SPECIFICATION AND PROTOTYPE IMPLEMENTATION OF AN OPEN SOURCE INTERACTIVE AMBIENT APPLICATION

October 2013
Fabrizio Fornari
Master of Science in Computer Science
SPECIFICATION AND PROTOTYPE IMPLEMENTATION OF AN OPEN SOURCE INTERACTIVE AMBIENT APPLICATION

Fabrizio Fornari
Master of Science
Computer Science
October 2013
School of Computer Science
Reykjavík University
University of Camerino

M.Sc. RESEARCH THESIS

ISSN 1670-8539
Specification and Prototype Implementation of an Open Source Interactive Ambient Application

by

Fabrizio Fornari

Research thesis submitted to the School of Computer Science at Reykjavík University and at the University of Camerino in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science from Reykjavík University and
Master of Science in Computer Science from University of Camerino

October 2013

Research Thesis Committee:

Kristinn R. Thórisson, Supervisor
Associate Professor, Reykjavík University

Luca Tesei, Supervisor
Associate Professor, University of Camerino

Hannes Högni Vilhjálmsson
Associate Professor, Reykjavík University

Marta Kristín Lárusdóttir
Assistant Professor, Reykjavík University
Copyright
Fabrizio Fornari
October 2013
The undersigned hereby certify that they recommend to the School of Computer Science at Reykjavík University for acceptance this research thesis entitled **Specification and Prototype Implementation of an Open Source Interactive Ambient Application** submitted by **Fabrizio Fornari** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Science**.

________________________________________

Date

________________________________________

Kristinn R. Thórísson, Supervisor
Associate Professor, Reykjavík University

________________________________________

Luca Tesei, Supervisor
Associate Professor, University of Camerino

________________________________________

Hannes Högni Vilhjálmsson
Associate Professor, Reykjavík University

________________________________________

Marta Kristín Lárusdóttir
Assistant Professor, Reykjavík University
The undersigned hereby grants permission to the Reykjavík University Library to reproduce single copies of this research thesis entitled *Specification and Prototype Implementation of an Open Source Interactive Ambient Application* and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the research thesis, and except as herein before provided, neither the research thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatsoever without the author’s prior written permission.

_________________________
Date

_________________________
Fabrizio Fornari
Master of Science
Abstract

LivingShadows is an ongoing art technology project run by the Icelandic Institute for Intelligent Machines (IIIM). The goal of the project is to develop new ways to connect distant parts of the world and bring people together through the use of digital technology. LivingShadows creates an ambient interaction space that, rather than being explicitly presented to its users and requiring their undivided attention, exists unobtrusively in their surroundings over months and even years, connecting remote places via interactive shadows. Our contribution to the project consisted first in verifying that the hardware and software we were using for the prototype were suitable to reach the largest set of developers that may decide to join the LivingShadows community. Then we defined how to manage the passers-by’s representation and its interactions inside a virtual environment populated by virtual objects and virtual creatures. We finally focused on the networking part of the project which consists in sharing the virtual world and the passers-by’s representation across the network. To prove our results, we developed some applications. In this document we describe aspects that may lead the LivingShadows project to be a starting point for the creation of a community of developers.
To my parents and my brother.
Acknowledgements

I want to thank my supervisor Kristinn R. Thórisson that has been a source of inspiration along this year spent in Reykjavík University. I thank also Hannes Hógni Vilhjálmsson that assisted me during the final part of my studies and Marta Kristín Lárusdóttir for the suggestions she gave me to improve this thesis. I thank also Luca Tesei, my supervisor from Unicam.

A special thank to Angelo Cafaro with whom I spent the last months sharing ideas opinions problems and solutions, speaking with him it has been a real source of inspiration. Moreover I thank the Erasmus programme that helped me to face the expenses for living in Iceland. thank to the University of Camerino, the University of Reykjavík, my colleagues, the ESN organization and all the new friends that I met during the experience of the Double Degree.

Last but not least a big thank to my family which supported my stay in Iceland and contributed with its affection to make me become the person I am.
Contents

List of Figures xi
List of Tables xiii

1 Introduction 1
   1.1 Brief Elucidation .................................................. 2
   1.2 Motivation ............................................................ 4
       1.2.1 The Shadow ....................................................... 5
       1.2.2 Virtual Reality .................................................. 6
       1.2.3 Augmented Reality .............................................. 6
       1.2.4 Spatial Augmented Reality ................................. 7

2 Project Technologies 9
   2.1 Hardware Requirements ............................................. 9
       2.1.1 3D sensors ..................................................... 10
       2.1.2 Computer ....................................................... 13
       2.1.3 Projector ....................................................... 13
   2.2 Software .......................................................... 15
       2.2.1 Unity3D ......................................................... 15

