Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE GRADO

La Gamificación como método de aprendizaje

Autor: Enrique Perez Soler

Director: Cristian Moral Martos

MADRID, JULIO 2019

ÍNDICE

INTRODUCTION

We sometimes don't see it or, maybe, we don't want to admit it, but there are issues with the current educational systems all around the world. It's true that not all are the same but it's unmistakable the fact that they all derive from the same way of ethical thinking that was settled in predominant historical periods of prosperity and also war.

In an education survey conducted by the World Innovation Summit for Education[1] (WISE) in 2015 (figure 1) [1], different countries were asked if they were overall satisfied with their educational system and it was made clear that a 73% of the countries subjected to the survey were unsatisfied with it.

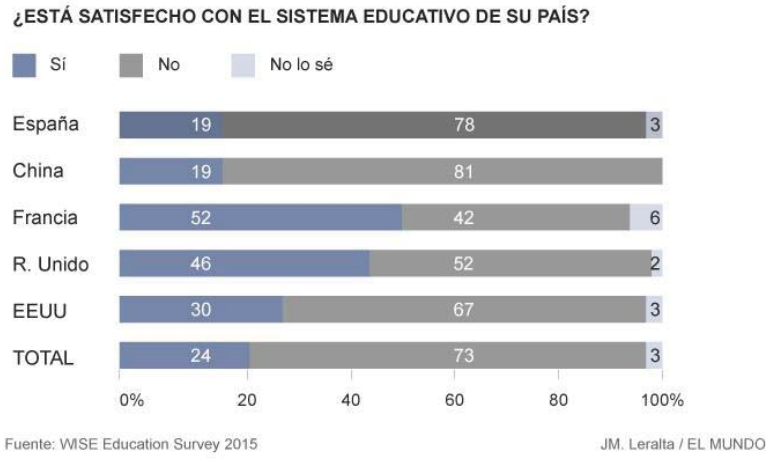


Figure 1.

Some of the current issues we find in educational systems have to do with the grading system itself. As addressed by Education.com's article *Are traditional grades a thing of the past* [2], "traditional factors like grading, extra credit, late work or class participation represent life skills which, while important, don't necessarily reflect a student's content knowledge". They also referred to the way certain work is evaluated explaining that "students' tests, quizzes and projects make up 90% of their knowledge grade, while practice work comprises only 10%."

That grading system is often subjected to a possibly outdated education system that many countries don't seem to realize delays the learning of the newer generations of students. In an article by *Instituto Inspirare*[2] director, Anna Penido [3], she addresses how, in Brazil, the education system is outdated with regard to the upcoming generation of children born in the turn of the millennium and whose future is influenced every day more and more by growing sectors like the technological and the international relationships; she utilizes the phrase "Just as the futures of students depend on their schools, so do the future of schools depend on the students",

[1] International initiative aimed at transforming education through innovation.
[2] Brazilian education al innovation advocate institute

meaning, one has to catch up to the other if the goal of preparing the students for the future that is to come would be fulfilled.

Nevertheless, not only does she addresses the issues in the educational system in Brazil but also provides solutions to it like allowing the kids to participate more in order to have a more "hands-on learning", and that can only be possible if the administration and the teachers share information and power. That way, the room is open for debate and collaboration leading to a more engaging atmosphere. By creating opportunities for listening to student's educational process, for choosing new ways of learning, for co-authoring to stimulate authorial productions and for giving a co-responsibility to the students to promote their participation in discussions and initiatives aimed at improving their daily school life; the schools would encourage applying new and innovative ideas in their educational system.

More ways to improve and innovate the education can be found in new ways of teaching like the Japanese multiplying method for kids to remember the carry[3] value or learn how to multiply in a different way (Figure 2) (You can find the explanation of the method here https://tapintoteenminds.com/japanese-multiplication/)

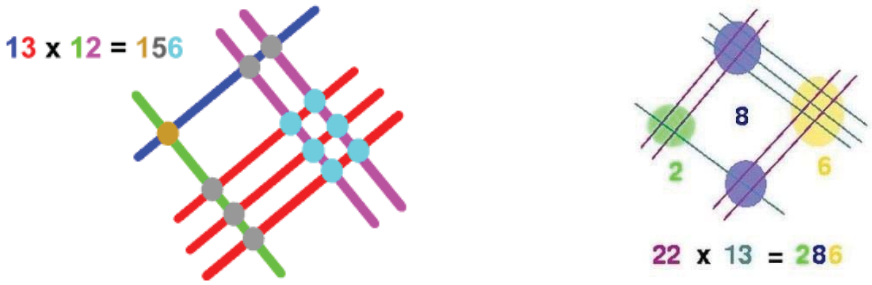


Figure 2.

Or like in the Algorithm Based o Numbers method for mathematical calculations [4] (ABN method), which appeared as an answer to the issues with our calculating system like a deficient mental calculation capacity that followed it. The method tries to grant the students or the mathematician the capacity to solve a certain problem in an "open way" that is, letting him or her use the most suitable strategy and approach that he or she deem most effective, even if it's not the same one other students made. Of course, always using a numerical solution, using numbers no matter their size or complexity (real, natural, rational or even binary!)

From all the possible methods to improve the educational systems we've found out there, there's one in particular that we felt allowed us to use our computer engineering skills and knowledge to prove the effectiveness of this method by creating a learning platform that made use of it, and that is the concept of Gamification.

Gamification is described in Wikipedia [5.1] as "the application of game-design elements and game principles in non-game contexts", meaning, it's a channel or a mean that uses games and game theory as a way of representing or addressing a

[3] A digit that is transferred from one column of digits to another column of more significant digits in arithmetic operations

problem/procedure to be presented to an audience/user. There is no actual date recorded of where nor when it started to be used but since the concept appear in 2008 in the context of computer software it has found its application in a range of different fields including army training, employee skill testing and even product quality testing.

But there's a field in particular it has found another use in that we thought would have a bigger and more everlasting impact, and that is education. Often referred to as "Gamification of Learning", this use of the concept tries to provide a new way of teaching in a more visual and somewhat appealing way. Given the screen-influenced students there are today as a result of the growing technological era that we have been witnessing ever since computers began to be domestically commercialized; we found an opportunity to encourage this sort of educational method as a better way of teaching this new generation and hopefully the ones to come.

Gamification of Learning has approached the educational efficiency by using a system based on "achievement rewards" to congratulate the students for their involvement in a certain activity or resolution of a problem presented in order to encourage them to continue learning and be more competitive. Nevertheless, many have developed a criticism towards it as it might cause other behavioral problems in younger users such as difficulties when socializing, permanent physical damages to the eyes and a sense of competition that might lead to unaccomplished self-esteems. Therefore, it's true that it can't be applied as an alternative to the conventional education methods in this field yet but it's on its way towards it and our goal with this project is to make our contribution to that cause.

In this project we intend to apply this specific concept use to design a learning platform consisting of a basic narrative-driven game where the main character, the user/player, is an alien princess sent to Earth by her father to learn about altruism exercising her problem solving skills using logic, intuition and a little math to prepare her so when the time comes for her to be queen she'd be ready to be a good one. The most common targeted age-range this sort of learning methods is oriented to are those in the ages of prime learning that is between primary school and high school, both included. Nevertheless, the intended focus on visual representation of the problems provide a trial-error learning process that allows the targeted range of age to be widened. But primarily, this platform will try to teach the students the values of solidarity and using the knowledge they have and will learn through the system in the resolution of problems with the hope that some might even be applicable to real life situations.

PREVIOUS WORK

As explained previously, there's no actual factual proof of a first implementation of the concept of Gamification in the educational field or other. However, even though the term "gamification" first appeared online in 2008 it was used earlier in the field of entertainment in Richard Bartle's 1978 first interactive multi-user 'game' MUD1 (figure 1) and Will Wright's 1989 SimCity (figure 2) [5.2], a city-building simulation game were the players were given the role of mayor of a city whose tasks were, as defined in Wikipedia, "to build up a city, providing basic transit links, power, and simplistic service needs for their residents, while watching out for problems and dealing with a multitude of disasters, most of which are based on real-life disasters. Alongside the option to make a city from scratch, the game also features scenarios that task players to oversee a pre-built city and deal with specific issues that it faces, most of which require the player to rebuild after a disaster."

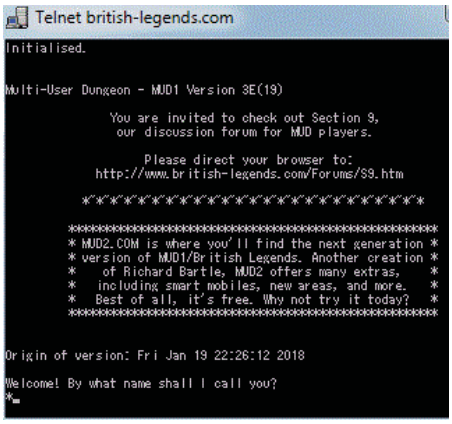


Figure 1: Screenshot of the MUD1

Figure 2: Screenshot of 1989 SimCity

Moreover, the concept of gamification, in other words, the application of a rewarding system that celebrated the achievements that the targeted users obtained, also took over other fields besides the educational and entertainment ones as it also provided certain improvements that could benefit them. Organizational productivity and positive engagement with employees were attractive aspects to business corporations as well as governments and other enterprises that sought a positive working environment.

Nevertheless, the most researched field application was naturally in the entertainment and educational aspect. It wasn't until 2002 that the training-and-learning-through-simulation oriented games, called Serious Games, came to be and established itself as the reference method for that purpose. It's characteristic and different from other approaches to gamification in that it's a subgenre of "serious storytelling" and is subsequently heavily story-driven, in other words, it enhances the teaching through a narrative that engages the user. Some examples include medical and flight simulations (figure 3), but already back in 1999 LeapFrog Enterprises introduced the LeapPad (figure 4), a hand-held paper-based book that kids could interact with and it became one of the first examples of the use of Serious Games.

Figure 3: Example of a flight simulation

Figure 4: Girl using a 1999 LeapPad

As I researched on the topic, I found no better way to portray the evolution of the concept throughout time that with Dustin Bethel's article *A brief history of Gamification* [6], where not only does he refers to an info graphic describing the history of gamification published by TechnologyAdvice[4] that discussed how the genesis of gamification techniques dated back to the late 1800s; but he provided a chronological list of the fundamental events that coined the term that we know and use today. Here I list those events:

1896 – The practice of rewarding started in the late 1800s. Stamps were sold by marketers to retailers in order to reward loyal customers.

1973 – Charles Coonradt began a consulting firm called "The Game of Work" which incorporated feedback loops in sports into professional workplaces. His work focuses on encouraging employee engagement, involvement, and motivation

1979 – MUD1 was created — the very first multi-user virtual game.

1980 – "What Makes Things Fun to Learn: A Study of Intrinsically Motivating Computer Games" was published by Thomas Malone, explaining what makes virtual games to enticing.

1981 to 1987 – Large companies like American Airlines, Holiday Inn, and National Car Rental implements programs to reward their customers and encourage brand loyalty.

1990 to 1996 – Household gaming gains popularity, and Richard Bartle publishes "Who Plays MUAs," standing for "multi-user adventures," dividing video gamers into four different types.

[4] TechnologyAdvice: an online platform that allows users to find, compare and buy business tools

2002 – A gaming initiative is put in place and links the electronic gaming industry to professions, including training, health, education, and public policy.

2003 – The term "gamification" is coined by Nick Pelling, a computer programmer (figure 5).

Figure 5: Nick Pelling at the Gamification World Congress

Figure 6: Scene of TV Show The Office

2007 – Bunchball launches a gamified website for the show "The Office" called "Dunder Mifflin Infinity", (figure 6). This website gets over eight million page views in one and a half months.

2009 – A game-based learning and educational environment called Quest to Learn welcomes a class of sixth graders (figure 7).

Figure 7: Quest to Learn engaging with kids

Figure 8: Mozilla's OpenBadges initiative

2010 – Gamification Co. hosts the first ever Gamification Summit in San Francisco.

2012 – Mozilla launches an open badge initiative to mark accomplishments online (figure 8).

2014 – It is predicted by M2 Research that gamification will grow to be a 2.8 billion dollar industry by 2016 (figure 9) [8].

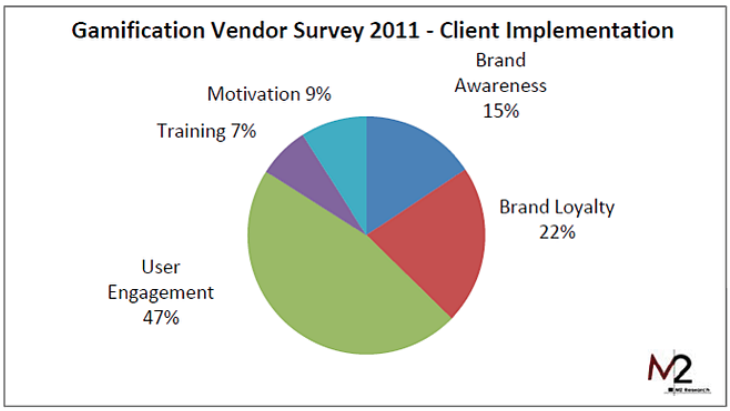


Figure 9: M2 Gamification Vendor Survey 2011

2016 – Some might not recognize it as an accomplished contribution to the gamification field, but reviewers of the popular location-based Augmented Reality game *Pokemon Go!* (figure 10), praised the game for promoting physical exercise[5](so technically the class of Physical Education). Especially to younger audiences as a way to "step out" of the recent popular indoor leisure activities promoted by the new technologies such as TV, Cinema and game consoles.

Figure 10: Gameplay Screenshot of Pokémon GO! app

[5] As said in the *Gamification* Wikipedia page in the Health section, 3rd paragraph; as example of a recent and impactful use of gamification in the entertainment industry

SOLUTION

We first decided to plan the different phases of the development of the platform having always in mind that getting to understand the tools we'd be using and the scope they had would be one of the very first things to take into account. We really wanted to learn and make use of all the advantages of the powerful tool that is *Unity* and the C# programming language needed to implement the different functionalities. *Unity* is a cross-platform game engine developed by Unity Technologies used to create three-dimensional, two-dimensional, virtual or augmented reality games as well as simulations and other experiences. It offers a primary scripting API in C# for both the Unity editor in the form of plugins, and games themselves, as well as drag and drop functionality.

Nevertheless, at the time of getting into it we realized it was of no use without a proper System Design, so we decided to research the different software project development phases to understand the thought process behind their planning. We finally created a Gantt Diagram (figure 1) to represent which phases there'd be in the development of this system as well as how long it would take and the dates of start and planned finish. To give more insight of the development in the *Work Plan* we also added a representation of the progress of each task/phase with a color that would symbolize the status of each phase: blue for Active, green for Completed, grey for Upcoming.

	Start Date	End Date	Timeline	Status
Gamification as a learning platform	02/02/2019	02/06/2019		
System Design	02/02/2019	01/04/2019		Complete
Tool Learning	05/03/2019	05/05/2019		Active
Implementation	05/03/2019	05/05/2019		Active
Technical Testing	25/04/2019	05/05/2019		Upcoming
User testing	05/05/2019	15/05/2019		Upcoming
Documentation	16/05/2019	02/06/2019		Upcoming

Figure 1. Captured stage the 27th of March 2019

- ***System Design Phase*** is where the *what, why and for whom* this system will be would be discerned as well as defining the software architecture and requirements through Software Engineering data representation tools and concepts like UML class diagrams, activity diagrams and flow charts.

- ***Tool Learning Phase*** this phase would develop almost parallel to the System Design Phase as new or better ideas for designs can appear once there is a better understanding of the tools used. It consists of learning the management and intricacies of the game engine Unity as well as the programming language C# and the physics engine and math behind. Also, the tools regarding the visual aspect of the platform to make it more attractive to the user (3D Modelling, concept and environment design, etc.)

- ***Implementation*** Phase in which the major development of the system will be done. This phase would represent most of the time spent in the project and it will begin as soon as the Tool Learning phase starts but will go on until the platform is fully implemented.

- ***Technical Testing Phase*** due to the UX-oriented nature of this project, a series of technical tests and performance capability will be implemented adjusting to the amount of time left. This phase might not be strictly necessary as in the Implementation Phase and given my programming methodology, we'd provide testing for most of the code to assure their correct performance.

