

Bachelor in Computer Science and Engineering  
2016/2017

*Bachelor Thesis*  
**Designing User Experiences: a Game  
Engine for the Blind**

---

Álvaro Cáceres Muñoz

Tutor/s: Teresa Onorati  
Thesis defense date: July 3<sup>rd</sup>, 2017



This work is subject to the Creative Commons **Attribution-NonCommercial-NoDerivatives 4.0** International Public License.

## Abstract

Video games experience an ever-increasing interest by society since their inception on the 70's. This form of computer entertainment may let the player have a great time with family and friends, or it may as well provide immersion into a story full of details and emotional content.

Prior to the end user playing a video game, a huge effort is performed in lots of disciplines: screenwriting, scenery design, graphical design, programming, optimization or marketing are but a few examples. This work is done by game studios, where teams of professionals from different backgrounds join forces in the inception of the video game.

From the perspective of Human-Computer Interaction, which studies how people interact with computers to complete tasks [9], a game developer can be regarded as a user whose task is to create the logic of a video game using a computer. One of the main foundations of HCI<sup>1</sup>. is that an in-depth understanding of the user's needs and preferences is vital for creating a usable piece of technology. This point is important as a single piece of technology (in this case, the set of tools used by a game developer) may – and should have been designed to – be used on the same team by users with different knowledge, abilities and capabilities. Embracing this diversity of users functional capabilities is the core foundation of accessibility, which is tightly related to and studied from the discipline of HCI.

The driving force behind this research is a question that came after considering game developers: *Could someone develop a video game being fully or partially blind?* Would it be possible for these users to be part of a game development team? What should be taken into account to cover their particular needs and preferences so that they could perform this task being comfortable and productive?

The goal of this work is to propose a possible solution that can assure inclusion of fully or partially blind users in the context of computer game development. To do this, a Used Centered Design methodology has been followed. This approach is ideal in this case as it *starts including people you're designing for and ends with new solutions that are tailor made to suit their needs* [106]. First, previously designed solutions for this problem and related works have been analyzed. Secondly, an exploratory study has been performed to know how should the target user be able to interact with a computer when developing games, and design insights are drawn from both the state of the art analysis and the study results. Next, a solution has been proposed based on the design insights, and a prototype has been implemented. The solution has been evaluated with accessibility guidelines. It has been finally concluded that the proposed solution is accessible for visually impaired users.

**Keywords** Human-Computer Interaction, User Centered Design, Accessibility, C++, Game development, Unity3D, Godot Engine, Unreal Engine

---

<sup>1</sup>Acronym. Stands for *Human-Computer Interaction*

## Acknowledgements

If someone asks about the author of this thesis, it may seem reasonable to answer with Álvaro Cáceres. However, this thesis would have never seen success (and much less to do so before the established deadline) if it had not been for the help of lots of people who helped, taught and motivated me through the whole process of writing the thesis. I thought it was appropriate to mention them here, so that the reader is aware of the fact that any piece of human knowledge is available to us nowadays thanks to friendship and teamwork.

First of all, I want to thank my bachelor thesis supervisor, Teresa Onorati. She has been extremely comprehensive, patient, supportive and willing to help. I know that it is easier to tell bachelor students to choose a research topic already defined by professors, however she has defended my ambitious ideas since the beginning. I am doubly thankful for all the wise pieces of advice she has given me during these months, not only about the thesis, but about the life of a researcher and lots of topics about life in general. Finally, I also have to thank her for praising my abilities and potential beyond my personal circumstances. During the whole bachelors degree, I have been splitting my time between two different studies, and I know from other professors how puzzling it is to listen to a student who has so many *crazy projects* in mind and so little time to keep up with everything.

Lots of people have also given me advice and solved serious problems that have arisen over the creation of this thesis. I firstly have to thank professors from Universidad Carlos III de Madrid. Telmo Zarraonandia perfectly explained the state of development of GREP, which has been essential to consider different alternatives for the design of the solution. José Antonio Iglesias gave me an extensive tutoring lesson about my university's bachelor thesis criteria, and it was extremely helpful. Ángel García also deserves my gratitude, as he told me about an accessibility project, GoAll, that was being developed at my university by the time I was writing the thesis. Secondly, I thank all people who had never heard of me and still helped me. Here I would include Francisco Monzón from ONCE, who despite of being so busy devoted his time to search for participants for the exploratory study. He also wanted to know about other projects and ideas I have, and he was extremely supportive for those. Aleksander Morgado made me not fear the DBus protocol anymore, and he was enthusiastic about my project as both the two of us defend Free Software principles. Alejandro Piñeiro also offered me extensive help regarding the ATK library; I can say I was even more scared of ATK than DBus, so his help has been greatly appreciated. Thirdly, I would like to thank people from Pinto Association for People with Disabilities, who made me save time redirecting me to the ONCE offices that were relevant for my thesis.

Both family and friends have also had a big impact in this thesis. My family have always supported me, and made me think that my project was close to receiving an award for saving humanity. My friends were equally enthusiastic, especially those who, as I do, love the Linux and Free Software world; they were excited of thinking that thanks to me, Godot Engine was not only going to be the first successful Open Source game engine, but also the first one that was fully accessible. I also wanted to mention Jorge Hidalgo, both for his support and for all his help about laws and user consents; he is amazingly skilled and good-hearted, and I know for sure that a great future awaits him,

both professionally and personally. I have to especially thank Carmen López for all the affection and encouragement she has given me, not only for this thesis but for all my goals and projects in my life. When I first devised this project (around two years before starting the bachelor thesis), I thought that the design of the solution was going to be a game engine built from scratch by me; I wanted to name the program Carmen, and for a good reason (*"Carmen V2.35 stable"... it definitely sounds exotic*).

Last but not least, I want to appreciate all the people that have participated in the exploratory study. As I have already said, I was absolutely unaware about all the things I was going to learn prior to starting writing my bachelor thesis, and most of the things I have learned come from their personal experiences, which have been priceless for me. Their encouraging messages have touched me several times. It feels great to know that this project is much more than a bachelor thesis for me, but it feels even better to know that people out there think that my project is going to help so many people. I sincerely hope this project goes much beyond a bachelor thesis, and that it actually gives way to accessible game development.

# Contents

<b>1</b>	<b>Introduction</b>	<b>8</b>
1.1	Motivation . . . . .	8
1.2	Socioeconomic environment . . . . .	9
1.3	Research method . . . . .	9
1.4	Proposed solution . . . . .	12
1.4.1	Input and output design . . . . .	12
1.4.2	Solution implementation . . . . .	12
1.5	Outline . . . . .	13
<b>2</b>	<b>State of the art</b>	<b>14</b>
2.1	Definitions . . . . .	14
2.1.1	Visual impairment . . . . .	14
2.1.2	Functional diversity . . . . .	15
2.1.3	Accessibility . . . . .	16
2.1.4	Assistive technology . . . . .	16
2.2	Assistive technologies for visually impaired users . . . . .	17
2.3	User interfaces for visually impaired users . . . . .	17
2.3.1	3D views . . . . .	17
2.3.2	3D audio games . . . . .	18
2.3.3	Cognitive mental models of 3D views . . . . .	19
2.4	Accessible software development tools for visually impaired users . . . . .	19
2.5	Game development tools for visually impaired users . . . . .	20
<b>3</b>	<b>Design insights</b>	<b>22</b>
3.1	Exploratory study . . . . .	22
3.2	Participants . . . . .	23
3.2.1	Participant profile . . . . .	23
3.2.2	Searching for participants . . . . .	24
3.2.3	Demographics . . . . .	24
3.3	Interviews . . . . .	26
3.4	Design insights . . . . .	33
<b>4</b>	<b>Design of the solution</b>	<b>36</b>
4.1	Chosen technology . . . . .	36
4.1.1	Available technologies comparison . . . . .	36
4.1.2	Godot Engine user interface analysis . . . . .	39
4.2	Design specification . . . . .	44
4.2.1	Input mode . . . . .	44
4.2.2	Output mode . . . . .	51
4.2.3	Configuration . . . . .	53
4.3	Implementation details . . . . .	55
4.3.1	New user interface elements . . . . .	55
4.3.2	Screen reader . . . . .	55
4.3.3	3D Audio . . . . .	56
4.4	3D view prototype . . . . .	57
4.4.1	Purpose . . . . .	57
4.4.2	Prototype implementation . . . . .	58

<b>5</b>	<b>Evaluation</b>	<b>60</b>
5.1	Evaluation method . . . . .	60
5.1.1	WCAG guidelines . . . . .	60
5.1.2	Future user evaluation with the 3D view prototype . . . . .	61
5.2	Evaluation with WCAG 2.0 checklist . . . . .	62
5.3	Results . . . . .	62
<b>6</b>	<b>Project management</b>	<b>64</b>
6.1	Regulatory framework . . . . .	64
6.1.1	Accessibility standards . . . . .	64
6.1.2	Software licenses . . . . .	64
6.1.3	User privacy . . . . .	67
6.2	Planning . . . . .	68
6.3	Budget . . . . .	72
<b>7</b>	<b>Conclusions</b>	<b>76</b>
7.1	Technical conclusions . . . . .	76
7.2	Future work . . . . .	76
7.3	Personal conclusions . . . . .	77
	<b>Glossary</b>	<b>80</b>
	<b>References</b>	<b>81</b>
<b>A</b>	<b>WCAG 2.0 checklist</b>	<b>88</b>
A.1	Perceivable . . . . .	88
A.1.1	Guideline 1.1: Text Alternatives . . . . .	88
A.1.2	Guideline 1.2: Time-based media . . . . .	89
A.1.3	Guideline 1.3: Adaptable . . . . .	90
A.1.4	Guideline 1.4: Distinguishable . . . . .	91
A.2	Operable . . . . .	92
A.2.1	Guideline 2.1: Keyboard Accessible . . . . .	93
A.2.2	Guideline 2.2: Enough Time . . . . .	94
A.2.3	Guideline 2.3: Seizures . . . . .	94
A.2.4	Guideline 2.4: Navigable . . . . .	95
A.3	Understandable . . . . .	96
A.3.1	Guideline 3.1: Readable . . . . .	96
A.3.2	Guideline 3.2: Predictable . . . . .	97
A.3.3	Guideline 3.3: Input Assistance . . . . .	98
A.4	Robust . . . . .	98
A.4.1	Guideline 4.1: Compatible . . . . .	99

## List of Figures

1	User Centered Design methodology . . . . .	11
2	UCD Methodology. Phase 1: State of the art . . . . .	14
3	UCD Methodology. Phase 2: Design insights . . . . .	22
4	Exploratory Study. Participant age distribution . . . . .	25
5	Exploratory Study. Visual impairment distribution . . . . .	25
6	Exploratory Study. Software development experience distribution . . . . .	26
7	Exploratory Study. Game development experience distribution . . . . .	26
8	UCD Methodology. Phase 3: Design of the solution . . . . .	36
9	Godot Engine: project manager screen . . . . .	40
10	Godot Engine: shortcuts tab . . . . .	41
11	Godot Engine: main screen . . . . .	42
12	Godot Engine: 3D view . . . . .	43
13	Mockup for the accessibility section . . . . .	54
14	3D view prototype running . . . . .	58
15	UCD Methodology. Phase 4: Evaluation . . . . .	60
16	Gantt chart . . . . .	72

## List of tables

1	Game development kits comparison . . . . .	38
2	Accessibility problems in Godot Engine . . . . .	44
3	Design specification: keyboard shortcuts for menus . . . . .	45
4	Design specification: keyboard shortcuts for the 3D view (1) . . . . .	47
5	Design specification: keyboard shortcuts for the 3D view (2) . . . . .	48
6	Design specification: keyboard shortcuts for the 3D view (3) . . . . .	49
7	Design specification: keyboard shortcuts for the 3D view (4) . . . . .	50
8	Design specification: keyboard shortcuts for the 3D view (5) . . . . .	51
9	Planning (1) . . . . .	69
10	Planning (2) . . . . .	70
11	Weekly dedicated hours . . . . .	71
12	Complete project costs (1) . . . . .	74
13	Complete project costs (2) . . . . .	75
14	Evaluation with the WCAG 2.0 checklist: guideline 1.1 . . . . .	88
15	Evaluation with the WCAG 2.0 checklist: guideline 1.2 (1) . . . . .	89
16	Evaluation with the WCAG 2.0 checklist: guideline 1.2 (2) . . . . .	90
17	Evaluation with the WCAG 2.0 checklist: guideline 1.3 . . . . .	90
18	Evaluation with the WCAG 2.0 checklist: guideline 1.4 (1) . . . . .	91
19	Evaluation with the WCAG 2.0 checklist: guideline 1.4 (2) . . . . .	92
20	Evaluation with the WCAG 2.0 checklist: guideline 2.1 . . . . .	93
21	Evaluation with the WCAG 2.0 checklist: guideline 2.2 . . . . .	94
22	Evaluation with the WCAG 2.0 checklist: guideline 2.3 . . . . .	94
23	Evaluation with the WCAG 2.0 checklist: guideline 2.4 (1) . . . . .	95
24	Evaluation with the WCAG 2.0 checklist: guideline 2.4 (2) . . . . .	96
25	Evaluation with the WCAG 2.0 checklist: guideline 3.1 . . . . .	96
26	Evaluation with the WCAG 2.0 checklist: guideline 3.2 . . . . .	97
27	Evaluation with the WCAG 2.0 checklist: guideline 3.3 . . . . .	98
28	Evaluation with the WCAG 2.0 checklist: guideline 4.1 . . . . .	99



# 1 Introduction

## 1.1 Motivation

Video games make a big impact on different levels of society. At first we have consumers, with more than 28% of world population playing video games on a regular basis [50]. Secondly we have game companies, which generate \$99.6 billion in revenues globally and imply the deployment of 2322 developer offices only in the USA [72]. Lastly, technological advances developed for computer gaming also turn into new market opportunities, a remarkable example being the inclusion of GPU's for scientific computation.

Of the different sections of society that are involved in video games, consumers have access to them irrespective of their level of visual accuracy. Although not all video games are accessible for visually impaired users, some games are being adapted or created from scratch taking into account different levels of visual accuracy [52]. Audio games are also a good solution for these users, as they only output sound instead of sound and image; in this way, users only need their ears to receive full information of the game, and sight no longer becomes a barrier to play computer games.

There is however another important section of society that cannot easily access the world of videogames when suffering from visual impairment: developers. It is true that it is technically possible for visually impaired users to develop the logic of a game without a graphical user interface; in this case all the behavior of the program is specified using code and files. There are visually impaired users who try creating video games from scratch using this approach [52]. However, the majority of professional video games are created using a game engine, which is a piece of software that automates several steps in the creation of video games. Game engines are based on graphical user interfaces, and they are not designed with accessibility in mind. For this reason, professional game development still poses a barrier to visually impaired users.

The range of jobs opportunities that the game industry offers to developers thus sets out the motivation for the research methodology followed across this document. From the viewpoint of the UCD<sup>2</sup> methodology, this motivation is formalized with three concepts: a target user, a set of necessities, and a problem that the user currently experiences. These concepts will be later addressed in more detail; for now, it suffices to know the following:

- The potential *target user* is a visually impaired user who could productively develop games in a professional game studio.
- The *necessities* are whatever means, tools, modes of interaction... that the target user may require to effectively achieve his/her goal. These necessities are still unknown: they will be discovered through the different phases of the applied UCD research methodology.
- The *problem* is that the target user cannot achieve his/her goals, given that his/her necessities have not been covered. In terms of accessibility, the user's problem is that the technology s/he is trying to use is not accessible; making this technology accessible would therefore solve the problem.

---

<sup>2</sup>Acronym. Stands for *User Centered Design*.

## 1.2 Socioeconomic environment

Accessible video game development for visually impaired users is an important mission to accomplish, as it would affect a huge number of users. More specifically, around 4% of world population had some degree of visual impairment in 2014, according to data from the World Health Organization [58]. While this percentage seems small, it translates to 285 million people. There are users whose visual accuracy may be temporarily reduced<sup>3</sup>, so the number of target users is actually much bigger.

One of the ways in which game development benefits society is related to its popularity. More consumption requires more workforce, and for products like videogames (which are relatively expensive) workforce is well remunerated. Game designers in the United States had an average salary of \$73,864 in 2014 [28]; this is slightly higher than the average salary of \$54,525 in the United States by that year [54]. Game programmers and engineers earned around \$93,251 each year by 2014 [28]: this is roughly twice the average salary of a citizen from the United States.

These job opportunities would imply substantial changes on visually impaired users' current quality of life. Demographic data taken from the United States in 2014 shows that an alarming majority of visually impaired adults struggle to find a job: only 40% were employed [36]. Global statistics are even more severe, with 90% of visually impaired users living with low incomes [58]. Probably, not all users with visual difficulties will have a professional or academic background in software development; and still, making this career available for them would drastically increase welfare for this sector of population.

However, these benefits and opportunities will only be possible if inclusion is guaranteed. This assertion is obviously linked to moral principles, but there are important practical reasons to take into consideration. A visually impaired user could –more or less comfortably– develop a game with just a text editor, if it is accessible. This workflow would likely not be accepted in a professional context, where more complex and automated tools are used by the whole development team. This example sets out the difference between accessibility and inclusion. If visually impaired users want to work in a professional team (ergo benefiting from better salaries), they must be able of using the *same* tools as game development studios use.

## 1.3 Research method

As it has been mentioned before, this research project has been based on a UCD research methodology. One of the main concepts of User Centered Design is that it makes technological designers (engineers, computer scientists...) think about the target user in first place. Technology has been traditionally designed with commercial or technical goals in mind. Users had to adapt their mental models to effectively use it, which often results in frustration and low self-esteem towards technology [25]. UCD advocates think that design *should support its intended users' existing beliefs, attitudes, and behaviors as they relate to the tasks that the system is being designed to support* [38]. This makes the designed product easier to use, which generates a feeling of calm, productiveness and even enjoyment in the target user.

---

<sup>3</sup>This idea is explained in more detail in section 2.1.

Quoting the Userfocus webpage [96], the UCD methodology is rooted in the International Usability Standard ISO 13407, which specifies the following main points about tasks and people involved in a design project of this kind:

- *The design is based upon an explicit understanding of users, tasks and environments.*
- *Users are involved throughout design and development.*
- *The design is driven and refined by user-centred evaluation.*
- *The process is iterative.*
- *The design addresses the whole user experience.*
- *The design team includes multidisciplinary skills and perspectives.*

User Centered Design is flexible [97] and allows to be used with slight variations. In this research project, the methodology has followed the typical four phases of UCD:

1. Understanding of the state of the art: this phase aims at answering to questions like: what do we already know about our target users? Have there been previous solutions to the problem we are trying to solve? And in case we have discovered this problem for the first time, is there any related knowledge that may be useful as a reference?
2. Design insights: how should we design the proposed solution? What are target users' needs, feelings and beliefs towards the problem to address? The outcome of this phase is similar to the *requirements* from software engineering methodologies. However, design insights are more flexible and, most importantly, they can change in time as the UCD methodology allows several iterations before reaching a final design decision. In order to get the design insights, I have used two sources of information:
  - Conclusions drawn from the state of the art
  - Exploratory study: this has been chosen as an additional way of gathering data from target users. A set of unstructured interviews has been followed. Despite of variability in users' answers, there are similarities in their views and beliefs towards the studied problem. This gives very useful information that is an essential part of the design insights. State of the art conclusions reinforce the results of this study, as they tend to have a theoretical or empirical foundation.
3. Design of the solution: in this phase, the solution is proposed. The UCD projects require an extensive amount of effort and time to understand users and the way in which the product they need should be designed. For this reason, *design* normally refers to a specification of the user interface of the desired product; implementation of the product may or may not be part of this design. Given that most research in UCD is done about graphical user interfaces<sup>4</sup>, the design of the solution is normally a sketch of that interface. However, the solution for the problem studied in this project cannot be fully described with sketches. This last point will be further described in section 4.

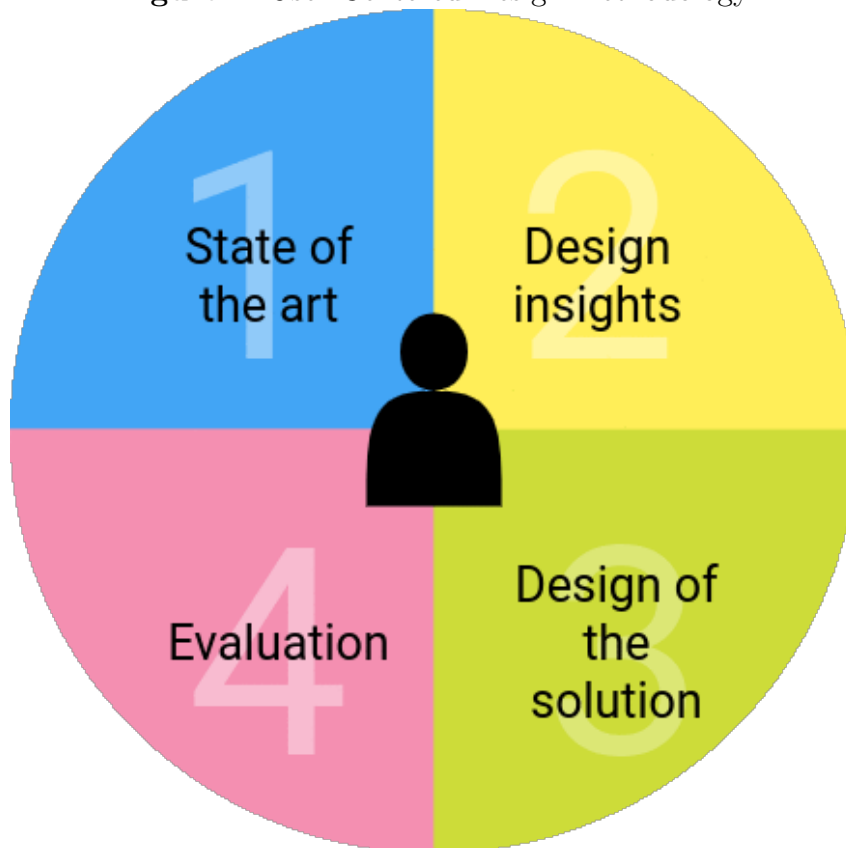
---

<sup>4</sup>A graphical user interface is a set of visual elements that enable humans to communicate with a computer. Examples of these elements are windows, buttons, menus. . .

4. Evaluation: finally, the proposed solution must be tested to make sure that it covers target users' needs. There are several ways of doing this. One of them is to evaluate users' satisfaction by asking them to perform tasks on a prototype of the proposed interface. For instance, user evaluation can involve previously interviewed users (from the exploratory study), or experts in the field of design. Another option is to evaluate the design against a well-designed set of design guidelines and heuristics. In short, these design artifacts have been discussed, validated and agreed by international design committees and researchers; hence their usefulness when evaluating a design. For this research project, accessibility guidelines have been chosen due to their level of detail.

The four phases of User Centered Design can be easily understood by looking at figure 1. They are displayed in a circle, because this research methodology is iterative<sup>5</sup>. As in a circle, each phase takes in inputs the outputs of the previous phase. The avatar of a person is placed in the center of the diagram, symbolizing that the user is involved in each phase of the methodology, either directly or indirectly.

**Figure 1:** User Centered Design methodology



The way this research project has been done is not only appealing because of the care taken towards users' needs and emotions. Hevner's viewpoint of scientific research as a reusable piece of knowledge [12] highlights the power of using this approach towards research. In this sense, the resemblance between software and research is remarkable. Software was originally duplicated across different companies and computers; if companies

<sup>5</sup>Due to the time constraints of a bachelor thesis, only one iteration has been performed in this project.

had not began to reuse software (with libraries, API's, frameworks, etc.), technology would not have advanced as much as it has done in the previous decades. In the same way, a thoughtful design can be later applied to several problems. To be more specific, let's use this project as an example. The proposed solution will consider a specific program, but the design specifications drawn from this project could be used to make other programs (i.e. other game development kits) accessible for visually impaired users.

## 1.4 Proposed solution

### 1.4.1 Input and output design

One of the biggest concerns about this work was to find the optimal modes of interaction. This means finding the best combinations of input and output modes.

Both the experience from users and the studied literature conclude what should be used for input. Visually impaired users take most advantage from keyboard interaction to input data to the game development software. Other means of interaction such as gestures or mouse input have been discarded; literature and users coincide with regard to this point. Moving hands from the keyboard to the mouse without a visual reference is disorienting to some extent. Also, using gestures would require custom hardware, webcams or devices such as the Xbox Kinect<sup>6</sup>; this automatically *labels* visually impaired users in the context of a work team. All this coincides with how sighted developers interact with the computer. They rarely take their hands out of the keyboard, or they do it as little as possible; this means that keyboard allows for more precise, faster interaction.

For receiving information –output– from common interface components (menus, windows, dialogue boxes), screen readers seem to be the most efficient output mode, and the one that can be used for users with any level of visual impairment. Some users with slight vision loss also make use of magnifiers, font size configuration and inverted colors for improving menu navigation. The 3D view of the video game is the most problematic part of the output system. It has been concluded that users prefer a mixture of auditory clues and verbal description of the 3D view (i.e. using the screen reader).

### 1.4.2 Solution implementation

The main outcome of this research has been the re-design specification of the user interface for a well-known game development kit. This specification is reusable and flexible, so it could be used for making any game development kit accessible for visually impaired users. In order to make the design specification accessible and to take into account the important auditory factor of this user interface, the design specification has been expressed in the form of graphical mockups and verbal descriptions.

Several game development kits have been analyzed for a future implementation. Godot Engine [79] has been selected due to its commitment to free and open source software. This means it has more chances than other game development kits to actually incorporate accessibility features in the future. For this reason, its user interface has been analyzed to

---

<sup>6</sup>This is a peripheral device that detects gestures in 3D space. See [78] for more details.

have a real-life example of the proposed redesign. Some implementation details regarding cross-platform accessibility have been discussed.

A small prototype has been designed to test a small part of the proposed user interface in the future [94]. It has been developed using Unity3D as it allows for fast prototyping of cross-platform applications. This prototype is just a small version of what the solution is, which it makes future experiment conditions more controlled.