3 Implementation 19
   3.1 Design and Applications Programming ......................... 19
       3.1.1 Basic scene ................................................... 19
   3.2 Models and Animation creation .................................. 22
   3.3 Object-Shadow Interaction ....................................... 23
       3.3.1 Interaction and Proportion Details ....................... 23
   3.4 Other ways to identify collisions .............................. 26
       3.4.1 Character 3D model .......................................... 26
       3.4.2 Edge detection ............................................... 29
3.5 Networking ................................................. 30
  3.5.1 Client .................................................. 31
  3.5.2 Server .................................................. 33

4 Applications ............................................... 37
  4.1 Burst Them All .......................................... 37
  4.2 Bouncing Ball .......................................... 38
  4.3 Client-Server .......................................... 40
  4.4 Lag ....................................................... 40

5 Related Works ............................................. 43
  5.1 Culture Walls ........................................... 43
  5.2 ShadowTalk ............................................. 44
  5.3 Scott Snibbe's art works .............................. 45
  5.4 Myron Krueger and Videoplace ..................... 46
  5.5 Passages ............................................... 48
  5.6 OneSpace .............................................. 49

6 Conclusions ................................................ 51
  6.1 Future Works ........................................... 52

References .................................................... 55

A 3D Sensor .................................................. 57
  A.1 Depth Camera .......................................... 57

B Noise Reduction ............................................ 61
List of Figures

1.1 LivingShadows logo. ............................................. 1
1.2 Passers-by representation. ................................. 2
1.3 Passers-by with their projected shadows and a virtual tree. ...................... 2

2.1 LivingShadows installation prototype. ......................... 10
2.2 Apple Mac mini. ............................................... 13
2.3 NEC U260W Ultra-Short-Throw projector. ...................... 13
2.4 Issues with front-projection. ................................. 14
2.5 Unity3D interface. ............................................ 16

3.1 Basic scene with 3D creatures and passer-by’s shadow. ..................... 20
3.2 Camera view. .................................................... 21
3.3 Gnome 3D model with Blender. ................................ 22
3.4 Motion capture with Ni-Mate plugin for Blender. ............................. 23
3.5 Unity3D default colliders. ..................................... 24
3.6 Capsule collider applied to a 3D model. ................................ 25
3.7 3D models. ........................................................ 26
3.8 Skeleton generated from the passer-by approaching installation. ....... 27
3.9 Skeleton generated from the passer-by walking through the installation. .... 28
3.10 Skeleton generated from the passer-by facing the 3D sensor. ............... 28
3.11 Passer-by silhouette represented using cubes. ............................ 29
3.12 Sphere inside the silhouette. .................................... 29
3.13 Client-Server Architecture. .................................... 30
3.14 Client Menú. .................................................... 31
3.15 Client flow-chart. .............................................. 32
3.16 Server Menú. .................................................... 33
3.17 ServerConnection flow-chart. .................................. 34
3.18 ServerInteractionHandler flow-chart. ................................ 36

4.1 Burst Them All. .................................................. 38
4.2 Bouncing Ball  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
4.3 Client-Server with two clients  . . . . . . . . . . . . . . . . . . . . . . . . 40
5.1 CultureWalls  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
5.2 Draft of the ShadowTalk installation  . . . . . . . . . . . . . . . . . . . . 44
5.3 Compliant Bodies distort a soft rectangle of light  . . . . . . . . . . . . . 45
5.4 Impression A screen that holds an imprint, like clay  . . . . . . . . . . . . 45
5.5 Deep Walls A cabinet of cinematic memories  . . . . . . . . . . . . . . . 46
5.6 Visceral Cinema Chien Interactive reinterpretation of surrealist film  . . . 46
5.7 Shadow Bag Shadows unpredictably chase, follow and collapse  . . . . . . 46
5.8 VIDEPLACE Users and creatures  . . . . . . . . . . . . . . . . . . . . . . 47
5.9 VIDEPLACE User and sphere  . . . . . . . . . . . . . . . . . . . . . . . 47
5.10 Passages A very close proximity media space  . . . . . . . . . . . . . . 48
5.11 OneSpace A video conferencing system that creates a depth mirror  . . . 49
5.12 LivingShadows Passer-by silhouette, virtual object sphere and real object chair  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
A.1 3D Sensor schema  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
A.2 Depth Camera Schema  . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
A.3 Infrared light pattern  . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
A.4 Extrapolated user’s silhouette  . . . . . . . . . . . . . . . . . . . . . . . 59
B.1 Infrared Reflection caused by a reflective object  . . . . . . . . . . . . . 62
List of Tables

2.1 Specifics of four 3D sensors. .............................................. 11
Chapter 1

Introduction

The LivingShadows project consists in a platform hardware and software that will allow people from all over the world to communicate using their own "shadows", setting up a new form of communication that is different from, for instance, a skype conference but with the same purpose of connecting people. The project is the result of work by Gunnar Steinn Valgardsson, Hrafn Th. Thórisson and Kristinn R. Thórisson at Reykjavik University and IIIM. They were also responsible for the realization of the first LivingShadows prototype. The work described in this thesis extends the project by (a) evaluating possible implementation, including choice of hardware and software for realizing various possible applications, (b) defining a way to handle interactions between virtual objects and the passers-by’s representation (shadows), and (c) developing a networked version so that two or more sites running a shared LivingShadows could be hooked together.