- ***User Testing Phase*** the focus has shifted t include the study of whether or not this project is not only profitable but also an improvement (which is also related to the profit in the possibility of investment); this phase would showcase the response of user test through User Tests in order to further prove the concepts of gamification and study the result of this new method of learning more consistently.

- ***Documentation Phase*** where all the knowledge used, and the final Work Memory will be included and completed in detail. If the time invested in the development of such document should be questioned, we would like to notify that every single advancement/step taken in the project would be reported daily so most of the documentation and phase would be advanced alongside the project itself.

Without further ado, we began to design the game's architecture. Through its lifetime it has suffered some changes, but it never really strayed away from its initial concept: 2 scenes (counting out the main menu/introduction scene), the first being a 3D environment to better engage the audience and give them a sense of "spacial presence"; and the second being a 2D logic puzzle were the main focus and the academic goal of the project will take place.

As an introduction and a way of giving some sort of context to the narrative aspect of the game we added an initial screen (figure 2). It's just a simple menu with two buttons to change language, Spanish and English for the moment, and concept art of who'd be the main character in the story. She'd explain who she is and what the mission or goal of this whole enterprise is.

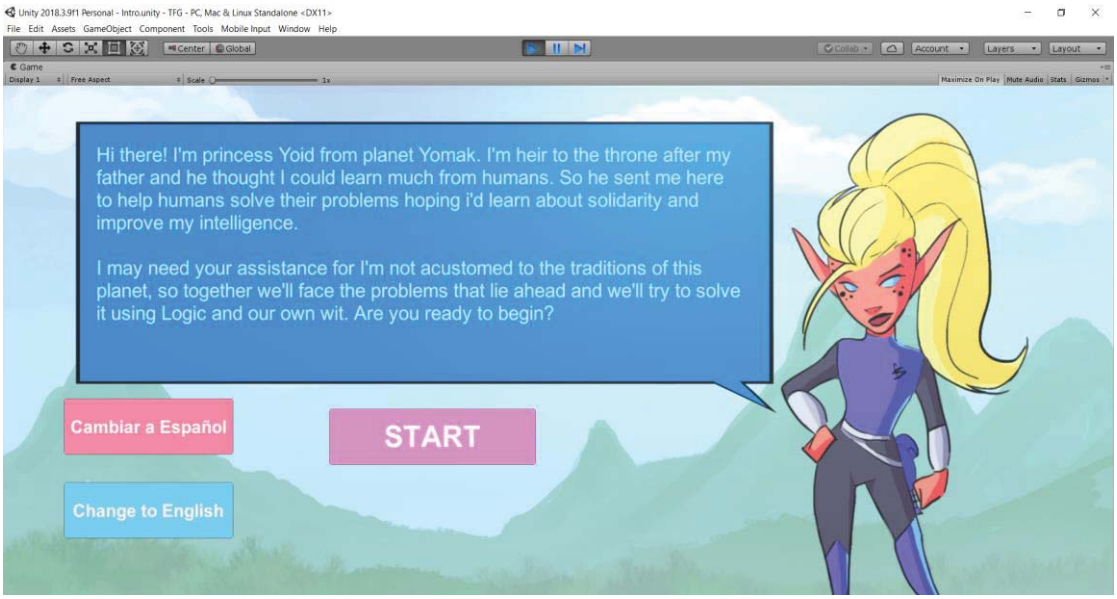


Figure 2. Initial Menu scene

The first scene will include the player as main agent or guider of the 3D experience, the environment will allow him or her to walk, jump and collide with objects, if we consider the components of this particular 3D-space engine (figure 3). The main goal of this scene is to, apart from the previously mentioned *creating an engaging first experience with the learning platform*; to also allow the player to feel like he or she is fully immersed in accompanying this new character (portrayed by Yoid from planet Yoka) through the field.

The player will be able to move her around using the arrow keys making her interact, in a small level, with the environment and making her follow the path that leads to the first problem she'd have to solve: The River crossing problem. Moreover, it was not a good practice to add too many details around the environment as it might drag the player off-course and distract him or her from the main objective in the game, especially if it has an educational purpose.

Further on, she would encounter a farmer at the end of the path with a sheep, a wolf and a cabbage that he has just bought from the town market. Once the player reaches him in the shore of the river that'd be the cue for the next scene: the 2D logic puzzle.

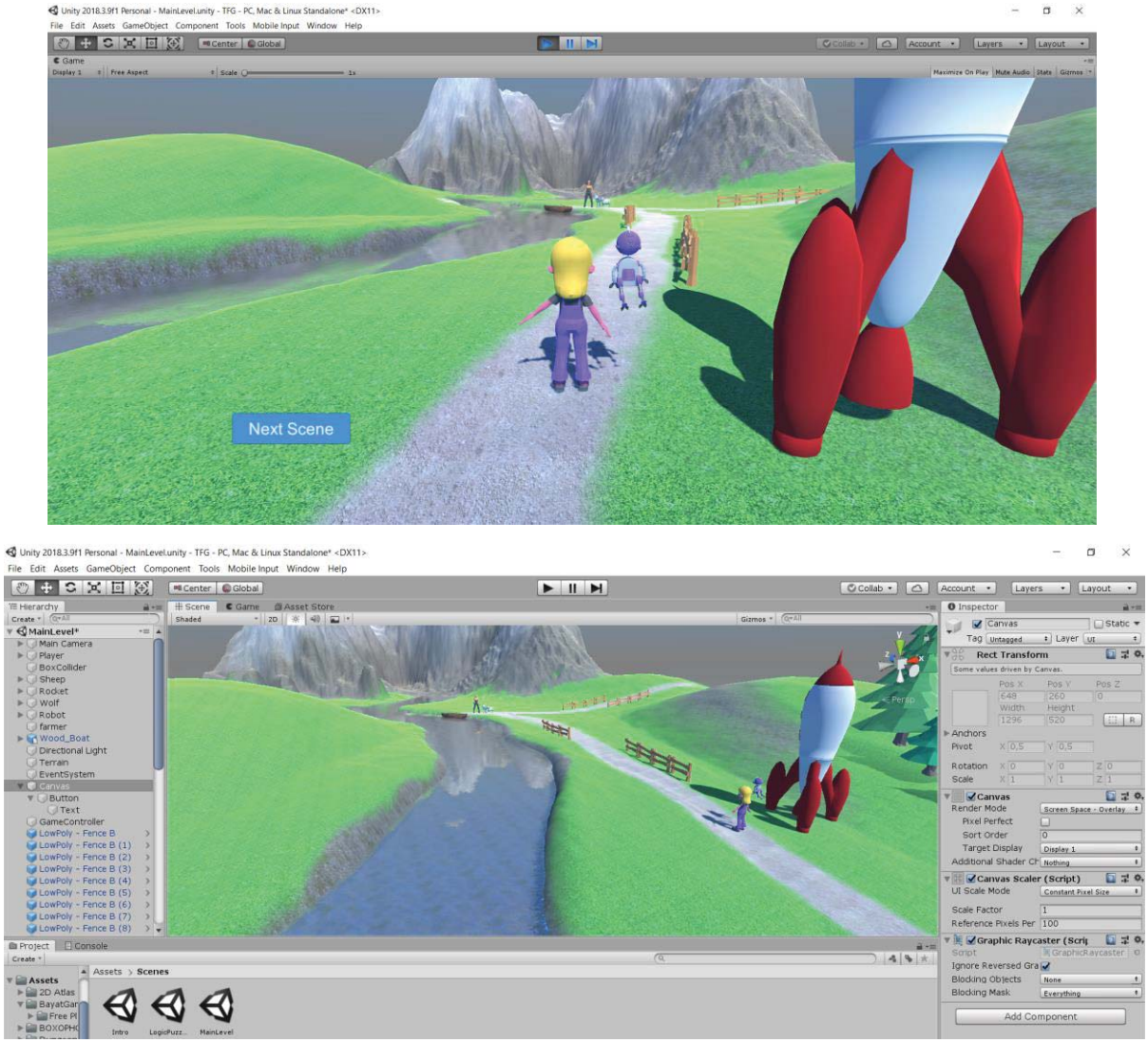


Figure 3. Screenshots of the 3D scene's Gameplay view and terrain perspective

Before we move on to the next scene, we'd like to better illustrate the different elements in the 3D scene and the different relationships there are between them through the following class diagram:

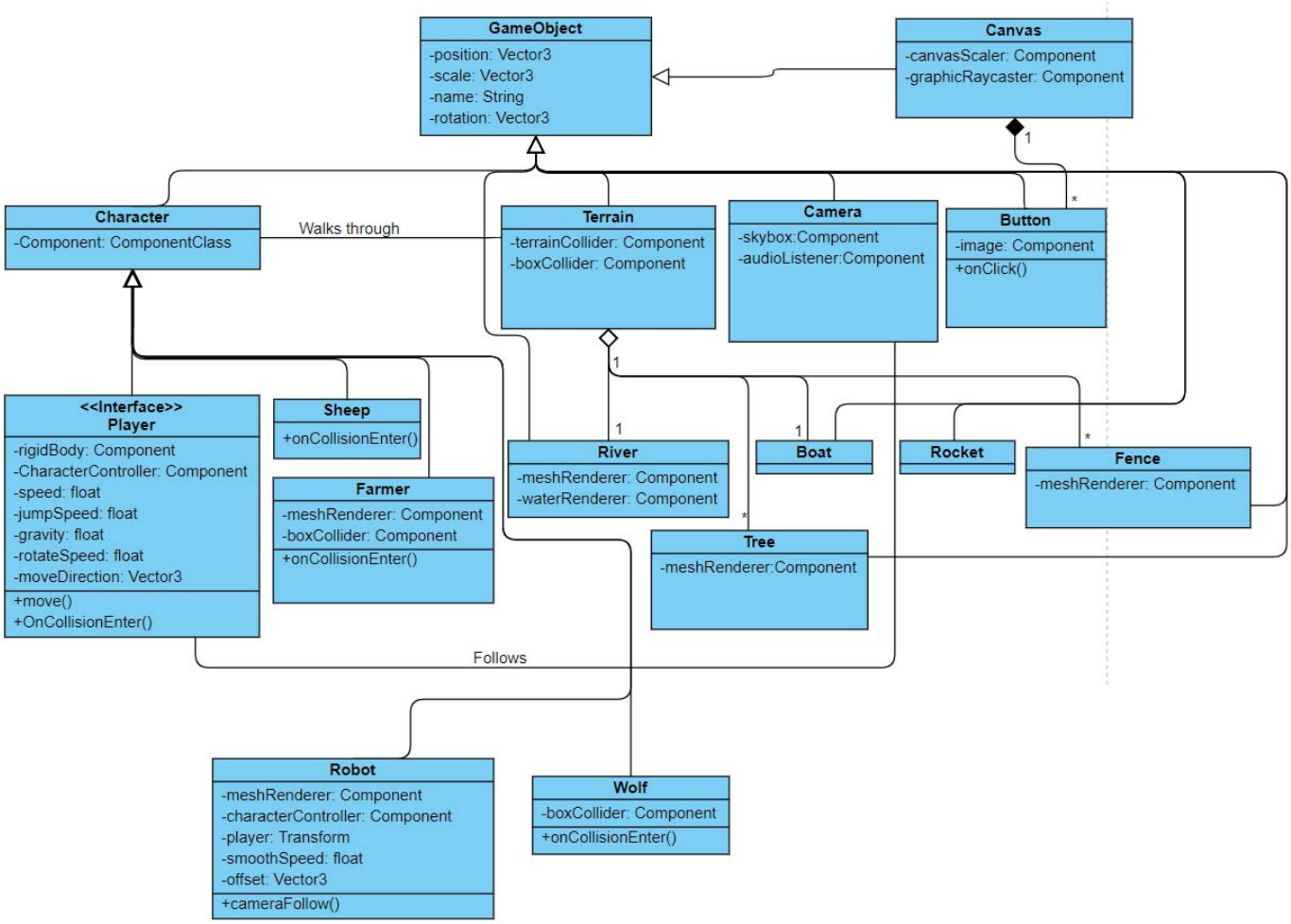


Figure 4. Class Diagram of 3D scene

This type of UML diagram was chosen because it perfectly captures the core representation of the system for its further understanding. In this diagram (figure 4), we can see the different classes, class attributes, operations and relationships of the system. With it, we can understand which classes are children to which classes, optimizing the representation of these as many have the same attributes; and by "classifying" the system and giving it a hierarchy it's easier to interpret the object-oriented inheritance concept implicit in it. Furthermore, we can showcase the different aggregations with the respective cardinalities, compositions and relations that connect different classes.

The environment was made to give a sense of real-life event and for the user to feel like they have actually encounter the problem (the next logic puzzle scene) and in order for that to happen some effects and environment elements were added to the scene, like the river or the fences in the road (figure 5)

Figure 5. Screenshot of gameplay footage to showcase the potential interaction with the environment.

Moving on to the following scenario, once the player reaches the farmer there'll be a triggering event that would make the transition to the new scene. This time it'll be a 2D scene, more oriented to the UX and with a heavily UI relied learning process.

In this scene, the player will be introduced to a text that would unravel character by character (letter by letter). This functionality was implemented this way in order to engage the player into reading the text and processing the information little by little. This option was added for a younger targeted audience, but another option to "skip the slow text" was added too for older audiences by clicking the *spacebar*; should the player want to skip the text unraveling, he/she has to simply click the spacebar and the entire text will appear followed by three buttons: *Sheep*, *Wolf* and *Cabbage* (figure 6).

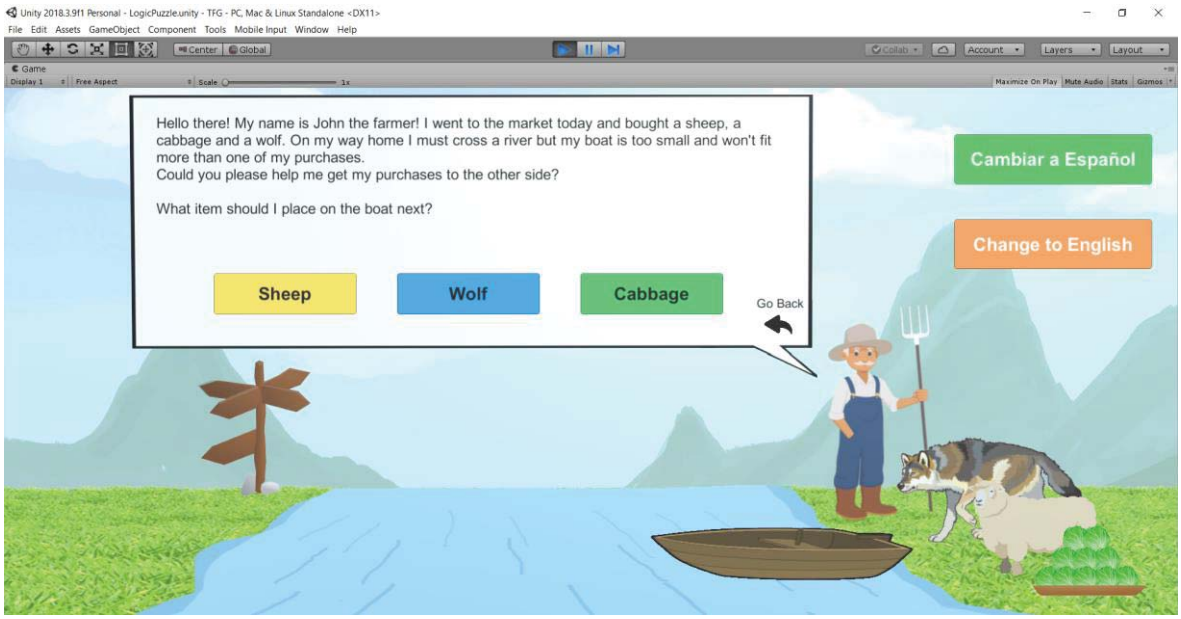


Figure 6. Screenshot of gameplay footage showcasing the first stage of the puzzle

The exercise is simple, as the farmer refers to you for help in deciding which product should be taken to the other side in his one-available seat only boat, the user/player should select from the three buttons which product to take. Once the player has decided a popup confirmation message will appear. This functionality was implemented this way in order to give the player a "on second thought" moment; making sure he or she has taken the time to think about the consequences of his or her decision.