## 1.5 Outline

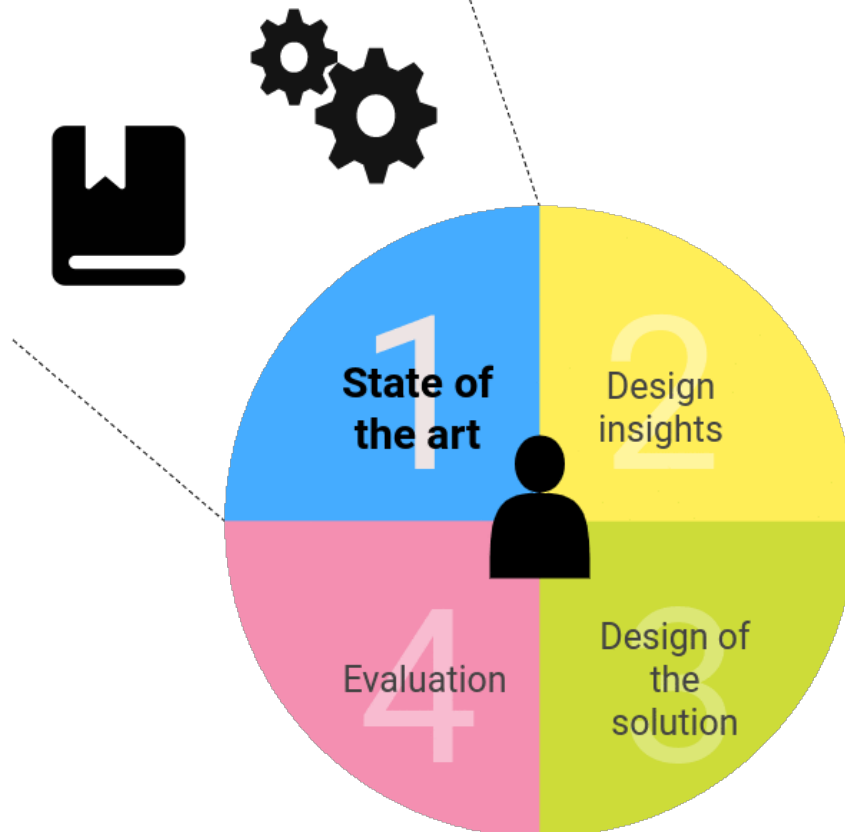
This document has been divided into the following sections:

- Section 1 explains the context in which this project was devised. Firstly, the reasons for proposing this project are shown. Secondly, a strong emphasis has been put on the contribution of this project to society. After this, the methodology followed along the project has been introduced to the reader. Then the proposed solution for the studied problem is described in general terms. Finally an outline (this section) serves as a quick reference of the contents of this document.
- Section 2 presents an overview of previous research and technologies that are related to this project. It also includes a description of key terms that are mentioned across the document, such as accessibility, visual impairment, functional diversity and assistive technologies.
- Section 3 delves into target users needs. The exploratory study required to gather this information is explained in detail, including how participants were chosen and how was it performed. Findings derived from this study, are combined with conclusions from the state of the art. This is used to formulate the problem of this project and a possible solution for that problem.
- Section 4 proposes a design specification that tries to solve target users' problem. Ideas for translating the design specification into a software implementation are also provided in this part of the document. Additionally, a small prototype for an isolated part of the user interface of the program has been developed.
- Section 5 describes how the proposed solution has been assessed in terms of accessibility. Known accessibility guidelines are presented and applied to the proposed solution.
- Section 6 addresses issues regarding planning, legal implications, development and cost of this project. The description of such topics is compliant with Universidad Carlos III de Madrid's bachelor thesis assessment criteria.
- Section 7 concludes the research efforts done during this project. This include conclusions learned from the different phases of the User Centered Design methodology, future improvements, and experiences gathered from a personal point of view.
- The appendix stores secondary but useful data related to the project. This includes a glossary explaining important terms, a full description of the applied evaluation checklist. . . .

## 2 State of the art

Figure 2 makes a reference to the content that is going to be discussed in this section. It is based on the original diagram, although some elements have changed. Colors and text are highlighted for phase 1. A book and a gearwheel come out of the phase 1 section. This symbolizes that both academic and technical solutions have been taken into account for this section.

Figure 2: UCD Methodology. Phase 1: State of the art



### 2.1 Definitions

#### 2.1.1 Visual impairment

It is crucial to understand the concept of visual impairment in order to follow the research work done in this project. Target users mentioned in section 1.1 are, in this case, users with some degree of visual impairment. Basically, a visually impaired user's eyesight cannot be corrected to a "normal level" [24]. In this sense, visual impairment is *the functional limitation of the eye, eyes or the vision system* [24]. The word *functional* from *functional limitation* comes from visually impaired users not being able to make full use of the functions provided by their vision system. The word *limitation* does not mean that they cannot use vision in any way, but rather than they cannot use it to its full potential, thus being limited. Because of this, there are not only people with or without visual impairment; there are different degrees of visual impairment, and designing a solution for visually impaired users requires understanding all these different degrees of functional limitation.

An intermediate concept that should be explained before delving into the types of visual impairment is the term *visual acuity*, because visual impairment is related to how good our visual acuity is. Through this document, numeric notation like 20/80 is used. This number indicates visual acuity, or how sharply the user can see. For instance, if a user has a visual acuity of 20/80, this means s/he can clearly see an object 20 feet away, while a normal person can see it being 80 feet away [71].

There are several classifications for visual impairment. Some people classify users according to the biological condition that causes their loss of visual acuity (amblyopia, retinitis pigmentosa, strabismus. . . ); some others classify them according to how can they use vision to perform tasks. This second categorization has been used because it better describes target user's needs –in terms of Human-Computer Interaction– than a medical condition. Citing [20], there are four degrees of visual impairment according to how it affects task solving:

1. Partially sighted: user's task-solving performance is adversely affected, *even when corrected to the extent possible* [20]
2. Low vision: user's visual acuity is between 20/70 and 20/160 and it cannot be corrected
3. Legally blind: user's visual acuity is between 20/200 and 20/400 (legally blind with severe low vision) or between 20/400 and 20/1000 (profound visual impairment, *which is very close to total blindness* [20])
4. Totally blind: user cannot perceive light

### 2.1.2 Functional diversity

As described by the World Health Organization, disability *"is an umbrella term, covering impairments, activity limitations, and participation restrictions"* [58]. It is also worth noting that *"almost everyone will be temporarily or permanently impaired at some point in life"* [18]. This leads to the concept of *functional diversity*. This term is almost self-explanatory; it describes a variety –diversity– of users with different capabilities, different ways to solve the same problem.

Introducing the concept of functional diversity highlights two ideas that are important for this project. Firstly, making a software product accessible for visually impaired users should not be considered an extra functionality, but a necessary one. At the end visually impaired users belong to the intricate set of (functionally diverse) users, and they have specific needs and preferences, just like everyone else does. Secondly, considering visually impaired users when designing a product also has advantages for other users; they may not have a permanent functional limitation, but at some point in time their visual system may be somehow limited, and the technology that enables interaction for visually impaired users will then be useful for these users. Preparing technology for this functional diversity is therefore vital for everyone; it can either improve our productivity, making our life more comfortable, or it can solve an otherwise serious problem (due to not having access to technology in a critical situation, for instance).



### 2.1.3 Accessibility

What was stated in the last section as preparing technology for functional diversity is equivalent to –and it is normally known as– making technology accessible. Accessibility, as its name indicates, tries to give users with disabilities access to technology [34]. If a product is not accessible, some users will not be able to use it, or they will only be able to use it to a certain degree, but not fully.

Accessibility has been heavily studied by companies and universities during the last decades. A proof of how important has accessibility become in the field of computer science is the adoption of accessibility guidelines by companies such as Microsoft [108], Google [105] and Apple [62]. In the end, accessible products attract more clients who could not use them before. Accessibility is beneficial both for clients and for companies.

This research work follows a User Centered Design methodology because of its connections with accessibility, which is part of Human-Computer Interaction. For instance, Design for All [103], which tries to make technology available to anyone (that is, without using dedicated accessible technology) originated in Human-Computer Interaction research. This shows that the UCD research methodology is appropriate for this project.

### 2.1.4 Assistive technology

At this point the reader should know who is the user we are looking for (visually impaired users), how does this extend to other users (functional diversity), and what disciplines are in charge of solving his/her current problem (Human-Computer Interaction, accessibility). A natural question that may arise now is, which technology can solve this problem? How does accessible technology look like?

This technology is known as *assistive technology*, because it supports –assists– users. Assistive technology is not necessarily a separate piece from the original product. Sometimes, assistive technology is designed to be used with almost any product, and sometimes it is specifically incorporated to a single product. This depends on several factors. If a separate assistive technology is clearly visible (like a braille display), it automatically labels the user to the rest of the world, which generates anxiety for him/her. However, there are separate assistive technology products that are only perceptible to the user; for instance, a screen reader. These technologies will be further discussed in following sections of this document.

An example of how assistive technologies can improve lives of a variety of users is speech recognition [56], which was originally designed to assist people with motor functional diversity and is nowadays used by almost any smartphone user, given that it allows for hands-free operating the smartphone; this is useful when performing another critical task with hands (e.g driving) or when being somewhat far from the device (e.g. when sitting on a couch with the phone being 2 meters away from you) and feeling unwilling or unable to move to reach the device at that moment.

Like speech recognition, a game engine that can be used without looking at a screen can be really useful for a wide variety of users. To clarify this, let's consider a common scenario in game development: the user (in this case, the developer) is editing a VR game in Unity3D, a videogame development kit widely used in the game industry [93]. Testing a VR game implies using a VR headset, for instance the Oculus Rift [83]. This workflow is not comfortable for the user because VR headsets are not recommended to be used for extended periods of time (even less for software development), so it is necessary to constantly take the headset on and off, which can generate eyestrain due to the change of light between the headset and the monitor. If it was possible to develop videogames without having to look at a screen, VR game developers could constantly wear the VR headset when working, turning it on to test the game and turning it off otherwise and perceiving the UI of the game editor with assistive technology.

## 2.2 Assistive technologies for visually impaired users

Nowadays there are thousands of assistive technology products that make visually impaired user's life much easier. Most commonly used technologies are the following:

- Screen magnifiers: either software or hardware-based, screen magnifiers increase the size of a text area on the screen. They cannot be used by totally blind users, as opposed to low vision users who can perceive light and shapes to a certain degree. Most widely used operating systems contain free screen magnifiers by default: *Zoom pane* on macOS, *Enhanced Zoom Desktop*, *gnome-mag* and *KMag* on Linux-based distributions, and *Magnifier* for Windows since Windows98.
- Screen readers: a screen reader is a software that receives information from the computer (UI's, operating system events, text...) and generates either synthesized voice or change in a braille display as an output. This piece of technology is extensively used by UVFD who experience a total lack of vision, but it can be used having partial lack of vision as well. There are several screen readers available; mostly used ones include JAWS and NVDA for Windows, VoiceOver for MacOS, and ORCA for Linux.
- Refreshable braille screens: these devices contain small rounded tips that are raised or lowered according to the focused information on the computer, creating an array of dots that display the text in braille [86]. They normally display one line of 80 characters, and they are expensive due to the materials they are built with. Users normally access the computer with the keyboard and receive most part of information from the synthesized voice of the screen reader; this, together with the small amount of data that can be shown on refreshable braille screens and their price, make them secondary tools that are either used when checking for specific symbols or directly omitted.

## 2.3 User interfaces for visually impaired users

### 2.3.1 3D views

Most part of the user interface of a program can be identical for sighted and visually impaired users. Assistive technologies shown in previous sections allow visually impaired users to interact with commonly used graphical user interfaces. When a program is

designed to establish communication with a screen reader, for instance, the program will look the same to any user; the only difference will reside in whether the program sends data about the user interface to the screen reader or not. In this sense, most part of the menu navigation from a graphical user interface is said to be easily accessible. In fact, several user interfaces are almost fully accessible by default, such as the GNOME desktop environment [29].

Despite of accessible menu navigation, there are still elements of the user interface that visually impaired users cannot easily access. Let's consider a game development kit. These toolkits combine classical elements from the graphical user interface (such as buttons, text fields or menus) with one or more 3D views. A 3D view is a three-dimensional graph with references like a grid and 3D axes. This 3D view allows to detect, at a glance, most relevant elements of a videogame scene. 3D views are normally placed in the center of the program's window; it is done this way because game designers tend to constantly look at it. This important part of a user interface is absolutely transparent for visually impaired users, since 3D views are user interface elements that are normally not known for screen readers.

### 2.3.2 3D audio games

3D audio games are interesting examples of 3D user interfaces that (normally) work for visually impaired users. The adverb *normally* was added to the former sentence because an audio game may not be fully accessible. In fact, the game developer of *Ear Monsters*, a 3D audio game, discusses in [26] the differences between making an audio game and making an accessible audio game. This blog post is also interesting because it highlights the importance of properly designing information for the audio game. After testing with users, the creator of *Ear Monsters* decided to drastically simplify conditions in the game; for instance, he changed the (originally numerous) amount of enemy spawning points to be small and fixed.

Researchers at the University of Bucharest [35] developed a 3D audio game for visually impaired users. They used realistic 3D audio techniques that generate the correct sensation of spatial location. It is worth noting that they also made a great effort designing icons (or rather, auditory icons). Auditory icons require being clearly distinguishable between them, but putting a high number of them can saturate user's short-term memory.

A third audio game worth noting is *Shades of Doom* [14]. It was specifically developed for visually impaired users and it is among the first person shooter audio games ever made. One of the causes of its success is the combination of realistic 3D audio, carefully chosen auditory icons, continuous synthesized sounds, and verbal speech.

Finally, [7] explains the creation of a 3D audio game similar to the formerly described. The key difference is the use of haptic interaction, that lets the user feel the touch of objects with a special piece of hardware. While this functionality allows more immersive gameplay, such hardware would be inappropriate in the context of developing games in a team. As described before when learning about assistive technologies, using this hardware would automatically label visually impaired users for the rest of the working team.

### 2.3.3 Cognitive mental models of 3D views

Searching over the Internet makes me think that no research has been done specifically for 3D views for design programs (e.g. graphic design programs, game development kits). However there is research work done about generic 3D interfaces, where the goal is to find objects, rather than editing and moving them. For instance, [6] shows the creation of a 3D audio user interface that allows to move around a room with some objects. Stereo headphones were used to generate the sensation of listening to audio sources that are located at a particular three-dimensional position, and to simulate room acoustics. One of the conclusions of this research paper is that 3D audio gives much more information than other assistive technologies, such as a screen reader; however, it is crucial to choose realistic 3D audio software.

Research has also been done to test how accurately blind children can model spatial structures in their minds [4]. Children first learned the shapes of objects by using a 3D audio program. Then they were asked to physically model these object (with sand, Lego's, etc). Results concluded that visually impaired users can generate pretty exact mental models of spatial information.

Another interesting paper [2] specifically addresses interaction with 3D views for visually impaired users. Firstly, they reinforced the statement that visually impaired users' hearing is more developed than in the case of sighted users. Secondly, they discovered that visually impaired users prefer auditory icons built with different notes and the same timbre (for instance, assigning note A4<sup>7</sup> of the piano to one icon, and the note G#6<sup>8</sup> of the piano to other icon and so on). It was also found that visually impaired users prefer simple timbres as opposed to the timbre of musical instruments. Finally, the study showed that visually impaired users find continuous sounds annoying. They made the additional additional remark that representing objects with the same timbre and different notes implies that users have to associate notes (frequencies) to objects; unless the user has absolute pitch<sup>9</sup>, distinguishing and remembering notes becomes unfeasible when the number of icons increase.

## 2.4 Accessible software development tools for visually impaired users

After having learned about technology and user interfaces for visually impaired users, it is possible to narrow perspective to software development programs. Currently, there are several text editors and IDE's that offer full or near-to-full accessibility for visually impaired users. Some examples are listed as follows:

1. Emacspeak: this editor is actually the Emacs text editor with an additional accessibility layer. Emacs is a cross platform, powerful and highly maintained text editor. The Emacspeak project takes advantage of both the features of Emacs and a painstaking research done on accessibility [104]. Despite of having lots of features, Emacs has a steep learning curve.

---

<sup>7</sup>This is a notation for the fourth note *La* of the piano

<sup>8</sup>This is a notation for the fourth note *Sol sharp* of the piano, which is one key above the fourth *Sol*.

<sup>9</sup>Absolute pitch is a rare condition by which a person can memorize the exact frequency of notes and daily sounds.

2. Microsoft Visual Studio: this IDE is widely used on the industry. Its main platform is Windows, but support for MacOS and Linux is on the way of being fully available. Visual Studio benefits from the NVDA addon that enables accessibility via screen readers and braille displays [63].
3. NotePad++: while having little functionality in comparison with Visual Studio or Emacs, this text editor for Windows is ideal for quickly editing text or code, and it contains a NVDA addon that makes it accessible [87].
4. Eclipse: this IDE is popular among Java developers and it can be used on either Windows, MacOS or Linux. It allows accessing information from screen readers, changing colors, magnifying text, and navigate menus using voice recognition software [45].

## 2.5 Game development tools for visually impaired users

Generic software development tools have been described as they are used by visually impaired users to develop games. In any case, these users could ideally have access to game engines (also called game development kits) that require writing less code and prevent simple mistakes. For this reason, tools specially designed to create videogames are described in this section.

Unfortunately, there are few tools of this kind. Experience gained by talking to target users across these months suggest that most of them create video games from pure code. Some others create their own game engines. This has clear drawbacks:

- Software is duplicated by many developers who want to achieve the same goal (as opposed to code reuse)
- Software is maintained by one person, and in a best-case scenario, by a small group of people
- Software cannot complexity cannot scale
- Errors are difficult to find and solve
- Most of this software finally gets abandoned

It must be noted, however, that some tools have been released, as shown below:

- Quorum [92]: this is a programming language that has been designed for blind children. It is interesting because its design has been based on a Human-Computer Interaction study, becoming *the world's first evidence-oriented programming language*, as stated on its webpage. One of the features of Quorum is that it contains libraries for game development. While it is possible for visually impaired users to create video games with Quorum, this software poses several issues. The most important of them is that no commercial games are being developed with this language; instead, game development kits with graphical user interfaces are used. This will not allow visually impaired users to develop games as part of professional game studios. Also, creating a whole new language with a reduced number of developers means that a smaller effort will have been put

on each component, including game and graphics optimization; if the goal was to make development accessible, the best option is to focus on making user interaction accessible, and plug that accessibility layer to an already implemented game engine.

- BGT (Blastbay Game Toolkit) [102]: this is a game development kit. It was originally created by Blastbay Studios, a game development company, to create their own audio games. After some time they started selling the game engine. Finally they abandoned BGT and released its source code for free. BGT is an interpreter<sup>10</sup> that reads source code and translates it into low level instructions for the computer. As long as the code is being edited with an accessible text editor (such as the ones described in 2.4), the task of creating a video game is feasible for visually impaired users. However, BGT is currently maintained by one person, it is not used in the game industry, and using it is just slightly more usable than programming a video game from scratch.
- Unity Accessibility Plugin [80]: this is not a full game development kit, but an extension that makes the Unity3D game development kit accessible. It is being maintained by Michelle K. Martin, a former software developer for CRYTEK (the company that owns Cry Engine<sup>11</sup>). She made a great decision trying to make Unity accessible, as most game developers (either professionals or amateurs) extensively use Unity, so that visually impaired users are closer to find a gap in the industry. Also, the code for this plugin is published for free with the MIT license, that gives developers freedom to contribute without restrictions. In fact this project is so well thought out that the mission of this thesis would be futile if it wasn't for some details. Firstly, this plugin is only maintained by two contributors as seen in [80]; even if M.K. Martin is a skilled developer, software is difficult to scale when workforce is so scarce. Secondly, it is clear that this plugin that works separately from (and depending on any development changes in) the Unity game engine. Although the plugin can be downloaded from the official Unity Store, installing it require a strange use of the program options, which reveals the fact that Unity developers are not considering M.K. Martin's ideas, irrespective of all her effort. Because of this, it may be possible, at any moment, that there is no way to connect Unity with this plugin; also, the installation process is too difficult for users, specially for inexperienced ones. Thirdly, this project is far away from being compatible with all major operating systems, partly due to the fact that only two developers are maintaining this project in their spare time. This project is definitely the closest visually impaired users currently are from being able to develop games in a professional context, but the way Unity has been built and licensed makes it a bad solution by now, and it is definitely not fully usable.

---

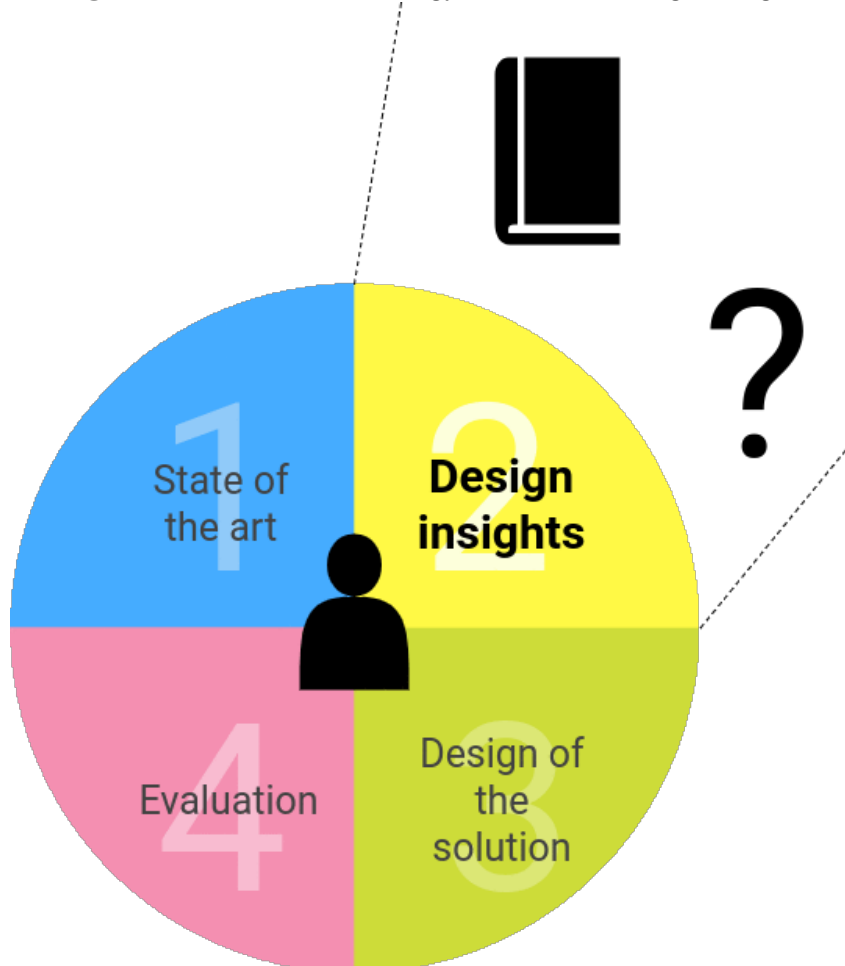
<sup>10</sup>An interpreter reads code written in a particular programming language and transforms it into basic instructions that the computer can understand. [76]

<sup>11</sup>Both Unity3D and Cry Engine are game two of the three most famous game development kits currently available. They will be further described in the following sections of this document.

### 3 Design insights

Figure 3 makes a reference to the content that is going to be discussed in this section. It symbolizes that conclusions from the state of the art and findings from the exploratory study have been considered in this section.

Figure 3: UCD Methodology. Phase 2: Design insights



#### 3.1 Exploratory study

There are several techniques for gathering information about users: questionnaires, focus groups or interviews are examples of commonly used techniques. The exploratory study, as its name indicates, tries to *explore* a concept with user information [32]. Several users are asked open questions in an interview. Then the researcher has to look for similarities between answers of all users. These similarities are formalized and yield information that can be used to better understand the problem and a possible solution for that problem. This formalized knowledge is usually known as *findings*; the term highlights the fact that this technique allows to discover complex information about user experiences.

Exploratory studies require more time and skills from the researcher, compared to other user information gathering techniques. Conversations are extensive and may span several days or weeks if users live in distant parts of the globe. This means that the

number of interviewed users will always be more reduced, specially when the interview is unstructured or semi-structured. Nevertheless, the information gathered from the study is rich and full of details that would otherwise have not been captured. As it has been mentioned before, this project is highly complex in terms of Human-Computer Interaction, because requires considering more interfaces than the classic Graphical User Interface (GUI). Every single detail from users is valuable in this case, even memories and experiences that may go beyond the scope of the interview. For this reason, an exploratory study has been used to gather knowledge from the users.

Users were first given a short description about this research project. They were told that alternative input modes are being considered: auditory icons, and hand / body gestures recorded with a webcam or special sensors. They were asked short questions regarding age, profession, etc. Finally, they were asked two open questions where they were encouraged to explain for as long as they wanted.

Short questions were the following:

1. what is your age
2. what visual problem do you have
3. how many years have you been developing software (in case you do)
4. how many years have you been developing video games (in case you do)
5. do you study or work?
6. do you develop software as part of your job or as a hobby (in case you do)?
7. do you develop video games as part of your job or as a hobby (in case you do)?

Open questions are the following:

1. How do you normally interact with your computer?
2. Would you feel comfortable with the interaction modes I have mentioned, together with keyboard interaction?

## 3.2 Participants

### 3.2.1 Participant profile

Finding the adequate users is a key point of the exploratory study. This research tries to analyze a specific type of user, so it is necessary to find users which match a particular profile. More specifically, exploratory study participants required to have:

- some degree of visual impairment
- experience developing games or
- experience developing software or using computers

*Experience* here accepts many degrees. Both users with basic knowledge and advanced capabilities are accepted. Also, it was irrelevant to accept students, hobbyists or professionals, as long as they fulfilled the required criteria.



### 3.2.2 Searching for participants

Searching for users was a long process. The number of users that match the profile described in section 3.2.1 is low, so it takes time to find these users.

One of the first places I went to search for users was Pinto Association for People with Disabilities<sup>12</sup> [43], as it belongs to my home town. This institute is an early assistance center, so users are too young to develop software or videogames (3 to 10 years). Also, most users suffered from cognitive disability; there was only one children with visual impairment, but s/he was particularly young.

After talking to volunteers and the center secretary, I was advised to contact two of the Madrid offices of the National Organization of Spanish Blind People (ONCE<sup>13</sup>). This organization is probably the biggest Spanish institution that defends the rights of the visually impaired. It took several hours to get to the right office, as each office addresses different topics. I initially visited the General Office and from there I was redirected to the ONCE Madrid office. In particular, I talked to the Social Affairs department, who told me to meet Francisco Monzón, in charge of the Culture section. I explained my project to him and he gave me advice about accessibility in general; he also praised my idea and thought that it could greatly help visually impaired users. Finally, he promised to search for users he knew from ONCE computer technology courses. Some weeks later, I managed to contact with two people found by Mr Monzón.

One of the users featured in this exploratory study was found from a web article. After finding his contact information, we stayed in touch for some days. S/he advised me to search in the Program-L mail list [74], as well as in the Audio Games Forums [68]. The first resource is a mail list for visually impaired developers and computer enthusiasts. The second resource is a set of forums about audio games; there are several topics such as gaming, development, announcements, etc. Most users were found at one of these forums.

Finally, I also found users after reading online articles and papers, as well as from advice received at the aforementioned places.