Figure 1.1: LivingShadows logo.
1.1 Brief Elucidation

The LivingShadows project is in its very early stages. One hub has been set up in a public hallway at Reykjavík University, with over 100 people walking past every day. Development of the shadow technology and AI techniques are also in the beginning stages, representing an opportunity for students who are interested in the project.

In a typical scenario, the shadows may come from people walking past a particular location in their home town, or from someone else in a far-away city, on the other side of the planet. In addition to shadows cast by real people, the space is also inhabited by shadows cast by invisible (virtual) creatures, who primarily exist in the onlookers’ imagination. Mixing art, design, technology, artificial intelligence and computer graphics, the system allows shadows to interact in interesting ways, and presents users with an experience that could only be mediated by modern technology.
This project also goes under the category of Ambient Intelligence (AmI) which refers to electronic environments that are sensitive and responsive to the presence of people. AmI makes use of electronic devices that operate in the background and communicate to each other reaching the capability of perceiving human presence and modifying the information provided to the user according to his needs.

An example of an environment enriched with Ambient Intelligence is a "smart home". Several artifacts and items in a house can be enriched with sensors to gather information about their use. In addition to investigating intelligent devices in a home, an example of Ambient Intelligence is allowing the home itself to possess intelligence and make decisions regarding its state and interactions with its residents.

Our project may be seen as part of a smart home or a building which attributes to walls features that they normally don’t have.
1.2 Motivation

The LivingShadows project aims to mix art and technology to provide an unique experience to passers-by walking through the installation, giving them the possibility to interact with something unexpected, something that they are not used to interacting with: their own shadow. In fact, we want to give the possibility for passers-by to "play" with their own shadows and challenge possible future developers to make the interaction as much realistic and entertaining as possible.

We also want to provide the possibility to create characters to animate the projected world and make the shadow interact with them creating effects that may induce people to interact with the platform. The shadow would be like an avatar in the sense that it can interact with other objects and characters in the virtual world and maybe in the future it may integrate some sort of artificial intelligence to make it move by itself. In this thesis we: (a) worked on the development of the interaction between shadows and virtual objects exploring different solutions. Moreover, the project aims to let people share this virtual environment across the network, and in this respect the contribution of this thesis work includes (b) exploring a way to introduce networking into the project, and describing an example of a Client-Server architecture implementation.

Another key aspect is the creation of a community of people working on the LivingShadows project that may exchange ideas, effects and creatures on the project’s website in such a way to have a living community that provides ideas and attracts more people inside the project. In this context (c) we contributed to ensure that the hardware and software we were using for the prototype were suitable to reach the largest set of developers that may join the LivingShadows community: in terms of hardware this consisted in exploring the various 3D sensors that are now on the market and in terms of software it consisted in using open-source or free software. Moreover (d) our work contributes to form an useful documentation for every future developers which may join the project.

Once the project will reach an optimal state, we may try to understand how much it is immersive. For example: if it happens an interaction between the passer-by’s shadow and a creature or an object in the virtual world, what does the passer-by feel? Does he feel that he has some control of the virtual environment or not?
1.3 The Shadow

Even if a shadow doesn’t have a proper consistency and then it doesn’t allow effective interactions with the world, in the history shadows have been used for specific purposes. For example, shadows have been used for sundials - devices that tell the time using an object’s shadow projected by the Sun; this was one of the methods used to measure the flow of time before the advent of modern watches. Shadows have also held an important role in art: Shadowgraphy or ombromanie is in fact the art of performing a story or show using images made by hand shadows. Albert Almoznino (2002), gave us an introduction to Shadowgraphy and some examples of hand shadows that everyone can do with some practice. A similar use of the shadows have been done in shadow play (or shadow puppetry) - an ancient form of storytelling and entertainment which uses flat articulated figures (shadow puppets) to create the impression of moving humans and other three-dimensional objects (further information about shadow puppets can be retrieved on puppetindia.com).

The shadow is also a synonymous of anonymity. Acting in the shadow can be seen as an advantage: it is the equivalent of sneaking avoiding exposure and remaining in a state of safety. Sometimes it is also used to overcome shyness: it is easier to express our own opinion anonymously rather than expose ourselves in first person to people’s judgement. Moreover in psychology the concept of shadow has been associated with negative characteristics. Especially for C.G.Jung (1959) the shadow is an archetype in more detail "The shadow personifies everything that the subject refuses to acknowledge about himself and yet is always upon him directly or indirectly-for instance, inferior traits of character and other incompatible tendencies". The shadow, then is composed by the collection of all our uncontrollable instincts that we try to repress every day, but still are always with us. Those attributions to the shadow can be noticed in the literature for instance in "The strange case of Dr.Jekyll and Mr.Hyde" Stevenson R. L. (1895) as well as in the theatre which incorporate the shadow archetype in operas like "Goethe’s Faust" Goethe (1808).

