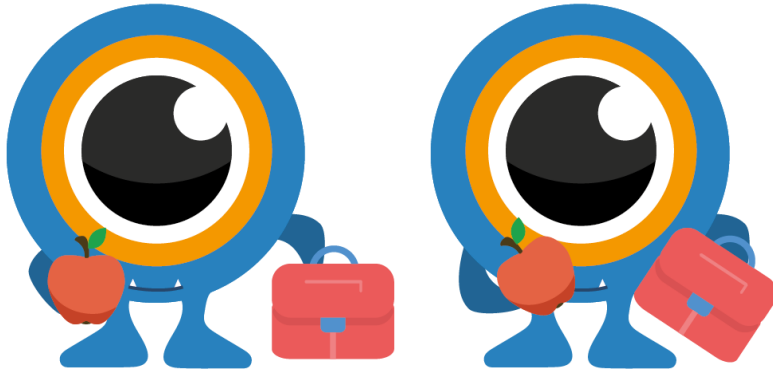
Values and Open Egg Background



Breakfast Animation



Cook Food Animation



Tea Time Animation

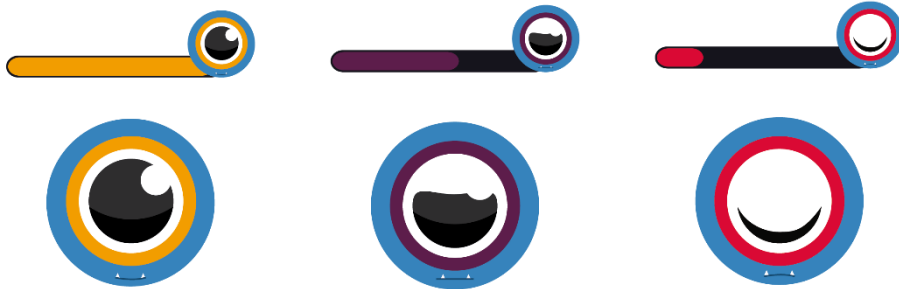
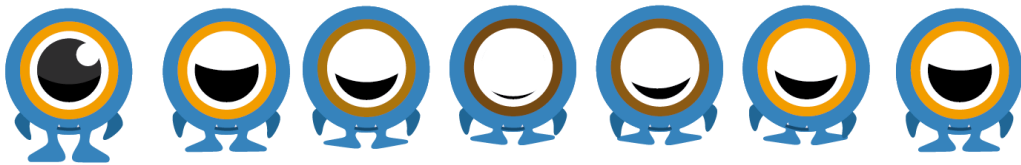
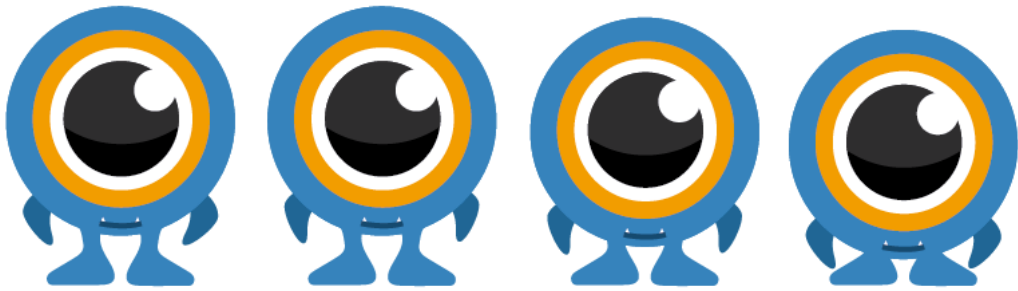