In order to better explain the overall behavior of the puzzle game we've decided to create UML flow charts and activity diagrams as from all the available methods in the Unified Modeling Language, these represent better the dynamic of the interaction between user and system. Other diagrams and charts of this language focus more on the different agents that assist in the interaction of the final user with the different components of the system or define a certain architecture with the outside resources used or hierarchy, but in our case our system relies heavily on decision-making events and depends almost entirely in the actions of the user so elements like the hierarchy or external resources feel irrelevant.

Due to the complexity of the project and in order to allow a better understanding of the charts and implementation of the project and to prevent repetition, the explanation of the solution will be simplified to an iteration of the final stages of the game. Once that's explained the rest will follow the same behavior with mild changes. To begin with, here are the flow charts of the relevant methods that belong to the *AreYouSure* class as they control the actions of the confirmation menu:

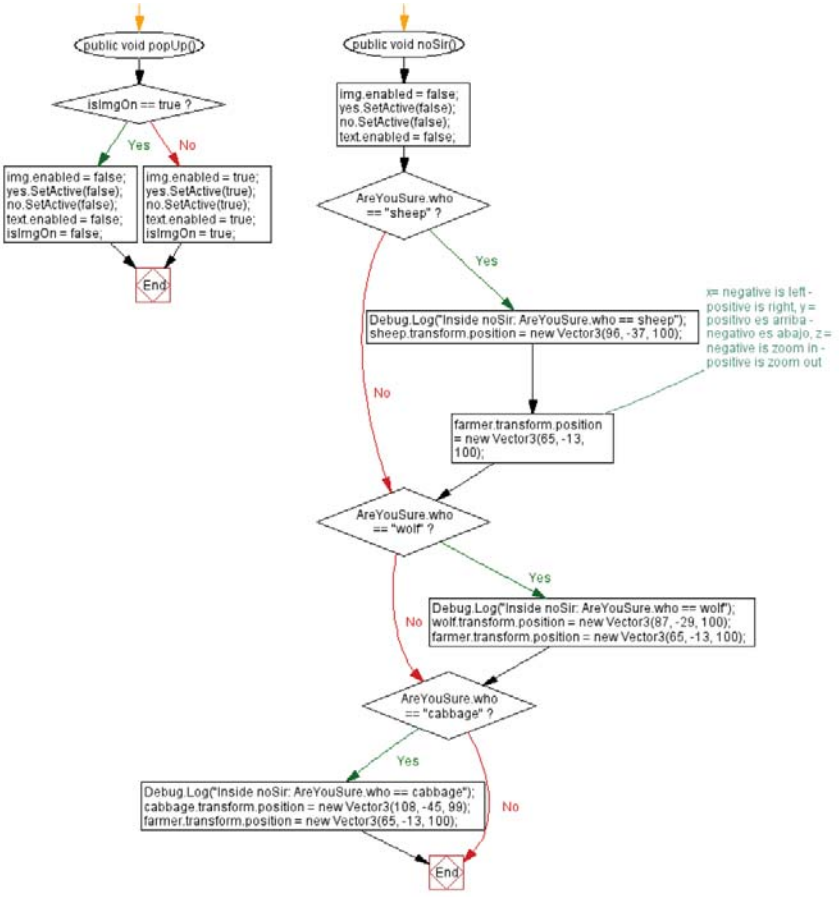


Figure 7. Implementation of the methods *popup()* and *noSir()*

The *popup()* method (figure 7) is in charge of enabling or disabling the first confirmation menu screen (*AreYouSure*) by checking the Boolean value of *isImgOn* ("is image on" referring to the image of the menu, the name of the component that the menu is made from).

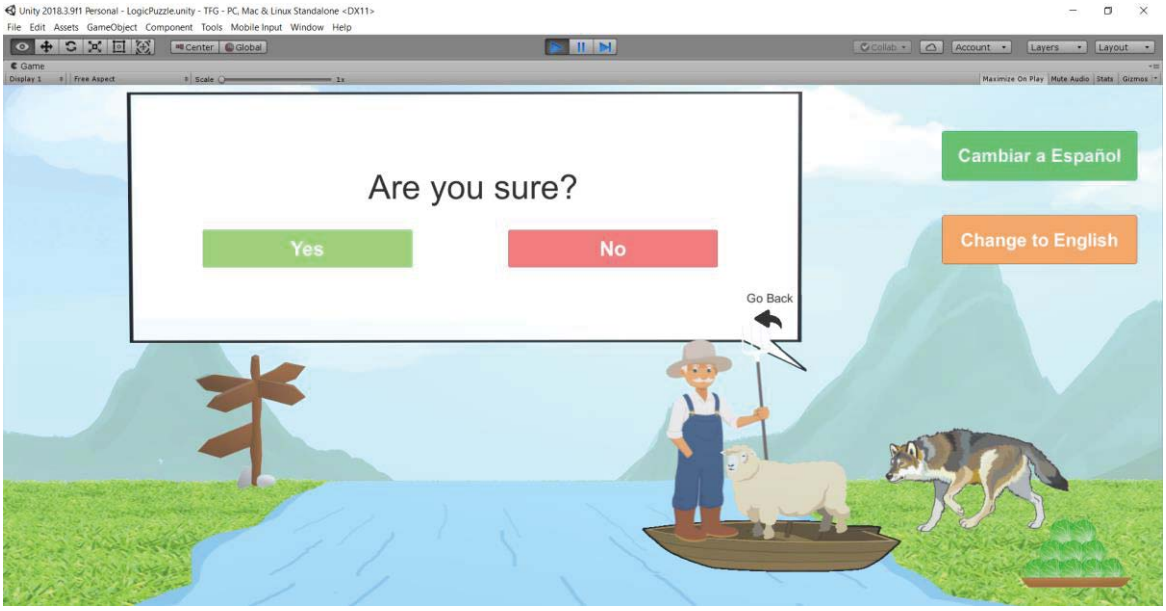


Figure 8. Screenshot of gameplay first selection menu screen.

Once the confirmation menu is enabled, the user has the choice of confirming the selection by clicking the YES button or returning by clicking the NO button. In this second case, the function *noSir()* (figure 7) is executed and after it disables the menu it checks the value of the selected purchase (*Sheep*, *Wolf* or *Cabbage*), and in each case it returns the selected purchase and the farmer back to their initial positions. It hasn't been mentioned but when a purchase is selected via their buttons (figure 8), it and the farmer are loaded into the boat as to further stress what would be left behind as the farmer takes that purchase; this was done in order to give the user a chance of realizing their potential mistake in their decision. That's why when the NO button is clicked in the confirmation menu it would take those two elements off the boat.

Should the player click YES in the confirmation menu *AreYouSure* (figure 11), the system will again evaluate the *who* global variable to see which purchase must be taken to the other shore with the farmer and the boat; and before that it also makes sure the menu is disabled to make way to the following screen with instructions to the player. As you can see, before it takes it checks the *AreYouSure.who* variable, it checks first the *who2* global variable, this one is a of Boolean type and it checks the stage of the system.

Should a purchase had already been taken to the other shore, if you understand the workings of the puzzle you'd realize in the first stage only the sheep can be taken to the other shore as if the cabbage is taken the wolf and the sheep would be left behind; and if the wolf is taken the sheep would be left with the cabbage. Therefore, once the system checks the stage the system is in, it decides which path to take. If it's the first stage and the wolf or the cabbage is selected, when clicking YES in the confirmation menu an error message will popup as it shouldn't be allowed to make

that decision for they would lose the game (figure 9). We decided not to work with any sort of "lives-based system" or similar as the main goal of the game is to allow the player to realize his or her mistake and correct it; that's why even if the player gets it wrong, we allow him or her to rethink his or her decision.

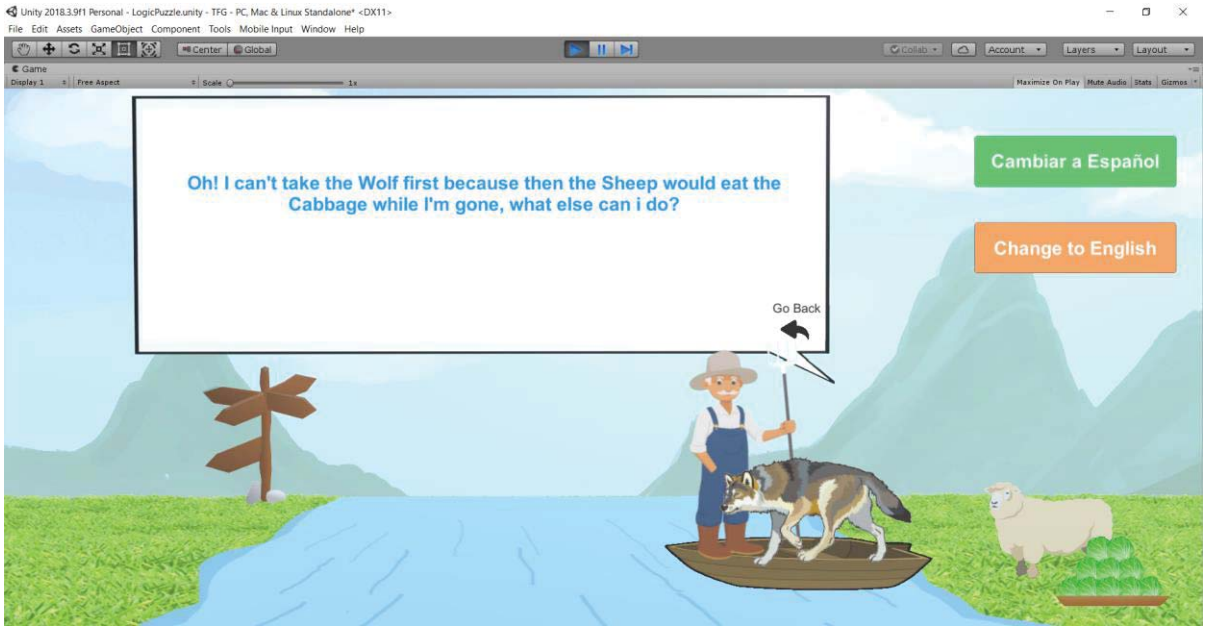


Figure 9. Screenshot of error message that shows when the player has chosen to take the wolf first

Once the player realizes only the sheep could be taken to the other shore without loosing any purchases in the process, the menu screen will showcase a new menu tailored to the solution (Figure 10) making way for the next "stage" of the problem

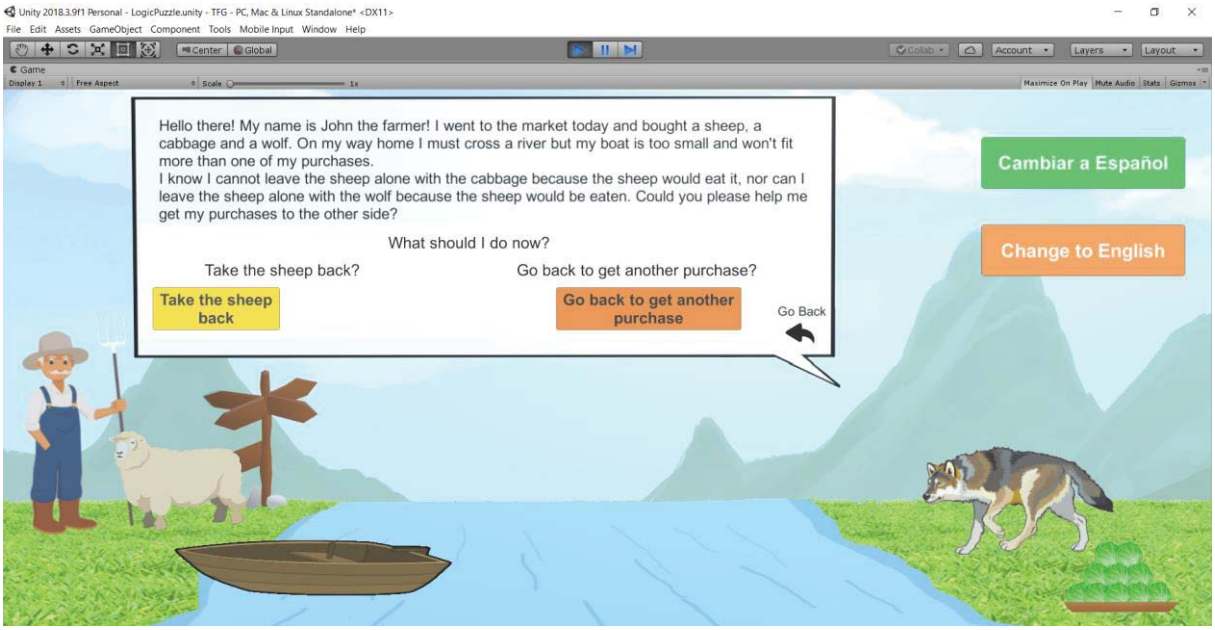


Figure 10. Screenshot of second menu (*combo*) that appears when sheep taken

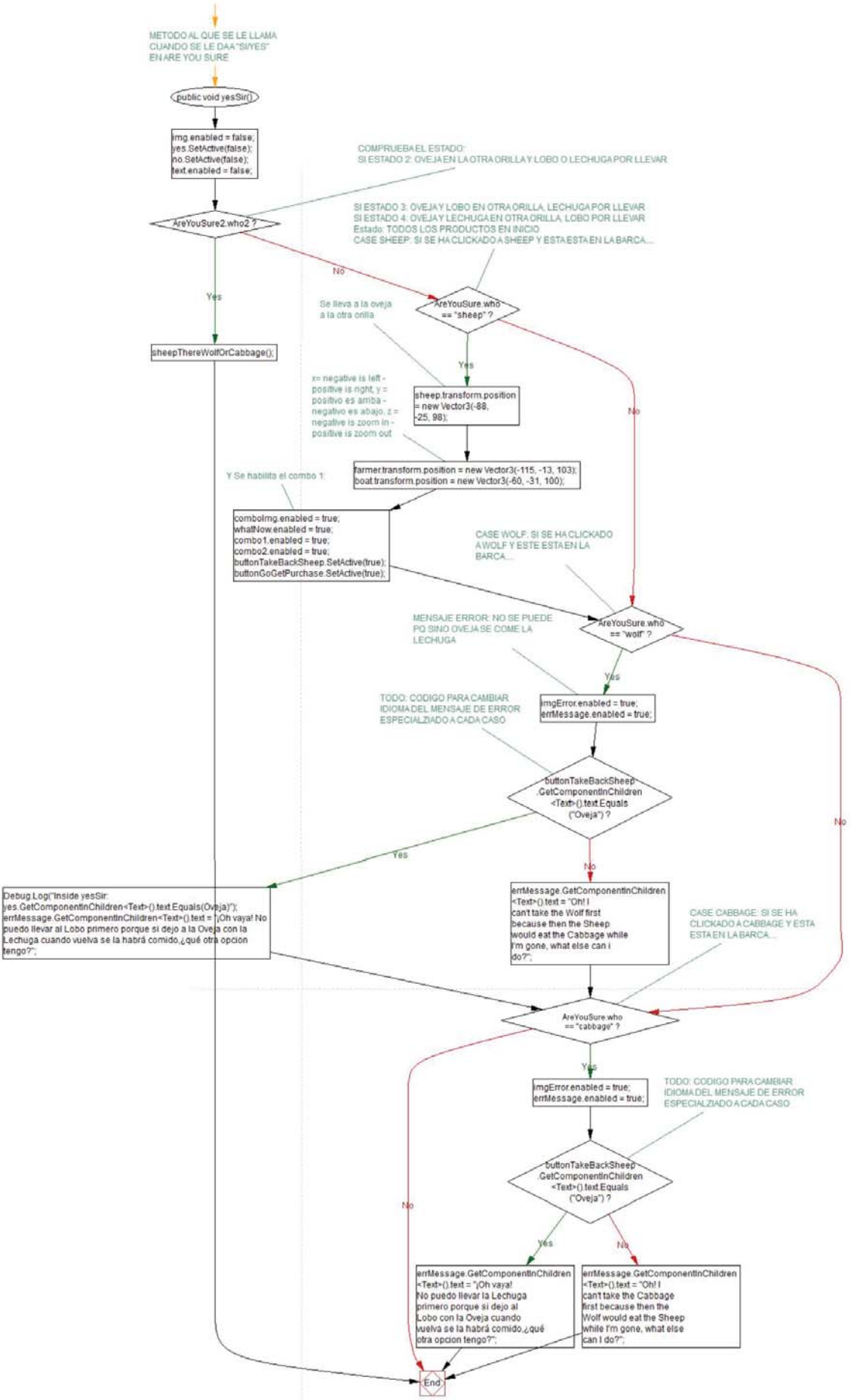


Figure 11. UML Flow chart of the *yesSir()* function of the *AreYouSure* class

As mentioned before, once the sheep is taken to the other shore the system offers the player a second menu (figure 10) where he or she can choose to take the sheep back or go back to the other shore and take a new purchase. Given that taking back the sheep at this stage of the puzzle will lead nowhere as it'd revert to the initial stage, we'll only study the returning to get a new product.