Nowadays with the advent of technology the shadow is back, except for rare cases (some remaining of what we cited before), to play a marginal role in the everyday life. With our project we want to bring the shadow to a positive meaning: our shadow will play a fundamental role representing a new form of interactive communication with users and developers from all over the world.

1 *archetype, (from Greek archetypos, "original pattern"), in literary criticism, a primordial image, character, or pattern of circumstances that recurs throughout literature and thought consistently enough to be considered a universal concept or situation. Definition from Enciclopedia Britannica www.britannica.com
1.4 Virtual Reality

A comprehensive definition of Virtual Reality may be found on Wikipedia: "Virtual reality (VR) is a term that applies to computer-simulated environments that can simulate physical presence in places in the real world, as well as in imaginary worlds. Furthermore, virtual reality covers remote communication environments which provide virtual presence of users with the concepts of telepresence and telexistence or a virtual artifact (VA) either through the use of standard input devices such as a keyboard and mouse, or through multimodal devices such as a wired glove and the use of head- or body-attached displays. The simulated environment can be similar to the real world in order to create a lifelike experience for example, in simulations for pilot or combat training or it can differ significantly from reality, such as in VR games".

In Steuer J. (1992) the author gives us a comprehensive definition of virtual reality quoting several experts on the field and analysing which are the factors that bring to telepresence: vividness and interactivity.

Steuer then classified various media technologies coming from the reality and the world of science-fiction. According to those factors he rated the Bradbury’s Nursery from the Veldt - a short story we can find in Ray Bradbury (1950) - at the highest level of vividness and interactivity. The Bradbury’s Nursery consists in a virtual reality room which gives to the children the possibility to virtually represent their thoughts and reproduce their imaginary world in a so vivid way that somehow the world becomes real.

We took knowledge of the Steuer’s work and we decided to try to develop our project showing as much vividness and interactivity as possible.

1.5 Augmented Reality

Our project LivingShadows doesn’t comprehend only the concept of virtual reality but it actually uses virtual reality to enhance the real world and this goes under the definition of Augmented Reality. Bimber O. et al. (2005) about Augmented Reality (AR): "In contrast to traditional VR, in AR the real environment is not completely suppressed; instead it plays a dominant role. Rather than immersing a person into a completely synthetic world, AR attempts to embed synthetic supplements into the real environment (or into a live video of the real environment)." Azuma R. et al. (2001) define some properties of AR:

- combines real and virtual objects in real environment;
• runs interactively and in real time;
• registers (aligns) real and virtual objects with each other
• can potentially apply to all senses, including hearing, touch, and smell.

To summarize we can say that Augmented Reality increases the information normally perceived by an user adding graphics, sounds, haptic feedback and smell to the natural world. Everyone from kids to grown adults may, for example, benefit from the ability to place computer-generated graphics in their field of vision. One classical example of AR is the use of a device to literally see directions aligned on the real world.

1.6 Spatial Augmented Reality

The use of a projector to display the virtual environment on a real wall, in our case, makes us actually refer to a particular AR called Spatial Augmented Reality (SAR). SAR consists in the use of technological instruments to increase the expressive power of the real world transforming it in such a way to fuse together what is virtual and what is real; it differs from Augmented Reality because it doesn’t require any extra display device, such as monitors, head mounted displays, etc. but it makes use of real world objects like workbenches or walls as display screens.

As Bimber O. et al. (2005) reported: "...spatial displays offer many advantages and solve several problems that are related to visual quality (e.g., resolution, field-of-view, focus, etc.), technical issues (e.g., tracking, lighting, etc.), and human factors (e.g., cumbersome-ness, etc.), but they are limited to non-mobile applications...Due to the decrease in cost and availability of projection technology, personal computers, and graphics hardware, there has been a considerable interest in exploiting SAR systems in universities, research laboratories, museums, industry, and the art community".

For the reasons that Bimber reported, we decided to proceed with the implementation of our Spatial Augmented Reality.
Chapter 2

Project Technologies

The LivingShadows project is based on a combined use of components such as: a 3D sensor, a projector, a computer and a display surface (wall). The prototype requires a space of approximate $5m^2$ and good wiring since we need to access the network and to maintain the ability to add other devices like a camera that may be used for recording reactions of passers-by in order to carry out some future studies. Moreover the location of the equipment shall be such that the user’s attention is not drawn to the instruments rather than to the projection.

2.1 Hardware Requirements

The original idea for this project required the use of:

- a 3D sensor.
- a powerful computer capable of running graphics applications with a good frame rate.
- a Ultra-Short-Throw Projector.

Because of our installation (Figure 2.1) is a prototype, we don’t have particular restriction about the hardware components to use but it’s important to maintain at least the following requirements to not reduce the performances:

- A 3D sensor not less powerful than the XBOX 360 Kinect.
- A good computer with graphics capabilities such as to allow a smooth execution of graphics applications built with Unity3D.
An ultra-short projector that allows the user to reach the projection as closest as possible without that his real shadow interfere.