Snack Animation

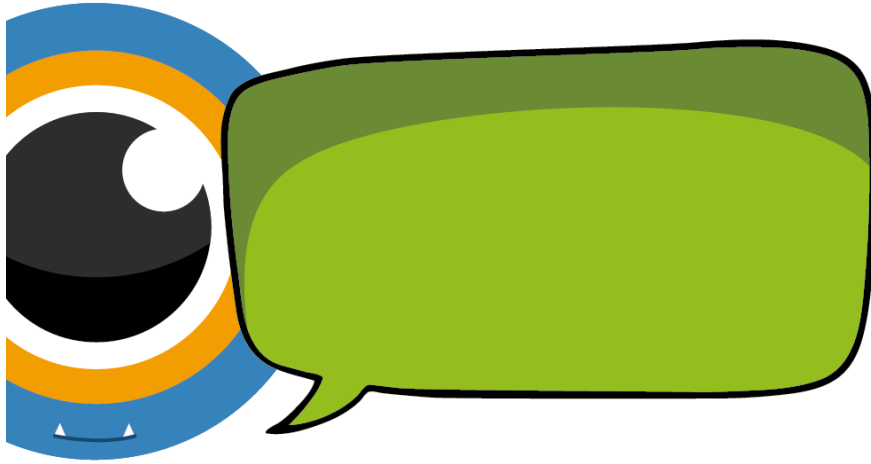


Dinner Animation

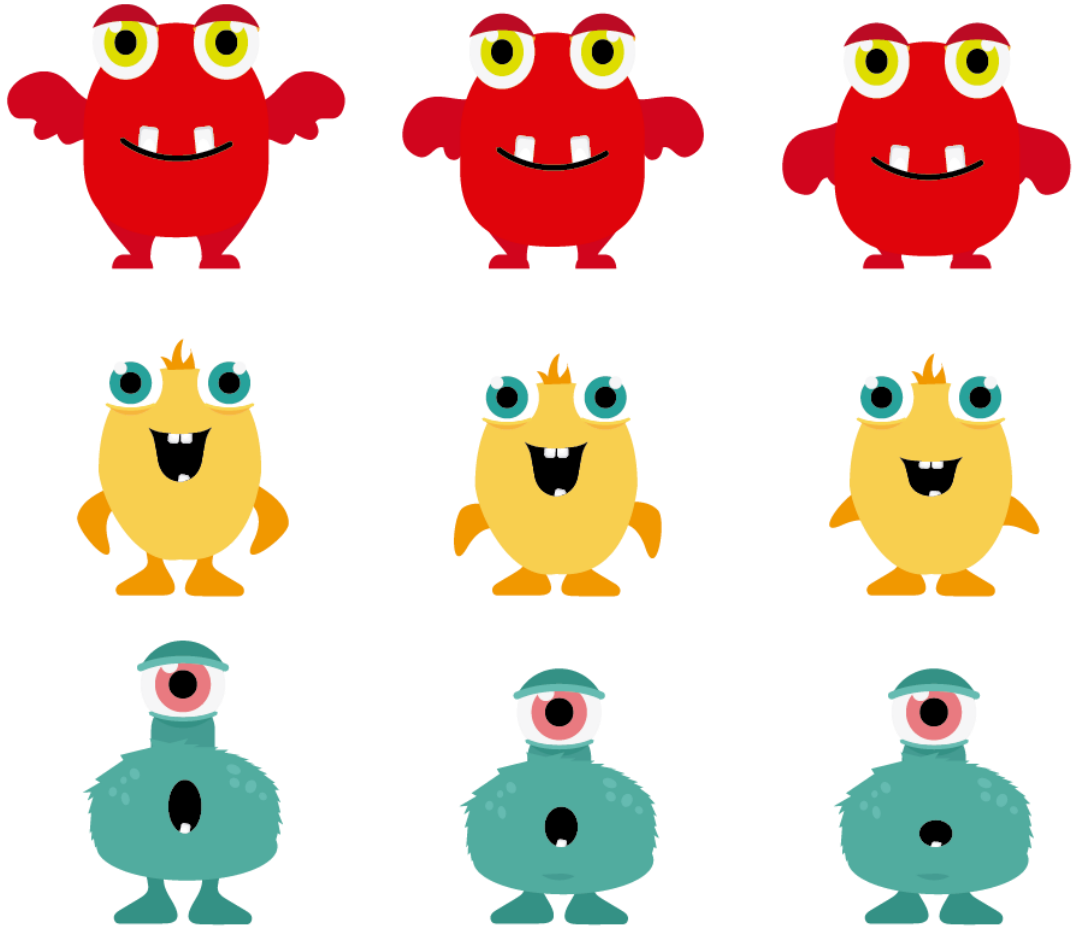




Happiness Bar



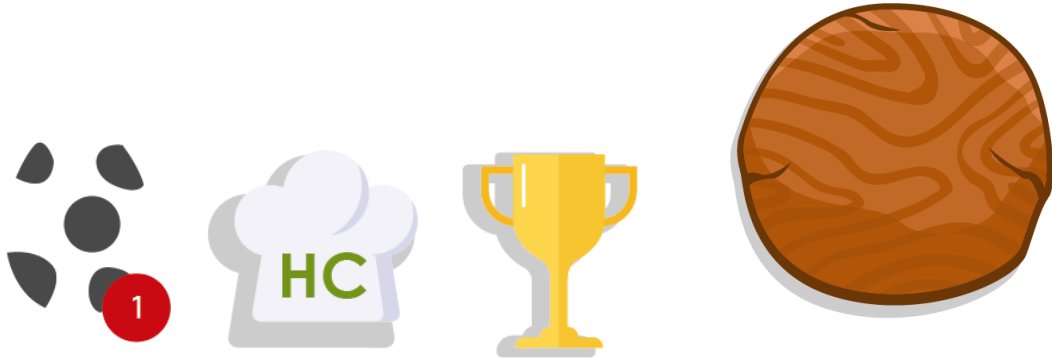
Talking Glu Animation



New Creatures Animation



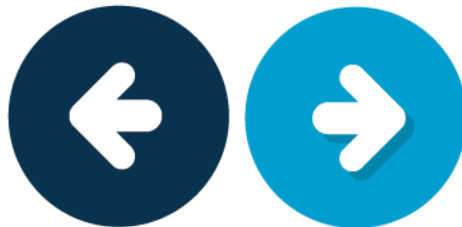
Open Egg Animation



Egg, Hc, Trophy and Add Values Buttons.



Options Buttons



Back and Next Buttons