### 3.2.3 Demographics

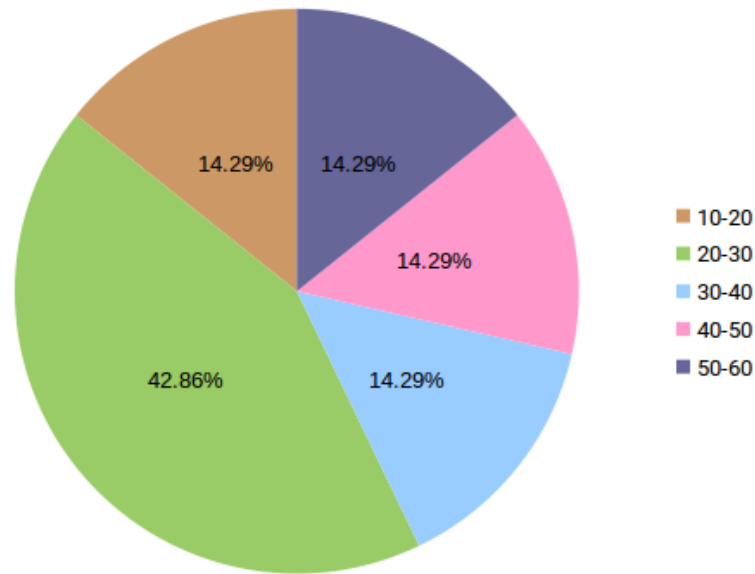
A total of 7 users were interviewed. They have ages between 17 and 52 years, with most of them being younger than 30 years. Average age is also around 30 years; no users younger than 10 or older than 60 were found. This age distribution makes sense, since visually impaired users have been using computers for the last generations. Figure 4 shows a distribution of age across these users:

---

<sup>12</sup>Translated from the official Spanish name, *Asociación de Minusválidos Pinto, Centro de Atención Temprana*

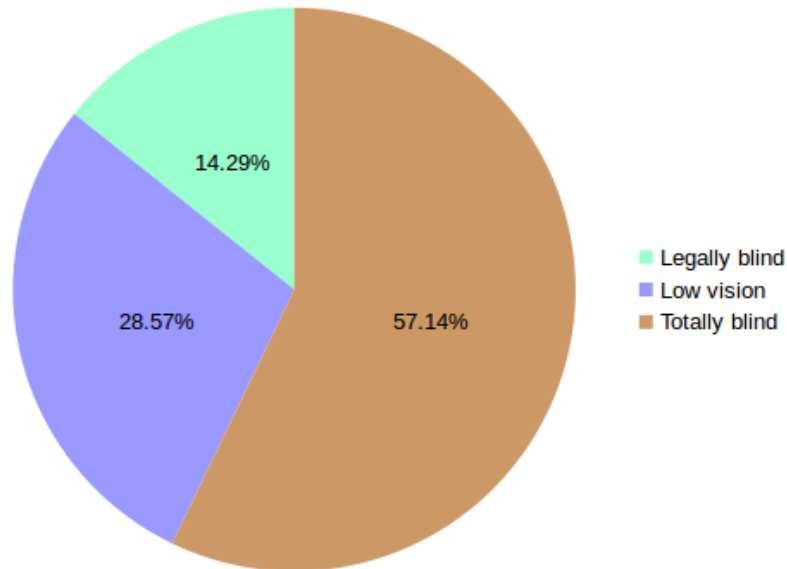
<sup>13</sup>Acronym. Comes from the Spanish term *Organización Nacional de Ciegos Españoles*.

**Figure 4:** Exploratory Study. Participant age distribution



Most of these users are totally blind, as shown in figure 5. Next ones in number were users with low vision. Only one was legally blind. No partially sighted users were found. Sighted users were not considered because they rarely know how to use assistive technologies, one of the key points in this research.

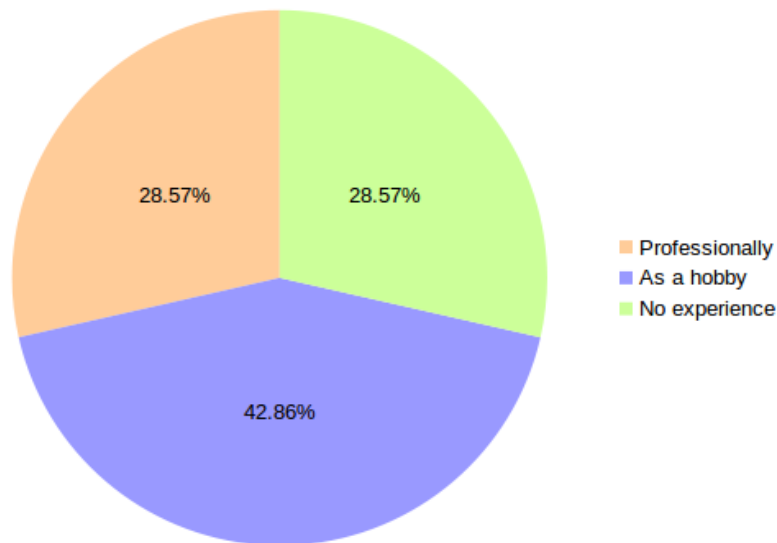
**Figure 5:** Exploratory Study. Visual impairment distribution



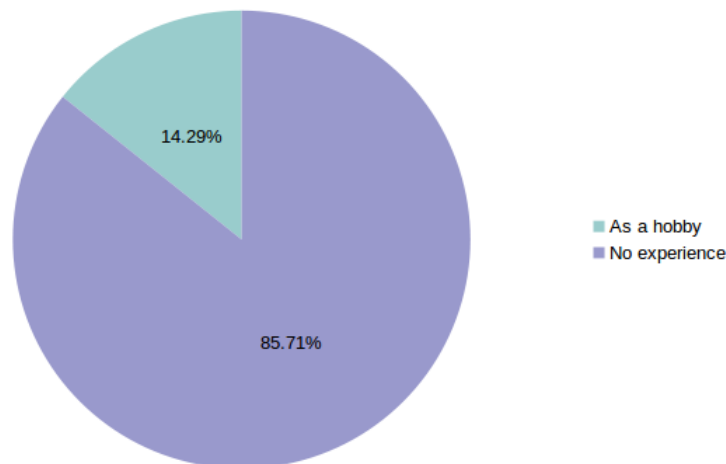
Most users had developed software, either professionally or as a hobby, for an average of 4 years and a half. However, only two users developed games, and only had 1 year of experience. It is also worth noting that these two participants who developed games only did as a hobby. One of them told me outside of the interview that s/he had designed an audio game and wanted to make it a commercial success, although s/he doubted that was possible. This data shows that there are few visually impaired people able to develop a game, and that in any case they could be easily hired in a game studio. This information

is summarized in figures 6 and 7

**Figure 6:** Exploratory Study. Software development experience distribution



**Figure 7:** Exploratory Study. Game development experience distribution



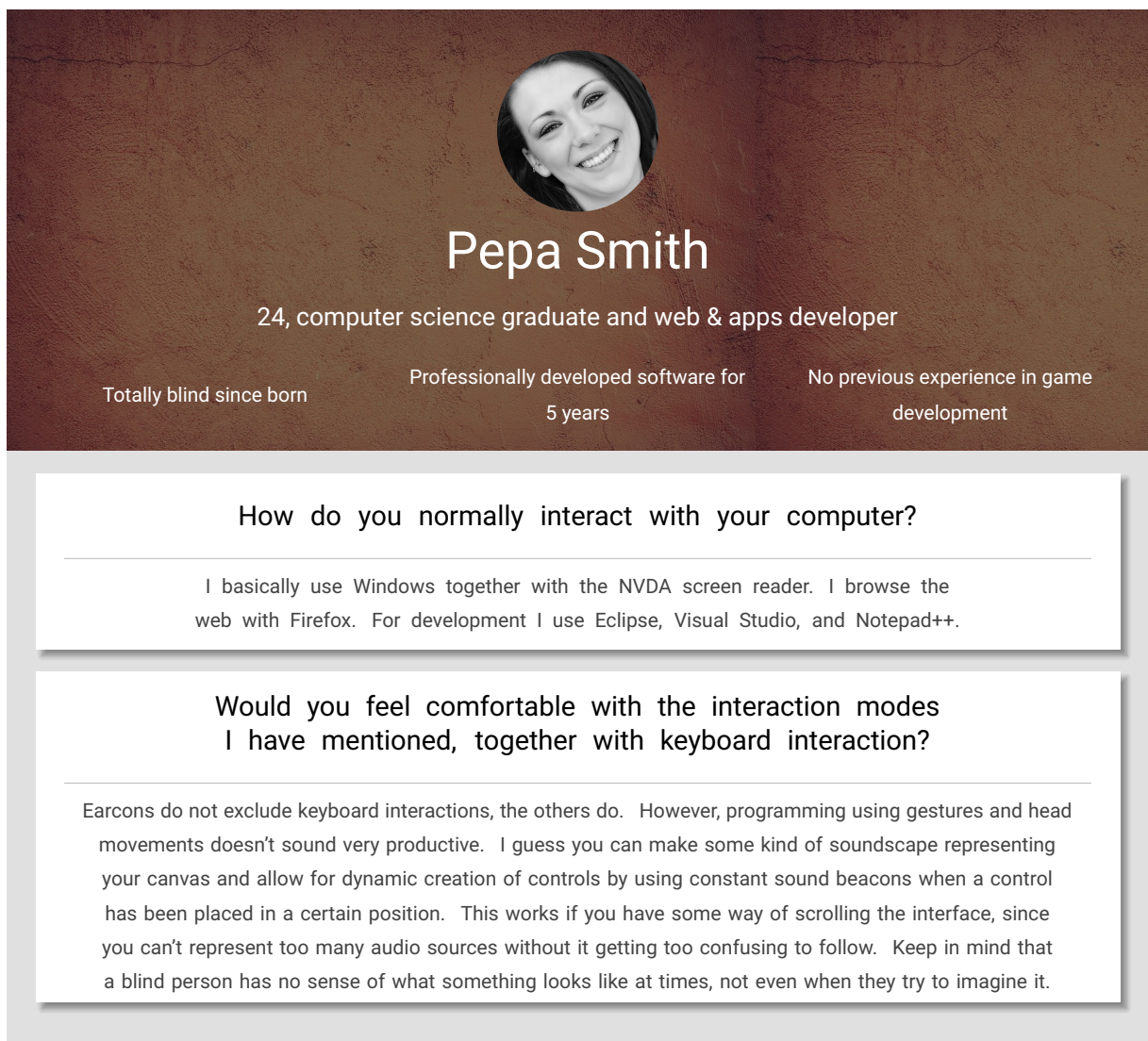
### 3.3 Interviews

A summary of the interviews is provided below in the form of cards. This way of representing the outcome of the interviews is slightly different from a simple dialogue. It has been based on the concept of *personas*<sup>14</sup>, which is used in the UCD methodology to collect information about user profiles. A persona is different from a person or the archetype of a person [27]. Personas are highlights of aspects of a person that are relevant for the problem being discussed, and depend on the context of the task being done. In this case, we present *real personas* as the described users are real and they try to describe the context in which a visually impaired developer does his/her daily life tasks.

<sup>14</sup>The name is correctly written; it comes from the Latin word for *person*.

The data shown in the following cards is real (except for the names and pictures<sup>15</sup>, which have been invented to preserve participants' privacy). However, it is correct to say that they are inspired by personas.

Specifically, the interviews have a semi-structured format. They follow this format because somehow the two questions were asked during the conversation, but in the end each interview has touched upon several topics and personal experiences. Contact with users has been mostly by email, some of them by Twitter private messages, and some of them by telephone. As a reminder, any resemblance in name, surname and picture to actual persons, living or dead, is purely coincidental.



The image shows a user profile card for Pepa Smith. At the top, there is a circular profile picture of a smiling woman with dark hair. Below the picture, the name "Pepa Smith" is written in a large, white, sans-serif font. Underneath the name, it says "24, computer science graduate and web & apps developer". At the bottom of the profile section, there are three pieces of information: "Totally blind since born", "Professionally developed software for 5 years", and "No previous experience in game development". Below the profile section, there are two white boxes with black text. The first box contains the question "How do you normally interact with your computer?" and the answer "I basically use Windows together with the NVDA screen reader. I browse the web with Firefox. For development I use Eclipse, Visual Studio, and Notepad++." The second box contains the question "Would you feel comfortable with the interaction modes I have mentioned, together with keyboard interaction?" and the answer "Earcons do not exclude keyboard interactions, the others do. However, programming using gestures and head movements doesn't sound very productive. I guess you can make some kind of soundscape representing your canvas and allow for dynamic creation of controls by using constant sound beacons when a control has been placed in a certain position. This works if you have some way of scrolling the interface, since you can't represent too many audio sources without it getting too confusing to follow. Keep in mind that a blind person has no sense of what something looks like at times, not even when they try to imagine it."

**Pepa Smith**  
24, computer science graduate and web & apps developer

Totally blind since born      Professionally developed software for 5 years      No previous experience in game development

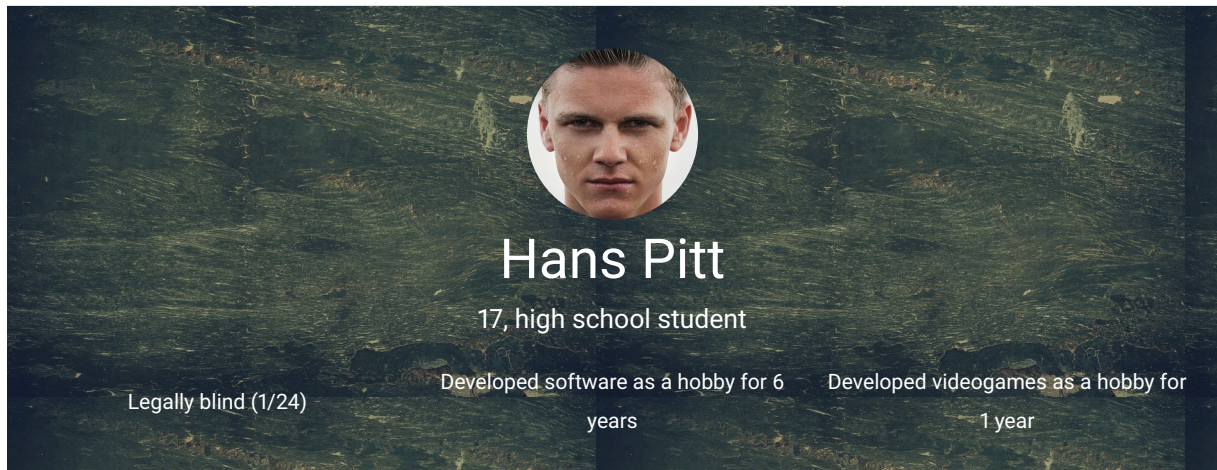
**How do you normally interact with your computer?**

I basically use Windows together with the NVDA screen reader. I browse the web with Firefox. For development I use Eclipse, Visual Studio, and Notepad++.

**Would you feel comfortable with the interaction modes I have mentioned, together with keyboard interaction?**

Earcons do not exclude keyboard interactions, the others do. However, programming using gestures and head movements doesn't sound very productive. I guess you can make some kind of soundscape representing your canvas and allow for dynamic creation of controls by using constant sound beacons when a control has been placed in a certain position. This works if you have some way of scrolling the interface, since you can't represent too many audio sources without it getting too confusing to follow. Keep in mind that a blind person has no sense of what something looks like at times, not even when they try to imagine it.

<sup>15</sup>The second user's profile picture is a picture of myself. The rest of pictures have been legally downloaded from Pixabay [73], a webpage to download free, non-copyrighted pictures



A profile card for Hans Pitt. At the top center is a circular portrait of a young man with short, light-colored hair. Below the portrait, the name "Hans Pitt" is written in a large, white, sans-serif font. Underneath the name, "17, high school student" is written in a smaller, white, sans-serif font. At the bottom of the card, there are three white text elements: "Legally blind (1/24)" on the left, "Developed software as a hobby for 6 years" in the center, and "Developed videogames as a hobby for 1 year" on the right. The background of the card is a dark, textured image of a forest floor.

### How do you normally interact with your computer?

1.1: General use: I use screen reading technology to use and control my computer. I, for the most part, use keyboard only. 1.2: software development. To find IDEs (Integrated development environments) that are fully accessible with screen reading technology is often hard without some kind of special hacky method. So for the biggest part I use Eclipse (because it's fully accessible) and Notepad++ with a command line. Site note: When you're blind and developing software and / or video games you will need to use the command line / terminal at some point in time.

### Would you feel comfortable with the interaction modes I have mentioned, together with keyboard interaction?

I don't really know how the 2nd option would work. I guess it could work but I'm not sure. As for the first option, I think it could be useful if done right.



Erik Jiménez

42, computer technician

Low vision

Developed software as a hobby for 10 years

No previous experience in game development (s/he plays games)

### How do you normally interact with your computer?

I normally use keyboard and mouse because I only have residual vision, instead of total blindness. If my vision was slightly worse I would only use keyboard; as a side note, JAWS only works via keyboard, partly because it is more efficient than mouse, partly because at least all visually impaired users can work with a screen readers. I also use Windows Magnifier as it is the lightest tool in terms of CPU and RAM usage. Finally I use a custom magnifier made by ONCE, but it has more bugs and it takes more computer resources than Windows Magnifier. There are moments where there is too much or too little ambient light and I cannot clearly see the screen. In these situations I use keyboard exclusively. I also do this when the computer I am working with does not have any magnification software installed. A last comment about screen readers...when you use the screen reader you have much more functionality compared to the screen magnifier. It is not unusual for me to interact with keyboard and screen reader; it is the most powerful combination for me.

### Would you feel comfortable with the interaction modes I have mentioned, together with keyboard interaction?

I think earcons could be useful, although users should be able to switch them on and off whenever they want, either one by one or globally. It should also be possible to change earcon volume, again for each one or for all of them. Being able to *discriminate* earcons by configuring them is really important; if not they could become annoying. I personally don't like the idea of gestures; I would prefer manipulating 3D objects with keyboard shortcuts, although I would include it as other users may find it useful. By the way, if you used gestures with webcams I would find troubles testing your solution because I don't have a webcam. Maybe you could assume that other users with visual impairment will neither have webcam, so it would be ideal if you could control the program by just using keyboard.



## Natalie Duemilanove

52, photographer and computer technician

Low vision

No previous experience in software  
development

No previous experience in game  
development

### How do you normally interact with your computer?

Although I am not a software developer, I do computer maintenance work. As a photographer, I use image editing software, which I think is similar to the software you are talking about. I use the screen reader and similar tools, but I can also directly look at the screen (only that it is not super-precise this way).

### Would you feel comfortable with the interaction modes I have mentioned, together with keyboard interaction?

In my particular case, and as a user with low vision, earcons would only be useful for specific user experiences, like when moving to places where there is no visual signals. Gestures are different; probably you and I would immediately associate them with people with reduced mobility, but I am sure that we won't be looking at screens in the future as we do nowadays. We will probably communicate in different ways with interfaces. Speech recognition is an option; what about gestures? To my mind they would be perfect for specific actions. In fact we can see this kind of commands nowadays, for instance on Android. Samsung Galaxy S6 and newer models offers gesture commands, even if people don't use them a lot by now...



## Hassan Pacheco

26, linguistics student and freelance developer

Low vision on 1 eye till age 10; totally blind since then

Professionally developed software for 9 years

Developed games as a hobby for 1 year


### How do you normally interact with your computer?

I'm using screen readers to interact with computers (NVDA and Jaws on Windows, and Orca on Linux). I've learned scripting for NVDA and Jaws in order to make non-accessible programs accessible for my needs. I do not find useful those solutions which require voice commands or any kind of sensors.

### Would you feel comfortable with the interaction modes I have mentioned, together with keyboard interaction?

I don't think that webcam or other kinds of sensors will be useful in fixing accessibility issues. Accessibility should mean doing a job as conventionally as possible, or in compliance with the industry standards. Unfortunately, while the blind have difficulty in explaining screen readers to their boss sometimes, such a specialized workstation will not help the blind find a job, I guess. Instead I advice sticking to keyboard commands that would simulate mouse motions, and 3D audio to indicate objects' position or movements as well as color tones and saturations on real time. One such example can be seend at <https://github.com/nvaccess/audioScreen>. I experienced one such situation when I applied to a Java developer position. They wanted me to code with IntelliJ, but it was not so accessible those years, and I was using Eclipse instead. Even such a slight IDE preference was an issue to be addressed. If you are a freelancer or self-employed it does not pose a problem unless you collaborate with a team remotely. However, you need to be able to work with industry standards if you want to work in an office environment.





**Susana Ellis**  
34, game & software tester and sound designer

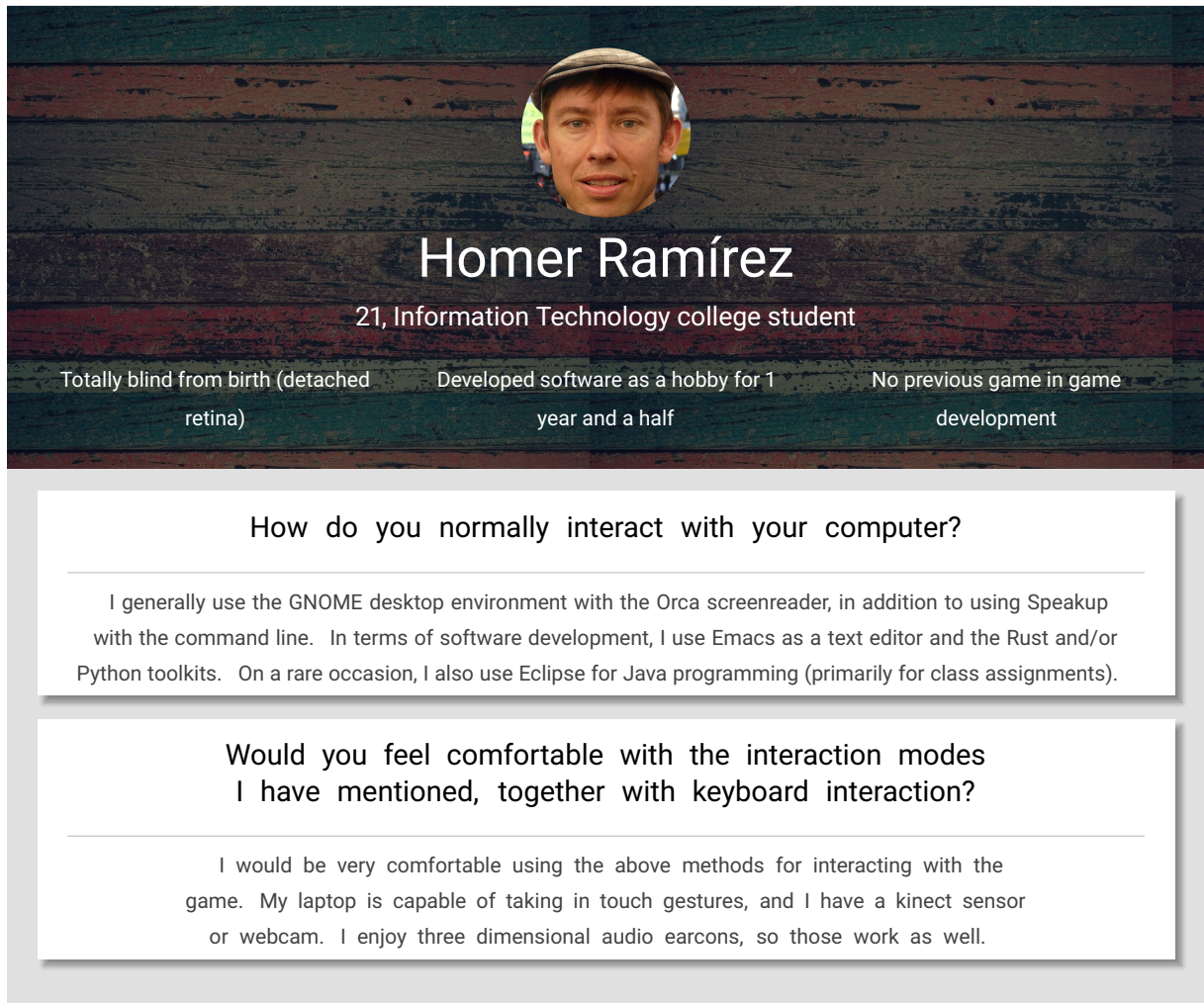
Totally blind since birth      No previous experience in software development      No previous experience in game development

### How do you normally interact with your computer?

I interact with my computer with a keyboard, sometimes I use a mouse but mostly it's a combination of keyboard and a screen reader called NVDA. I play games in my computer, with keyboard only. I have not hit the tactile or console scene just yet, but touch may happen eventually, who knows.

### Would you feel comfortable with the interaction modes I have mentioned, together with keyboard interaction?

I would be happy with any mode. Audio cues are useful. I have also played some games which use 3D audio and the overall feeling is quite good. I don't use expensive 3D hardware or extra game pads or things like that, because space on my bench is limited and I cannot afford those pieces of equipment either.



**Homer Ramírez**  
21, Information Technology college student

Totally blind from birth (detached retina)      Developed software as a hobby for 1 year and a half      No previous game in game development

**How do you normally interact with your computer?**

I generally use the GNOME desktop environment with the Orca screenreader, in addition to using Speakup with the command line. In terms of software development, I use Emacs as a text editor and the Rust and/or Python toolkits. On a rare occasion, I also use Eclipse for Java programming (primarily for class assignments).

**Would you feel comfortable with the interaction modes I have mentioned, together with keyboard interaction?**

I would be very comfortable using the above methods for interacting with the game. My laptop is capable of taking in touch gestures, and I have a kinect sensor or webcam. I enjoy three dimensional audio earcons, so those work as well.

### 3.4 Design insights

Together, the state of the art and the exploratory study have produced a remarkable amount of information. This information, known as design insights, determine who our *target user* is, what are the user's *needs*, and what is the *problem* to address.

One of the findings gathered from the exploratory study is that participants have used the same visual impairment classification than the one used in section 2.1.1. This indicates that this classification can be considered as a real instrument for users, as it describes the way in which they can or cannot fulfill a specific task.

In terms of input interaction, most participants agreed in their preference for keyboard. No extra hardware should be used, according to majority's preferences. According to Hassan, using special hardware could automatically label the user as *different* or *inferior* because they are using special hardware. Pepa says that switching between special equipment and the keyboard would not be very productive, given her experience as a software developer who spends many hours typing code. Only Susana and Natalie view gestures as useful, but they do not make any reference to how *flashy* would it be to use them. Also, [7] proposes using haptic technology to enhance input and output interaction, but this research is focused on immersive games, where it is not unusual to see increasingly *flashy* gadgets.

All participants have similar opinions regarding output mode. Most of them use auditory icons with other applications and audio games they play, so they are happy with them. However, Erik pointed out that they should be easily configurable. This includes changing the sound sample associated to a particular icon, changing its volume, or turning one or all of them off, in case the user is annoyed by the sound. This coincides with [35], where one of the conclusions of the study was that auditory icons should be easy to distinguish one from another; if users can change auditory icons as they want, they will easily distinguish them. Finally, and based on [14], it seems that output information could be either sound samples, continuous sounds (like a sine wave) or verbal descriptions that are sent to the screen reader.

Pepa also made an interesting comment regarding 3D views, and it should be taken into consideration given its relevance to the research. She said that users can get confused when lots of acoustic information is presented at once. Also, she pointed out that visually impaired users sometimes may not have an idea of what an object looks like, so it is better to reduce the number of simultaneous objects that are being shown in the 3D view. Although [4] states that visually impaired users have very precise mental models of objects in space, it is also true that the rest of research papers and games considered in the State of the Art highlight that 3D spatial information should be as much simplified as possible.

Erik talked about how she can switch between different input and output modes. This is particularly true for users with smaller degrees of visual impairment. However, all users agree on the fact that keyboard plus screen reader is the most productive interaction *combo*, and it can be used by people with any degree of visual impairment; this is crucial for making the final product *inclusive*.