In the following subsections we focus in more detail on each of those components.

### 2.1.1 3D sensors

We looked at the 3D sensors that are now on the market and we collected here some information we found on the internet to compare them and understand which one could be more suitable for our purposes. We took in example four 3D sensors: XBOX 360 Kinect, Windows Kinect, Asus Xtion PRO Live and Carmine Prime Sense.
Below, we synthesize their major characteristics:

<table>
<thead>
<tr>
<th></th>
<th>XBOX 360 Kinect</th>
<th>Windows Kinect</th>
<th>Asus Xtion PRO Live</th>
<th>Carmine 1.08 PrimeSense</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Field of View</strong></td>
<td>57°H, 43°V, 27 to +27 degree tilt up/down</td>
<td>57.5°H, 43.5°V, 27 to +27 degree tilt up/down</td>
<td>58° H, 45°V, 70° D</td>
<td>57.5°H, 45°V, 69°D</td>
</tr>
<tr>
<td><strong>Range</strong></td>
<td>1.2m - 3.5m</td>
<td>0.8m - 3.5m</td>
<td>0.8m - 3.5m</td>
<td>0.8m - 3.5m</td>
</tr>
<tr>
<td><strong>Sensor</strong></td>
<td>RGB &amp; Depth &amp; Microphone*4, 3-axis accelerometer</td>
<td>RGB &amp; Depth &amp; Microphone*4, 3-axis accelerometer</td>
<td>RGB &amp; Depth &amp; Microphone*2, 3-axis accelerometer</td>
<td>RGB &amp; Depth &amp; Microphone*2, 3-axis accelerometer</td>
</tr>
<tr>
<td><strong>Depth Image</strong></td>
<td>VGA (640x480): 30 fps, QVGA (320x240): 30 fps</td>
<td>640x480, 320x240, 80x40: 30 fps</td>
<td>VGA (640x480): 30 fps, QVGA (320x240): 60 fps</td>
<td>VQA (640x480): 30 fps</td>
</tr>
<tr>
<td><strong>Color Image</strong></td>
<td>VGA 640x480: 30 fps</td>
<td>1280x960 RGB: 12 fps, 640x480 Raw YUV: 15 fps, 640x480: 30 fps</td>
<td>SKXGA 1280x1024: 30 fps</td>
<td>VQA 640x480: 30 fps</td>
</tr>
<tr>
<td><strong>Dimensions</strong></td>
<td>27.9 x 6.4 x 7.6 cm</td>
<td>27.9 x 6.4 x 7.6 cm</td>
<td>18 x 3.5 x 5 cm</td>
<td>18 x 2.5 x 3.5 cm</td>
</tr>
<tr>
<td><strong>Data Interface</strong></td>
<td>USB 2.0</td>
<td>USB 2.0</td>
<td>USB 2.0</td>
<td>USB 2.0 / 3.0</td>
</tr>
<tr>
<td><strong>Price</strong></td>
<td>$599.99</td>
<td>$149.99</td>
<td>$159.99</td>
<td>$200.00</td>
</tr>
</tbody>
</table>

Table 2.1: Specifics of four 3D sensors.

The specifics come from the websites: primesense.com, asus.com, msdn.microsoft.com and xbox.com.

Field of view and range refer to the space that the 3D sensor perceives. H,V,D stand for Horizontal, Vertical, Diagonal.

Sensor refers to the set of components that form the sensor: RGB cameras, Depth sensors, microphones and accelerometers.

Depth Image refers to the resolution and frame rate at which the depth images can be perceived.

Color Image refers to the resolution and frame rate at which the color sensor (a normal camera) can process images.

Dimension is the physical space that each 3D sensor occupies.

Data Interface corresponds to the interface that enables the communication between the sensors and a computer.

Price is the cost of each sensor and it is updated to October 2013 and refers to shipping in the U.S.
From the specifics in Table 3.1, we can notice the sensors are similar to each other. The main differences may be seen on: physical dimensions, price and near mode.

- The Asus Xtion Pro Live and the Carmine 1.08 PrimeSense are really small compared to the Kinects and for our prototype is better to use a small sensor because it reduces the surface that may distract passers-by from the projection itself. This may lead us to test one of those sensors for future installations.

- The price, is one of the factor we consider important since we want to use a sensor that has the same performances of the others but maintaining the cost for future installation as low as we can in such a way that everyone may afford it. Then in this case the XBOX 360 Kinect resulted to be the cheapest sensor.

- The near mode is a feature that only the Windows Kinect presents and it allows the sensor to have a shorter range that allows objects or people to be perceived even if they are 0.5m from the sensor. This is not possible with the other sensors that can perceive objects or people starting from 0.8m.

However the few differences we cited before, anyone of the 3D sensors seemed to be suitable for our purposes. Then for our prototype we choose to use the XBOX 360 Kinect we had available. One more difference in using the XBOX 360 Kinect instead of one of the other devices is the License, in fact this sensor can be used to develop applications but it doesn’t have a commercial license which means if you want to commercialize your work you need to buy a Windows Kinect or another sensor.