When the *Go Back to get a new purchase* button is clicked, a new confirmation menu (*AreYouSure2*) shows up with the following methods:

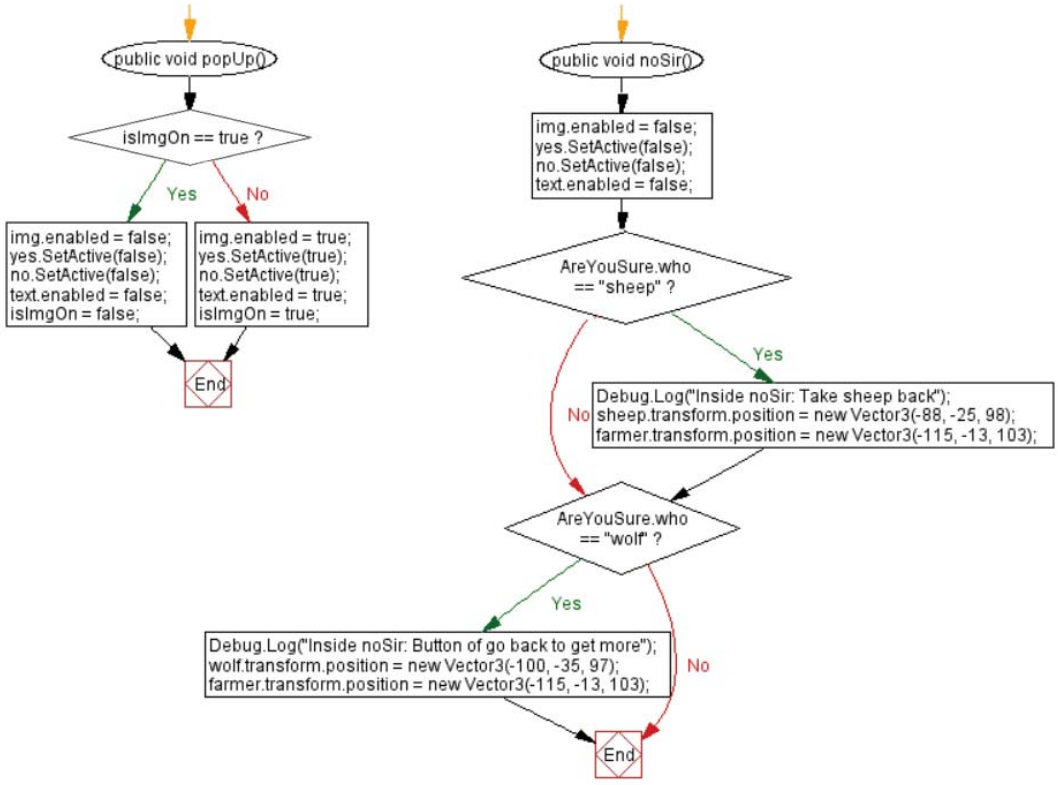


Figure 12. UML Flow charts of *popup()* and *noSir()* methods of *AreYouSure2* class

The interaction with it is like the first confirmation menu (*AreYouSure*) with the only difference being that if the player decided to go get another purchase, this time the sheep will stay in the other shore and we'd enter the official **second stage** of the puzzle. The way we refer to this stage is the same way we call the method that controls the interactions in this stage: *sheepThereWolfOrCabbage()*. Although this method is located in the *AreYouSure* class it's only accessed when we enter the stage 2 thanks to the verification of the variable *who2* when the player clicks YES in the second confirmation menu (*AreYouSure2*).

Nevertheless, the events that are triggered when the YES button is clicked and the system verifies the player meant for the farmer to go back to get another purchases (by clicking the button Go Back that would set the variable *who* to *wolf*) are: taking the farmer and the boat to their initial position, disabling the *Sheep* button as it's no longer in the origin shore, modifying the positions of the last two buttons (*Wolf* and *Cabbage*) and then setting the stage 2 by making *who2* = true (figure 13).

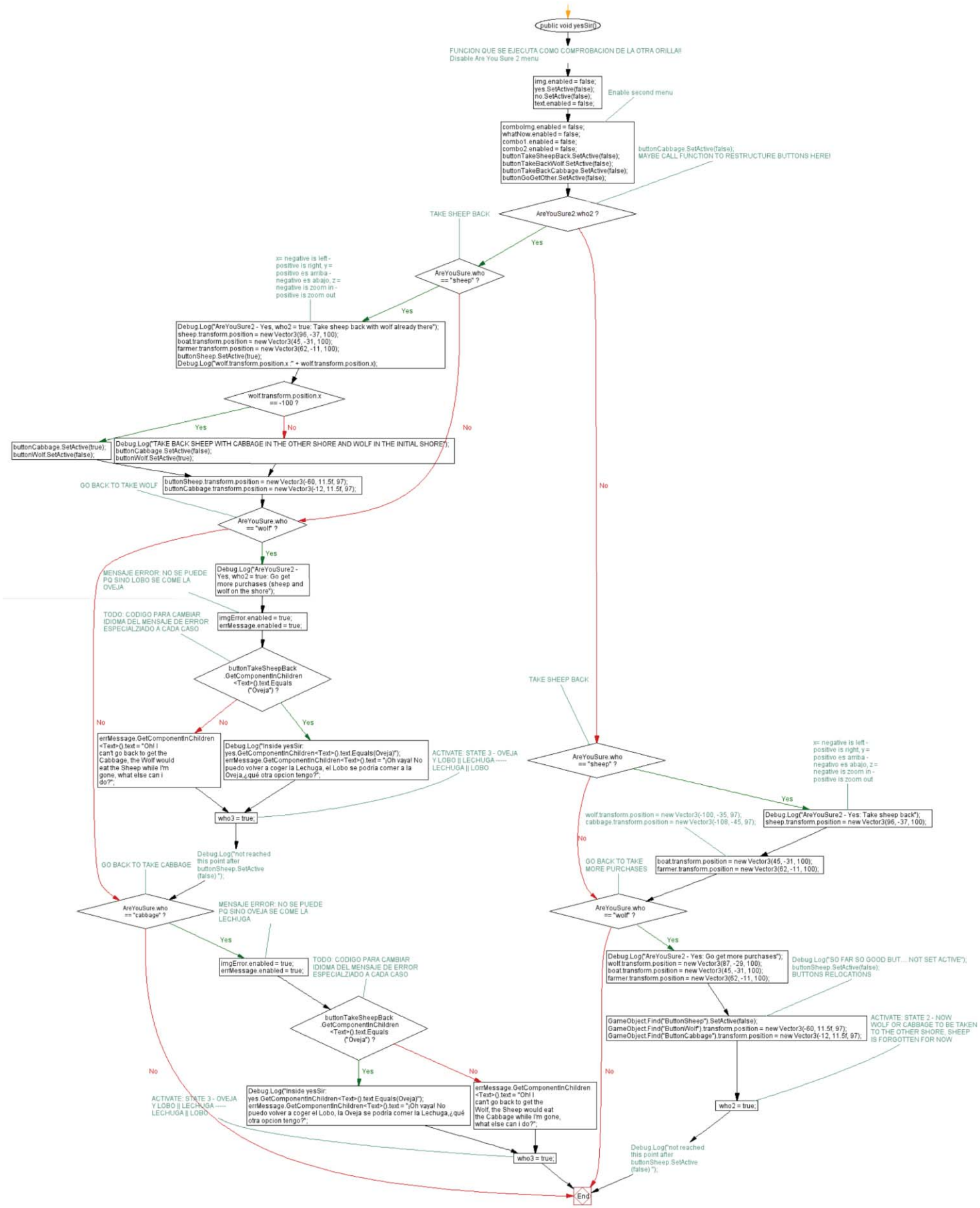


Figure 13. UML Flow chart of the *yesSir()* function of the *AreYouSure2()* class.

From now on it might seem tricky and messy to follow the sequence of execution of methods and verification of conditions, but we'd try to more or less guide the execution through the first occurrences of the methods and explain the repetition cycle with the parts that will be modified.

So now, we are back to kind of square one with the only difference that one of the purchases is in the other shore. The player must choose now if he or she will take the wolf or the cabbage and in either case the behavior will be the same: the purchase will be loaded in the boat along with the farmer, the first confirmation menu (*AreYouSure*) will *popup* and if we confirm our selection it will take the purchase to the other side.

This time the execution will branch in the *AreYouSure*'s *yesSir()* code to the if statement where the condition of *who2* equals true and that leads to the execution of the *sheepThereWolfOrCabbage()* function (Figure 14). Here the method will discern if it was the wolf or the cabbage the selected one and whichever it was it'll take it to the other side and enable the next combo menu where we'll have once again to choose between several options*: Take the Sheep back, Take the Wolf/Cabbage Back* or *Go back to get another purchase* (Figure 15)

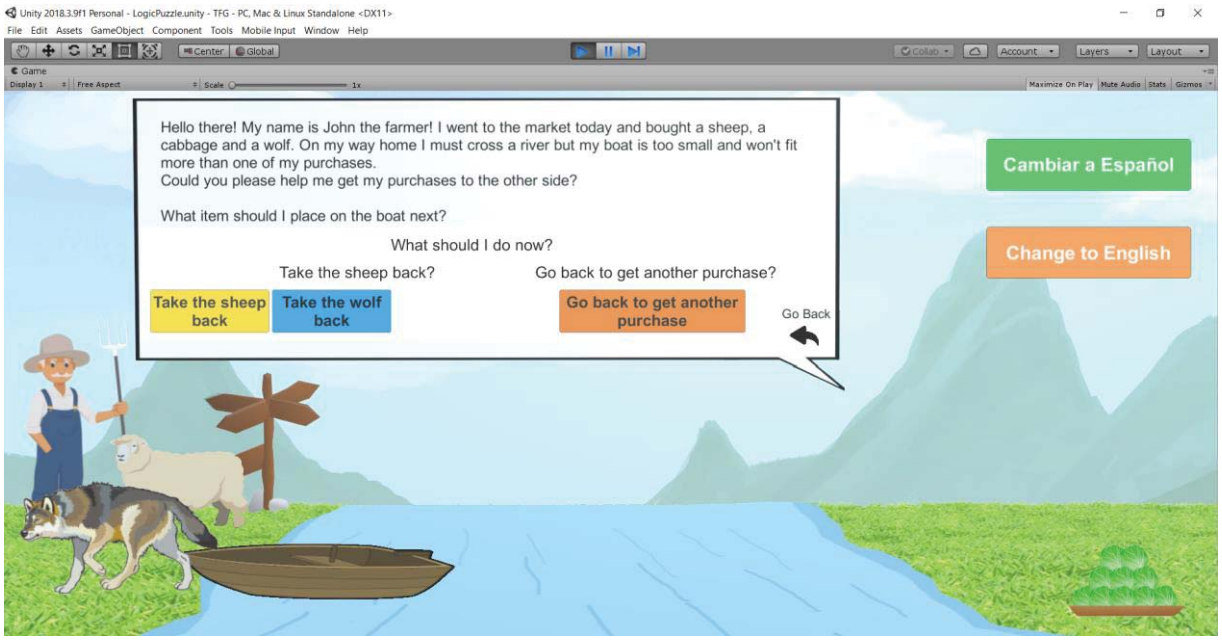


Figure 15. Screenshot of the stage 2 menu option screen

When this situation happens, we decided to make this the **stage 3** and final stage. The reason we consider it the final stage is because from this stage on the logic behind the system's interaction and decision-making procedure will repeat scenarios with the ones we have already seen.

Of course, there'd be differences as the combination of purchases in each shore will vary but as for the actual functionality of the system it will behave the same. The system will always check of course to make sure the farmer doesn't leave potential danger scenarios in any shore like, for instance, in figure 15 we see that one of our three options is return to get the cabbage. This decision will result fatal as if we

indicate the farmer to go back just by himself the Wolf would have a chance of eating the Sheep. An error message will pop up to warn the player of his or her decision.

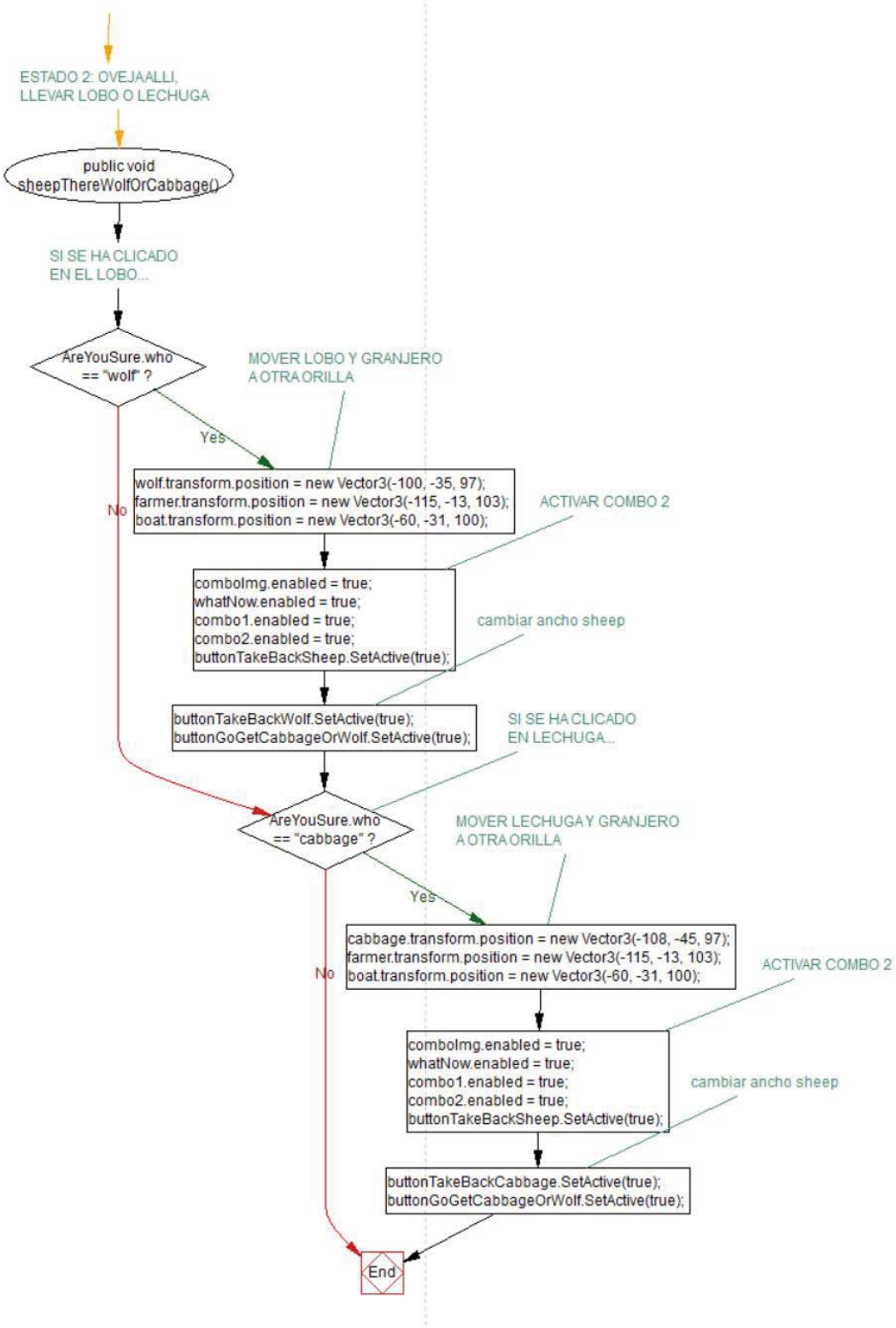


Figure 14. UML Flow chart of the *sheepThereWolfOrCabbage()* function of the *AreYouSure()* class

From this point on the code very much grows to be a bigger state-case scenario where the system analyses which products will be taken where and always making sure the conditions, or in this case, restrictions are always followed.