Having analyzed products like Visual Studio highlight a current trend in the creation of accessible software development tools. As pointed out in [30], it is preferable to not reinvent the wheel and to add accessibility as an additional module to existing and maintained solutions, ensuring all users can work using the same up-to-date tools. This is the approach used in almost all big software projects. In fact, among the products shown in section 2.5, the most promising solution is the Unity accessibility plugin, and part of its success comes from the developer obeying the principle of software reuse. Certainly, the solution proposed in this document will also comply with this software reuse principle. For this reason, the proposed design solution will be focused on an accessibility layer, which will be attached to an already existing game development kit.

Thanks to the valuable information gained with the design insights, it is now possible to define the following design elements:

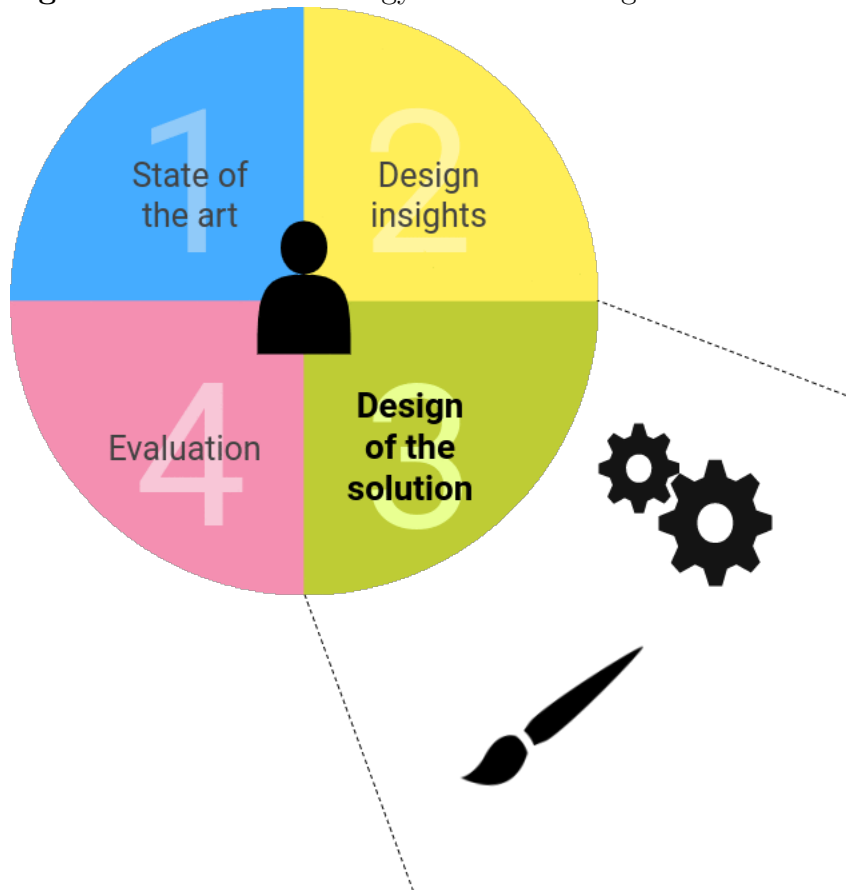
- Target user: s/he is a visually impaired user. This includes partially sighted users, users with low vision, legally blind users, and totally blind users. The user knows how to perform daily tasks in a computer at least, but it is common that s/he also has some experience with software development. In terms of age, the target user is generally young: either a grown teenager (15 to 25 years approximately) or an adult (25 to 55 years approximately).

- User needs: the user cannot use a graphical user interface without additional software, that is, s/he cannot depend on visual output only. The user may see the screen contents blurred, have difficulties with particular colors, or may just not see anything at all. In order to consider the worst-case scenario, we can say that the user normally cannot make use of a screen and graphical peripherals (mouse, touch screen, joystick, etc). The user therefore prefers to navigate the computer with the keyboard, and prefers to receive feedback from the computer with a screen reader. S/he may also use auditory icons, provided s/he could change their volume or deactivate them at any moment. The user's hearing gets saturated if there are many simultaneous sounds being played, so s/he prefers to have a partial view of the program and the scroll it. The user may also need a screen magnifier or a color inversion tools that already come with the operating system. In order to correctly perceive game objects from the 3D view, s/he needs stereo headphones.
- Problem: the target user wants to develop a computer game in a game development studio. Additionally, s/he may want to develop a game as a side project. However, available game development software used at companies do not let the user perform this task without fulfilling his/her needs and preferences.

## 4 Design of the solution

Figure 8 makes a reference to the content that is going to be discussed in this section. It symbolizes that the outcome of this section is both a design specification and a set of implementation details.

**Figure 8:** UCD Methodology. Phase 3: Design of the solution



Design insights have been crucial to have an overview of what features should the program have. Taking these high-level details into consideration, it is now easy to come up with a design specification that *captures* these insights. This section first describes the software taken as a reference for the design of the solution. Then the design specification is explained in detail. After this, implementation suggestions for the design are provided, considering the software taken as a reference. Finally, a prototype is designed for future user evaluation of the design.

### 4.1 Chosen technology

#### 4.1.1 Available technologies comparison

Several programs have been considered to have flexibility when choosing the technology for this design. All of them can be considered game development kits, that is, programs that contain all necessary tools to create a game (design tools, programming tools, animation tools, etc.). Game development kits are also known for automating several game development tasks, which make game development easier, less error-prone and faster.

There are thousands of game development kits. Only a few have been selected, either because of their popularity in the game industry, or because of special features that make them interesting for this research project. They are described below:

- Unity 3D [94]: Unity is by far the most intuitive game development kit in the market, and one of the three most used, together with Unreal Engine and Cry Engine (see below). It has been widely adopted by companies and hobbyists because of its ease of use and the number of platforms games can be exported to.
- Unreal Engine [101]: this game development kit created by Epic Games Inc. is arguably the one with most powerful graphical capabilities. Most high budget videogames are developed with Unreal Engine. It is less usable than Unity, but it has more functionality and it has been further optimized, so it is best suited for expert game developers.
- Godot Engine [79]: this game development kit was originally created by two game developers to release their own games; the source code was released some years after they founded their game studio, and now it is the most complete free and open source game development kit available. They have recently received 20,000\$ by the Mozilla Awards [47], that encourage the use of free and open source software.
- Cry Engine [70]: graphic capabilities of this software, created by Crytek GmbH, are really close to those of Unreal Engine. It has been extensively used in games that have later turned into commercial success, like *Crysis 3* [23].
- Amazon Lumberyard [66]: after Amazon signed contract with Crytek GmbH, they created a version of Cry Engine with extended support for Amazon products. It is still in development and several years will pass until this software is used by game studios, but it looks promising.
- GREP [33], [41]: this game creation environment targets educators without previous technical experience who want to build educational games featuring virtual reality. It is based on a Unity game, so that educators can create the game in virtual reality while they play it.

Table 1 shows most important features for each one of the game development kits considered above:

Criteria	Unity	Unreal Engine	Godot Engine	Cry Engine	Amazon Lumberyard	GREP
FOSS			X			X
Source code available	?	X	X	X	X	
Windows support	X	X	X	X	X	X
OSX support	X	X	X			X
Linux support			X			
Other OS support			X			
Number of export platforms	31	9	9+	9	5	3
Editor implemented in	C, C++	C++	C, C++, Python	C++, Lua, C#	C++, Lua, C#	Unity, XML
Games implemented in	C#, Boo	C++	C++, GDScript	C++, Lua, C#	C++, Lua, C#	XML, graphic, Unity project
Documentation	9/10	8/10	7/10	4/10	0/10	2/10
Maturity (years)	18	11	2	14	0	6
Asset store	X	X	X	X		
Number of developers	700	938	337	600	?	1

**Table 1:** Game development kits comparison

Each game development kit has been assessed by several criteria. There are some things that should be clarified about the table. *FOSS* is an acronym for *Free and Open Source Software*; this criteria means whether if a particular software is free and open source software or not. *Support* means if a particular software can be used in a particular operating system. *Export platform* refers to the operating systems and devices where developed games can be played once they are published. *Documentation* is a subjective criterion that gives a grade out of 10 to the documentation of a particular software; it has been based on the documentation size, tutorials and lessons and ease of use of the programming reference. GDScript is a programming language based on Python and specifically used for Godot Engine. The *Number of developers* criterion is a rough estimation; it is based on the number of contributors at the online code repositories where each software is being stored. *OS* is an acronym for *Operating System*. *9+* in

Godot Engine's number of export platforms means that this number is actually higher than 9, as custom platform builds can be programmed.

Godot Engine has been chosen to consider the design of the solution for the following reasons:

- It is the only game development kit that is free and open source and has its source code available online. This makes future incorporation of accessibility features highly probable, compared to privative software<sup>16</sup>.
- It is the game development kit that can be used in the widest range of operating systems, with the special addition of Linux (which is not available for any other game development kit).
- It has been implemented in known and up-to-date programming languages (this is shared by other game development kits, with the exception of GREP).
- Its documentation is good and the number of developers is high, taking into account that it is a very *young* game development kit. In just 1/9 of the time Unity has been active, Godot Engine has half the number of contributors Unity has got nowadays; these figures indicate that Godot Engine community is growing fast, and this translates into more reliability, more features and less errors.

#### 4.1.2 Godot Engine user interface analysis

Before generating the design specification, it is important to consider how is the chosen technology (i.e., Godot Engine) designed. This will mostly cover details about the user interface of Godot Engine, as these information is crucial for the design specification; implementation details will be considered later in section 4.3.

Godot Engine is organized between two screens:

- *Project manager* screen: this is the starting screen where saved projects can be loaded. There are two tabs: one with the projects list, and another one with projects and templates uploaded by other users.
- *Main* screen: this is the screen where most activity takes place. Main screen is similar to the *Project manager* screen. It is divided between 5 panes: a *FileSystem* pane, an *Output / Debugger / Animation* pane, a *2D / 3D* view, a *Scene* pane, a *Node* pane and an *Inspector* pane. There is an upper menu bar, but it is divided in 5 separated parts: a *Scene / Import / Tools / Export* menu, a *2D view / 3D view / Script / AssetLib* menu, a menu with playing and testing actions, a system load bar and a small menu with *Settings* and 2 indicators, one of them representing changes in the file system, and another one representing changes in the 2D/3D view.

Despite of not having been designed with accessibility in mind, most elements of Godot Engine's graphical user interface can be easily made accessible for visually impaired users. Most part of the program's windows are navigable by keyboard, which already removes an important barrier for these users. There are some exceptions, however.

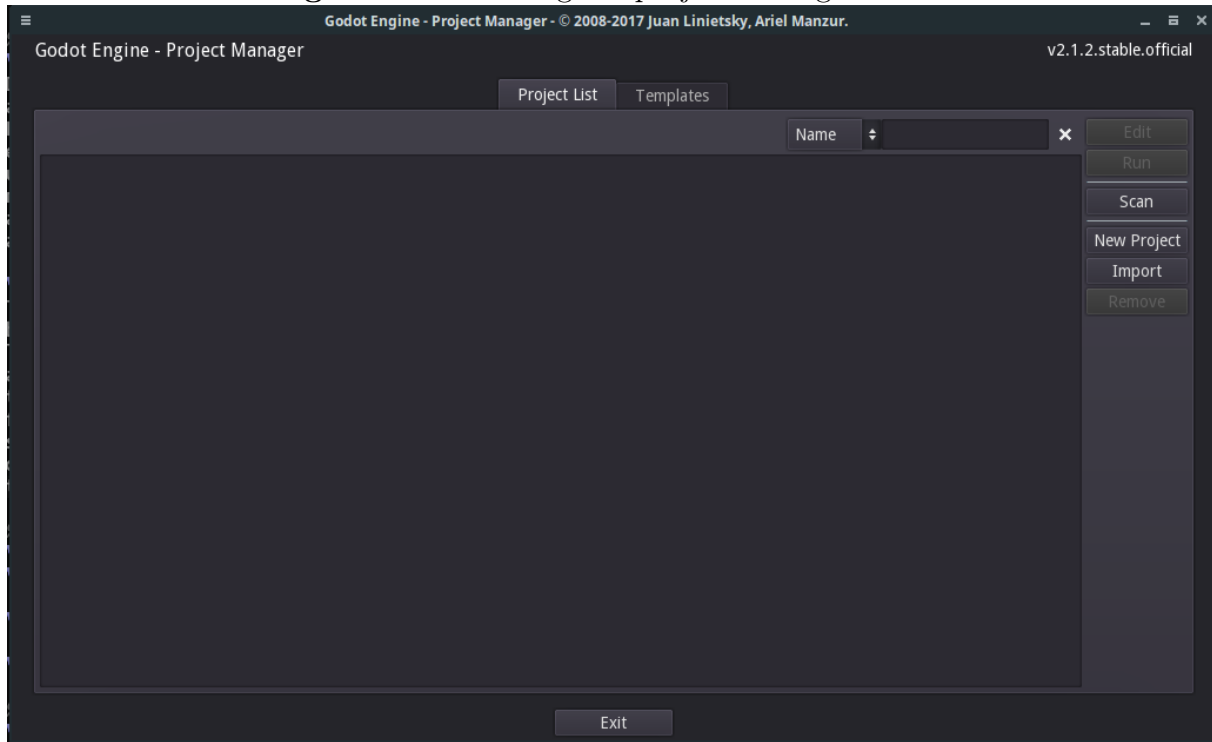
---

<sup>16</sup>Refers to software that is neither free software nor open source software.



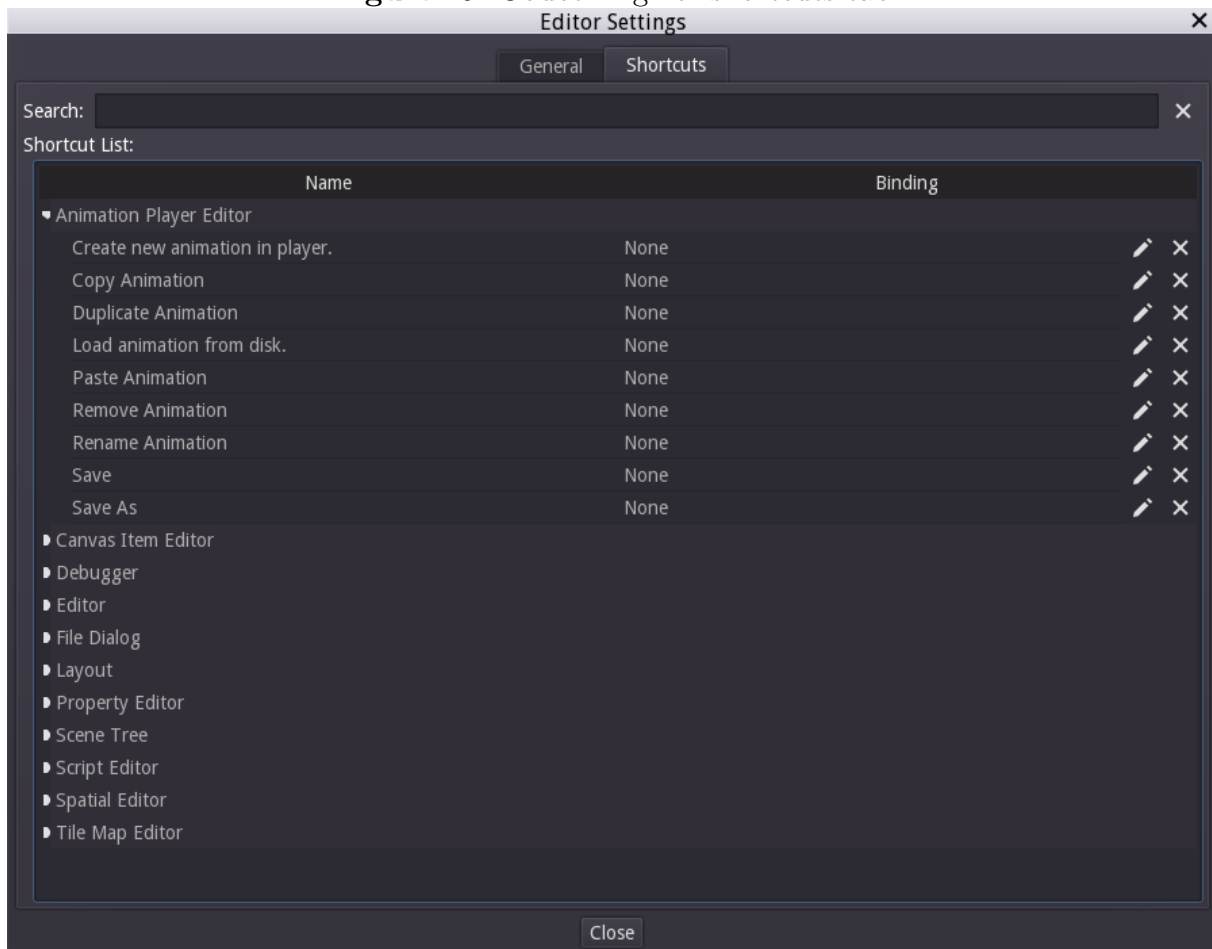
The *Project manager* screen only allows to move between the list of saved objects. The side bar with available actions for that screen, or another tab with game templates, are not accessible from the keyboard. However, most part of this screen becomes tab-navigable after clicking on the *Scan* action button located at the side bar. Almost no tooltips have been included. A screenshot of this screen is shown in figure 9.

**Figure 9:** Godot Engine: project manager screen



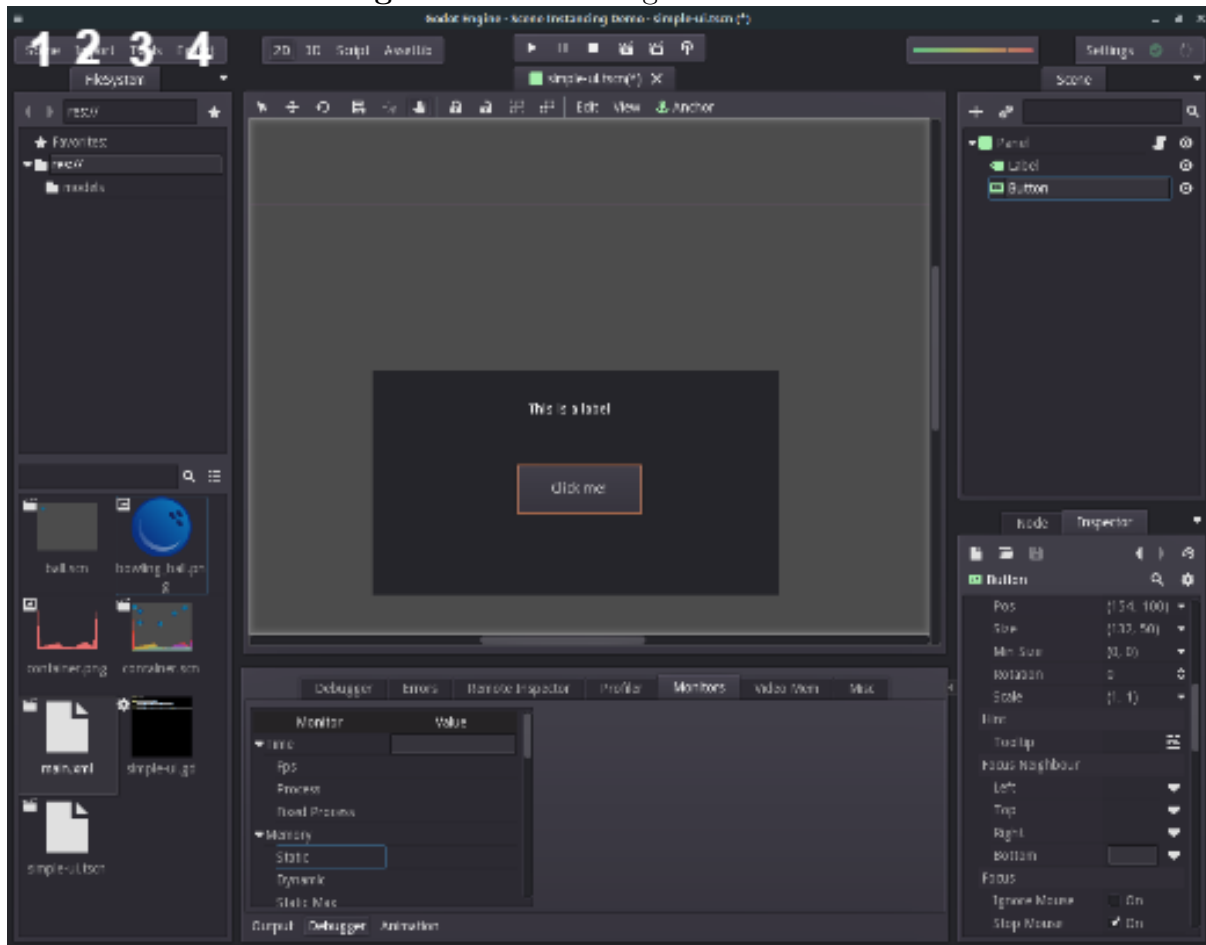
*Settings* menu is interesting because it contains a *Keyboard shortcuts* tab, where users can configure all keyboard shortcuts that can be used in the editor. This feature is extremely important for visually impaired users, who replace mouse gestures with keyboard shortcuts. It is highly probable that all these shortcuts are read to this menu from a configuration file. In this sense, adding new shortcuts for efficient keyboard navigation looks like an easy task. A screenshot of this menu is shown in figure 10.

Figure 10: Godot Engine: shortcuts tab



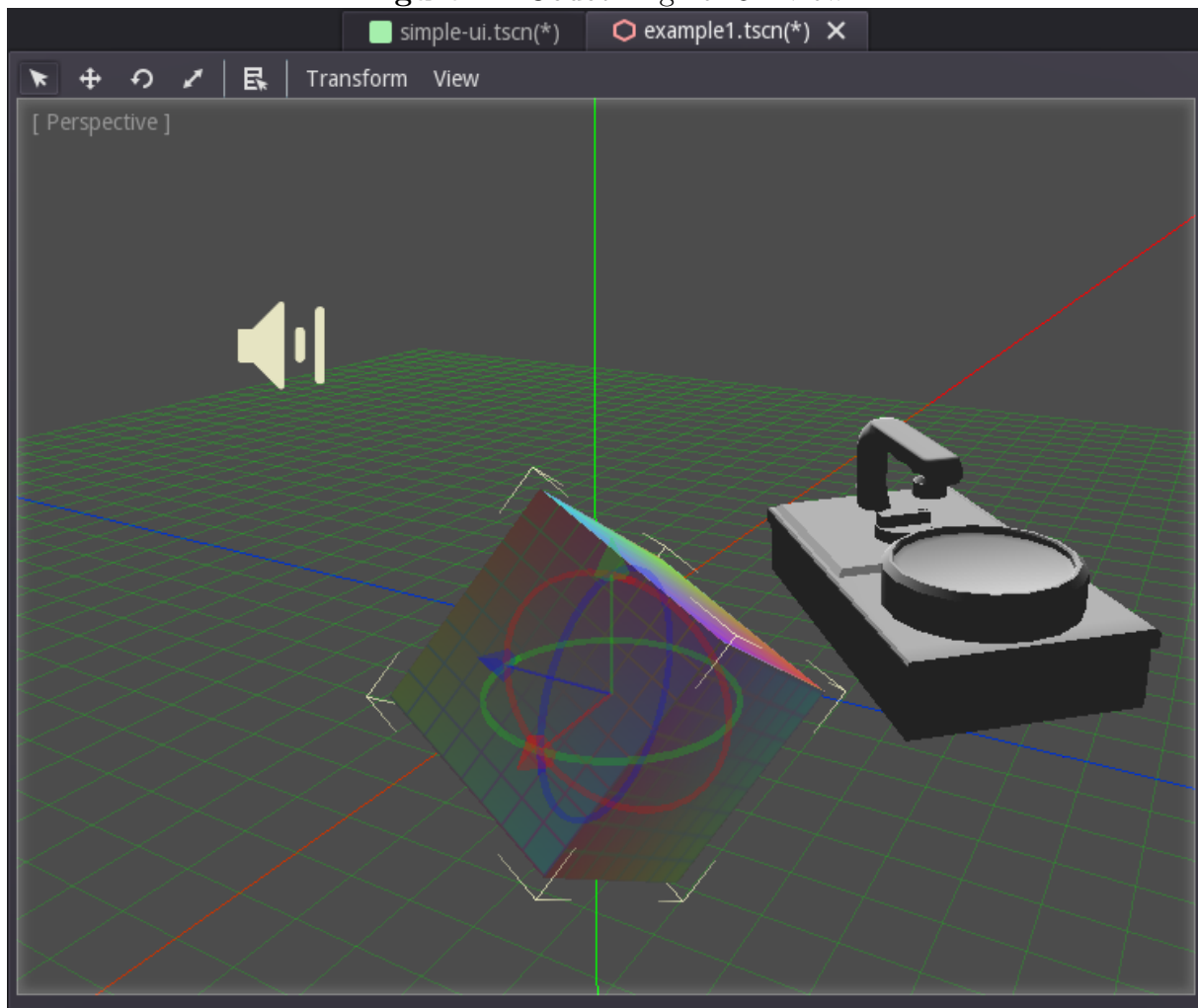
Main screen elements permit tab navigation up to some degree, but some parts cannot be accessed by the keyboard. For instance, parts 1, 2, 3 and 4 of the upper menu (from left to right and starting from 1, as indicated in figure 11) cannot be accessed by keyboard. Moving between tabs does not work, even if there are shortcuts for that (**Ctrl-Tab** and **Shift-Ctrl-Tab**, as in most programs). There are almost no tooltips. When pressing **Shift-Tab** to go to the previously visited menu element, menu focus is put on a different menu element, which is not a logical navigation. Pressing arrow keys has the same result than using **Tab** and **Shift-Tab**, which again is not logical. For some panes like the *Inspector* pane, keyboard navigation is not logical and it does not allow to access drop-down menus associated to each element. Each pane has a drop-down menu to change its position or to attach it to other panes; this feature is interesting but it cannot be accessed with the keyboard. Finally, there is no way to know if status indicators (those located at the top-right part of the upper menu) are changing or not, as they depend on vision. A screenshot of this screen is shown in figure 11.

Figure 11: Godot Engine: main screen



The 3D view shows three axis for reference and a grid that represents an horizontal plane intersecting with coordinate  $(0,0,0)$ . Tab switching works in this part of the user interface; it lets switching between different open scenes. Basically, the user can rotate view, change zoom and move in any direction by using different combinations of mouse movement and mouse clicks. The 3D view also changing properties from game objects: principally, they allow to translate, rotate and scale the object; these are properties that can also be changed from the *Inspector* pane with keyboard, although they require selecting the desired game object (either from the 3D view or from the *Scene* pane) and getting to the *Inspector* pane by using Tab and arrow keys. It is also possible to select one or several objects; this makes the *Inspector* pane consider properties of a group of objects, instead of considering them separately. As of the panoramic view a sighted user gets when looking at the 3D view, visually impaired users could receive similar information by selecting objects from the *Scene* pane and checking their properties at the *Inspector* pane one by one, but this is clearly a very inefficient way of interacting with the program. A screenshot of a 3D view with three objects (a sound source, a cube, and an imported 3D object) is shown in figure 12.