See Appendix A for more details about what a 3D sensor is and how it works.
2.1.2 Computer

Since computers may be significantly different one an other and they require an higher cost as their power increase, we decided to use a Mac-Mini we had available which consist in a computer of medium-high performances with the singularity of being really small and suitable for its installation in a common area. In fact, it can be easy hidden avoiding to distract the attention of the passers-by from the projection.

Figure 2.2: Apple Mac mini.

2.1.3 Projector

The Ultra-Short-Throw projector is another essential component of our prototype; it allows the passer-by to walk trough the installation without having his real shadow projected on the display surface. The one we used for the prototype is the NEC U260W.

Figure 2.3: NEC U260W Ultra-Short-Throw projector.
With a common front-projector we go through the issues illustrated in the picture below (Fig. 2.4). In Summet J. et al. (2007) the author gives a rich explanation of those issues and they describe some techniques to overcome them maintaining the use of front-projectors.

![Figure 2.4: Issues with front-projection.](image)

On the picture we can see a front-projector projecting light on a display surface and an occluder object, that in our case may be a passer-by, which obstacles the projection with the results of having the actual occluder’s shadow projected on the display surface. An obvious alternative to the projector that ensure there would be no shadow interference between the real passer-by’s shadow and the projection, may consist in using a Liquid-Crystal Display (LCD). In this way we would eliminate the necessity to use any kind of projector having the application displayed directly on the LCD. However, this solution would limit us to the LCD size which still remains smaller compared to the surface that a projector can reach and of course an LCD of large dimensions is more expensive of a projector. Moreover we would like to maintain the project under the category of Spatial Augmented Reality without the use of any extra display device.
2.2 Software

During the first months of study, we focused our attention in establishing which software would have been better to use for the project.

Requirements for the choice of the software have been:

- **Software cost.** Everything should have been open source or free, to facilitate developers to be able to develop applications without any cost in terms of money.

- **Software portability.** It consist on the possibility of running the same software on different platforms (Operating systems: Windows, Mac, Ubuntu). It is a fundamental factor that will limit the range of developers that may be interested in working on the project.

- **Software flexibility.** For flexibility we refer, in general, to the capability of the software to adapt accordingly to the user’s requirements and the possibility of upgrading the software itself to get software improvements and more functions that benefit our developers.

It was due to the previous points that we had to deeply analyse several kind of software such as: Unity3D, Windows Kinect SDK, OpenNI, Zigfu SDK etc.

2.2.1 Unity3D

The only software that has been a fixed point since the beginning of the project is Unity3D - a well known game engine for the development of games. It captured our attention because it is easy to use, it provides several functions, and the free version allows to work without significant limitations (referring to our purposes). Moreover it provides the possibility to be upgraded to the Pro version at the price of $1500 or with a license that expires after a year (a cheaper solution if we need to use Unity3D only for a short amount of time). Unfortunately Unity3D is still not available for Ubuntu users, this limits the range to Windows or Mac users.

Unity3D can be downloaded on the official website unity3d.com where we can find a large community of users sharing experiences and advices. Moreover we can find the documentation, the API, some tutorials for beginners and much more. Because the Unity3D community is quiet large, we can find materials spread all over the internet.
If the decision of using Unity3D has been clear since the beginning, the other software required several weeks of analysis and tests before we choose. We especially refer to the software for handling the transfer of data between the computer and the Kinect. The most used software for this purpose are: libfreenect, Windows Kinect SDK and OpenNI. The main difference between them refers to the compatibility of each software with the Operating Systems that are now available (Windows, Mac, Ubuntu) and with Unity3D.

- **Libfreenect** it’s a library compatible with Windows, Linux and Mac developed by the OpenKinect community. Libfreenect provides a good solution to allow users to connect to the Kinect and access to its data stream. But it doesn’t provide an simple way to integrate it in Unity3D.

- **Windows SDK Kinect** provides a really good environment to interact with the Kinect with several functions, examples and documentation by Microsoft. Unfortunately the Windows SDK Kinect is only compatible with Windows operating systems and it provides a Unity3D wrapper package that has several bugs. Moreover the use of Microsoft software limits the choice of the 3D sensor to the Kinect and of course to Windows operating systems.

- **OpenNI** is an open source SDK that allows to handle the communication with a 3D sensor, that may be a Kinect or one of the other we previously cited. OpenNI provides several functions and examples to handle the interaction with the 3D sensor. The integration with Unity3D may be easily done through a library called Zigfu that acts as a wrapper between OpenNI and Unity3D in such a way to allow the use of the Kinect inside Unity3D. The Zigfu library provides a better integration with
Unity3D than the Windows Kinect wrapper, with more examples and less bugs.