This project leads us to believe that from the rest of the possible representations of data or system behavior out there, like the ones provided by UML, there's one that while complimentary to the flow chart's design represents the interactions with a more concise approach, and that is the UML activity diagrams (figure 15)

Part of the reason we decided to use these diagrams was because we felt we needed a more detailed view of the dynamic of the system. Not as much how the behavior would branch depending on our decisions but rather understand what options there were. This was important to design a correct and successful user experience and fulfill our goal of making it an alternative to what it's already out there.

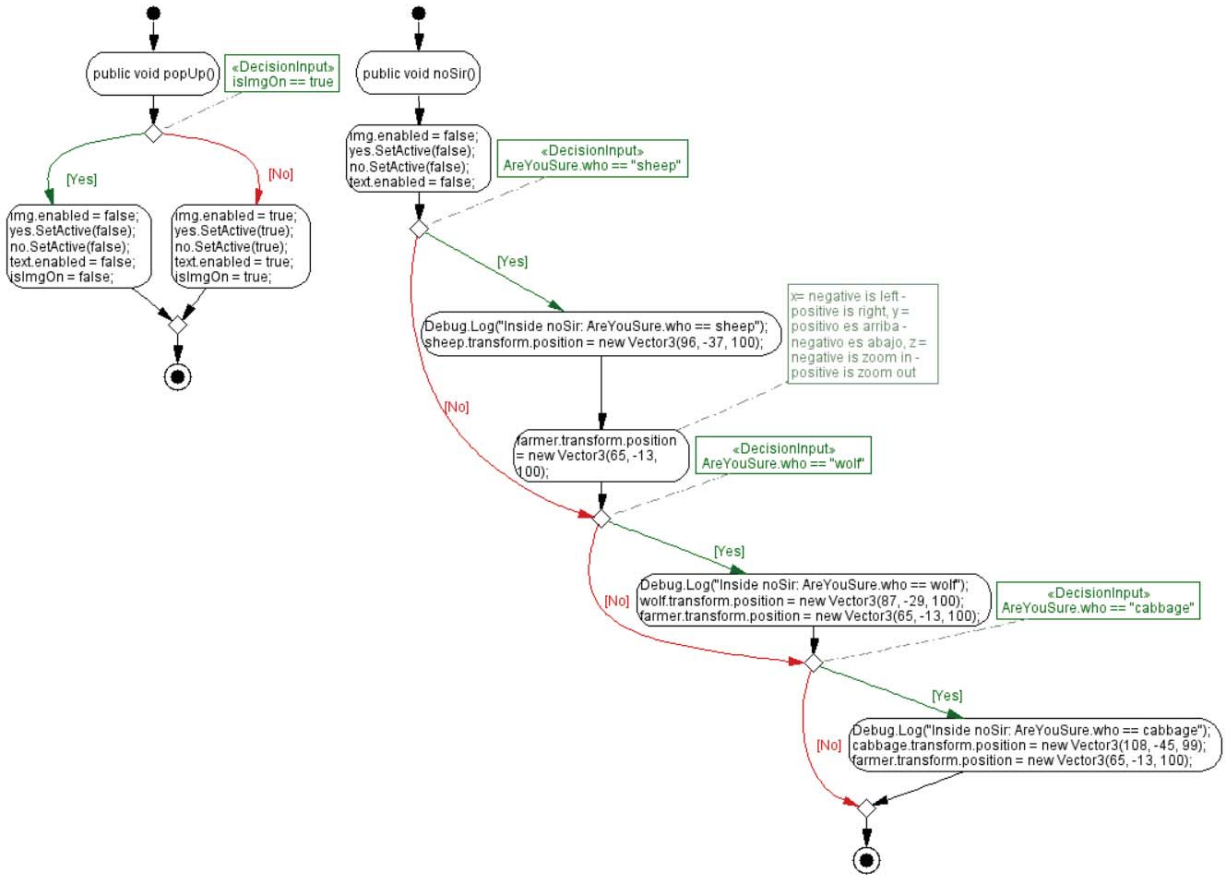
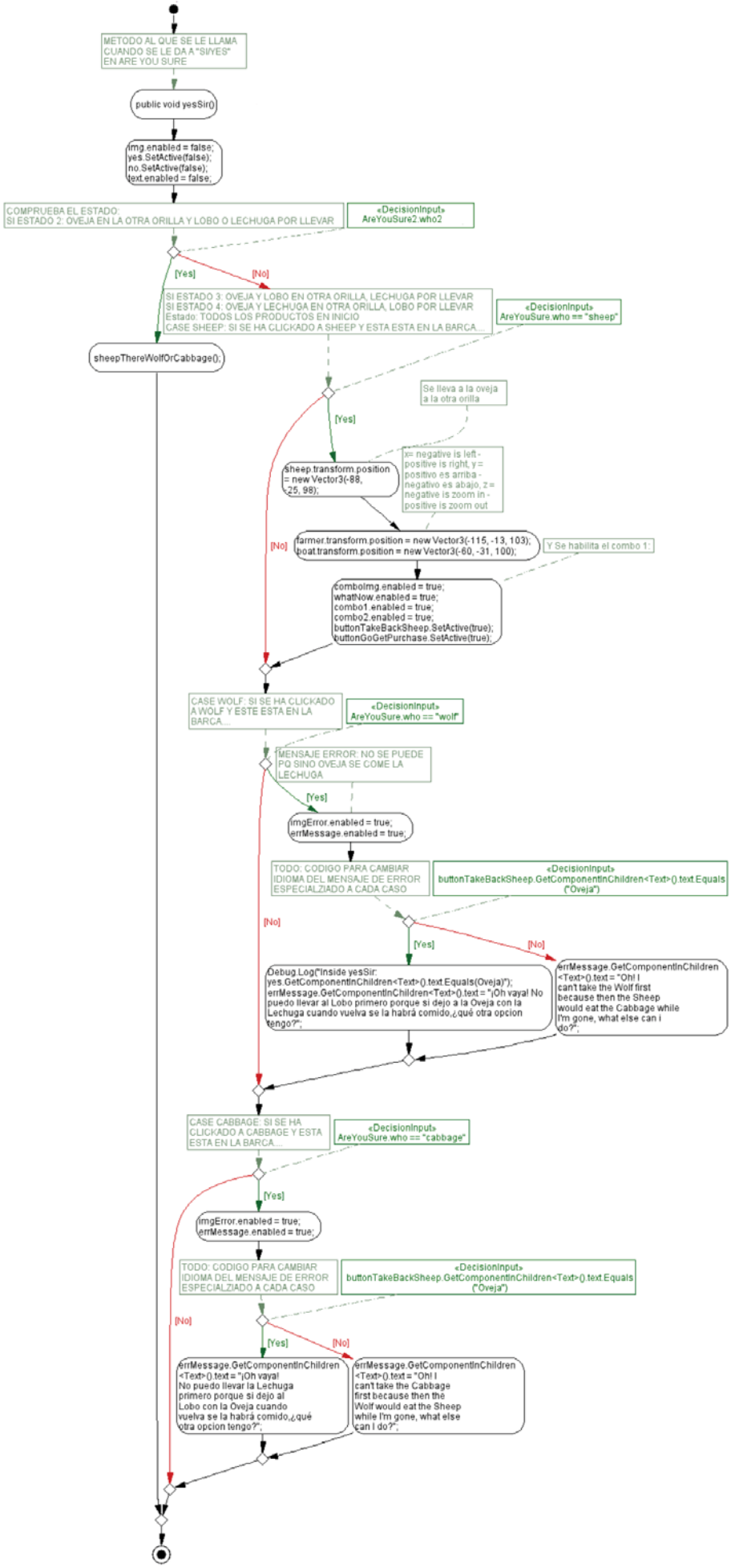


Figure 15.1. Activity Diagrams of the methods *popup()* and *noSir()* of *AreYouSure* class

As we can see in the figure 15.1, we have a similar structure to the flow chart diagram but with the main difference being the depiction of all the available options of the user or in this case player.

The other reason we felt these diagrams will be useful is in order to clarify our chain of thought, to realize the weaker points of the interaction with the system and try to make it better. The option of making it as scalable as possible was always in our minds; to allow the future us or even a newcomer to look at what we had done and be able to understand the different activities that take place on our system.


METODO AL QUE SE LE LLAMA CUANDO SE LE DA A "SIIYES" EN ARE YOU SURE
public void yesSir()
img.enabled = false;
yes.SetActive(false);
no.SetActive(false);
text.enabled = false;
COMPRUEBA EL ESTADO:
SI ESTADO 2: OVEJA EN LA OTRA ORILLA Y LOBO O LECHUGA POR LLEVAR
«DecisionInput»
AreYouSure2.who2
[Yes]
[No]
SI ESTADO 3: OVEJA Y LOBO EN OTRA ORILLA, LECHUGA POR LLEVAR
SI ESTADO 4: OVEJA Y LECHUGA EN OTRA ORILLA, LOBO POR LLEVAR
Estado: TODOS LOS PRODUCTOS EN INICIO
CASE SHEEP: SI SE HA CLICKADO A SHEEP Y ESTA ESTA EN LA BARCA....
«DecisionInput»
AreYouSure.who == "sheep"
sheepThereWolfOrCabbage();
Se lleva a la oveja a la otra orilla
[Yes]
sheep.transform.position = new Vector3(-88, -25, 98);
x= negative is left - positive is right, y = positivo es arriba - negativo es abajo, z = negative is zoom in - positive is zoom out
[No]
farmer.transform.position = new Vector3(-115, -13, 103);
boat.transform.position = new Vector3(-60, -31, 100);
Y Se habilita el combo 1:
comboImg.enabled = true;
whatNow.enabled = true;
combo1.enabled = true;
combo2.enabled = true;
buttonTakeBackSheep.SetActive(true);
buttonGoGetPurchase.SetActive(true);
CASE WOLF: SI SE HA CLICKADO A WOLF Y ESTE ESTA EN LA BARCA....
«DecisionInput»
AreYouSure.who == "wolf"
MENSAJE ERROR: NO SE PUEDE PQ SINO OVEJA SE COME LA LECHUGA
[Yes]
imgError.enabled = true;
errMessage.enabled = true;
TODO: CODIGO PARA CAMBIAR IDIOMA DEL MENSAJE DE ERROR ESPECIALIZIADO A CADA CASO
«DecisionInput»
buttonTakeBackSheep.GetComponentInChildren<Text>().text.Equals ("Oveja")
[No]
[Yes]
[No]
Debug.Log("inside yesSir: yes.GetComponentInChildren<Text>().text.Equals(Oveja)"); errMessage.GetComponentInChildren<Text>().text = "¡Oh vaya! No puedo llevar al Lobo primero porque si dejo a la Oveja con la Lechuga cuando vuelva se la habrá comido,¿qué otra opcion tengo?";
errMessage.GetComponentInChildren <Text>().text = "Oh! I can't take the Wolf first because then the Sheep would eat the Cabbage while I'm gone, what else can i do?";
CASE CABBAGE: SI SE HA CLICKADO A CABBAGE Y ESTA ESTA EN LA BARCA....
«DecisionInput»
AreYouSure.who == "cabbage"
[Yes]
imgError.enabled = true;
errMessage.enabled = true;
TODO: CODIGO PARA CAMBIAR IDIOMA DEL MENSAJE DE ERROR ESPECIALIZIADO A CADA CASO
«DecisionInput»
buttonTakeBackSheep.GetComponentInChildren<Text>().text.Equals ("Oveja")
[No]
[Yes]
[No]
errMessage.GetComponentInChildren <Text>().text = "¡Oh vaya! No puedo llevar la Lechuga primero porque si dejo al Lobo con la Oveja cuando vuelva se la habrá comido,¿qué otra opcion tengo?";
errMessage.GetComponentInChildren <Text>().text = "Oh! I can't take the Cabbage first because then the Wolf would eat the Sheep while I'm gone, what else can I do?";