Figure 12: Godot Engine: 3D view



The aforementioned issues regarding accessibility in Godot Engine's user interface are summarized in table 2:

Problem	Place in the user interface
No full keyboard navigation	<i>Project manager</i> screen, main screen
Almost no tooltips are included	<i>Project manager</i> screen, main screen
No logical ordering of Tab navigation	Main screen
Arrow keys and Tab have the same behavior	<i>Project manager</i> screen, main screen
Information cannot be perceived without vision	Status indicators, 3D view
Multiselection is not efficient without a mouse	3D view

**Table 2:** Accessibility problems in Godot Engine

## 4.2 Design specification

Knowing in detail how does the user interface of Godot Engine work for the most part, and remembering the design insights discovered in section 3.4, the process of writing a design specification is simple, logical, and almost instantaneous.

Design specifications are normally drawn in what interaction designers call *sketches* or *mockups*. These drawings show the main elements of the design in a visual way, and they are drawn with tools such as Axure [53] or Adobe Illustrator [69]. However, this design workflow is not practical for this solution, because the graphical user interface is already built, and what has to be re-designed is the interaction, which is a behavior. Also, most part of this behavior is related to audio, something that is difficult to implement with Axure or Illustrator; these two programs are thought not to design auditory user interfaces but graphical user interfaces, which is what most designers work with. In addition, providing such a design would not be accessible for visually impaired users unless a verbal description was given. This design specification thus requires to be written instead of drawn. However, some drawings will be provided as supportive material.

The design specification is divided into three sections: *Input mode*, *Output mode* and *Configuration*. The *Input mode* section describes all hardware and software required to enable accessible input for the target user. The *Output mode* section describes all hardware and software required to enable accessible output for the target user. The *Configuration* section explains how to modify the *Settings* view of Godot Engine to include accessibility features that users can easily change at will.

### 4.2.1 Input mode

#### Menu input

Menus can be navigated by keyboard only or with keyboard and mouse: this enables

visually impaired users with severe vision loss to efficiently navigate menus, and it also allows sighted users and visually impaired users with some degree of visual perception to make use of mouse gestures provided by Godot Engine. Keyboard shortcuts can be changed in the *Shortcuts* tab. Table 3 shows relevant keyboard shortcuts that are added or redefined. They are sorted in ascending order of specificity according to the hierarchy defined in Godot Engine graphical user interface (panes, tabs, pane contents...):

Shortcut	Behavior
F10	Changes menu focus to the first element of the upper menu ( <i>Scene</i> ). This shortcut is normally used to access upper menu in programs.
Alt- $n$	Changes menu focus to the first element of pane number $n$ . This enables users to easily switch between panes without having to use <b>Tab</b> for several seconds, and to change data from one pane remembering data from another pane for a short amount of time. For instance, the user may be listening to the 3D view, select an object, then changing to the <i>Inspector</i> pane and precisely change this object's position by entering numeric values at the <i>Translation</i> field.
Ctrl-Tab	Changes menu focus to the first element of the tab adjacent to focused tab in the focused pane.
Tab	Changes menu focus to the first element of the next group inside of the focused pane.
↑	Changes menu focus to the element above the focused element inside of the focused group in the focused pane; this includes elements from upper menu.
↓	Changes menu focus to the element below the focused element inside of the focused group in the focused pane; this includes elements from upper menu.
→	Changes menu focus to the element at the right of the focused element inside of the focused group in the focused pane; this includes elements from upper menu.
←	Changes menu focus to the element at the left of the focused element inside of the focused group in the focused pane; this includes elements from upper menu.

**Table 3:** Design specification: keyboard shortcuts for menus

### 3D view input

3D view can be navigated by keyboard only or with keyboard and mouse. Keyboard shortcuts can be changed in the *Shortcuts* tab.

In this design specification, two new modes have been added to the *Select*, *Move*, *Rotate* and *Scale* modes: *ExploreMove* and *ExploreRotate*. These modes are the keyboard equivalent of mouse gestures for visually impaired users. It is important to bear in mind that modes are mutually exclusive, and only one of them can be active at a time.

The shortcut design has tried to generate a unified mental model for the target user. Actions are specified with **Ctrl**, followed by a letter that is the initial of the action: *F* for *Find*, *O* for *Origin*, *M* for *Multiselection*... New modes are specified with plain letters, and they are close to the natural left hand position in the keyboard (the *A* and *S* letters), just like the letters for already existing modes were assigned. Finally, arrows have been chosen following the most intuitive spatial model.  $\uparrow$ ,  $\downarrow$ ,  $\rightarrow$  and  $\leftarrow$  keys symbolize  $+Z$ ,  $-Z$ ,  $+X$  and  $-X$  axes, so the arrow keys are placed representing the *floor*, i.e. the  $(X,Z)$  plain. In order to specify height with the *Y* axis,  $\uparrow/\downarrow$  are pressed with the **Shift** key; they were chosen for vertical movement because  $\uparrow$  and  $\downarrow$  are mentally related both with moving vertically or to the front/back.

Tables 4, 5, 6, 7 and 8 show relevant keyboard shortcuts that are added or redefined:

Shortcut	Behavior
Shift-↑	<ul style="list-style-type: none"> <li>• If the <i>Select</i> mode is selected and the <i>ExploreMove</i> and <i>ExploreRotate</i> modes are toggled off, it finds the object that is closest in the Y axis to and above the user's position in the 3D view and selects it.</li> <li>• If the <i>Move</i> mode is selected, the <i>ExploreMove</i> and <i>ExploreRotate</i> modes are toggled off and an object is selected, the object is moved up by a small amount of units.</li> <li>• If the <i>Rotate</i> mode is selected, the <i>ExploreMove</i> and <i>ExploreRotate</i> modes are toggled off and an object is selected, the object is rotated in the Y axis with a positive angle by a small amount of units.</li> <li>• If the <i>ExploreMove</i> mode is selected, the user moves up in the 3D space.</li> <li>• If the <i>ExploreRotate</i> mode is selected, the user rotates positively in the Y axis in the 3D space.</li> </ul>
Shift-↓	<ul style="list-style-type: none"> <li>• If the <i>Select</i> mode is selected and the <i>ExploreMove</i> and <i>ExploreRotate</i> modes are toggled off, it finds the object that is closest in the Y axis to and below the user's position in the 3D view and selects it.</li> <li>• If the <i>Move</i> mode is selected, the <i>ExploreMove</i> and <i>ExploreRotate</i> modes are toggled off and an object is selected, the object is moved down by a small amount of units.</li> <li>• If the <i>Rotate</i> mode is selected, the <i>ExploreMove</i> and <i>ExploreRotate</i> modes are toggled off and an object is selected, the object is rotated in the Y axis with a negative angle by a small amount of units.</li> <li>• If the <i>ExploreMove</i> mode is selected, the user moves down in the 3D space.</li> <li>• If the <i>ExploreRotate</i> mode is selected, the user rotates negatively in the Y axis in the 3D space.</li> </ul>

**Table 4:** Design specification: keyboard shortcuts for the 3D view (1)



Shortcut	Behavior
↑	<ul style="list-style-type: none"> <li>• If the <i>Select</i> mode is selected and the <i>ExploreMove</i> and <b>ExploreRotate</b> modes are toggled off, it finds the object that is closest in the Z axis and in front of the user's position in the 3D view and selects it.</li> <li>• If the <i>Move</i> mode is selected, the <i>ExploreMove</i> and <b>ExploreRotate</b> modes are toggled off and an object is selected, the object is moved forward by a small amount of units.</li> <li>• If the <i>Rotate</i> mode is selected, the <i>ExploreMove</i> and <b>ExploreRotate</b> modes are toggled off and an object is selected, the object is rotated in the Z axis with a positive angle by a small amount of units.</li> <li>• If the <i>Scale</i> mode is selected, the <i>ExploreMove</i> and <b>ExploreRotate</b> modes are toggled off and an object is selected, the object is scaled with a factor greater than 1 by a small amount of units.</li> <li>• If the <i>ExploreMove</i> mode is selected, the user moves forward in the 3D space.</li> <li>• If the <i>ExploreRotate</i> mode is selected, the user rotates positively in the Z axis in the 3D space.</li> </ul>

**Table 5:** Design specification: keyboard shortcuts for the 3D view (2)

Shortcut	Behavior
↓	<ul style="list-style-type: none"> <li>• If the <i>Select</i> mode is selected and the <i>ExploreMove</i> and <b>ExploreRotate</b> modes are toggled off, it finds the object that is closest in the Z axis and behind the user's position in the 3D view and selects it.</li> <li>• If the <i>Move</i> mode is selected, the <i>ExploreMove</i> and <b>ExploreRotate</b> modes are toggled off and an object is selected, the object is moved backward by a small amount of units.</li> <li>• If the <i>Rotate</i> mode is selected, the <i>ExploreMove</i> and <b>ExploreRotate</b> modes are toggled off and an object is selected, the object is rotated in the Z axis with a negative angle by a small amount of units.</li> <li>• If the <i>Scale</i> mode is selected, the <i>ExploreMove</i> and <b>ExploreRotate</b> modes are toggled off and an object is selected, the object is scaled with a factor smaller than 1 by a small amount of units.</li> <li>• If the <i>ExploreMove</i> mode is selected, the user moves backward in the 3D space.</li> <li>• If the <i>ExploreRotate</i> mode is selected, the user rotates negatively in the Z axis in the 3D space.</li> </ul>

**Table 6:** Design specification: keyboard shortcuts for the 3D view (3)

Shortcut	Behavior
	<ul style="list-style-type: none"> <li>• If the <i>Select</i> mode is selected and the <i>ExploreMove</i> and <i>ExploreRotate</i> modes are toggled off, it finds the object that is closest in the X axis and to the right of the user's position in the 3D view and selects it.</li> <li>• If the <i>Move</i> mode is selected, the <i>ExploreMove</i> and <i>ExploreRotate</i> modes are toggled off and an object is selected, the object is moved to the right by a small amount of units.</li> </ul>
→	<ul style="list-style-type: none"> <li>• If the <i>Rotate</i> mode is selected, the <i>ExploreMove</i> and <i>ExploreRotate</i> modes are toggled off and an object is selected, the object is rotated in the X axis with a positive angle by a small amount of units.</li> <li>• If the <i>ExploreMove</i> mode is selected, the user moves to the right in the 3D space.</li> <li>• If the <i>ExploreRotate</i> mode is selected, the user rotates positively in the X axis in the 3D space.</li> </ul>
	<ul style="list-style-type: none"> <li>• If the <i>Select</i> mode is selected and the <i>ExploreMove</i> and <i>ExploreRotate</i> modes are toggled off, it finds the object that is closest in the X axis and to the left of the user's position in the 3D view and selects it.</li> <li>• If the <i>Move</i> mode is selected, the <i>ExploreMove</i> and <i>ExploreRotate</i> modes are toggled off and an object is selected, the object is moved to the left by a small amount of units.</li> </ul>
←	<ul style="list-style-type: none"> <li>• If the <i>Rotate</i> mode is selected, the <i>ExploreMove</i> and <i>ExploreRotate</i> modes are toggled off and an object is selected, the object is rotated in the X axis with a negative angle by a small amount of units.</li> <li>• If the <i>ExploreMove</i> mode is selected, the user moves to the left in the 3D space.</li> <li>• If the <i>ExploreRotate</i> mode is selected, the user rotates negatively in the X axis in the 3D space.</li> </ul>

**Table 7:** Design specification: keyboard shortcuts for the 3D view (4)

Shortcut	Behavior
A	Toogles the <i>ExploreMove</i> mode, that allows to move around the 3D space.
D	Toogles the <i>ExploreRotate</i> mode, that allows to rotate in the 3D space.
Enter	If the 3D view search object menu is active, the focus changes to the next object whose name matches the search field (in case there is only 1 match, found object stays focused).
Ctrl-+	Zoom is increased.
Ctrl- -	Zoom is decreased.
Ctrl-F	A search floating menu (like the <i>Editor Settings</i> menu) appears with the title "Search object in the 3D view". This search menu contains a text box where the user can write the name of the object s/he is looking for in the 3D view. If there is any object in the 3D view whose name partially matches the search field, focus changes to that object.
Ctrl-D	The X-Y-Z distance from the user's position in the 3D view to the focused object is sent to the screen reader.
Ctrl-M	The focused object is selected. Several objects can be selected in this way (multi-selection)
Ctrl-S	Restores selection / multi-selection (no object is selected)
Ctrl-R	Restores user's rotation to (0,0,0).
Ctrl-O	Restores user's position to the origin, (0,0,0).
Ctrl-A	Plays all auditory icons. Synthesized sounds will be reloaded; sound samples will be played again.
Ctrl-P	Plays the auditory icon/s of the selected/multi-selected object/s. Synthesized sounds will be reloaded; sound samples will be played again.

**Table 8:** Design specification: keyboard shortcuts for the 3D view (5)

#### 4.2.2 Output mode

##### Screen reader

All elements of the graphical user interface should be defined with a tooltip that briefly describes their function, and a node type that will be read by the screen reader. When the focus of the menu or the 3D view changes, the screen reader will output the name, type, tooltip and value (if it has value) of the new focused element. Graphical user interface elements ordering must be logical: it must go from upper-left corner to the lower-right

corner<sup>17</sup>.

### Status indicators

Status indicators communicate data to the user as follows:

- CPU status indicator: if CPU load doesn't rise more than 15% for 0.5 seconds, no action is performed. Else, a continuous sound synthesizer is played. The synthesizer uses a square wave function; this wave function can be changed at the *Accessibility* menu. The frequency of the synthesizer starts with a C5 note for the lowest level of CPU load (i.e, when the CPU load starts increasing steeply), and it can get up to a C6 note for the highest level of CPU load. If the CPU load is more than 50% for more than 1 second, the following message is sent to the screen reader: *"CPU load higher than 50%"*. Both of the two alarm behaviors can be turned off at the *Accessibility* menu. Synthesizer level can be changed at the *Accessibility* menu.
- Resource change indicator: if any resource changes, the following message is sent to the screen reader: *"Resource changed%"*. This alarm behavior can be turned off at the *Accessibility* menu.
- Editor window repainting indicator: if the window is repainted (i.e, any element of the user interface has changed), a sound synthesizer is played for 0.1 seconds. The synthesizer uses a square wave function; this wave function can be changed at the *Accessibility* menu. A square wave has been used (as it was stated with the CPU status indicator) to associate this timbre with status indicators from the top-right part of the upper menu. The synthesizer sounds with half the volume of the CPU status indicator alarm. The synthesizer frequency is a C4 note, which resides at a different octave than C5-C6, the octave range for the CPU status indicator alarm. This is translated to a short, low-pitched note, which makes the alarm less unpleasant, taking into account that the window is repainted often. The volume for this alarm can be changed at the *Accessibility* menu. This alarm can be turned off at the *Accessibility* menu.

### 3D views - verbal descriptions

Verbal descriptions are those that give the distance between the user's 3D position and the focused object. As previously described, this data is sent to the screen reader. Here it is a sample verbal description: *"Object enemy13 is at 5 units X axis, 6 units Y axis and 1 unit Z axis away from you"*.

### 3D view - auditory icons

Objects in the 3D view should be represented by auditory icons, which provide the user with a spatial representation of objects. These auditory icons may be either sound samples (with a finite duration), synthesized sounds with a finite duration, or continuous synthesized sounds. Given that Godot Engine *knows* which is the type of each game object beforehand, it would be possible to assign default sounds to each type of game object. Whether if a particular game object is associated to a sound sample or a synthesized sound,

---

<sup>17</sup>Users with an occidental background have been considered, as they read from top-left to bottom-right. However, this ordering should be changed depending on the nationality and culture of the user. Simplifications like these one have been taken not to go outside the scope of a bachelor thesis.

and the configuration of any of the two, should be configurable from the *Accessibility* menu.

### 4.2.3 Configuration

#### Accessibility menu

The *Accessibility* menu contains all settings related to accessibility for Godot Engine, except for the keyboard shortcuts, which are configured in the *Shortcuts* tab. It would be located at *Menu* → *Settings* → *Editor settings* → *General*, as one of the many sections of this tab (examples of other sections in the *General* tab are the *Text editor* section, the *Scenetree Editor* section...). This section will be divided into several subsections; their contents are described in the following paragraphs.

#### Auditory icons settings

This subsection contains an entry for each element in the user interface that incorporates auditory icons, including the status indicators from the upper-right part of the upper menu bar. They would be organized in a tabular form; columns would be as follows:

1. Auditory icon name
2. Auditory icon type
3. Auditory icon volume
4. Whether if the icon is muted or not
5. Button; gives access to a file manager in case the auditory icon is a sound sample (otherwise, the button will be shown as non-clickable)
6. Button; opens a small toolkit to configure the synthesizer options in case the auditory icon is a synthesizer (otherwise, the button will be shown as non-clickable). This toolkit would contain the following options:
  - Waveform type
  - Minimum frequency in Hertz
  - Maximum frequency in Hertz
  - Whether if the user wants the synthesizer to be continuous or not
  - Duration in milliseconds, in case the synthesizer is finite and not continuous (otherwise, this field will be non-editable and a value of infinity will appear)
  - Decay function, in case the synthesizer is not continuous (otherwise, this field will be non-editable)

#### Screen reader settings

Operating systems already provide a settings menu for the screen reader. For this reason, the *Accessibility* menu should not contain screen reader settings, in order to avoid conflicts with the operating system settings.

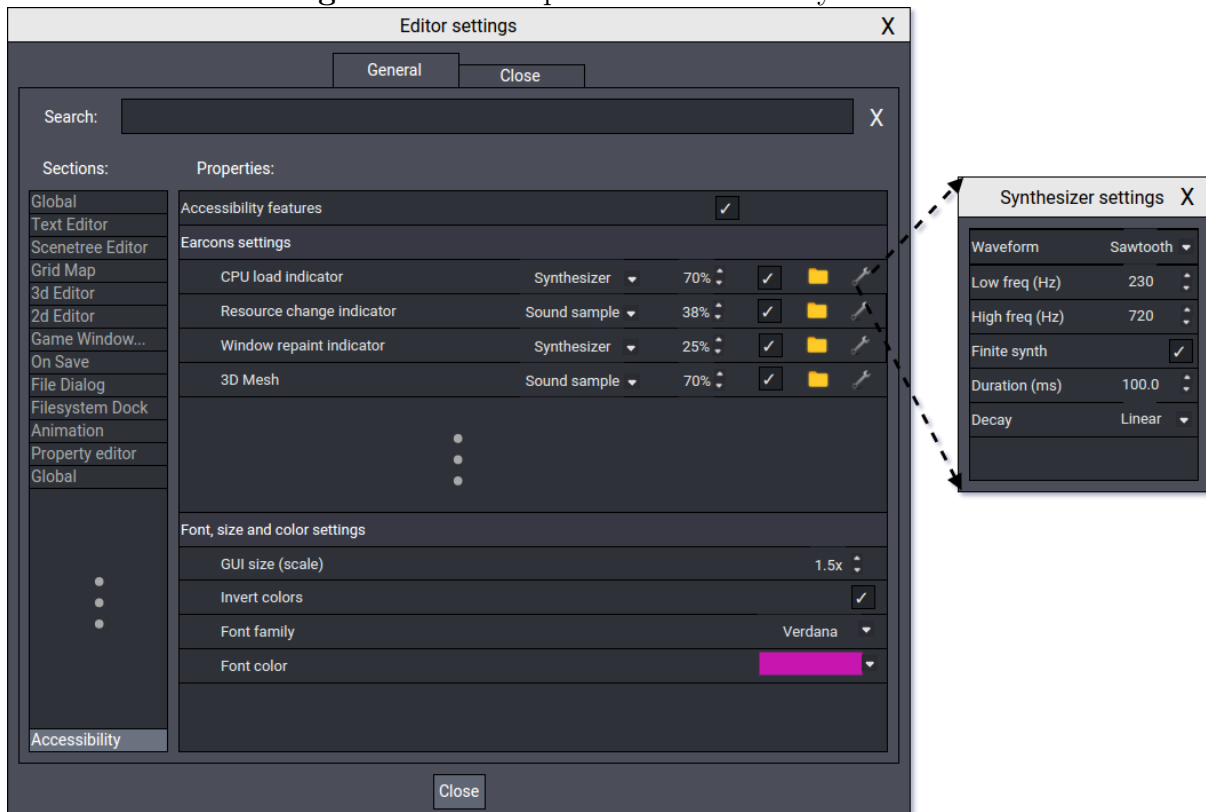
## Font, size and color settings

This section of the *Accessibility* menu should contain the following elements:

- Graphical user interface size: numeric input field. It changes the scale of the graphical user interface elements, relative to its original size.
- Graphical user interface color: already available through the *Global* section.
- Invert colors: check box. When ticked, this option inverts colors of all the elements of the graphical user interface.
- Font family: drop-down menu. It contains different font families (e.g. Comic Sans, Times New Roman...)
- Font size: already available through the *Global* section
- Font color: color picker drop-down menu. It changes the color of the text.

In order to complete the design specification, and to clarify the verbal description that has been given through these sections, a mockup of the *Accessibility* section is provided in figure 13. Vertical groups of three dots symbolize a group of elements of the same type that have been omitted (i.e., sections from the *Sections* side bar and auditory icon entries from the *Properties*). A *Synthesizer settings* popup has been also considered; dashed arrows coming from a settings icon mean that this popup appears on the screen when configuring a particular auditory icon. *Earcon* is an abbreviation for *auditory icon* that is commonly used in accessible programs.

Figure 13: Mockup for the accessibility section



## 4.3 Implementation details

### 4.3.1 New user interface elements

The *Accessibility* section added to Godot Engine's user interface is based on the user interface elements that were already implemented in Godot Engine (drop-down menus, color pickers, numeric input fields...). Adding this section to the program is straightforward because it does not require implementing new types of user interface elements. Also, this part of the solution could be reused with other game development kits.

### 4.3.2 Screen reader

Godot Engine (and ideally, any program) should maintain the same functionality for the most popular operating systems; namely, Windows, MacOS and Linux. Cross-platform compatibility can be difficult to achieve when different operating systems don't have the same base software at their disposal. This is the case of screen readers. Each operating system has a different screen reader and, although they all follow known accessibility standards, they receive, process and output information in different ways. Additionally, these screen readers must somehow establish communication with programs, so that they can read the user interface contents. This communication is done by a set of code known as accessibility API<sup>18</sup>, and each operating system has its own accessibility API. A quick overview of screen readers and accessibility API's for the most popular operating system is given below, as well as development details on how to make them establish communication.

- Windows: there is a wide variety of screen readers that can be used under this operating system. The two most commonly used screen readers are JAWS, from Freedom Scientific [77] and NVDA<sup>19</sup>, originally developed by Michael Curran and James Teh [82]. These screen readers communicate with Windows using Windows accessibility API's. Currently, Windows supports two accessibility API's: IAccessible2 [42] and MSAA<sup>20</sup> [81]. The former is more complete but its implementation details are somewhat less clear; the later contains less functionality, but Microsoft offers an extensive documentation for using the API in C and C++. This last point is interesting because Godot Engine's graphical user interface is written in C++, so that communicating Godot Engine with MSAA would be relatively easy.
- MacOS: this operating system uses VoiceOver [98] as its screen reader. This screen reader communicates with the operating system using Apple's own accessibility API. This API, known as AppKit [67], is actually in charge of many other services related to graphical user interface creation. One of these services is the NSAccessibility protocol [64], which is the piece of AppKit that specifically deals with accessibility. As part of AppKit, NSAccessibility is developed in Objective-C, which has been increasing compatibility with C++ over the last years [22]. Therefore, it would be possible to connect Godot Engine to VoiceOver using NSAccessibility.
- Linux: the most commonly used screen reader is the Orca screen reader [31]. Highest degree of accessibility requires applications' user interfaces to be implemented

---

<sup>18</sup>Acronym. Stands for *Application Programming Interface*.

<sup>19</sup>Acronym. Stands for *Non Visual Desktop Access*.

<sup>20</sup>Acronym. Stands for *MicroSoft Active Accessibility*.



with the GIMP Tool Kit [91], known as GTK for its acronym. Linux allows to choose different window managers, as opposed to Windows or MacOS; the most popular window manager is the Gnome desktop environment [75], for which more accessibility effort has been done. The Godot Engine's graphical user interface has been developed from scratch, and not with GTK; most game development kits have custom implementations of the graphical user interface, in order to optimize system resources (RAM memory and CPU). Despite of this, it is possible to communicate Godot Engine's user interface with the Orca screen reader thanks to the accessibility API used by the Gnome desktop environment, called ATK<sup>21</sup> [29]. ATK is an specification rather than a pure implementation, but this specification has already been programmed in several programming languages such as C or Python. Given that Godot Engine's graphical user interface is programmed in C++ (which is compatible with C), it is possible to load the ATK C implementation into Godot Engine, which would successfully communicate the game development kit with the Orca screen reader.

### 4.3.3 3D Audio

The proposed design specification requires auditory icons to be heard while the user is editing the video game. Therefore, software that outputs real time 3D audio is required, not only when testing games that use sound, but also when editing these games. Fortunately, Godot Engine already contains several C/C++ libraries that handle 3D audio and playing sound files. They are mentioned below, together with a link to their location in Godot Engine's GitHub source code repository:

- libogg [60]: handles files with the .ogg file format. Source code can be found in [48].
- libvorbis [59]: it is highly related to the libogg library. Source code can be found in [49].
- opus [51]: it handles audio streaming over the internet. Source code can be found in [84].
- rtaudio [44]: it handles real time audio. Source code can be found in [55].

These four libraries contain code that can be used to instantiate 3D audio objects. Each game object in the 3D view would have an associated 3D audio object, that would be instantiated at the (X,Y,Z) coordinates of the game object. The listener's point of view would coincide with the user's (X,Y,Z) position in the 3D view. The sound generated for a specific game object would be determined with the user's auditory icons settings from the *Accessibility* menu. These libraries are already supported by Godot Engine for gameplay, so calling them for editor runtime would be an easy task. It is worth noting that this idea could be also implemented in any other game development kit and with any other 3D audio library, which makes the proposed design flexible and reusable.

---

<sup>21</sup>Acronym. Stands for *Accessibility Toolkit*.

## 4.4 3D view prototype

### 4.4.1 Purpose

This prototype only tests a small part of the user interface of the chosen game development kit: the 3D view. It obtains information about user's perception of auditory information, and about the accuracy of his/her mental models as opposed to the real displayed information.

In this case the prototype only considers the user's ability to move in a 3D space and to look for a particular auditory icon. The prototype is presented to the user as an experiment where s/he has to accomplish a goal. Several objects have been placed in this 3D space: 5 cubes, 2 cylinders, and 1 sphere. The goal is to guess the number of cubes and cylinders and to reach the sphere, by using auditory icons associated to them. When the user reaches the sphere, the prototype stops its execution; user's guess about the number of cubes and cylinders is directly discussed with the researcher. User satisfaction is measured by having an interview with him/her about the experience of having used the prototype.

The prototype allows to use the rotation, movement and mode shortcuts defined in section 4.2.1. It is also possible to reset position and rotation, play all auditory icons in a sequential fashion, and to activate an audio message with help. Shortcuts for these actions have followed those proposed in section 4, to test if users are comfortable with these keyboard mappings.