After a test of those software we decided to use OpenNI and the Zigfu library which guarantee to reach the largest set of developers that may be interested in collaborating with the LivingShadow project. Then, established the hardware and software to use, we started to implement some of our ideas.
Chapter 3

Implementation

After we ensured that the hardware and software, we were using for the prototype, were the most suitable to reach our purposes, we started exploring how we could handle: the shadow, the creatures that would populate the virtual environment, and the networking part for the communication and the virtual world sharing. Before we contributed to the project, the only feature that LivingShadows presented was the simple projection of virtual objects and passers-by’s shadow on the display surface without any interaction. Moreover the networking was completely absent.

3.1 Design and Applications Programming

We based our project on Unity3D which allows us to create a 3D virtual environment that we can modify as we prefer. Since we use Unity3D we had to create scenes - collections of objects that populate the virtual environment - on which our applications are based on. We decided for convenience to program in C# and we developed a basic scene, described below, that is used for all our applications.

3.1.1 Basic scene

We started creating a new scene in Unity3D with a game object called plane on which we applied a texture and the script ZigUserViewer that is provided by the Zigfu library. A texture is an image or movie file that we lay over or wrap around objects inside the Unity3D environment. A script is instead a piece of code that can be attached or it can refer to an object changing its status. In our case the script
ZigUserViewer allows us to have a texture representing the passers-by silhouette captured by a 3D sensor. In some cases we also populate the environment with virtual objects or animated creatures we imported from Blender, Maya or other software for 3D modelling. The creatures are placed in front of the texture in such a way that when the application is projected on a wall, the passer-by’s silhouette (we call it "shadow") appears to be on the same plane of the creatures. This prospective, should give the impression to the passer-by that he can interact with the creatures in the projection.

![Image of basic scene with 3D creatures and passer-by’s shadow.](image)

Figure 3.1: Basic scene with 3D creatures and passer-by’s shadow.

In Fig. 4.1 we can see the basic scene we described before with the addiction of modified cubes on the edges of the plane to confining the virtual objects in that area and the unity object camera. On the bottom-right of the picture we can see the camera preview that corresponds to the resulting image which will be projected. An example of the camera view is reported in Fig. 4.2.
Once the scene is projected on the display surface it gives us the feeling that the virtual creatures and the shadow are actually on the same plane.
3.2 Models and Animation creation

This part of the project is still underdevelopment and consists in creating 3D models and animations that will be used to populate the virtual environment: the objects or creatures will have the possibility to move inside the scene and interact with the passer-by’s shadow.

The creation of new 3D models requires specific software. Some of them may be: Blender, Maya and 3DS Max. In our first attempts we chose to use Blender that resulted to be one of the best free software for 3D Modeling.

![Figure 3.3: Gnome 3D model with Blender.](image)

The most important point to consider speaking about 3D modeling is that it requires time to create a 3D model, and this depends on the complexity of the model we want to develop and on the designer’s ability of using the software. Moreover, also the creation of animations for the models is a time consuming task. In this phase particular attention is needed when we export the models and the animations from our 3D modelling software to Unity3D to avoid incompatibility problems.

We can even create animations using directly our 3D sensor and a process called "motion capture" which consists in recording movements of objects or people that may then be applied to virtual models. An example of software for motion capture is Ni-Mate which provides also a plugin to be used in Blender.
We tested this software and it resulted to be a fast and easy way to generate great human-like animations but with the drawback of adding noise to the animation itself (the noise generated by using the 3D sensor). The animation then requires to be manually fixed inside Blender to reduce the noise.

3.3 Object-Shadow Interaction

Once we have the basic scene ready, we can modify it applying other scripts or modifying the one that we have already applied. In our case we choose to modify the script ZigUserViewer that we applied on the plane of our basic scene. This script corresponds to the code which updates the texture on the plane showing the passers-by shadows formed by a set of black pixels. We modified it in such a way that once the texture is applied we scan the texture and we check if there is at least one black pixel with the same coordinates of a virtual object or creature. If so, we say that a collision is detected and we handle it based on which object is "colliding" with the black pixel.

3.3.1 Interaction and Proportion Details

In our case an interaction happens as a consequence of a collision between passers-by's shadows and creatures or objects in the scene. The collision detection between
the objects and the shadow is implemented using a proportion between the coordinates of the pixels forming the shadow inside the texture and the coordinate of the objects in the Unity3D world.

First of all we calculate the dimension of a pixel inside the texture. This value is given dividing the dimensions of the texture in the Unity3D coordinates by the resolution of the texture. For instance if the resolution is 160x120 and the dimensions of the plane in Unity3D, where the texture is applied on, are 0.4x0.3 we’ll have that a pixel occupies a virtual Unity3D space of 0.4/160 x 0.3/120 = 0.0025 x 0.0025.

We decided to fix the plane dimensions in the scene in such a way to maintain this proportion. Then when we refer to a pixel inside the texture, we know which position the pixel occupies and applying the proportion we can calculate its coordinates in the scene.

The area covered by an Unity3D object is given by the dimensions of the attached collider. Colliders are used to detect collisions between objects inside Unity3D. The colliders that we are using are: sphere, box and capsule (Fig. 4.5); for them we created some functions that allow the interaction with the passer-by’s shadow.