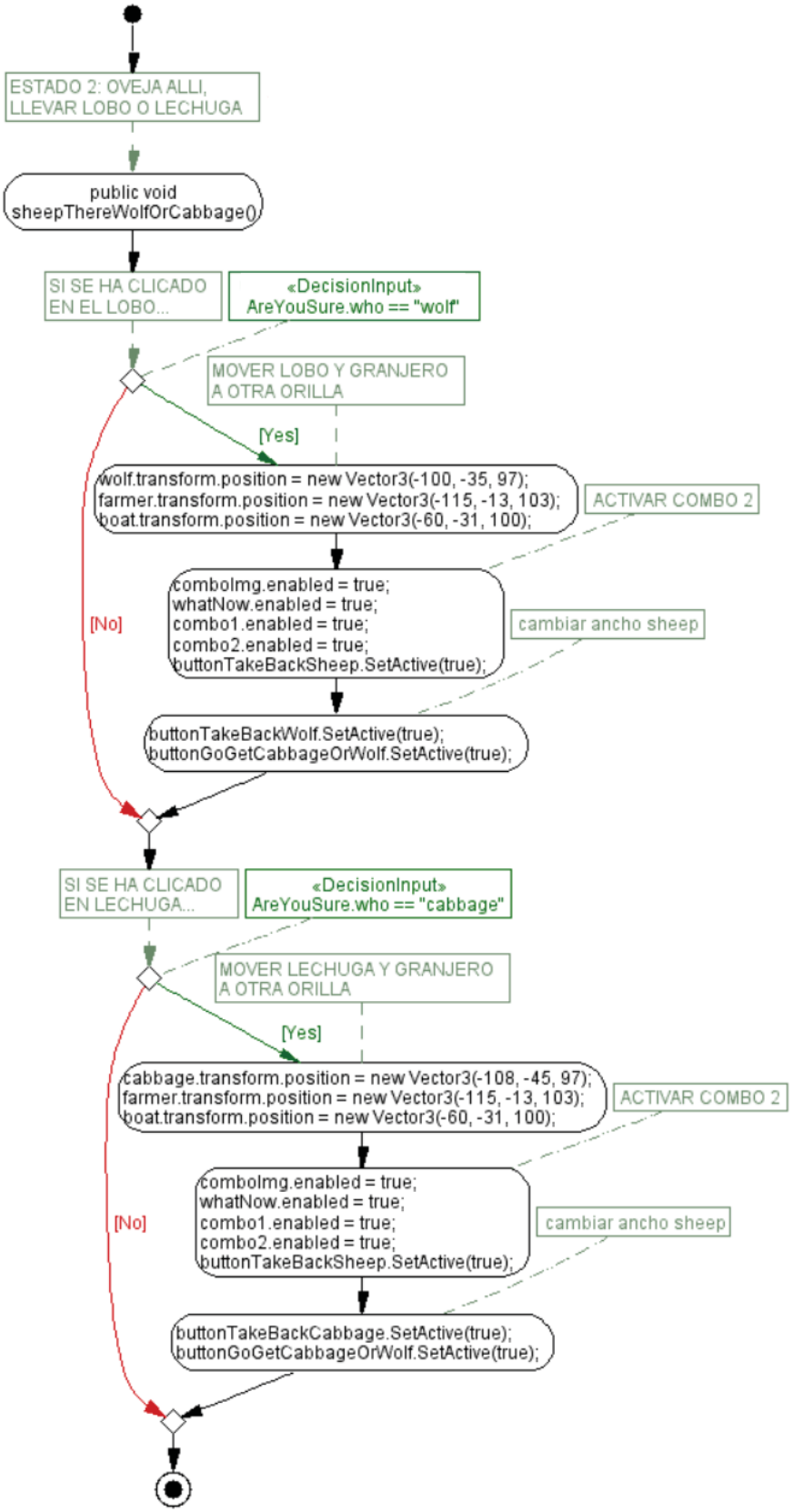
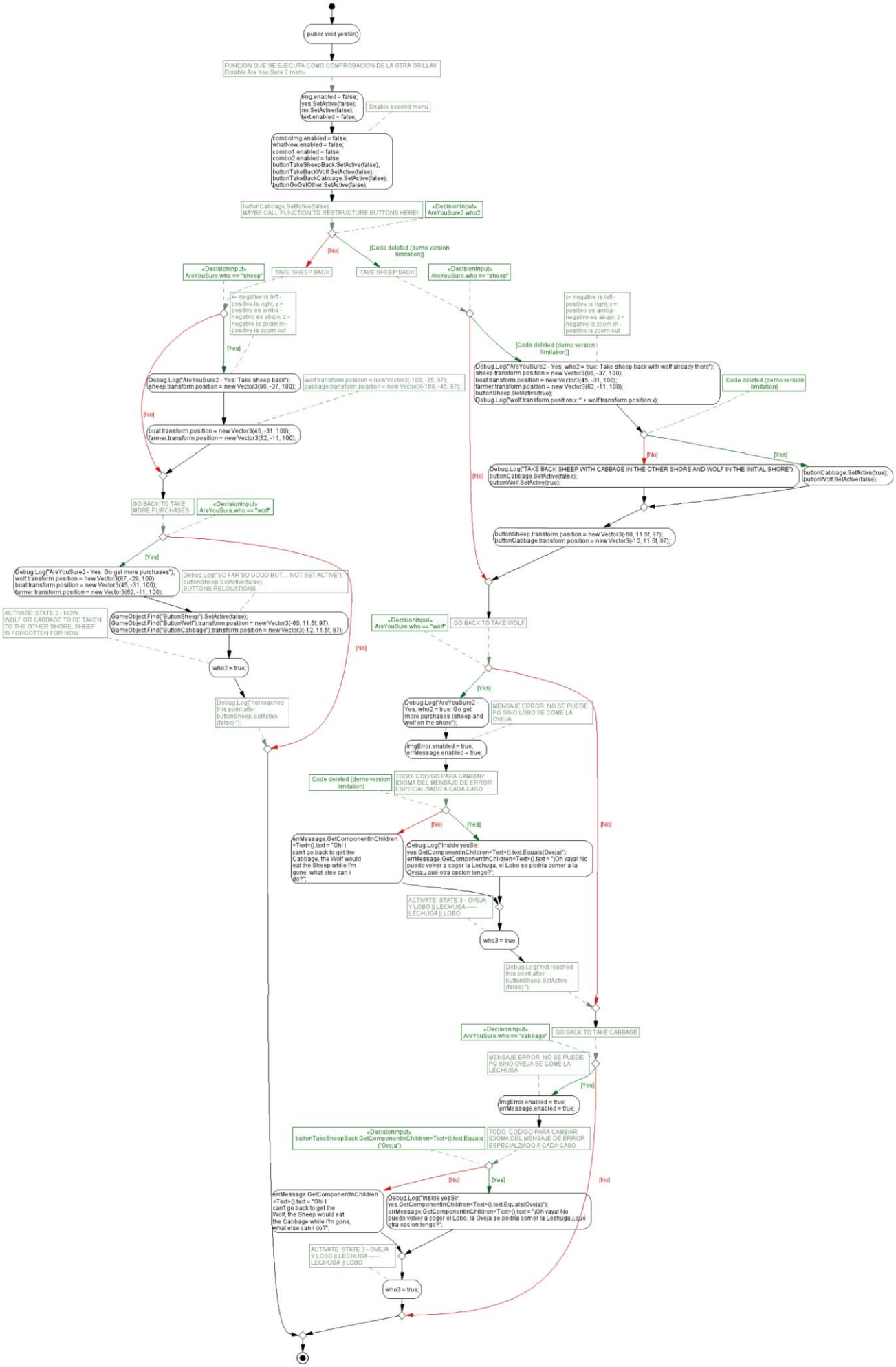


Figure 15.2. Activity diagrams of the methods of *yesSir()* and *sheepThereWolfOrCabbage()* methods of *AreYouSure* class


public void yesSir()
FUNCION QUE SE EJECUTA COMO COMPROBACION DE LA OTRA ORILLA!! Disable Are You Sure 2 menu
img.enabled = false; yes.SetActive(false); no.SetActive(false); text.enabled = false;
Enable second menu
comboImg.enabled = false; whatNow.enabled = false; combo1.enabled = false; combo2.enabled = false; buttonTakeSheepBack.SetActive(false); buttonTakeBackWolf.SetActive(false); buttonTakeBackCabbage.SetActive(false); buttonGoGetOther.SetActive(false);
buttonCabbage.SetActive(false); MAYBE CALL FUNCTION TO RESTRUCTURE BUTTONS HERE!
«DecisionInput» AreYouSure2.who2
[No]
[Code deleted (demo version limitation)]
«DecisionInput» AreYouSure.who == "sheep"
TAKE SHEEP BACK
TAKE SHEEP BACK
«DecisionInput» AreYouSure.who == "sheep"
x= negative is left - positive is right, y = positivo es arriba - negativo es abajo, z = negative is zoom in - positive is zoom out
[Yes]
Debug.Log("AreYouSure2 - Yes: Take sheep back"); sheep.transform.position = new Vector3(96, -37, 100);
wolf.transform.position = new Vector3(-100, -35, 97); cabbage.transform.position = new Vector3(-108, -45, 97);
[No]
boat.transform.position = new Vector3(45, -31, 100); farmer.transform.position = new Vector3(62, -11, 100);
GO BACK TO TAKE MORE PURCHASES
«DecisionInput» AreYouSure.who == "wolf"
[Yes]
Debug.Log("AreYouSure2 - Yes: Go get more purchases"); wolf.transform.position = new Vector3(87, -29, 100); boat.transform.position = new Vector3(45, -31, 100); farmer.transform.position = new Vector3(62, -11, 100);
Debug.Log("SO FAR SO GOOD BUT.... NOT SET ACTIVE"); buttonSheep.SetActive(false); BUTTONS RELOCATIONS
ACTIVATE: STATE 2 - NOW WOLF OR CABBAGE TO BE TAKEN TO THE OTHER SHORE. SHEEP IS FORGOTTEN FOR NOW
GameObject.Find("ButtonSheep").SetActive(false); GameObject.Find("ButtonWolf").transform.position = new Vector3(-60, 11.5f, 97); GameObject.Find("ButtonCabbage").transform.position = new Vector3(-12, 11.5f, 97);
[No]
who2 = true;
Debug.Log("not reached this point after buttonSheep.SetActive (false) ");
Debug.Log("AreYouSure2 - Yes, who2 = true: Take sheep back with wolf already there"); sheep.transform.position = new Vector3(96, -37, 100); boat.transform.position = new Vector3(45, -31, 100); farmer.transform.position = new Vector3(62, -11, 100); buttonSheep.SetActive(true); Debug.Log("wolf.transform.position.x " + wolf.transform.position.x);
Code deleted (demo version limitation)
[No]
[Yes]
Debug.Log("TAKE BACK SHEEP WITH CABBAGE IN THE OTHER SHORE AND WOLF IN THE INITIAL SHORE"); buttonCabbage.SetActive(false); buttonWolf.SetActive(true);
buttonCabbage.SetActive(true); buttonWolf.SetActive(false);
buttonSheep.transform.position = new Vector3(-60, 11.5f, 97); buttonCabbage.transform.position = new Vector3(-12, 11.5f, 97);
«DecisionInput» AreYouSure.who == "wolf"
GO BACK TO TAKE WOLF
[Yes]
Debug.Log("AreYouSure2 - Yes, who2 = true: Go get more purchases (sheep and wolf on the shore");
MENSAJE ERROR: NO SE PUEDE PQ SINO LOBO SE COME LA OVEJA
imgError.enabled = true; errMessage.enabled = true;
Code deleted (demo version limitation)
TODO: CODIGO PARA CAMBIAR IDIOMA DEL MENSAJE DE ERROR ESPECIALIZADO A CADA CASO
[No]
[Yes]
errMessage.GetComponentInChildren <Text>().text = "Oh! I can't go back to get the Cabbage, the Wolf would eat the Sheep while I'm gone, what else can i do?";
Debug.Log("Inside yesSir: yes.GetComponentInChildren<Text>().text.Equals(Oveja)"); errMessage.GetComponentInChildren<Text>().text = "¡Oh vaya! No puedo volver a coger la Lechuga, el Lobo se podria comer a la Oveja,¿qué otra opcion tengo?";
ACTIVATE: STATE 3 - OVEJA Y LOBO || LECHUGA ------ LECHUGA || LOBO
who3 = true;
Debug.Log("not reached this point after buttonSheep.SetActive (false) ");
[No]
«DecisionInput» AreYouSure.who == "cabbage"
GO BACK TO TAKE CABBAGE
MENSAJE ERROR: NO SE PUEDE PQ SINO OVEJA SE COME LA LECHUGA
[Yes]
imgError.enabled = true; errMessage.enabled = true;
«DecisionInput» buttonTakeSheepBack.GetComponentInChildren<Text>().text.Equals ("Oveja")
TODO: CODIGO PARA CAMBIAR IDIOMA DEL MENSAJE DE ERROR ESPECIALIZADO A CADA CASO
[No]
[Yes]
errMessage.GetComponentInChildren <Text>().text = "Oh! I can't go back to get the Wolf, the Sheep would eat the Cabbage while I'm gone, what else can i do?";
Debug.Log("Inside yesSir: yes.GetComponentInChildren<Text>().text.Equals(Oveja)"); errMessage.GetComponentInChildren<Text>().text = "¡Oh vaya! No puedo volver a coger el Lobo, la Oveja se podria comer la Lechuga,¿qué otra opcion tengo?";
ACTIVATE: STATE 3 - OVEJA Y LOBO || LECHUGA ------ LECHUGA || LOBO
who3 = true;

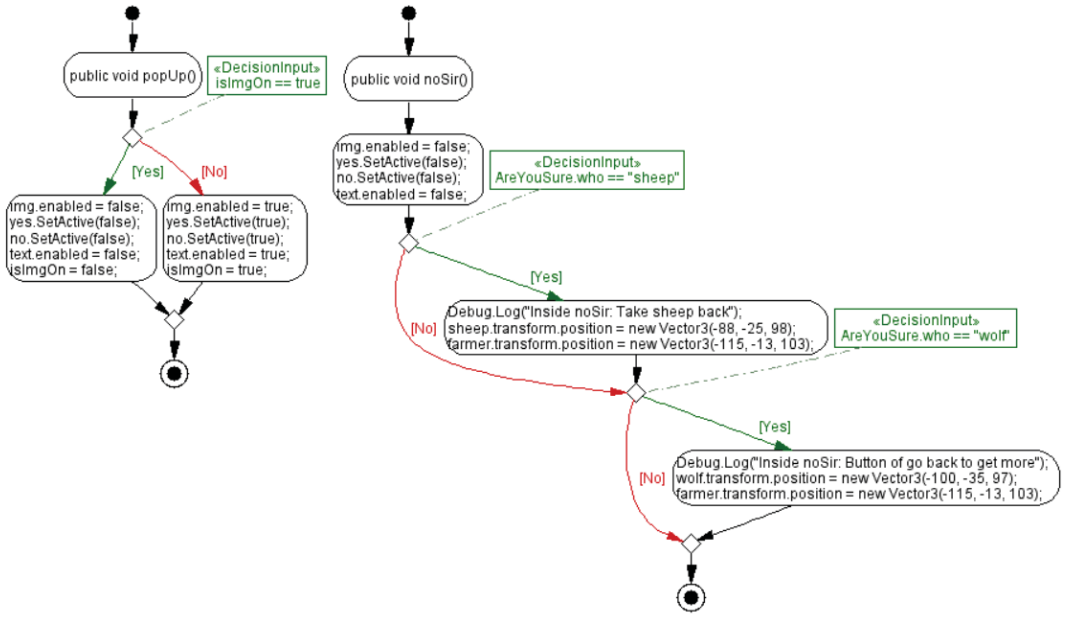


Figure 15.3 Activity diagrams of the methods of *yesSir(), pop()* and *noSir()* methods of *AreYouSure2* class

As a final exposition of the ideas behind the solution we developed for our particular issue: to propose an alternative to the actual education system that we believe must be updated and improved in order to teach more efficiently to the upcoming generations; we want to stress the system's heavy reliance on product position-checking in order to validate or not the decision that the player will make.

Moreover, as we kept learning more and more about the workings of Unity and the C# programming language, we began to realize most of the functions, variables and attributes played with the virtual space and time. We learned that when programming graphics and implementing scene renderings there was a lot of math behind it. It made use of 3D space vectors, mathematical curves and matrixes to represent objects, or better said, pixels in the screen. So, in a sense, we were aware that in order to generate movements we had to play with the virtual space we were given, especially for the 3D scene.

Nevertheless, movement had to be synchronized with the user interface, not only through input commands like the arrow keys and the mouse clicks, but through the communication and event triggers of each of those actions. Unity had specific ways to trigger events with the C# language and at the time of addressing those interactions we found a lot of issues of synchronization as we wanted to order the movements in a way it made sense and which followed the activity diagrams we had generated (figure 15). But in order to do that we had to understand how the method calling order worked and that was hard to test.

Due to another important aspect of games itself, that is, the visualization and artistic design, we took this opportunity to develop more a second interest we had apart from our programming and engineering skills and that is the 3D modeling and Digital Art. We designed some conceptual art pieces of the main character and it was even helpful in planning the representation of the terrain (figures 5 and 6); and would have very much used them in the game should we have counted with more time to

dedicate to the visual aspect. Things like animations, special effects and overall making the experience more engaging. Nevertheless, the project was ambitious enough as we planned on giving it a scalable potential for future improvements upon it. The designs were also chosen to appear more amicable and appeal to younger audiences as well as more adult ones.

Figure 5: Character designs for the main player

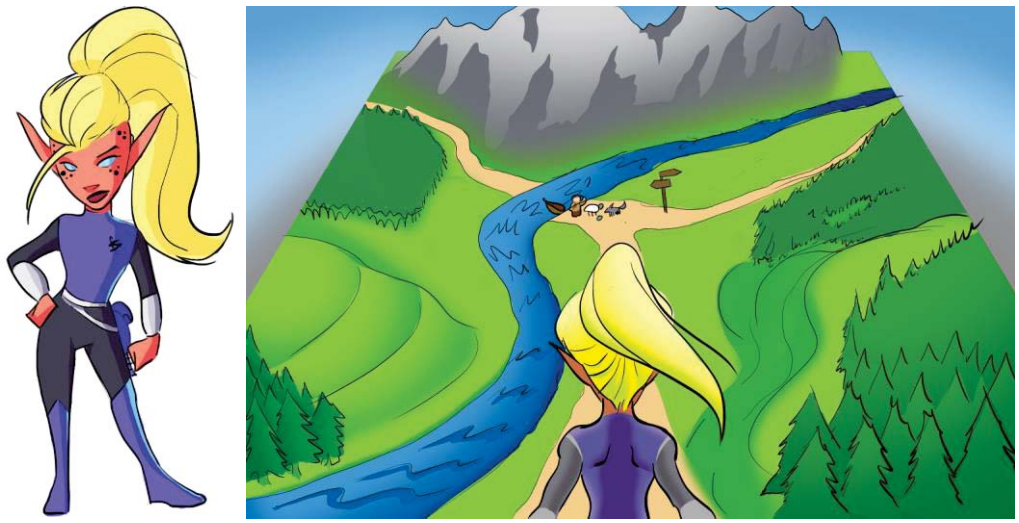


Figure 6: Final Character and Environment design

The main intention behind dedicating time to the design of this concepts and environment was to provide the system with a more appealing and attractive façade. And this was one of the main goals of the system, providing a mathematical or educational topic in a more visually attractive form so that the students would feel some sort of deeper connection to it and maybe could learn better by stimulating their visual memory.