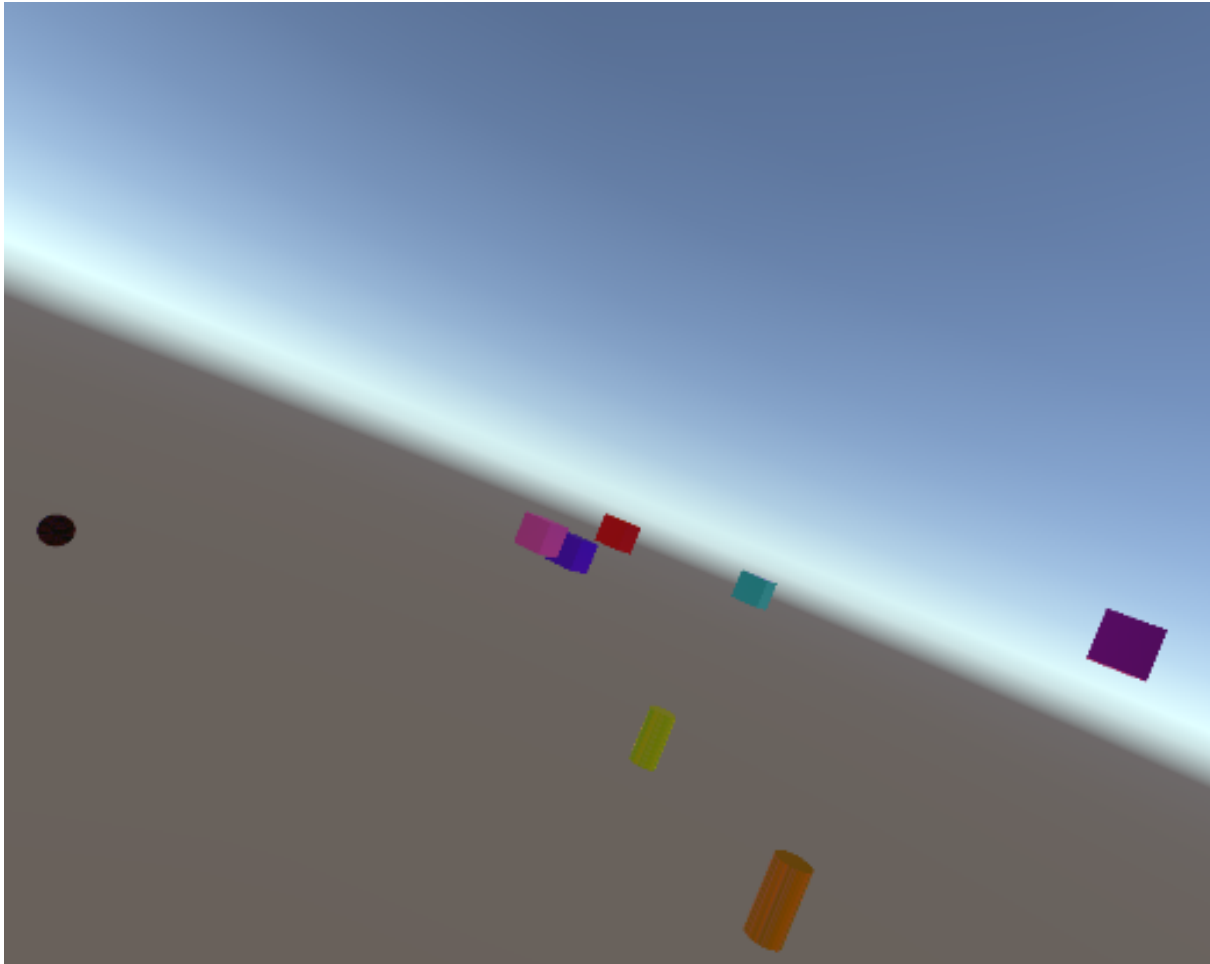
Pre-recorded synthetic voice has been used to simulate output from the screen reader. In this way, the following messages are generated:

- When the user activates the *ExploreMove* mode: *ExploreMove mode activated.*
- When the user activates the *ExploreRotate* mode: *ExploreRotate mode activated.*
- When the user activates the *ExploreRotate* mode: *ExploreRotate mode activated.*
- When the user restores his/her position: *position restored.*
- When the user restores his/her rotation: *rotation restored.*
- When the program starts and when the user presses F1 for help: *You are placed in a 3D space. There are objects in this space: cubes, cylinders and one sphere. Objects have associated auditory icons. These icons make sound, so that you know where objects are placed in space. Cubes sound like a double bass. Cylinders sound like a piano. The sphere sounds like a metallic mesh falling to the ground. Your goal is to find how many cubes are, to find how many cylinders are, and to reach the sphere. To move in space, you have to enter the ExploreMove mode by pressing A. To rotate, you have to enter the ExploreRotate mode by pressing D. You can move or rotate in any direction by pressing arrow keys. Up and down keys are used for front back direction (Z axis). Right and left keys are used for right left direction (X axis). Shift up, and shift down, mean up down direction (Y axis). Press F1 to repeat this help message. Press Control O to reset your position to (0,0,0). Press Control R to reset your rotation. Press Control P to play the auditory icons. Good luck!*

- When the user touches the sphere: *Congratulations! You have reached the sphere. The program will now exit. Thank you for your time.*

Figure 14 shows a screenshot of the prototype running. Both the *ExploreMove* and the *ExploreRotation* modes were used. Several game objects are shown in the 3D space.

**Figure 14:** 3D view prototype running



#### 4.4.2 Prototype implementation

Unity3d has been used to create the prototype for several reasons. Firstly, Unity workflow is simple and documentation is detailed, which enables developers to create prototypes, videogames or even applications really fast. Secondly, I have professional experience with Unity and I felt comfortable implementing the prototype with it. Thirdly, Unity allows to export executables to a wide variety of operating systems without changing any configuration. Lastly, Unity allows to use 3D sound in a very intuitive way; this is ideal to try the part of the design specification regarding 3D views.

Auditory icons have been based on sound samples, as Unity provides a good mechanism for easily using sound samples (called Audio Source or Audio Clip in Unity). Sound samples have been recorded with acoustic instruments from my home and a smartphone. They have been later processed and cut with Audacity, a free audio editor. Several sound samples were considered initially, including:

- a double bass played with a bow
- a double bass played using pizzicato
- an African djembe played in the center of the drum head
- a piece of wood knocked with the hand
- a metallic mesh hit against the ground
- a grand piano played without pedals or effects
- a piece of glass hit with the nail

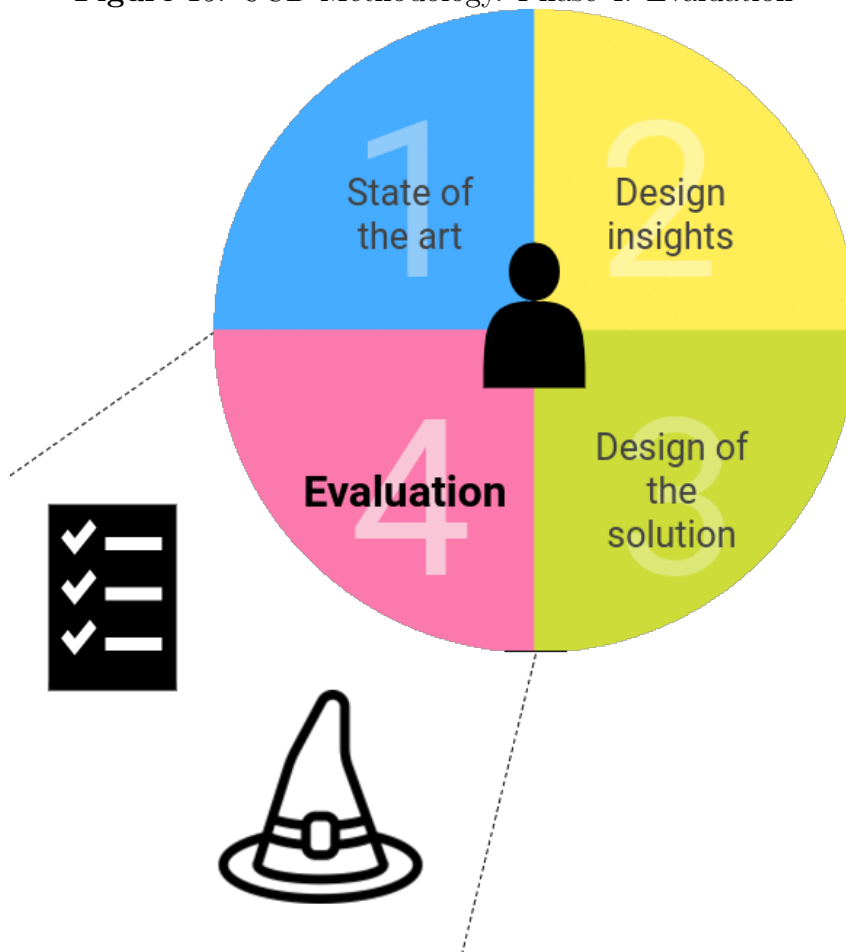
The bowed double bass was chosen for the cubes. The piano note was assigned to cylinders, and the metallic mesh was associated to the sphere. These three sound samples were the ones with most different timbres. Speech has been generated with an online text to speech tool.

The Unity project for the prototype has been uploaded to Mega, an online storage webpage. Executables have also been uploaded there, allowing to try the prototype without launching Unity. The download link can be found at [65]. This download is encrypted with a password in order to avoid plagiarism. The password can be requested by email to my university email address. The project could be made open source in the future, after bachelor thesis has been assessed.

## 5 Evaluation

Figure 15 makes a reference to the content that is going to be discussed in this section. It symbolizes that design and accessibility guidelines have been used for this section, and that the Wizard of Oz technique [11] has been considered as a possible additional evaluation technique to be used in the future.

Figure 15: UCD Methodology. Phase 4: Evaluation



Two possible techniques have been considered for the evaluation of the solution designed in section 4. The first one has been used to assess the quality of the design specification given its robustness and acceptance in the world of accessibility. The second one has been considered for future improvement; in this way, this phase of the research methodology can be later reused and expanded with more evaluation data.

### 5.1 Evaluation method

#### 5.1.1 WCAG guidelines

Web Content Accessibility Guidelines [89] are part of the Web Accessibility Initiative [56], which is the most comprehensive and maintained knowledge base about Web accessibility. WCAG<sup>22</sup> is focused on Web content, and it encompasses a wide range

<sup>22</sup>Acronym. Stands for *Web Content Accessibility Guidelines*.

of disabilities and technologies. In fact, WCAG have been consciously made technology-independent, in the same fashion as the User Centered Design methodology; because of this, they can be used to evaluate a software program, even if it is not Web-based. Section 2.0 of WCAG has been used in this evaluation, as it has been revised and contains more detailed evaluation criteria that will yield more useful evaluation results. As a side note, WCAG states that fulfilling these standards has benefits for all users, because features that make software accessible have been proven to also make it more usable [13]; this confirms the argument used for this thesis, where it has been argued that making a game engine accessible for visually impaired users is indeed beneficial for sighted users as well.

As explained later in section 6.1.1 with more details, there are many accessibility standards and guidelines that could have been used for this research. Also, because this project has been strongly based on Human-Computer Interaction, there are evaluation methods coming from this discipline that can be used, such as Nielsen usability heuristics [1]. However, WCAG is by far the most accepted set of accessibility guidelines, they are not tied to federal or legal regulations like other standards do<sup>23</sup>, and the evaluation process is particularly systematic and clear when using these guidelines. For these reasons, WCAG 2.0 is going to be used as the main evaluation criteria of the proposed solution.

WCAG evaluates three general criteria, that are later subdivided in smaller criteria called guidelines. In order for Web pages (and in the case of this project, for software) to be considered accessible, the following criteria should be fulfilled:

- *Perceivable: web content is made available to the senses - sight, hearing, and/or touch*
- *Understandable: Content and interface are understandable*
- *Robust: Content can be used reliably by a wide variety of user agents, including assistive technologies*

### 5.1.2 Future user evaluation with the 3D view prototype

The idea of considering users' opinion as one of the evaluation criteria comes naturally in Human-Computer Interaction research. Actually, users' opinion has already been used in section 3 to formulate the three design elements (*target user*, *needs* and *problem*) of this research project. Prototypes try to get user feedback about the proposed solution before having finished the design process. Designers need to put users in conditions close to real-life ones without an implementation. To test prototypes, a possible technique the Wizard of Oz [11], and it allows to learn about the quality of the design before making risky financial investments in the implementation.

The prototype discussed in section 4.4 covers a small part of the user interface proposed in the design specification. It is worth noting that there are many different ways of designing a prototype, with some of them covering even smaller sections of a user interface, and some others being close to the final product in terms of functionality and design. Prototypes are divided among horizontal prototypes, which show several features but none of them being fully developed, and vertical prototypes, where only one or two features are

---

<sup>23</sup>See section 6.1.1 for more details.

presented with close-to-product quality. This particular prototype can be considered a vertical prototype because it already implements shortcuts, navigation and the use of 3D sound for the 3D view as it would be done in a future implementation. Such a proof of concept is useful to know if the proposed solution for 3D views is adequate; the proposed 3D view redesign is a rather experimental user interface for visually impaired users, and no guidelines apply in this case.

## 5.2 Evaluation with WCAG 2.0 checklist

WCAG constitutes an extensive document that considers Web accessibility in detail. However, the Web Accessibility Initiative released last year a checklist [57] that is being widely deployed across accessibility experts and interaction designers, because it is concise and it covers all points of the guidelines. For the sake of clarity, this checklist will be used to assess the quality of the proposed solution. The evaluation checklist is long, so it has been included in the Appendix, section A.

Each one of the guidelines that compose WCAG is subdivided in several sections. Each section contains several recommendations. As its name indicates, they are not strict requirements, and a section can be considered fulfilled if several recommendations are fulfilled. While WAI does not offer details about how should grades be applied, the evaluation has considered three possible grades in this case:

- Tick for a recommendation that is fulfilled
- Cross for a recommendation that is not fulfilled
- *N/A* for a recommendation that does not apply, either because it makes a explicit reference to Web pages that cannot be considered in a game development kit, or because the studied program does not contain such element described in the recommendation.

## 5.3 Results

Considering only recommendations that apply to software<sup>24</sup>, and most specifically to Godot Engine with the proposed design of the solution, 61 recommendations out of 63 have been fulfilled<sup>25</sup>, and all sections from all guidelines can be considered fulfilled. This means that the levels of accessibility provided by the design of the solution are satisfactory and allow visually users to have a pleasant user experience.

Some grades require explanation. Guideline 1.2 about time-based media<sup>26</sup> is filled with *N/A* grades. This means that videos are not played as part of game development, so these recommendations did not apply in this case. All recommendations regarding contrast levels have been marked as fulfilled because font color can be configured from Godot Engine's *Editor settings* menu. Elements like *webpage* or *form* have been translated to its application counterparts: *graphical user interface, set of input fields...* Second

---

<sup>24</sup>That is, those recommendations that have been written by WAI for webpages but which can also be considered for a desktop application.

<sup>25</sup>See the Appendix (section A) for more information.

<sup>26</sup>See section A.1.2 for more details.

recommendation of guideline 1.3.3<sup>27</sup> is marked with a cross because some instructions actually rely upon sound in order for visually impaired users to be able of using the game development kit; however, this recommendation could be considered fulfilled because all auditory information in the design specification has a visual counterpart. Guideline 3.3.5<sup>28</sup> is marked with a cross because Godot Engine's help system has not been developed up to that extent, and the design specification did neither consider this possibility.

It should be noticed that the design specification only takes into account accessibility for visually impaired users. Although most part of the WCAG 2.0 recommendations have been fulfilled, any researcher willing to reuse this document for future work should keep this in mind. WCAG guidelines may change in the future and add more detailed recommendations for users with auditory disability, motor disability. . . Also, if future work is done about Godot Engine, its functionality should be checked at that moment; it is not unusual for programs to increment their functionality or to loose small parts of it after some time.

---

<sup>27</sup>See section A.1.3 for more details.

<sup>28</sup>See section A.3.3 for more details.



## 6 Project management

### 6.1 Regulatory framework

#### 6.1.1 Accessibility standards

Most accessible software is designed taking into account the Web Accessibility Initiative (WAI) guidelines [40]. WAI<sup>29</sup> tries to make the web accessible and inclusive. For this, a series of standards have been released through the years, including:

- Web Content Accessibility Guidelines (WCAG): recommendations for developing accessible web sites.
- Authoring Tool Accessibility Guidelines (ATAG): recommendations for making web design tools more accessible to creators. Hence the term *authoring tool*, as it makes reference to authors.
- User Agent Accessibility Guidelines (UAAG): recommendations for making web content rendering software (document readers, music players, etc) accessible.
- Accessible Rich Internet Applications (WAI-ARIA): recommendations for making interactive web applications accessible.

WAI design guidelines have been created with the World Wide Web in mind. This means they are not tailored to desktop or mobile applications. However, the WAI guidelines constitute the de facto standard for software accessibility in general, as they are complete and have been maintained for a long time. For this reason, the methodology followed in this work uses WAI guidelines.

In 2000, the United States Access Board released a law that requires technology at United States federal agencies to be accessible. This law is known as the Section 504 of the Rehabilitation Act, or simply the Section 508 technical standards [5]. The United States Department of Justice also published the 2010 ADA Standards for Accessible Design [16], which is analogue to the Section 504. Neither of these laws target research projects as in the case of this project, so there is no legal responsibility to apply them in this case.

Similar to the Section 508, Spanish government takes into consideration accessibility for technology that is released or used by national institutions. The UNE<sup>30</sup> Law 139803 [19] was released in 2012 and discusses accessibility requirements for Web content. Again there is no legal obligation to follow this law for this research project, as it targets Spanish government websites. It is worth mentioning that this law shares similarities with the WAI standards, as pointed out in the Spanish government website [21]. This highlights the wide acceptance of WAI guidelines.

#### 6.1.2 Software licenses

Several technologies have been considered and used while undergoing this User-Centered Design research project. Some of them could have been used to implement the solution,

---

<sup>29</sup>Acronym. Stands for *Web Accessibility Initiative*.

<sup>30</sup>Acronym. Stands for *Una Norma Española*, a Spanish term for *A Spanish Rule*.

but the restrictive nature of their software licenses made them inconvenient. Some others guaranteed more flexibility in the way they could be used, modified and distributed. Finally, there is also software that has not been used for the solution, but has been used to write this report, find information, etc. It is therefore important to mention the software licenses involved in this research and how do they affect future work.

Let's first consider technologies that were originally considered for the solution of this project but were finally dismissed because of their software licenses:

- Unreal Engine: this game development kit released its source code for its version 4. The development team of Unreal Engine has also released an online software repository that allows volunteers around the globe to request new features, correct bugs and even program new functionality that could be later added to the game development kit. However, these hypothetical contributions would depend upon the Unreal Engine End User License Agreement [95]. From sections 1.b) (*Distribution to other licensees*), 1.f) (*Distribution of Non-C++ Programming Language Integration*), *UE-Only Content, Non-Compatible Licenses*, 2) (*User License*), 3) (*Versions and Content*), 5) (*Royalty*), 8) (*Support*), 9) (*Feedback and Contributions*), 10) (*Third Party Software*), 11) (*Ownership*) and 12) (*Proprietary notices and attribution*) of this end user license agreement, it can be noticed that making Unreal Engine accessible (that is, making a contribution to Unreal Engine) for this research project does not necessarily mean that users are free to use this accessibility layer in the future. Also, this end user license agreement explicitly prohibits the addition of software that uses free software licenses such as the GNU General Public License, or the Creative Commons Attribution-ShareAlike License. This means that, in case that the proposed solution of this UCD research project is no longer maintained in the future and it was based on Unreal Engine (or if Unreal Engine abandoned this solution after having accepted it), users and volunteers would find difficulties trying to enhance this solution, specially if external software with any of these licenses was needed; the main problem is that these free software licenses are present in most modern software<sup>31</sup>.
- Unity 3D: this is another game development kit. Unity Technologies SF, owner of Unity3D, has recently released the source code of this program. However, not all of its source code is freely released, and full source code is only available after purchasing a *Pro* plan or an *Enterprise* plan [90] [15]. This poses even more problems than Unreal Engine; not only it is questionable that Unity Technologies would ever accept and maintain a contribution such as an accessibility layer, but it does not allow experienced users and developers to enhance it in the future unless paying a fee.
- Cry Engine: this game development kit's source code is published online and can be accessed for free. However, CRYTEK GmbH, the company that owns Cry Engine, *"requires users to purchase a commercial license to access certain parts of its code, and the legal restrictions of that license will still apply"* [107]. This poses the same problem as in the case of Unity 3D and Unreal Engine.

---

<sup>31</sup>It must be noted, however, that less restrictive software licenses are allowed such as the MIT License, the BSD License and the Apache License, which are not uncommon in software development.

There are also technologies which licenses allowed for easy contribution for –and distribution of– the solution proposed in this research project:

- Amazon Lumberyard: this game development kit uses the Apache License 2.0 [8]. This software license allows for commercial use, modification, distribution, patent use and private use. The only two provided restrictions are to include the license file and to let the creators know (with proper documentation) in case of changing the available software.
- Godot Engine: this game development kit comes with the MIT License [10]. This allows commercial use, modification, distribution and private use of this piece of software. The MIT License is slightly less restrictive than the Apache License provided with Amazon Lumberyard, as it only requires including the license file on the source code in case of modifying it.
- GREP: this is a virtual reality game development kit for non-technical users. As far as it has been known from tutoring lessons with its creator (a researcher from Universidad Carlos III de Madrid), there is no associated license to its source code since it is an academic project.

Finally, some programs were used while creating this project. The following list shows what restrictions applied for personal use and publication of documents:

- ShareLaTeX: an online text editor for the LaTeX document markup language. No restrictions for academic use.
- GitHub: a website for storing, sharing and tracking changes to software development projects. No restrictions for downloading projects.
- Ubuntu Studio 16.04.2 LTS: an operating system. Free of fees and restrictions.
- Debian Stable: an operating system. Free of fees and restrictions.
- Atom 1.17.2: a text editor for software development. Free of fees and restrictions.
- LibreOffice Calc 5.1.6.2: a spreadsheet editing software. Free of fees and restrictions.
- Microsoft Office PowerPoint: a presentation editing software. Paid (provided by university); free of additional fees and restrictions for distributing created documents.
- Orca 3.18.2: a screen reader. Free of fees and restrictions.
- Oracle VirtualBox: a virtual machine environment. Free of fees and restrictions.
- Skype: a text messaging and video / audio chat software. Free of fees and restrictions for personal use (video conference with Bachelor thesis supervisor).
- Whatsapp: a text messaging and video / audio chat software. Free of fees and restrictions for personal use.
- Google Chrome: a web browser. Free of fees and restrictions.

- Unity3d: a game development kit. Free of fees. No restrictions for the use it has been given (creating a prototype for academic purposes).
- Draw.io: a webpage for diagram drawing. Free of fees and restrictions.
- Mendelej Desktop 1.17.10: a program for handling bibliographic references in LaTeX.
- Wordreference.com: online dictionary. Free of fees and restrictions.
- Lunapic.com: online image editor. Free of fees and restrictions.
- Windows 10: an operating system. Provided for free with my laptop.
- GanttProject: a desktop program to create Gantt charts. Free of fees and restrictions.
- Audacity: an audio editing tool. Free of fees and restrictions.
- From Text To Speech: text to speech webpage. Free of fees and restrictions.
- Voice Recorder: voice recorder app for the Samsung Galaxy Note 3 smartphone. Free of fees and restrictions.

None of these tools required paying a fee using the standard and university student price plans available to me by the time this document was written. Also, no restrictions apply on the way this document and other related files have been created and published, principally because none of them have been created for commercial or illegal purposes, and the paid software was given by the university.

### 6.1.3 User privacy

A group of users have been interviewed for the exploratory study (shown in section 3). First, two open questions have been asked about the way in which they interact with the computer on a daily basis and about their thoughts on alternative modes of interaction I have proposed. Also, data has been gathered regarding their age, profession and degree of visual accuracy. Their names have not been included on the document; instead, invented names have been used in order to preserve their anonymity.

This complies with the Spanish organic law on the protection of personal data [3] (known as LOPD 15/1999<sup>32</sup>) and the Spanish organic law on the civil protection of the rights to honor, personal and familiar intimacy and one's own image [17] (known as LO 1/1982<sup>33</sup>). Users were notified about the data that was going to be gathered prior to the interview. Additionally, a proof of consent in the form of audios and / or signed documents have been kept.

---

<sup>32</sup>Acronym. Stands for the Spanish term *Ley Orgánica de Protección de Datos*.

<sup>33</sup>Acronym. Stands for the Spanish term *Ley Orgánica*.

## 6.2 Planning

Tables 9 and 10 show how tasks for this research project have been organized in time. Main tasks correspond to known phases of the research methodology and additional required work. These are divided into several subtasks for the sake of clarity. Following the notation used by most Gantt diagram plotting software, the amount of effort devoted to each task is specified in days. Most effort has been devoted to the design insights and the design of the solution, since they define the core of User-Centered Design research (understanding users' needs and designing to fulfill these needs). Time devoted to the design insights is higher than the time needed for the design of the solution because of the nature of this research, where target users are difficult to find.

Task	Duration (days)	Start date	End date
<b>1. State of the art</b>	<b>18</b>	<b>03/09/2016</b>	<b>26/12/2016</b>
1.1. Definitions	3	15/09/2016	01/10/2016
1.2. Assistive technologies	4	03/10/2016	20/11/2016
1.3. Accessible user interfaces	5	03/09/2016	10/10/2016
1.4. Accessible software development tools	2	05/09/2016	20/09/2016
1.5. Accessible game development tools	4	13/10/2016	26/12/2016
<b>2. Design insights</b>	<b>123</b>	<b>01/12/2016</b>	<b>07/04/2017</b>
2.1. Participants search	53	01/12/2016	15/03/2017
2.2. Participants demographic data gathering	30	24/02/2017	01/04/2017
2.3. Participants interviews	35	24/02/2017	06/04/2017
2.4. Design insights formulation	5	27/12/2016	07/04/2017
<b>3. Design of the solution</b>	<b>62</b>	<b>09/04/2017</b>	<b>07/06/2017</b>
3.1. Game development kits search	3	09/04/2017	16/04/2017
3.2. Game development kits comparison	2	17/04/2017	19/04/2017
3.3. Godot Engine user interface analysis	5	20/04/2017	30/04/2017
3.4. Input mode specification	12	02/05/2017	18/05/2017
3.5. Output mode specification	10	02/05/2017	18/05/2017
3.6. Configuration specification	11	19/05/2017	02/06/2017
3.7. Mockup design	4	03/06/2017	07/06/2017
3.8. Implementation details	6	19/05/2017	25/05/2017
3.9. 3D view prototype design	2	19/05/2017	21/05/2017
3.10. 3D view prototype implementation	5	22/05/2017	02/06/2017

Table 9: Planning (1)

Task	Duration (days)	Start date	End date
<b>4. Evaluation</b>	<b>5</b>	<b>04/06/2017</b>	<b>12/06/2017</b>
4.1. Evaluation methods	2	04/06/2017	07/06/2017
4.2. Evaluation method selection	1	08/06/2017	09/06/2017
4.3. WCAG 2.0 checklist evaluation	1	10/06/2017	11/06/2017
4.4. Evaluation conclusions	1	12/06/2017	13/06/2017
<b>5. Documentation</b>	<b>129</b>	<b>01/09/2016</b>	<b>16/06/2017</b>
5.1. Project structuring	7	01/09/2016	10/09/2016
5.2. Abstract	3	05/11/2016	12/06/2017
5.3. Introduction	7	25/03/2017	12/04/2017
5.5. Interviews LaTeX card design	8	15/04/2017	27/04/2017
5.6. Project management	2	13/03/2017	15/06/2017
5.7. Conclusions	2	12/06/2017	14/06/2017
5.8. Acknowledgements	1	15/10/2016	22/10/2016
5.9. Glossary	20	05/10/2016	21/05/2017
5.10. Reference search	10	05/09/2016	11/06/2017
5.11. Reference formatting	2	12/06/2017	13/06/2017
5.12. WCAG 2.0 checklist	1	10/06/2017	12/06/2017
5.13. Terminology check	5	15/04/2017	20/04/2017
5.14. Image search	6	10/03/2017	17/05/2017
5.15. Mockup and UCD images creation	8	27/05/2017	02/06/2017
5.16. LaTeX formatting	40	05/09/2016	20/05/2017
5.17. Section order coherence revision	3	24/11/2016	13/06/2017
5.18. Errata correction	4	10/09/2016	16/06/2017

Table 10: Planning (2)

An approximation of the number of work hours per week is summarized in table 11. This table also contains the total number of hours for each month, assuming an average of 4 weeks per month approximately. The project has lasted for the duration of

a bachelors degree academic year (September 2016 to June 2017). Notice that less hours were dedicated to the project during the first half of the year, and that the number of weekly hours significantly increases during the second half of the year. The reason for this workload distribution is that the bachelor thesis technically starts in the second half of the year; however, and given the difficulty of the project and the fact that it was a topic personally proposed by myself, I agreed with my bachelor thesis supervisor to start looking for information and contacts during the first half of the year, predicting this phase of the research project was going to be particularly slow.

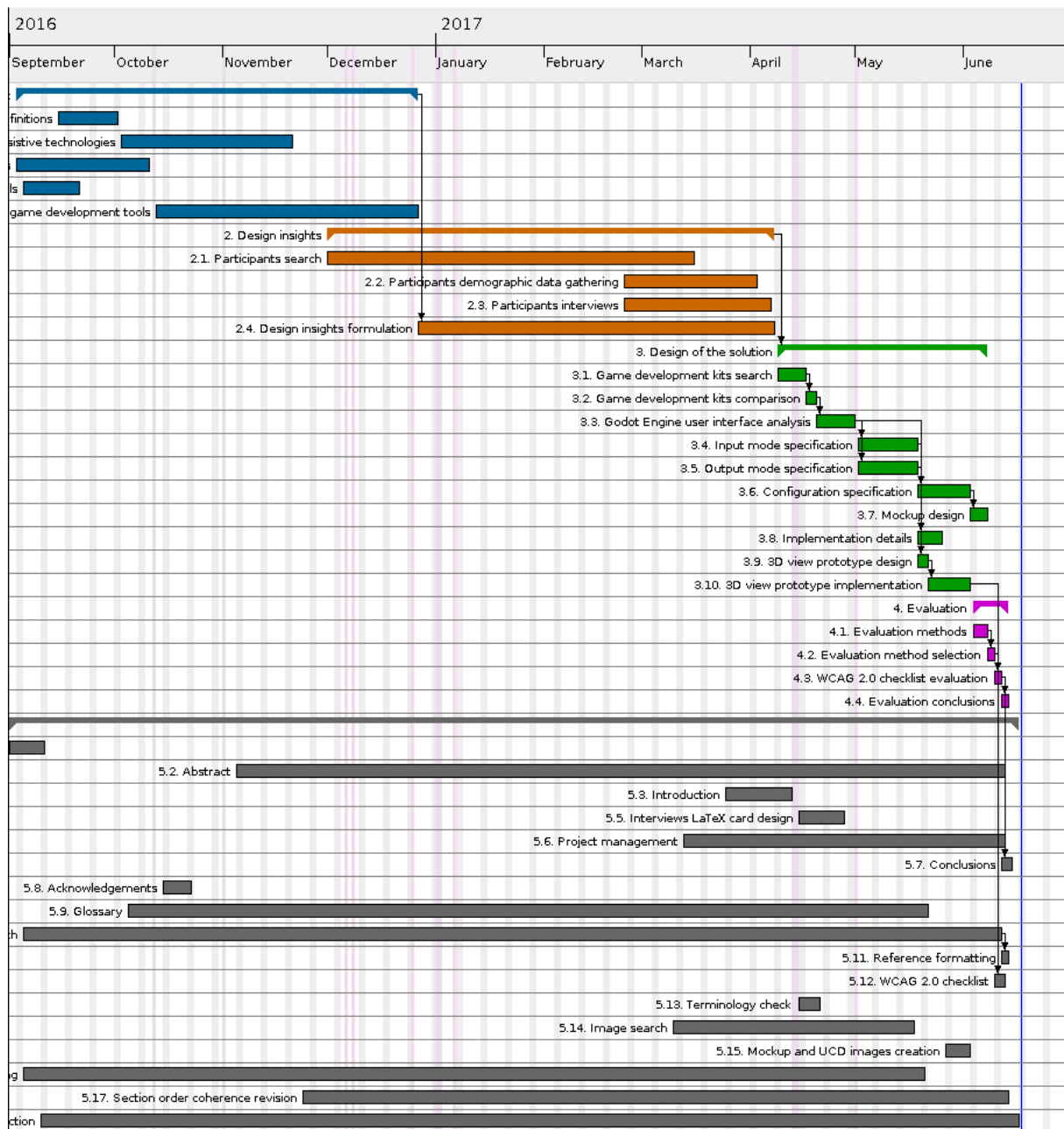
Month	Hours / week	Total hours
September 2016	10	40
October 2016	5	20
November 2016	5	20
December 2016	10	40
January 2017	5	20
February 2017	20	80
March 2017	20	80
April 2017	20	80
May 2017	20	80
June 2017	10	40
<b>Total</b>	<b>125</b>	<b>500</b>

**Table 11:** Weekly dedicated hours

Finally, a Gantt chart (figure 16) shows and organizes the aforementioned data organized in a more intuitive way. It gives an overview of the whole project; in addition, it represents task dependencies and whether if they have been done sequentially or in parallel. It is worth noting that the documentation task was done in parallel with the research methodology phases. Dependencies between tasks have also been represented, and they highlight how each phase of the User-Centered Design methodology depends on its previous phase, although some degree of overlapping is allowed for some tasks.



Figure 16: Gantt chart



### 6.3 Budget

Budget for this project has taken into account two factors: human resources and non-human resources. Human resources describe costs derived from the work done by employees (in this case, a junior computer scientist). Non-human resources refer to material costs: software licenses, hardware, electricity, Internet connection, etc.

Three different, yet related job positions have been considered to calculate the hourly wage. The work for this thesis has required working in the fields of user interface design, software development and academic research; thus, I thought it was more realistic to take into account an average salary drawn from these three job positions. Average hourly wage

for a junior front-end developer in Spain is about 20€ as of 2014 [37]. Similar data can be extracted from global economic studies, with an approximate salary of 22€ per hour for a European junior software developer in 2014 [39]. The 2017 estimate of the average hourly wage for a Spanish researcher in the field of technology is about 20€ as well [88]. Therefore, it seems reasonable to assume a hourly wage of 20€ in my case.

Tables 12 and 13 show all costs associated to this project, including both human resources and non-human resources. Power consumption has been based on an average power tariff from a popular Spanish electric utility company [85]; average cost is about 0,11789€/KWh. Main means of transport has been metro, commuter train and bus; Spanish people from the Community of Madrid whose age is under 26 have access to a special transport fare that includes unlimited use of these transport means for 20€ per hour [61]. Original laptop price (around 700€) was higher than the one written down; this means that its value has decreased over the years. Perishable products include pens, pencils, rubber, sharpener, and sheets of paper. Spanish generic VAT had a value of 21% at the time this thesis was being written [46]; all costs of this project are associated with generic VAT. As explained in section 6.1.2, no software licenses have been paid. Phone calls, 4G internet connection and home broadband internet connection have been based on a combined plan offered by British telephony company Vodafone [99]; this internet+phone plan costs 42.40€ per month.

Type	Cost
Developer / researcher	10,000€
Power consumption	58.95€
Laptop (Packard Bell Easy Note TM85 JO 460SP)	300€
Transport fares	200€
Phone calls + 4G + broadband internet	424€
Perishable products	30€
Amazon Lumberyard	0€
Godot Engine	0€
GREP	0€
ShareLaTeX	0€
GitHub	0€
Ubuntu Studio 16.04.2 LTS	0€
Debian Stable	0€
Atom 1.17.2	0€
LibreOffice Calc 5.1.6.2	0€
Orca 3.18.2	0€
Oracle VirtualBox	0€
Skype	0€
Whatsapp	0€
Google Chrome	0€
Microsoft Office PowerPont	0€

**Table 12:** Complete project costs (1)

Type	Cost
Unity3d	0€
Draw.io	0€
Mendeley Desktop 1.17.10	0€
Wordreference.com	0€
Lunapic.com	0€
Windows 10	0€
GanttProject	0€
Audacity	0€
From Text To Speech	0€
Voice Recorder	0€
<b>Total</b>	<b>11,012.95€</b>
<b>Total with VAT</b>	<b>13,325.67€</b>

**Table 13:** Complete project costs (2)

Overall cost of the project is low, thanks to the use of free software<sup>34</sup>. Also, nor hardware neither other materials have been required, which further reduces costs.

<sup>34</sup>Referring to free software as software that can be used for free, without having to pay any fee. This is different from software that fulfills the Four Essential Freedoms of Software, published by the Free software Foundation [100].

## 7 Conclusions

### 7.1 Technical conclusions

After having gone through the four phases of the User Centered Design methodology, it can be concluded that there is a possible solution for the target user's problem. In other words, it *is* possible for visually impaired users to develop games efficiently and using professional tools that are used in the video game industry. The conclusion of this research project has positive consequences for visually impaired users: they now can participate in professional game development and they can also create their own games without fully depending on programming. Till now, it was assumed by most people that video game development was beyond reach of these users; I have found this behavior both in sighted users and visually impaired users. However, paying special attention to users' needs has successfully turned into a detailed specification of input/output modes (how does the user interact with the computer) and logic behavior of the software (what does the computer to satisfy users requests), all of that fulfilling design guidelines that guarantee that the final product will be accessible.

This study shows that most part of the problem was common to that of other programs that are not accessible by default. Godot Engine and Microsoft Word, for instance, are almost identical as they consist of common graphical user interface elements. Making any of them 80% accessible only requires following accessible design principles and evaluating the design of the solution against accessibility guidelines.

What makes this research work innovative is the way in which the 20% remaining part of the program has been made accessible. This part of the user interface is the 3D view, which does not fit into most accessibility standards as it is normally not present in most software and web pages. The conducted exploratory study reveals how users prefer to navigate through this three-dimensional space. It has been shown that visually impaired users prefer to use keyboard navigation for input and a mixture of screen reader and auditory icons for output. Another important conclusion drawn from this research project is that the considered target user may have different degrees of visual impairment, and this requires that the designed user interface is flexible and allows for different interaction modes, specially in terms of output mode.

### 7.2 Future work

The high degree of *reusability* of this research project, possible thanks to the User-Centered Design methodology that has been used, makes room for future improvements that would make the proposed solution much more useful, usable and efficient.

One of the things that would enhance the quality of the solution would be to use several iterations. This means to start the design phases again, considering what has been discovered previously. This is only possible with more time. A bachelor student has much less time than a PhD student to finish the corresponding thesis (one semester compared to three years almost exclusively focused on that thesis). Nevertheless, it is possible for any researcher to take this project as I have left it, and perform one or more cycles, so that the final product really complies with users' needs and preferences.

Again, more time would have been translated to more users for the exploratory study. This information gathering technique will always yield less users compared to other techniques that are more automated; however, increasing the number of participants to several tenths or hundreds of people would have significant benefits for the quality of the findings from the exploratory study. In this case, it is not only statistical quantity what makes findings better, but also the possibility to find more personal, unique details from users.

Including user tests in the evaluation phase would also provide more informed conclusions. The reason for designing the interactive prototype is precisely to conduct a complete user evaluation in the future. Ideally, new iterations of the UCD methodology would include both an evaluation with accessibility guidelines and a user evaluation in this phase of the research.

This research has been focused on Human-Computer Interaction. As described before, normally the *product* of such a research work is not a machine or a piece of software, but a design specification. More time is devoted to analyzing users' needs, preferences and mental models, and a smaller amount of time is used to actually implement prototypes. Adding more iterations to the User-Centered Design research methodology could generate a solution quite close to a final product. Fortunately, the results of this research can be easily reused and modified to get an implementation.

Finally, there is one aspect that is still to be solved regarding accessibility in video game creation. The outcome of this research work allows visually impaired users to develop the logic and behavior of computer games, but this does not include working as a graphic artist or as an animation designer. Making these tasks accessible would be very hard, especially for graphic design, and they are outside of the scope and extent of this document. However, this research will be useful as a reference for addressing these problems, as they are highly related to the problem studied in this document.

### 7.3 Personal conclusions

I personally believe that the bachelor thesis is quite different from other subjects taught at the bachelor degree. Not only students gain extensive knowledge during the bachelor thesis, but they also mature as people. To me, this project has been difficult and it has required much more patience and time than other subjects; despite of this, and remembering conversations with other classmates who are also writing their bachelor thesis, it just feels better than any other subject we have done before. In this sense, all the effort I have made is totally worth it. Before closing the narrative of this thesis, I would like to let the reader know about all the things I have learned and all valuable experiences I have felt during these months.

One of the most important things I have learned while writing this thesis has been how to defend a personal project. This thesis spins around a theme that has been proposed and devised by me more than two years before entering the last course of the bachelors degree. It is not unusual for students to select a research topic proposed by a professor from the university. However, I was totally obsessed about this idea because, to my mind, it can give lots of people the possibility of making their creative projects come to reality,

just as I do when I design my games with Unity, or when I worked in game development last summer. Since some years ago, I am concerned about the fact that most work done in computer science nowadays goes to *commodities*, that is, products that do not solve serious problems, but which instead make life more comfortable or provide entertainment. I think of this bachelor thesis as the possibility to finish my bachelors degree feeling that I am prepared to help others as a computer engineer. For this reason, I have tried to justify each and every decision I have taken during the development of this document, so that the solution I *believe* in can be considered as valid.

I have also learned how to write an academic research document. I am heavily inclined towards research and I hope to work as a researcher in the future, so this knowledge is extremely valuable for me. Prior to taking this subject, I was not conscious about the effort researchers put into making a document structured and linked from one section to the next one. I have learned to shape my mindset when I write for an academic paper, so that I am objective and positive when I defend my arguments and hypotheses.

Communication skills is another a key aspect that I have developed when writing this thesis. During these months, and just for bachelor thesis-related subjects, several hundreds of emails have been sent. I have also had lots of calls and meetings with people from Spain who could help me with my thesis. In fact this communication workflow has been exhausting for me, as I had to maintain similar conversations over different mail lists, phone calls and private messages with lots of people. Also, it was especially difficult to find participants for the exploratory study, and it was even more difficult to get answers from them as each one lives in separate parts of the globe; most of them are busy either with studies or with work, so communication process was slow and with interruptions. However, I am happy about this experience, since I have learned when can I automate communication (and how to do it), and when do I have to exchange more personal messages with people.

Thanks to this communication process, I have been gifted with wonderful conversations of all kinds. Interaction with participants from the exploratory study has been especially interesting for me, as interviews often ended in digressions about related topics; these conversations have not been included as they were more personal and off-topic, but I can guarantee that they are invaluable. In one of these conversations, I discovered how visually impaired users think of themselves, and how do they react to the terms people use in order to be polite. It shocked me to see that these polite words can sometimes feel worse for them than using the term that describes who they are and what they are. Occidental society has developed an instinctive tendency to explode in unnecessary empathy towards disabled people. Contrary to what people normally think, this makes disabled people be more aware that they are being regarded as different. For one of the participants I was talking to during that conversation, this game of words is easier for people than actually finding a balance between accepting their disability and being respectful to them. I have also felt shocked about how people sometimes mistreat disabled people in a professional context; several users told me about working experiences where they felt this way. All this should make us reconsider how do we communicate disabled people. We should not forget that they are, simply, people, and they deserve to be treated like that; not with excessive empathy, and not with disdain. These insights have shaped the way in which this project has been developed; after rediscovering the meaning of *inclusion*, my efforts

have been directed towards reaching this goal, which is nothing more and nothing less than allowing people to live as what they are: people.



## Glossary

**API** Acronym for *Application Programming Interface*. 55, 56

**ATK** Acronym for *Accessibility Toolkit*. 56

**GTK** Acronym for *GIMP Toolkit*. 56

**GUI** Acronym for *Graphical User Interface*. 23

**HCI** Acronym for *Human Computer Interaction*. 1

**IDE** Acronym for *Integrated Development Environment*. 19, 20

**MSAA** Acronym for *MicroSoft Active Accessibility*. 55

**NVDA** Acronym for *Non-Visual Desktop Access*. 55

**ONCE** Acronym for *Organización Nacional de Ciegos Españoles*; Spanish term for *National Organization of Spanish blind people*. 24

**UCD** Acronym for *User Centered Design*. 8–10, 16, 26, 65, 77

**UI** Acronym for *User Interface*. 17

**UNE** Acronym for *Una Norma Española*; Spanish term for *A Spanish Rule*. 64

**VR** Acronym for *Virtual Reality*. 17

**WAI** Acronym for *Web Accessibility Initiative*. 62, 64

**WCAG** Acronym for *Web Content Accessibility Guidelines*. 60–63

## References

- [1] Jakob Nielsen. *10 Usability Heuristics for User Interface Design*. Nielsen Norman Group, 1995. URL: <https://www.nngroup.com/articles/ten-usability-heuristics/>.
- [2] Stephen W. Mereu and Rick Kazman. “Audio Enhanced 3D Interfaces for Visually Impaired Users”. In: *ACM SIGCAPH Computers and the Physically Handicapped* (1997), pp. 10–15. DOI: 10.1145/250025.250029. URL: <http://www.sigchi.org/chi96/proceedings/papers/Mereu/rnk-txt.htm>.
- [3] *Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal*. 1999. URL: <https://www.boe.es/buscar/pdf/1999/BOE-A-1999-23750-consolidado.pdf> (visited on 06/05/2017).
- [4] J Sánchez and M Lumbreras. “Usability and Cognitive Impact of the Interaction with 3D Virtual Interactive Acoustic Environments by Blind Children”. In: *Proceedings of the 3rd international conference on disability, virtual reality and associated technologies (ICDVRAT 2000)*. Santiago de Chile: The University of Reading, 2000, pp. 99–105. DOI: 10.1.1.99.4424. URL: [http://centaur.reading.ac.uk/19120/1/ICDVRAT2000%7B%5C\\_%7DFull%7B%5C\\_%7DProceedings%7B%5C\\_%7D3rd%7B%5C\\_%7DConf.pdf%7B%5C#%7Dpage=99](http://centaur.reading.ac.uk/19120/1/ICDVRAT2000%7B%5C_%7DFull%7B%5C_%7DProceedings%7B%5C_%7D3rd%7B%5C_%7DConf.pdf%7B%5C#%7Dpage=99).
- [5] United States Access Board. *Section 508 Standards*. 2000. URL: <https://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards> (visited on 05/09/2017).
- [6] Christopher Frauenberger and Markus Noistering. “3D audio interfaces for the blind”. In: *Proceedings of the 9th International Conference on Auditory Display (ICAD03)*. Austria, 2003, pp. 280–283. URL: <https://pdfs.semanticscholar.org/0203/0c97aa1e989568a49b3af224f3c1e55de2b6.pdf>.
- [7] John Wood, Mark Magennis, Elena Francisca, et al. “The Design and Evaluation of a Computer Game for the Blind in the GRAB Haptic Audio Virtual Environment”. In: (2003). URL: <http://www.eurohaptics.vision.ee.ethz.ch/2003/35.pdf>.
- [8] *Amazon Lumberyard LICENSE*. 2004. URL: <https://github.com/PlayFab/LumberyardSDK/blob/master/LICENSE>.
- [9] Alan Dix, Janet Finlay, Gregory D Abowd, et al. *Human-Computer Interaction*. 3rd. Harlow: Pearson Education Limited, 2004. ISBN: 978-0-13-046109-4.
- [10] Juan Linietsky and Ariel Manzur. *Godot Engine - LICENSE.txt*. 2004. URL: <https://github.com/godotengine/godot/blob/master/LICENSE.txt> (visited on 06/07/2017).
- [11] *Wizard of Oz method*. 2006. URL: <http://www.usabilitynet.org/tools/wizard.htm> (visited on 06/19/2017).
- [12] Alan R Hevner. “A Three Cycle View of Design Science Research”. In: *Scandinavian Journal of Information Systems © Scandinavian Journal of Information Systems* 19.192 (2007), pp. 87–92. URL: <http://aisel.aisnet.org/sjis>.
- [13] Ben Caldwell, Michael Cooper, Loretta Guarino Reid, et al. *Web Content Accessibility Guidelines (WCAG) 2.0*. 2008. URL: <https://www.w3.org/TR/WCAG20/> (visited on 06/19/2017).

- [14] *Shades of Doom Version 2.0*. 2008. URL: <http://www.gmagames.com/sod.shtml> (visited on 06/15/2017).
- [15] Rune Skovbo Johansen. *Why You Probably Don't Need a Source Code License*. 2009. URL: <https://blogs.unity3d.com/2009/03/20/why-you-probably-dont-need-a-source-code-license/> (visited on 06/07/2017).
- [16] *2010 ADA Standards for Accessible Design*. 2010. URL: <https://www.ada.gov/regs2010/2010ADASTandards/2010ADASTandards.htm> (visited on 06/07/2017).
- [17] *Ley Orgánica 1/1982, de 5 de mayo, de protección civil del derecho al honor, a la intimidad personal y familiar y a la propia imagen*. 2010. URL: <http://www.boe.es/buscar/pdf/1982/BOE-A-1982-11196-consolidado.pdf> (visited on 06/05/2017).
- [18] World Health Organization. *World Report on Disability*. Tech. rep. Malta: WHO Press, 2011. URL: <http://www.who.int/disabilities/world%20report/2011/report.pdf>.
- [19] *AENOR - Requisitos de accesibilidad para contenidos en la Web*. 2012. URL: <http://www.aenor.es/aenor/normas/normas/fichanorma.asp?tipo=N&codigo=N0049614#.WUSRn36G000>.
- [20] Cheryl Gabbert and Sarah Malburg. *Common Types and Characteristics of Visual Impairments*. 2012. URL: <http://www.brighthubeducation.com/special-ed-visual-impairments/35103-common-types-of-visual-impairment-in-students/> (visited on 06/14/2017).
- [21] *PAe - Normas Accesibilidad*. 2012. URL: [https://administracionelectronica.gob.es/pae\\_Home/pae\\_Estrategias/pae\\_Accesibilidad/pae\\_normativa/pae\\_eInclusion\\_Normas\\_Accesibilidad.html#.WUSQTX6G000](https://administracionelectronica.gob.es/pae_Home/pae_Estrategias/pae_Accesibilidad/pae_normativa/pae_eInclusion_Normas_Accesibilidad.html#.WUSQTX6G000) (visited on 06/07/2017).
- [22] Phillip Jordan. *Mixing Objective-C, C++ and Objective-C++: an Updated Summary*. 2012. URL: <http://philjordan.eu/article/mixing-objective-c-c++-and-objective-c++> (visited on 06/19/2017).
- [23] *Crysis 3*. 2013. URL: <http://www2.ea.com/crysis-3> (visited on 06/17/2017).
- [24] Ananya Mandal. *What is visual impairment?* 2013. URL: <http://www.news-medical.net/health/What-is-visual-impairment.aspx> (visited on 06/15/2017).
- [25] Don Norman. *The Design of Everyday Things: Revised and Expanded Edition*. New York: Basic Books, 2013. ISBN: 978-0-465-05065-9.
- [26] Brian Schmidt. *Making Ear Monsters: Developing a 3D Audio Game*. 2013. URL: <http://www.gamasutra.com/blogs/BrianSchmidt/20130617/194489/Making%20Ear%20Monsters%20Developing%20Da%203D%20Audio%20Game.php> (visited on 06/15/2017).
- [27] Mads Soegaard and Rikke Friis Dam. *The Encyclopedia of Human-Computer Interaction, 2nd Ed*. The Interaction Design Foundation, 2013. URL: <https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed>.
- [28] *Gamasutra Salary Survey 2014*. 2014. URL: <http://www.gamasutra.com/salariesurvey2014.pdf> (visited on 06/11/2017).

- [29] *How Accessibility Works in GNOME*. 2014. URL: <https://developer.gnome.org/accessibility-devel-guide/stable/gad-how-it-works.html.en> (visited on 06/15/2017).
- [30] Samuel Thibault. *Where does accessibility plug into the graphical desktop stack?* 2014. URL: <https://www.youtube.com/watch?v=DxVrIVQKHEg> (visited on 05/11/2017).
- [31] *Welcome to Orca*. 2014. URL: <https://help.gnome.org/users/orca/stable/introduction.html.en> (visited on 06/19/2017).
- [32] Robert K. Yin. *Case Study Research: Design and Methods*. Thousand Oaks, California: SAGE Publications, Inc., 2014. ISBN: 978-1452242569.
- [33] Telmo Zarraonandia, Paloma Diaz, Ignacio Aedo, et al. “Designing educational games through a conceptual model based on rules and scenarios”. In: *Multimedia Tools and Applications* 74.13 (2014), pp. 4535–4559. ISSN: 15737721. DOI: 10.1007/s11042-013-1821-1.
- [34] *Accessibility Basics*. 2015. URL: <https://www.usability.gov/what-and-why/accessibility.html> (visited on 06/15/2017).
- [35] Oana Balan, Florica Moldoveanu, Alin Moldoveanu, et al. “Developing a navigational 3D audio game with hierarchical levels of difficulty for the visually impaired players”. In: *Romanian Conference on Human Computer Interaction*. September. Bucharest, 2015.
- [36] Edward C. Bell and Natalia M. Mino. “Employment Outcomes for Blind and Visually Impaired Adults”. In: *Journal of Blindness Innovation and Research* 5.2 (2015). DOI: 10.5241/5-85. URL: <https://nfb.org/images/nfb/publications/jbir/jbir15/jbir050202abs.html>.
- [37] *Developers Salary Guide: October 2015 (Barcelona)*. 2015. URL: <https://www.jobfluent.com/blog/developers-salary-guide-october-2015-barcelona> (visited on 06/11/2017).
- [38] *Introduction to User-Centered Design*. 2015. URL: <http://www.usabilityfirst.com/about-usability/introduction-to-user-centered-design/> (visited on 06/12/2017).
- [39] *Software Engineer Salary Guide 2014*. 2015. URL: <http://fundersandfounders.com/software-engineer-salary-2014/> (visited on 06/11/2017).
- [40] *WAI Guidelines and Techniques*. 2015. URL: <https://www.w3.org/WAI/guid-tech> (visited on 06/07/2017).
- [41] Panayiotis Zaphiris and Andri Ioannou. “Learning and Collaboration Technologies: Second International Conference, LCT 2015 Held as Part of HCI International 2015 Los Angeles, CA, USA, August 2015, 2015 Proceedings”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9192 (2015), pp. 537–548. ISSN: 16113349. DOI: 10.1007/978-3-319-20609-7. arXiv: arXiv:1011.1669v3.
- [42] *Accessibility: IAccessible2*. 2016. URL: <https://wiki.linuxfoundation.org/accessibility/iaaccessible2/start> (visited on 06/19/2017).