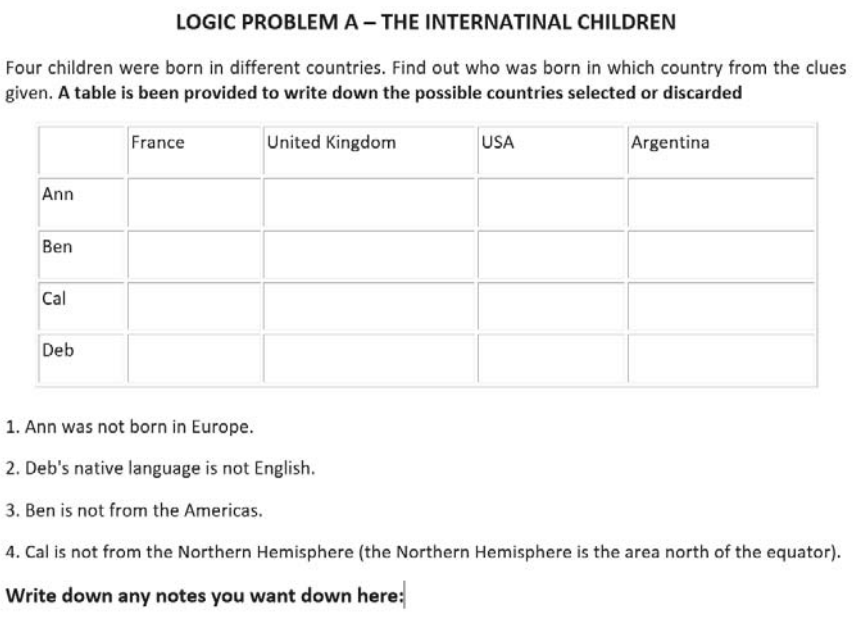
Of course, the main priority above all was to make a fully functioning system rather than a genuine and fancy one. At a certain point we realized we were solving smaller problems that had little to no hurry of being solved. But we managed to stop and focus on what was really important: to implement a system that would support our thesis that a more visual and less explicitly informative problem formulation encourages the thinking and reasoning of the student more than simply allowing him or her to work with all the data given and just solving through an mechanical or procedural process.

EVALUATION

The main purpose of this project is to provide to the world a new and improved way of teaching using the tools that we have learned in our degree. We considered all the skills we've learned of Artificial Intelligence, Programming, Software Engineering, Human-Computer Interaction, etc.; and we've tried our best to apply the ones that best suited our project in a more implicit or explicit way.

In order to better test the efficiency and effectiveness of what our system is trying to accomplish, that is, provide a better, more efficient and more attractive solution to the inefficiency of the teaching methods of the actual education system which, as we have seen and proven in the introduction of this thesis, is an issue that exists today; we have planned out some user tests that we thought were the ideal ones among all the available ones to fully proof if this system reaches the main goal we set it to. These tests will be carried out in order as follows:

1. **A/B usability test –** The goal of this testing is to ask the participants to compare both logic problem representation format, that is:
    a. **A paper formulation format** most commonly used nowadays with all necessary information (restrictions, conditions, instructions, etc.) provided to the user (figure 1) of an approximate equivalent problem to the logic problem presented in our system, which is above all the main educational aspect of it.



LOGIC PROBLEM A – THE INTERNATINAL CHILDREN

Four children were born in different countries. Find out who was born in which country from the clues given. **A table is been provided to write down the possible countries selected or discarded**

	France	United Kingdom	USA	Argentina
Ann				
Ben				
Cal				
Deb				

1. Ann was not born in Europe.
2. Deb's native language is not English.
3. Ben is not from the Americas.
4. Cal is not from the Northern Hemisphere (the Northern Hemisphere is the area north of the equator).

Write down any notes you want down here:

Figure 1. Paper-based formulation format

b. The format that my system is trying to test, that is, provide a visual representation of the problem with almost no clues nor all the information available thus encouraging the user to observe and learn from their mistakes in a trial-error way. This, in order to test if the format of figure 2, the paper format version (A), would be more or less effective or better than the visual one (B). The river crossing paper formulation is provided with two possible solution process: the step by step and the status-checked version (in this case there's only one unit of each so the items in the boat can be represented with letters or in binary, 1 for when the item is in the shore and 0 for when it isn't)

LOGIC PROBLEM B – THE RIVER CROSSING PROBLEM

A farmer went to the market today and bought a sheep, a pile of cabbage and a wolf. On his way home he must cross a river but hi boat is too small and won't fit more than one of his purchases.

If the farmer leaves on any of the shores the wolf and the sheep alone the wolf would eat the sheep and the same would happen with the pile of cabbage and the sheep.

How should the farmer take all his purchases to the other shore of the river?

To solve this exercise, you can use the following method:

1. Take the Sheep over
2. Return
3. Take the Wolf/Cabbage over
4. Return with the Sheep
5. Take the Cabbage/Wolf over
6. Return
7. Take Sheep over

Thus, there are seven crossings, four forward and three back

Or with this other:

Origin Shore					Destination Shore			
Farmer	Sheep	Wolf	Cabbage		Farmer	Sheep	Wolf	Cabbage
F	S	W	C					
		W	C	R	F	S		
F		W	C	I		S		
			C	V	F	S	W	
F	S		C	E			W	
	S			R	F		W	C
F	S						W	C
					F	S	W	C

Figure 2. A format, paper format, of the river crossing problem

We decided to use an equivalent instead of testing the user with the exact equivalent of the problem presented in our system (river crossing problem) in paper format because with this test we are also trying to test the efficiency and effectiveness of each of them. Therefore, we've selected a similar resolution-process logic problem that we have called *The International Children* and which shares the same state and condition-checking-driven chain of thought (as, through the clues given in figure 1, the state of each kid is to be modified and checked each time new information is processed and

conclusions are implied) just like state and condition-checking process that goes behind the shores in every trip the farmer does.

2. **A Survey –** we will carry out a survey with questions oriented to knowing what the users felt was a better way of presenting the problem. We would then provide them with a paper version format of our system's logic puzzle, we'd explain the differences and purpose behind both tests, and we will ask them, with the recently acquired knowledge after testing both formats, if they would still choose the one the chose before fully understanding the purpose of the A/B test.

So, all in all, through the A/B testing we will objectively test which format is more efficient measuring several things while the participants take the tests. We will use the methods provided in Justin Mifsud's article *Usability Metrics – A Guide To Quantify The Usability Of Any System* [9]. The following formula will be used to calculate the effectiveness or completion rate of both formats:

$$Effectiveness = \frac{Number\ of\ tasks\ completed\ successfully}{Total\ number\ of\ tasks\ undertaken}\ x\ 100\%$$

The task at hand is only one and that is the logic problems, so the completion rate is calculated by assigning a binary value of '1' if the test participant manages to complete a task and '0' if he/she doesn't.

We will also measure the number of mistakes made by the user, including unintended actions, slips or omissions, while attempting a task; and a time-based efficiency using Justin Mifsud's formula:

$$Time\ Based\ Efficiency = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N}\frac{nij}{tij}}{NR}$$

Where:
N = The total number of tasks (goals)
R = The number of users
nij = The result of task i by user j; if the user successfully completes the task, then Nij = 1, if not, then Nij = 0
tij = The time spent by user j to complete task i. If the task is not successfully completed, then time is measured till the moment the user quits the task

We would have carried out a different user testing, like a software performance testing, but we felt that the main focus of our project was to understand if, regardless or not of the implementation of the code, if the way the problem was presented, the way the design of the system succeeded in what we were trying to achieve with it. Moreover, we had also in mind carrying out a hallway usability testing but we realized these tests were done to prove the correct design and intuitiveness of the system, something we really wanted to get feedback from but that, when looking at the bigger picture that is our main goal of the project, we realize it wasn't what we were trying to test. It would, of course, matter if the user feels comfortable with the system, if it feels that it is appealing and even enjoyable to use; but with this thesis we wanted to get the feedback from them regarding if whether or not it was more effective to them to use our system or the other outdated one.

According to designer Jeff Bryant's article *Who, What, and Why – A Guide to User Testing Methods*[10], the optimum number of participants in usability testing is settled by the book *A Mathematical Model of the Finding of Usability Problems* which shows that with only five usability testing participants 85% of the issues within the tested UI will be uncovered, thus deeming the number ideal enough to measure our system's effectiveness. Nevertheless, according to figure 2, for best results, this should be performed with a total of 15 participants.

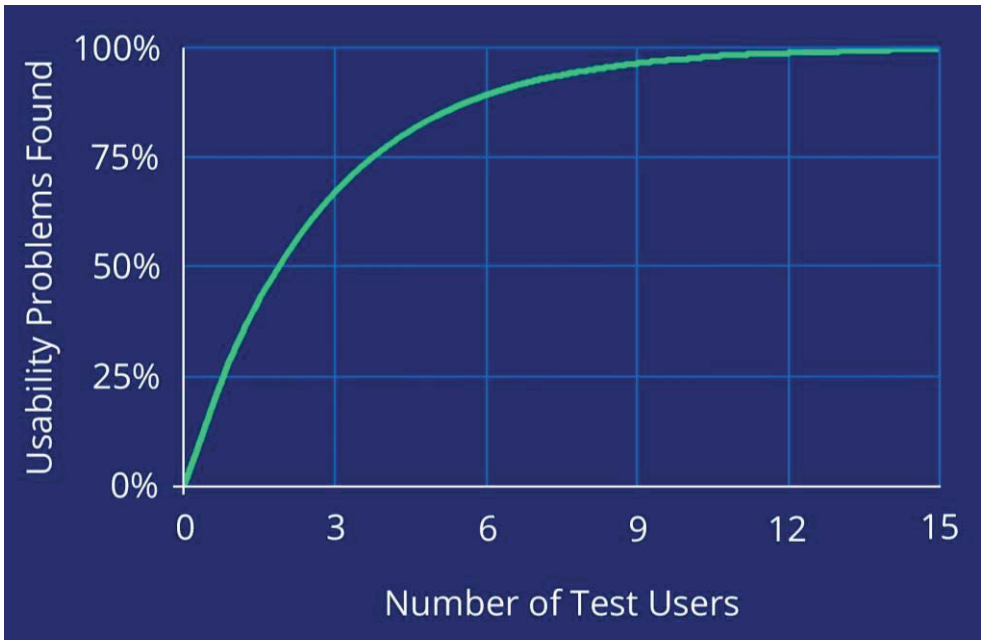


Figure 2. Result of a usability study according to the book *A Mathematical Model of the Finding of Usability Problems*

So, we proceeded to test a minimum of 5 participants to get a high percentage of effectiveness and time-based efficiency but ended up testing 7 in the age range of 13 to 27, the thought of possible targeted age range. It's not an ideal 15 but this number pretends to uncover as many issues as possible with the system at hand but our goal with this A/B usability test is to demonstrate that the problem formulation format our system implements is more effective at teaching than the paper formulation one.

Also it is important to clarify that even though we would be using an almost equivalent formulation of the river crossing puzzle to test which format is more effective, we would make the calculations on the basis of the survey we carried out of whether or not they would have chosen and if they would been able to complete the river crossing problem with the A format formulation. Therefore, after we carried out the tests, we got the following results:

Given the formula of Effectiveness seen before:

$$Effectiveness = \frac{Number\ of\ tasks\ completed\ successfully}{Total\ number\ of\ tasks\ undertaken}\ x\ 100\%$$

Number of tasks completed successfully = Number of participants that completed the problem (A or B)

Total number of tasks undertaken = Number of tests (of problem A or B) done in total

The effectiveness of each format has resulted in:

$$Effectiveness\ of\ format\ A = \frac{4\ participants\ completed\ the\ test}{7\ participants\ took\ the\ test}\ x\ 100\%$$

$$\boldsymbol{Effectiveness\ of\ format\ A = 57,1428571\%}$$

$$Effectiveness\ of\ format\ A = \frac{6\ participants\ completed\ the\ test}{7\ participants\ took\ the\ test}\ x\ 100\%$$

$$\boldsymbol{Effectiveness\ of\ format\ B = 85,7142857\%}$$

According to a study carried out by Jeff Sauro [11], the average Task Completion rate is 78%, therefore, format B (our system's) turned out to have a more than average success task completion rate and proved to be 61% more effective (figure 3)

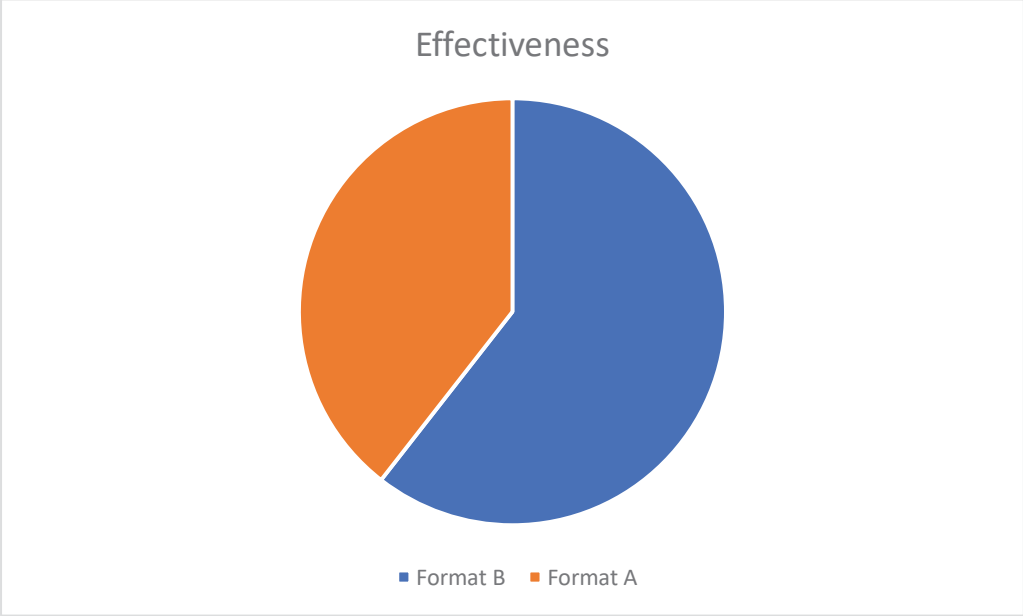


Figure 3. Effectiveness pie chart as result of the A/B usability testing carried out

The following measure we will take is the Time-Based Efficiency, given:

$$Time\ Based\ Efficiency = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N}\frac{nij}{tij}}{NR}$$

Where:
N = The total number of tasks (goals) == 1
R = The number of users == **7**
nij = The result of task i by user j; if the user successfully completes the task, then Nij = 1, if not, then Nij = 0
tij = The time spent by user j to complete task i. If the task is not successfully completed, then time is measured till the moment the user quits the task (in *minutes*)

$$Time\ Based\ Efficiency\ of\ Format\ A\ = \frac{\left(\frac{1}{5{,}33} + \frac{1}{3{,}13} + \frac{1}{6{,}56} + \frac{1}{3{,}53} + \frac{0}{7{,}6} + \frac{0}{7{,}18} + \frac{0}{6{,}01}\right)}{7}$$

$$\boldsymbol{Time\ Based\ Efficiency\ of\ Format\ A\ = 0{,}1346\ (tasks/sec)}$$

$$Time\ Based\ Efficiency\ of\ Format\ B\ = \frac{\left(\frac{1}{5} + \frac{1}{4{,}23} + \frac{1}{2{,}86} + \frac{1}{2{,}72} + \frac{0}{5{,}6} + \frac{1}{2{,}28} + \frac{1}{2{,}45}\right)}{7}$$

$$\boldsymbol{Time\ Based\ Efficiency\ of\ Format\ B\ = 0{,}2857\ (tasks/sec)}$$

The results show that Format B has a better tasks per second ratio than Format A, meaning it has a better projection of scalability when and if the system should grow and be added more tasks (in this case, mathematical or logical problems) to it.