- [43] *Asociación de Minusválidos Pinto, Centro de Atención Temprana*. 2016. URL: <http://sid.usal.es/centrosyservicios/discapacidad/1632/2-1-2-2/asociacion-de-minusvalidos-pinto-centro-de-atencion-temprana-zona-pinto.aspx> (visited on 06/14/2017).
- [44] Gary P. Scavone. *The RtAudio Home Page*. 2016. URL: <https://www.music.mcgill.ca/~gary/rtaudio/> (visited on 06/19/2017).
- [45] *Help - Eclipse Platform*. 2016. URL: <https://help.eclipse.org/neon/index.jsp>.
- [46] *IVA - Impuesto de Valor Añadido 2016*. 2016. URL: <http://www.datosmacro.com/impuestos/iva> (visited on 06/11/2017).
- [47] Juan Linietsky. *Mozilla awards Godot Engine as part of the MOSS "Mission Partners" program*. 2016. URL: <https://godotengine.org/article/mozilla-awards-godot-engine-part-moss-mission-partners-program> (visited on 06/17/2017).
- [48] *libogg*. 2016. URL: <https://github.com/godotengine/godot/tree/master/thirdparty/libogg>.
- [49] *libvorbis*. 2016. URL: <https://github.com/godotengine/godot/tree/master/thirdparty/libvorbis>.
- [50] Newzoo. *2016 Global Games Market Report - An Overview Of Trends & Insights*. 2016. URL: [https://cdn2.hubspot.net/hubfs/700740/Reports/Newzoo\\_Free\\_2016\\_Global\\_Games\\_Market\\_Report.pdf](https://cdn2.hubspot.net/hubfs/700740/Reports/Newzoo_Free_2016_Global_Games_Market_Report.pdf).
- [51] *Opus Codec*. 2016. URL: <http://opus-codec.org/> (visited on 06/19/2017).
- [52] Parham Doustdar. *An Autobiography of a Blind Programmer*. 2016. URL: <https://www.parhamdoustdar.com/2016/03/27/autobiography-blind-programmer/> (visited on 05/28/2017).
- [53] *Prototypes, Specifications, and Diagrams in One Tool | Azure Software*. 2016. URL: <https://www.azure.com/> (visited on 06/18/2017).
- [54] *Real Median Household Income in the United States*. 2016. URL: <https://fred.stlouisfed.org/series/MEHOINUSA672N> (visited on 06/11/2017).
- [55] *rtaudio*. 2016. URL: <https://github.com/godotengine/godot/tree/master/thirdparty/rtaudio>.
- [56] Shai Abou-Zahra. *Voice Recognition - Web Accessibility Perspectives Videos*. 2016. URL: <https://www.w3.org/WAI/perspectives/voice.html> (visited on 06/15/2017).
- [57] *WCAG 2.0 Checklist*. 2016. URL: <http://webaim.org/standards/wcag/WCAG2Checklist.pdf>.
- [58] World Health Organization. *Disabilities*. 2016. URL: <http://www.who.int/topics/disabilities/en/> (visited on 05/04/2017).
- [59] *Xiph.org*. 2016. URL: <https://xiph.org/vorbis/> (visited on 06/19/2017).
- [60] *Xiph.org: Ogg*. 2016. URL: <https://www.xiph.org/ogg/> (visited on 06/19/2017).
- [61] *Abono 30 días - Tarifas*. 2017. URL: <http://www.crtm.es/billetes-y-tarifas/billetes-y-abonos/abono-transportes/abono-treinta-dias.aspx> (visited on 06/11/2017).

- [62] *Accessibility for Developers*. 2017. URL: <https://developer.apple.com/accessibility/> (visited on 06/15/2017).
- [63] *Accessibility in Visual Studio Code*. 2017. URL: <https://code.visualstudio.com/docs/editor/accessibility> (visited on 05/16/2017).
- [64] *Accessibility Programming Guide for OS X: The OS X Accessibility Model*. 2017. URL: [https://developer.apple.com/library/content/documentation/Accessibility/Conceptual/AccessibilityMacOSX/OSXAXmodel.html#//apple\\_ref/doc/uid/TP40001078-CH208-TPXREF101](https://developer.apple.com/library/content/documentation/Accessibility/Conceptual/AccessibilityMacOSX/OSXAXmodel.html#//apple_ref/doc/uid/TP40001078-CH208-TPXREF101) (visited on 06/18/2017).
- [65] Álvaro Cáceres. *MEGA*. 2017. URL: <https://mega.nz/#!cXo0nDgK>.
- [66] *Amazon Lumberyard - Free AAA Game Engine*. 2017. URL: <https://aws.amazon.com/lumberyard/> (visited on 06/17/2017).
- [67] *AppKit | Apple Developer Documentation*. 2017. URL: <https://developer.apple.com/documentation/appkit> (visited on 06/19/2017).
- [68] *AudioGames, your resource for audiogames, games for the blind, games for the visually impaired!* 2017. URL: <http://audiogames.net/> (visited on 06/14/2017).
- [69] *Buy Adobe Illustrator CC*. 2017. URL: <http://www.adobe.com/products/illustrator.html> (visited on 06/18/2017).
- [70] *CRYENGINE | The complete solution for next generation game development by Crytek*. 2017. URL: <https://www.cryengine.com/> (visited on 06/17/2017).
- [71] Maureen A. Duffy. *Low Vision and Legal Blindness Terms and Descriptions*. 2017. URL: <http://www.visionaware.org/info/your-eye-condition/eye-health/low-vision/low-vision-terms-and-descriptions/1235> (visited on 06/14/2017).
- [72] Entertainment Software Association. *Essential Facts About The Computer and Video Game Industry*. 2017. URL: [http://www.theesa.com/wp-content/uploads/2017/04/EF2017\\_FinalDigital.pdf](http://www.theesa.com/wp-content/uploads/2017/04/EF2017_FinalDigital.pdf).
- [73] *Free Images - Pixabay*. 2017. URL: <https://pixabay.com/> (visited on 06/14/2017).
- [74] *FreeLists / V.I. Programmers Discussion List*. 2017. URL: <https://www.freelists.org/list/program-1> (visited on 06/14/2017).
- [75] *GNOME – An easy and elegant way to use your computer, GNOME 3 is designed to put you in control and get things done*. 2017. URL: <https://www.gnome.org/> (visited on 06/19/2017).
- [76] *Interpreter (computing) for Kids*. 2017. URL: [http://kids.kiddle.co/Interpreter\\_\(computing\)](http://kids.kiddle.co/Interpreter_(computing)) (visited on 06/16/2017).
- [77] *JAWS Screen Reader - Best in Class*. 2017. URL: <http://www.freedomscientific.com/Products/Blindness/JAWS> (visited on 06/19/2017).
- [78] *Kinect for Xbox One*. 2017. URL: <http://www.xbox.com/en-US/xbox-one/accessories/kinect> (visited on 06/13/2017).
- [79] Juan Linietsky and Ariel Manzur. *Godot Engine - Free and open source 2D and 3D game engine*. 2017. URL: <https://godotengine.org/> (visited on 06/17/2017).
- [80] Michelle K. Martin. *Screenreader Accessible Unity Template*. 2017. URL: <https://github.com/frastlin/ScreenreaderAccessibleUnityTemplate>.

- [81] *Microsoft Active Accessibility (Windows)*. 2017. URL: [https://msdn.microsoft.com/en-us/library/windows/desktop/dd373592\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/dd373592(v=vs.85).aspx) (visited on 06/19/2017).
- [82] *NV Access*. 2017. URL: <https://www.nvaccess.org/> (visited on 06/19/2017).
- [83] *Oculus Rift*. 2017. URL: <https://www.oculus.com/rift/> (visited on 05/31/2017).
- [84] *opus*. 2017. URL: <https://github.com/godotengine/godot/tree/master/thirdparty/opus>.
- [85] *Real-time price of electricity*. 2017. URL: <https://www.endesaclientes.com/price-electricity-vpsc.html> (visited on 06/11/2017).
- [86] *Refreshable Braille Displays*. 2017. URL: <http://www.afb.org/prodBrowseCatResults.aspx?CatID=43> (visited on 05/11/2017).
- [87] Derek Riemer. *Notepad++ Add-on for NVDA* No Title. 2017. URL: <https://github.com/derekriemer/nvda-notepadPlusPlus> (visited on 05/16/2017).
- [88] *Salario: el sector de Investigador en España*. 2017. URL: <http://espana.jobtonic.es/salary/26526/74915.html> (visited on 06/11/2017).
- [89] Shawn Lawton. *WCAG Overview*. 2017. URL: <https://www.w3.org/WAI/intro/wcag> (visited on 06/19/2017).
- [90] *Store*. 2017. URL: <https://store.unity.com/> (visited on 06/07/2017).
- [91] *The GTK+ Project*. 2017. URL: <https://www.gtk.org/> (visited on 06/19/2017).
- [92] *The Quorum Programming Language*. 2017. URL: <https://quorumlanguage.com/> (visited on 06/16/2017).
- [93] *Unity - Fast Facts*. 2017. URL: <https://unity3d.com/public-relations> (visited on 05/31/2017).
- [94] *Unity - Game Engine*. 2017. URL: <https://unity3d.com/> (visited on 06/17/2017).
- [95] *Unreal® Engine End User License Agreement*. 2017. URL: <https://www.unrealengine.com/eula> (visited on 06/07/2017).
- [96] *User centred design*. 2017. URL: <http://www.userfocus.co.uk/consultancy/ucd.html> (visited on 06/12/2017).
- [97] *User-Centered Design Basics*. 2017. URL: <https://www.usability.gov/what-and-why/user-centered-design.html> (visited on 06/12/2017).
- [98] *Vision Accessibility*. 2017. URL: <https://www.apple.com/accessibility/mac/vision/> (visited on 06/19/2017).
- [99] *Vodafone One 50Mb S con 20% de descuento*. 2017. URL: <https://www.vodafone.es/tienda/particulares/es/one-todo-en-uno/fibra-ono-movil/?mostrarGE=true> (visited on 06/20/2017).
- [100] *What is free software?* 2017. URL: <https://www.gnu.org/philosophy/free-sw.en.html> (visited on 06/11/2017).
- [101] *What is Unreal Engine 4*. 2017. URL: <https://www.unrealengine.com/what-is-unreal-engine-4> (visited on 06/17/2017).
- [102] *BGT (Blastbay Game Toolkit)*. URL: <http://www.blastbay.com/bgt.php> (visited on 06/16/2017).

- [103] *Design for All is design tailored to human diversity*. URL: <http://designforall.org/design.php> (visited on 06/15/2017).
- [104] Emacspeak Inc. *Emacspeak –The Complete Audio Desktop*. URL: <http://emacspeak.sourceforge.net/> (visited on 05/11/2017).
- [105] Google. *Accessibility - Usability - Material design guidelines*. URL: <https://material.io/guidelines/usability/accessibility.html> (visited on 05/09/2017).
- [106] IDEO. *Human Centered Design*. URL: <http://www.designkit.org/human-centered-design> (visited on 05/24/2017).
- [107] *Limited License Agreement for the Use of The CryEngine*. URL: <https://www.cryengine.com/ce-terms> (visited on 06/07/2017).
- [108] Microsoft. *Accessibility Design Guidelines for Software*. URL: [https://msdn.microsoft.com/en-us/library/aa291308\(v=vs.71\).aspx](https://msdn.microsoft.com/en-us/library/aa291308(v=vs.71).aspx) (visited on 05/09/2017).



## A WCAG 2.0 checklist

### A.1 Perceivable

Web content is made available to the senses - sight, hearing, and/or touch

#### A.1.1 Guideline 1.1. Text Alternatives

Success criteria	Recommendations	Check
1.1.1 Non-text Content (Level A)	All images, form image buttons, and image map hot spots have appropriate, equivalent alternative text.	✓
	Images that do not convey content, are decorative, or with content that is already conveyed in text are given null alt text (alt="") or implemented as CSS backgrounds. All linked images have descriptive alternative text.	N/A
	Equivalent alternatives to complex images are provided in context or on a separate (linked and/or referenced via longdesc) page.	N/A
	Form buttons have a descriptive value.	✓
	Form inputs have associated text labels.	✓
	Embedded multimedia is identified via accessible text.	N/A
	Frames are appropriately titled.	✓

**Table 14:** Evaluation with the WCAG 2.0 checklist: guideline 1.1

## A.1.2 Guideline 1.2. Time-based media

Success criteria	Recommendations	Check
1.2.1 Prerecorded Audio-only and Video-only (Level A)	A descriptive text transcript (including all relevant visual and auditory clues and indicators) is provided for non-live, web-based audio (audio podcasts, MP3 files, etc.).	N/A
	A text or audio description is provided for non-live, web-based video-only (e.g., video that has no audio track).	N/A
1.2.2 Captions (Prerecorded) (Level A)	Synchronized captions are provided for non-live, web-based video (YouTube videos, etc.)	N/A
1.2.3 Audio Description or Media Alternative (Prerecorded) (Level A)	A descriptive text transcript OR audio description audio track is provided for non-live, web-based video	N/A
1.2.4 Captions (Live) (Level AA)	Synchronized captions are provided for all live multimedia that contains audio (audio-only broadcasts, web casts, video conferences, Flash animations, etc.)	N/A
1.2.5 Audio Description (Prerecorded) (Level AA)	Audio descriptions are provided for all video content. NOTE: Only required if the video conveys content visually that is not available in the default audio track.	N/A
1.2.6 Sign Language (Prerecorded) (Level AAA)	A sign language video is provided for all media content that contains audio.	N/A
1.2.7 Extended Audio Description (Prerecorded) (Level AAA)	When an audio description track cannot be added to video due to audio timing (e.g., no pauses in the audio), an alternative version of the video with pauses that allow audio descriptions is provided.	N/A
1.2.8 Media Alternative (Prerecorded) (Level AAA)	A descriptive text transcript is provided for all pre-recorded media that has a video track.	N/A

Table 15: Evaluation with the WCAG 2.0 checklist: guideline 1.2 (1)

Success criteria	Recommendations	Check
1.2.9 Audio-only (Live) (Level AAA)	A descriptive text transcript (e.g., the script of the live audio) is provided for all live content that has audio.	N/A

**Table 16:** Evaluation with the WCAG 2.0 checklist: guideline 1.2 (2)

### A.1.3 Guideline 1.3: Adaptable

Success criteria	Recommendations	Check
1.3.1 Info and Relationships (Level A)	Semantic markup is used to designate headings (<h1>), lists (<ul>, <ol>, and <dl>), emphasized or special text (<strong>, <code>, <abbr>, <blockquote>, for example), etc. Semantic markup is used appropriately.	N/A
	Tables are used for tabular data. Headings, where necessary, are used to associate data cells with headers. Data table captions and summaries are used where appropriate.	✓
	Text labels are associated with form input elements. Related form elements are grouped with fieldset/legend.	✓
1.3.2 Meaningful Sequence (Level A)	The reading and navigation order (determined by code order) is logical and intuitive.	✓
1.3.3 Sensory Characteristics (Level A)	Instructions do not rely upon shape, size, or visual location (e.g., "Click the square icon to continue" or "Instructions are in the right-hand column").	✓
	Instructions do not rely upon sound (e.g., "A beeping sound indicates you may continue.>").	✗

**Table 17:** Evaluation with the WCAG 2.0 checklist: guideline 1.3

## A.1.4 Guideline 1.4: Distinguishable

Success criteria	Recommendations	Check
1.4.1 Use of Color (Level A)	Color is not used as the sole method of conveying content or distinguishing visual elements.	✓
	Color alone is not used to distinguish links from surrounding text unless the luminance contrast between the link and the surrounding text is at least 3:1 and an additional differentiation (e.g., it becomes underlined) is provided when the link is hovered over or receives focus.	✓
1.4.2 Audio Control (Level A)	A mechanism is provided to stop, pause, mute, or adjust volume for audio that automatically plays on a page for more than 3 seconds.	✓
1.4.3 Contrast (Minimum) (Level AA)	Text and images of text have a contrast ratio of at least 4.5:1.	✓
	Large text - at least 18 point (typically 24px) or 14 point (typically 18.66px) bold has a contrast ratio of at least 3:1.	✓
1.4.4 Resize text (Level AA)	The page is readable and functional when the text size is doubled.	✓
1.4.5 Images of Text (Level AA)	If the same visual presentation can be made using text alone, an image is not used to present that text.	✓
1.4.6 Contrast (Enhanced)	Text and images of text have a contrast ratio of at least 7:1.	✓
	Large text - at least 18 point (typically 24px) or 14 point (typically 18.66px) bold has a contrast ratio of at least 4.5:1.	✓
1.4.7 Low or No Background Audio (Level AAA)	Audio of speech has no or very low background noise so the speech is easily distinguished.	✓

Table 18: Evaluation with the WCAG 2.0 checklist: guideline 1.4 (1)

Success criteria	Recommendations	Check
	Blocks of text over one sentence in length:	
1.4.8 Visual Presentation (Level AAA)	Are no more than 80 characters wide.	✓
	Are NOT fully justified (aligned to both the left and the right margins).	✓
	Have adequate line spacing (at least 1/2 the height of the text) and paragraph spacing (1.5 times line spacing).	✓
	Have a specified foreground and background color. These can be applied to specific elements or to the page as a whole using CSS (and thus inherited by all other elements).	✓
	Do NOT require horizontal scrolling when the text size is doubled.	✓
1.4.9 Images of Text (No Exception) (Level AAA)	Text is used within an image only for decoration (image does not convey content) OR when the information cannot be presented with text alone.	✓

**Table 19:** Evaluation with the WCAG 2.0 checklist: guideline 1.4 (2)

## A.2 Operable

Interface forms, controls, and navigation are operable

## A.2.1 Guideline 2.1: Keyboard Accessible

Success criteria	Recommendations	Check
2.1.1 Keyboard (Level A)	All page functionality is available using the keyboard, unless the functionality cannot be accomplished in any known way using a keyboard (e.g., free hand drawing).	✓
	Page-specified shortcut keys and accesskeys (accesskey should typically be avoided) do not conflict with existing browser and screen reader shortcuts.	✓
2.1.2 No Keyboard Trap (Level A)	Keyboard focus is never locked or trapped at one particular page element. The user can navigate to and from all navigable page elements.	✓
2.1.3 Keyboard (No Exception) (Level AAA)	All page functionality is available using the keyboard.	✓

Table 20: Evaluation with the WCAG 2.0 checklist: guideline 2.1

### A.2.2 Guideline 2.2: Enough Time

Success criteria	Recommendations	Check
2.2.1 Timing Adjustable (Level A)	If a page or application has a time limit, the user is given options to turn off, adjust, or extend that time limit. This is not a requirement for real-time events (e.g., an auction), where the time limit is absolutely required, or if the time limit is longer than 20 hours.	N/A
2.2.2 Pause, Stop, Hide (Level A)	Automatically moving, blinking, or scrolling content that lasts longer than 5 seconds can be paused, stopped, or hidden by the user. Moving, blinking, or scrolling can be used to draw attention to or highlight content as long as it lasts less than 5 seconds.  Automatically updating content (e.g., automatically redirecting or refreshing a page, a news ticker, AJAX updated field, a notification alert, etc.) can be paused, stopped, or hidden by the user or the user can manually control the timing of the updates.	N/A  ✓
2.2.3 No Timing (Level AAA)	The content and functionality has no time limits or constraints.	✓
2.2.4 Interruptions (Level AAA)	Interruptions (alerts, page updates, etc.) can be postponed or suppressed by the user.	✓
2.2.5 Reauthenticating (Level AAA)	If an authentication session expires, the user can re-authenticate and continue the activity without losing any data from the current page.	N/A

**Table 21:** Evaluation with the WCAG 2.0 checklist: guideline 2.2

### A.2.3 Guideline 2.3: Seizures

Success criteria	Recommendations	Check
2.3.1 Three Flashes or Below Threshold (Level A)	No page content flashes more than 3 times per second unless that flashing content is sufficiently small and the flashes are of low contrast and do not contain too much red.	✓
2.3.2 Three Flashes (Level AAA)	No page content flashes more than 3 times per second.	✓

**Table 22:** Evaluation with the WCAG 2.0 checklist: guideline 2.3

### A.2.4 Guideline 2.4: Navigable

Success criteria	Recommendations	Check
2.4.1 Bypass Blocks (Level A)	A link is provided to skip navigation and other page elements that are repeated across web pages.	✓
	If a page has a proper heading structure, this may be considered a sufficient technique instead of a "Skip to main content" link. Note that navigating by headings is not yet supported in all browsers.	✓
	If a page uses frames and the frames are appropriately titled, this is a sufficient technique for bypassing individual frames.	✓
2.4.2 Page Titled (Level A)	The web page has a descriptive and informative page title.	✓
2.4.3 Focus Order (Level A)	The navigation order of links, form elements, etc. is logical and intuitive.	✓
2.4.4 Link Purpose (In Context) (Level AA)	The purpose of each link (or form image button or image map hotspot) can be determined from the link text alone, or from the link text and it's context (e.g., surrounding paragraph, list item, table cell, or table headers).	✓
	Links (or form image buttons) with the same text that go to different locations are readily distinguishable.	✓
2.4.5 Multiple Ways (Level AA)	Multiple ways are available to find other web pages on the site - at least two of: a list of related pages, table of contents, site map, site search, or list of all available web pages.	N/A
2.4.6 Headings and Labels (Level AA)	Page headings and labels for form and interactive controls are informative. Avoid duplicating heading (e.g., "More Details") or label text (e.g., "First Name") unless the structure provides adequate differentiation between them.	✓
2.4.7 Focus Visible (Level AA)	It is visually apparent which page element has the current keyboard focus (i.e., as you tab through the page, you can see where you are).	✓
2.4.8 Location (Level AAA)	If a web page is part of a sequence of pages or within a complex site structure, an indication of the current page location is provided, for example, through breadcrumbs or specifying the current step in a sequence (e.g., "Step 2 of 5 - Shipping Address").	✓

Table 23: Evaluation with the WCAG 2.0 checklist: guideline 2.4 (1)



Success criteria	Recommendations	Check
2.4.9 Link Purpose (Link Only) (Level AAA)	The purpose of each link (or form image button or image map hotspot) can be determined from the link text alone.  There are no links (or form image buttons) with the same text that go to different locations.	✓  ✓
2.4.10 Section Headings (Level AAA)	Beyond providing an overall document structure, individual sections of content are designated using headings, where appropriate.	✓

**Table 24:** Evaluation with the WCAG 2.0 checklist: guideline 2.4 (2)

### A.3 Understandable

Content and interface are understandable

#### A.3.1 Guideline 3.1: Readable

Success criteria	Recommendations	Check
3.1.1 Language of Page (Level A)	The language of the page is identified using the HTML lang attribute ( <code>&lt;html lang="en"&gt;</code> , for example).	N/A
3.1.2 Language of Parts (Level AA)	When appropriate, the language of sections of content that are a different language are identified, for example, by using the lang attribute ( <code>&lt;blockquote lang="es"&gt;</code> )	N/A
3.1.3 Unusual Words (Level AAA)	Words that may be ambiguous, unknown, or used in a very specific way are defined through adjacent text, a definition list, a glossary, or other suitable method.	✓
3.1.4 Abbreviations (Level AAA)	Expansions for abbreviations are provided by expanding or explaining the definition the first time it is used, using the <code>&lt;abbr&gt;</code> element, or linking to a definition or glossary. NOTE: WCAG 2.0 gives no exception for regularly understood abbreviations (e.g., "HTML" on a web design site must always be expanded).	✓
3.1.5 Reading Level (Level AAA)	A more understandable alternative is provided for content that is more advanced than can be reasonably read by a person with roughly 9 years of primary education.	✓
3.1.6 Pronunciation (Level AAA)	If the pronunciation of a word is vital to understanding that word, its pronunciation is provided immediately following the word or via a link or glossary.	✓

**Table 25:** Evaluation with the WCAG 2.0 checklist: guideline 3.1

## A.3.2 Guideline 3.2: Predictable

Success criteria	Recommendations	Check
3.2.1 On Focus (Level A)	When a page element receives focus, it does not result in a substantial change to the page, the spawning of a pop-up window, an additional change of keyboard focus, or any other change that could confuse or disorient the user.	✓
3.2.2 On Input (Level A)	When a user inputs information or interacts with a control, it does not result in a substantial change to the page, the spawning of a pop-up window, an additional change of keyboard focus, or any other change that could confuse or disorient the user unless the user is informed of the change ahead of time.	✓
3.2.3 Consistent Navigation (Level AA)	Navigation links that are repeated on web pages do not change order when navigating through the site.	N/A
3.2.4 Consistent Identification (Level AA)	Elements that have the same functionality across multiple web pages are consistently identified. For example, a search box at the top of the site should always be labeled the same way.	✓
3.2.5 Change on Request (Level AAA)	Substantial changes to the page, the spawning of pop-up windows, uncontrolled changes of keyboard focus, or any other change that could confuse or disorient the user must be initiated by the user. Alternatively, the user is provided an option to disable such changes.	✓

Table 26: Evaluation with the WCAG 2.0 checklist: guideline 3.2

### A.3.3 Guideline 3.3: Input Assistance

Success criteria	Recommendations	Check
3.3.1 Error Identification (Level A)	Required form elements or form elements that require a specific format, value, or length provide this information within the element's label.	✓
	If utilized, form validation cues and errors (client-side or server-side) alert users to errors in an efficient, intuitive, and accessible manner. The error is clearly identified, quick access to the problematic element is provided, and user is allowed to easily fix the error and resubmit the form.	✓
3.3.2 Labels or Instructions (Level A)	Sufficient labels, cues, and instructions for required interactive elements are provided via instructions, examples, properly positioned form labels, and/or fieldsets/legends.	✓
3.3.3 Error Suggestion (Level AA)	If an input error is detected (via client-side or server-side validation), provide suggestions for fixing the input in a timely and accessible manner.	✓
3.3.4 Error Prevention (Legal, Financial, Data) (Level AA)	If the user can change or delete legal, financial, or test data, the changes/deletions are reversible, verified, or confirmed.	N/A
3.3.5 Help (Level AAA)	Provide instructions and cues in context to help in form completion and submission.	✗
3.3.6 Error Prevention (All) (Level AAA)	If the user can submit information, the submission is reversible, verified, or confirmed.	✓

**Table 27:** Evaluation with the WCAG 2.0 checklist: guideline 3.3

## A.4 Robust

Content can be used reliably by a wide variety of user agents, including assistive technologies

**A.4.1 Guideline 4.1: Compatible**

Success criteria	Recommendations	Check
4.1.1 Parsing (Level A)	Significant HTML/XHTML validation/parsing errors are avoided.	✓
4.1.2 Name, Role, Value (Level A)	Markup is used in a way that facilitates accessibility. This includes following the HTML/XHTML specifications and using forms, form labels, frame titles, etc. appropriately.	✓

**Table 28:** Evaluation with the WCAG 2.0 checklist: guideline 4.1

Results: 61 2 63