Lastly, we proceeded with the conduction of a survey in which we asked the participants which format they deemed:

- **Easier to understand:** *2 out of 7 said FORMAT B*
- **Easier to solve:** *5 out of 7 said FORMAT B*
- **More intuitive:** *7 out of 7 said FORMAT B*
- **Better presented:** *7 out of 7 said FORMAT B*
- **Taught the lessons to learn better:** *4 out of 7 said FORMAT B*

And we consequently decided to represent that information in a formatted way (figure 4)

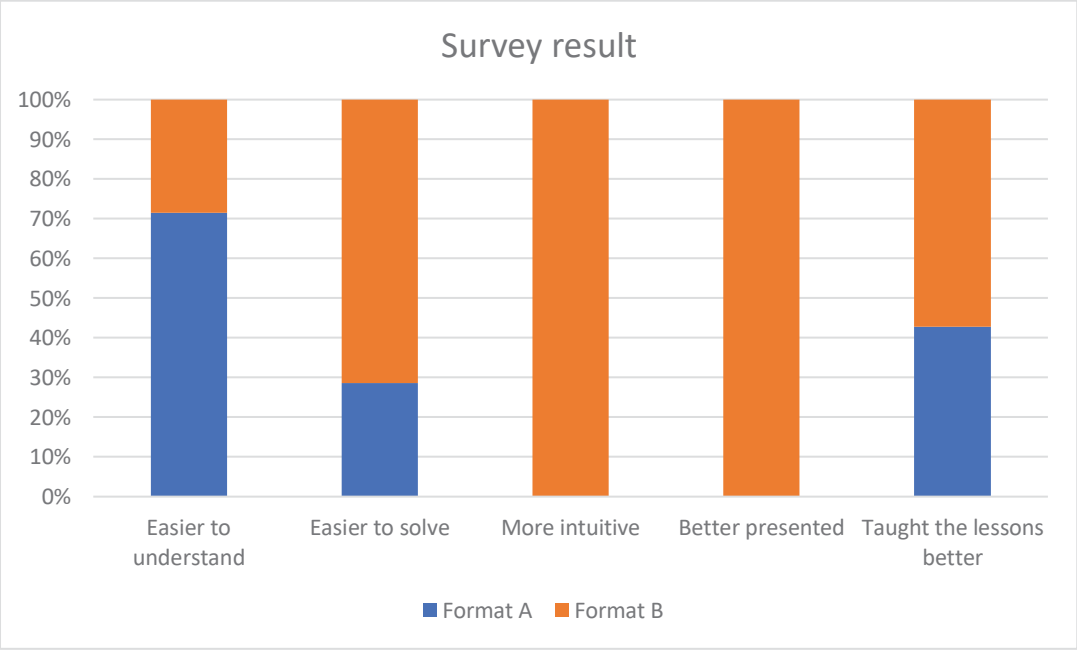


Figure 4. Stacked Column graph of the result of the survey conducted

As we can see in figure 4, the conclusions we can come to are that even though format A, the paper format formulation, resulted to be easier to understand than format B, digital format with objects and buttons; the reason behind is that in the paper format all the information: clues, restrictions, conditions, etc., is been given to

the user thus taking away the opportunity for the user to learn the reasons why that information exists and therefore, discouraging the student to think his or her way through the problem.

Another conclusion to get out of the figure is the fact that overall, most of the participants felt that Format B was more intuitive and better presented and although some considered the other format to be easier to solve and almost half of them thought the other format taught the lessons to learn just the same; it proved to be a better format for the better learning of the logic problem at hand.

To conclude with the evaluation of the system, we decided to make a final data exposition to contemplate other factors that must be taken into account but which might not influence the main parameters measured of Time Based efficiency and Task Completion Rate (or *Effectiveness*). These factors are: the number of mistakes and the participants preference of format (figure 5)

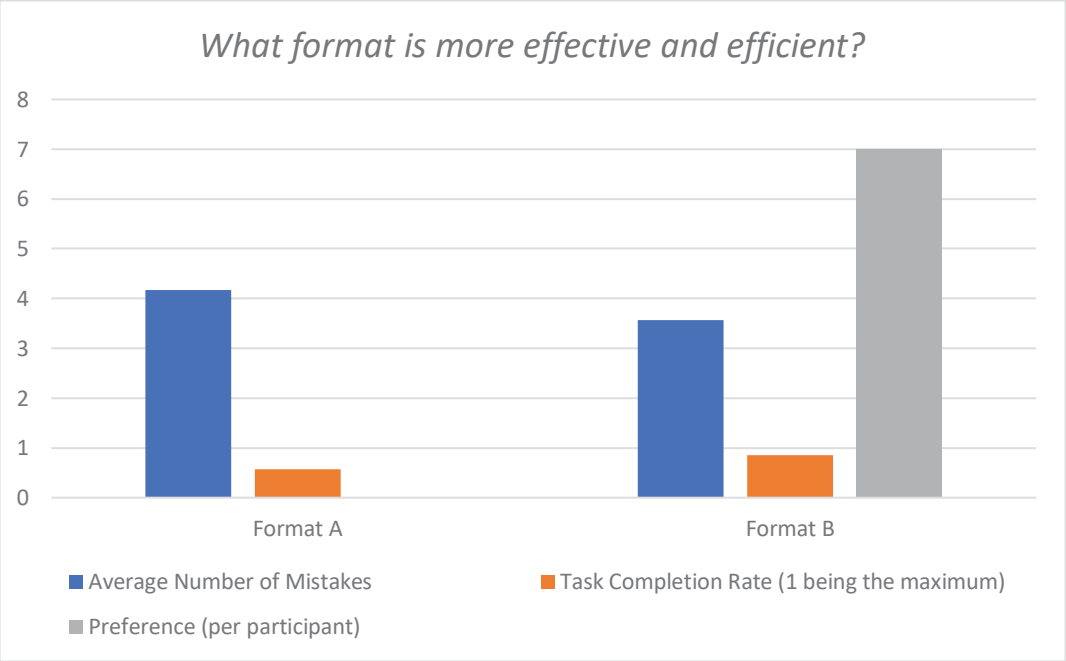


Figure 5. Bar chart comparing other relevant factors with the Task Completion Rate

The average number of mistakes are important because they not only show the mistakes made but they can also define many different things like the level of difficulty with which the problem is presented in their own format, how long the user takes to understand the problem based on how many mistakes it takes to learn the information necessary to solve it (in a trial-error sort of way).

The mistakes are put together with the Task Completion Rate in order to make a joint conclusion relative to the effectiveness of each format. Therefore, a lower number of mistakes and a high task completion rate means that format allows for a better self-learning process because the user has made (a) mistake/s and used this mistake/s to solve the problem, thus, learning from an error through trials.

With this in mind, we can see that format B has a lower number of mistakes and a higher task completion rate compared to format A and that, consequently, supposes a better and more efficient format for teaching. Plus, if we also take into account the user preferences, the format B is not only more efficient but also, it's a format preferred by the users that so far have tested it.

DISCUSSION AND CONCLUSIONS

Since the conception of this idea for a thesis project, we have embarked in a journey that has led us to learn, discover and prove more than we have ever imagined. We had the concept there, Gamification. We had heard about it before, we had thought of someday, once we knew more about programming and computer engineering, wanting to contribute to society through this really outstanding and potential concept use. And through this project we had established the main reasons why it could work.

Reviewing and considering once again the summary list of our list of goals we decided in the Work Plan and later improved in the Follow-Up reports:

- ***Creating an interactive and dynamic learning platform*** *as contribution to the research potential of a better teaching and learning methodology that is Gamification.*

- ***Creating a wide age-ranged learning platform*** *lacking a specific user-base in order to make this learning method accessible to anyone and everyone.*

- ***Adapting and applying the knowledge and skills acquired in the degree*** *for a practical use in the professional world.*

- ***Understanding the intricacies of a project of this complexity*** *and the application of Software Engineering skills and methodology learned in the degree to push the project forward.*

- ***Learning new and "state of the art" technologies*** *used in the development of Human-Computer interaction software with applications in the Computer Graphics field.*

- ***Acquire new skills and knowledge*** *through and of the software tools and programs that were necessary to implement this project and all its assets as well as any methodology or theories applied.*

- *Getting a closer look to the process of* ***designing a system to improve some aspect of society*** *and all its perks.*

- *Getting a* ***hands-on experience in the development of a software system*** *and what phases it goes through from its initial conception to its final release.*

- ***Study and research the possibility of making this project profitable*** *to academic entities and of whether they suppose and improvement in education in relation to an improvement-investment ratio.*

- *Provide a method of improvement to the education field to serve as a contribution to society.*

We can successfully conclude that, based on the results of the evaluation done of the system in the *Evaluation* section of this document, we have successfully fulfilled if not all, many of the objectives set out to fulfill, namely we have indeed:

- Succeeded in implementing a more than decent system to present a new educational model using technology we had little to no experience of use before we began the project and of which we now feel like we've gotten to learn more about, thus contributing to the use of *Gamification as a Learning Platform*, as our thesis states.

- Succeeded in testing the system through a A/B usability tests with different aged participants in order to get a wider range of age for our system to be used in and proven to be more effective than other formats of problem formulation (as seen in the *Task Completion rate (Effectiveness)* and the *Time-Based Efficiency* tested in the Evaluation phase)

- Succeeded in using our computer engineering skills and the knowledge acquired in many of the courses of our degree to develop a complex system with a real-life application.

- Succeeded in applying our software engineering knowledge to properly structure, build and document out system in a formal and professional way.

- Succeeded in learning in a bigger or smaller scale, the complexity and the workings of advance tools related to the computer graphics world that is *Unity* and also a new programming language that is C#.

- Succeeded in having a first-hand experience in the development of a big and serious software project that involved a lot of work, dedication, order and above all learning.

- Succeeded in creating a learning platform that could rival the actual methods used in educational systems and could substitute it in the future in order to provide a better and more efficient education.

Some objectives might have not been reached as they may have been discarded due to their lack of connection with the main goal and purpose of the system. Objectives like analyzing and studying the profit the system could have compared to other methods, a case study of its possible benefits and performance in the global market were not considered as the thesis focused mainly in providing a better education and whether it was possible and if it, in the end did.

Even though based on the evaluation done gave a relatively worst result of the format A, the paper-based formulation format of the problem of Crossing the River, it's fair to point out that the participants mentioned they preferred the other one only because this format would take more time to solve as they would also have to plan out how to represent the solution (something the system with its implemented format B was saving for them); and because they felt they "learned less" with the other one.

Our system was able to solve many of the objectives we set out because it made use of basic principles we learn in our degree. We learned to evaluate the system, to come up with an optimal design, to learn what it needed and what it didn't, set up the objectives, fears, possibilities. We took an issue that we saw was real, that it was happening in the world that was the outdated education system's teaching methods (in our particular case, a logic or mathematical problem) and we used our knowledge in software engineering and implementation and we provided an alternative, an improvement.

We never meant to undermine the actual method of today but rather provide an innovative and better method. And as we saw in the evaluation, as carried out the A/B usability testing we kept seeing how our system had a potential to be successful.

Nevertheless, some objectives were not met, and we knew that it was going to be difficult to fulfill all of them. We had the feeling the system itself was too ambitious, we felt there was too many new things or experiences we had never experienced that were going to weight on us later on. We had never worked with a powerful game engine like *Unity*, or have never planned, structured and developed a software development project before; so throughout all the phases of our project, from the System Design to the User Testing, we felt like that not only were we nervous and stressed for doing it right but also we were learning so much in the process.

We feel proud of how our system ended up looking and its performance in the User Testing phase. But we of course are aware of some aspects that were left to be improved or never implemented. Some include the addition of more scenes and, therefore, levels of difficulty; the optimization of the code behind the system, the improvement of the UX and interactions of the elements, the addition of new and more visually attractive features like animations and special effects, and so on.

Up to a point where we began to stress out as we weren't learning certain functionalities that we needed, it didn't matter how much we'd research on the topic, no solution seemed to work. Without realizing it, we began to lose focus of the main goal, not because we were growing tired of the project itself but because, for the implementation of something as simple as a transition or an *onClick()* event triggering, we had to invest a lot of time and solve a lot of errors, most of them related to how Unity handles referencing objects and scripts; but the majority had to do with the handling of objects through the various scripts which we needed to make them work in sync but had a hard time doing it.

Something we thought would be easy like adding the grass texture to the terrain became a complicated task as it had a thousand ways to do it, each more complicated than the next, and it didn't help that with the updating of the engine and the programming language at hand or even the addition of plugins and other applications that required a further knowledge to learn, there was never a solution compatible with what we already had implemented.

A clear example of this can be found on the camera smoothness functionality. When implementing the camera to follow behind the player it sometimes shook too much increasing the risk of giving mild motion sickness, which was not beneficial for its use with kids should it ever wind up being used in education like we intended; so we

learned of ways of adding smoothness to the camera movement but when following them to implement such functionality one of two things would always happen: either it overstepped on the camera movement code we already implemented or it was meant for a different kind of player perspective or gameplay so it was no use as it forced me to change the player code instead.

Unity is a versatile and complex tool and getting to learn it in its entirety was no easy task and impossible in the time we were given, nor was it our intention to learn it all in this project. However, should we have had more experience with the tool and of course time, we could have engineered ways to implement many of the code we had, to improve the system we already had and to make it more visually appealing and more scalable through refactoring.

Some ideas were also scrapped due to the lack of experience and time but never left forgotten. If anything, the system was implemented to leave room for improvement, and it was structured so that on top of it more and more functionalities would be added. Nevertheless, amongst other things, we intended to add a second scene with a higher level of difficulty, a fully functioning NPC (Non-Playable Character, some sort of AI that'd talk to the user) like the seen robot professor that could guide and give instructions to the user through the main character. Also, we never took the time to properly model, with a 3D sculping tool, the main character or any of the characters and elements that we had designed specifically for this system.

But we don't feel like the journey ends here. We know that with all the effort we put in this project and all the interest, knowledge and time spent in it we have plans to continue working on it. To get it ready to be presented to society and provide a better and more effective learning platform newer and older generations, impacted by the technological era we are witnessing, would enjoy. And all of that made possible by applying the concept of Gamification but implementing it with the skills and knowledge of a computer engineer.

BIBLIOGRAPHY/REFERENCES

[1] Olga R. San Martín. (2015). *Los cinco grandes problemas del profesorado español* [Online]. Available: https://www.elmundo.es/sociedad/2015/11/03/5637c9dc268e3e02488b456c.html

[2] Education.com. (November 2010). *Are traditional grades a thing of the past* [Online]. Available: https://www.education.com/magazine/article/traditional-grades/

[3] Anna Penido. (2018). *Many students are unengaged in their learning—how schools can create opportunities for participation* [Online]. Available: https://www.brookings.edu/blog/education-plus-development/2018/03/20/many-students-are-unengaged-in-their-learning-how-schools-can-create-opportunities-for-participation/

[4] Anonymous. (2016). *Algoritmos ABN: por unas matemáticas sencillas, naturales y divertidas* [Online]. Available: https://observatorio.profuturo.education/blog/2016/08/08/algoritmos-abn-por-unas-matematicas-sencillas-naturales-y-divertidas/

[5] Anonymous. (Last edited June 2019). Wikipedia Source:

[5.1] *Gamification* [Online]. Available: https://en.wikipedia.org/wiki/Gamification

[5.2] *SimCity (1989 video game)* [Online]. Available: https://en.wikipedia.org/wiki/SimCity_(1989_video_game)

[6] Dustin Bethel. (2017). *A brief history of Gamification* [Online]. Available: https://bit-lever.com/brief-history-gamification/

[7] Bryant Nielson. (2018). *The History and Direction of Gamification* [Online]. Available: https://www.yourtrainingedge.com/the-history-and-direction-of-gamification/

[8] Jared Polites. (2017). *Why the gamification market industry will grow 500% to $11 billion by 2020* [Online]. Available: https://blog.atrivity.com/why-gamification-industry-will-grow-to-11-billion-by-2020

[9] Justin Mifsud. (2015). *Usability Metrics – A Guide to Quantify the Usability of Any System* [Online]. Available: https://usabilitygeek.com/usability-metrics-a-guide-to-quantify-system-usability/

[10] Jeff Bryant. *Who, What, and Why – A Guide to User Testing Methods* [Online]. Available: https://www.toptal.com/designers/ux/user-testing-methods

[11] Jeff Sauro. (2011). *What is a good Task-Completion rate?* [Online]. Available: https://measuringu.com/task-completion/

Este documento esta firmado por

Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
Fecha/Hora	Tue Jul 02 11:16:45 CEST 2019
Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
Numero de Serie	630
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)