![Figure 3.5: Unity3D default colliders.](image)

We use the coordinate of objects and colliders to detect the collision with the shadow. For each type of collider we provided a way to detect the collision:

- For the **sphere collider** we need to know the position of the object and the collider radius, in this way we can calculate the surface covered by the sphere collider and compare it with the position of the black pixel representing the shadow.
For the **box collider** we need to know the position of the object and the collider height so to calculate the surface covered by the collider and make the comparison with the passer-by’s shadow coordinates.

For the **capsule collider** is easier to see the collider as a composition of one box collider and two sphere colliders. In this way we will double check some parts but the algorithm will remain simple.

Knowing the coordinates of: black pixels, colliders and objects; we created a link between the passer-by’s shadow displayed in the texture and the creatures or objects in the scene. In this way the collisions can be detected and handled to give the possibility to the passer-by to interact with the virtual world.

This way of handle the collision it’s quiet easy to apply but it’s not completely efficient because we use predefined colliders such as: box, sphere and capsule. The predefined colliders may not exactly cover the entire surface of the models, as shown in Fig. 4.6, and this may results in false collision detections. However, for our applications, this method resulted fairly efficient especially because our models are small in size compared to the passers-by’s shadows then the fake collisions are difficult to notice.

![Figure 3.6: Capsule collider applied to a 3D model.](image)
3.4 Other ways to identify collisions

Using the shadow applied on a texture makes the applications responsiveness affected by the time that the algorithm requires to verify if a collision between black pixels and virtual objects has occurred or not. After some attempts, we realized that: if the passer-by makes not too fast movements and also if the objects and the creatures inside the virtual environment are moving in a not too high speed (the threshold is not explicitly defined, it depends mainly on the computer performances), this way of identify the collisions is quite responsive.

However with a not powerful computer we can notice problems when the major part of the texture is occupied by the passers-by: more black pixel are in the texture more controls the algorithm must do to verify the collisions, and the controls are increased also by the amount of creatures that populate the virtual world. In this way only with few passers-by and few creatures we are able to execute a pretty responsive application, otherwise we may have some lag\(^1\) and loss of responsiveness.

To improve responsiveness, we considered the idea of replacing the texture with something else to handle the passer-by’s shadows.

3.4.1 Character 3D model

We may try to use a 3D character which will be projected instead of the passer-by’s silhouette. The difference in terms of collision detection, in this way, is that the collision will happen between "real" objects in Unity3D instead of using the coordinates of the pixels to detect collisions.

\(^1\) In real-time applications, the term is used when the application fails to respond in a timely fashion to inputs. Lag is also often used in reference to video games to describe the delay (or latency) between an action by a player and the reaction of the game. Definition from wikipedia.org
However in this case we will lose representation fidelity because the model will not be similar to each passer-by’s shadow. Another way it may be to combine the two methods using the texture to visualize the passer-by’s shadow as real as possible while using the 3D character to detect the collisions. In those cases the only aspect that can determine strange behaviors of the system is that the connection with the 3D character may be not stable and causing strange movements of the 3D character which will affect the virtual environment. Another cons may be the fact that a passer-by walking through the platform with a high speed may be not recognized and linked immediately to the 3D character losing then responsiveness. Moreover, because of the passers-by will walk on a side instead of facing the sensor (or in our case giving it their back) until they decide to interact with the virtual world, the information coming from the skeleton may be not available or distorted.

We made some tests and we verified that, unfortunately, the algorithm of skeleton tracking that is used for linking the passer-by to a 3D character works only in optimal condition: the passer-by must be entirely in front of the 3D sensor facing it (or giving it his back).

Figure 3.8: Skeleton generated from the passer-by approaching installation.
From the pictures we can see how the algorithm generates a skeleton representation which can be used to control a 3D character. The yellow dots are the joints and the green lines are used to represent the bones. In Fig. 4.8 and Fig. 4.9, we can see that some lines are missing which means that some parts of the body are not recognized. Then, when the passer-by is still approaching the sensor and he is not completely inside its field of view or he is not facing it, the link with a 3D character results to be unstable causing unpredictable behaviors for the 3D model.

After this, we concluded that for our project we cannot rely on the use of skeleton tracking and a 3D character to represent the passer-by.
3.4.2 Edge detection

To improve responsiveness, we can also consider the idea of applying some image processing to extract only the edges of the passer-by’s silhouette and instantiate virtual objects on those edges (Fig. 4.11). In this way we allow the edges to collide with other objects inside the virtual world without using any algorithm to identify collisions but only relying on the Unity3D engine. Doing so we avoid the controls between all the black pixels and the objects.

![Figure 3.11: Passer-by silhouette represented using cubes.](image)

![Figure 3.12: Sphere inside the silhouette.](image)

However, with this technique if a collision is not detected, because it happens too fast or because the edges are generated too slow, the object may go "inside" the passer-by’s silhouette (Fig. 4.12) and generate unpredictable behaviors.

For the moment then we decided to continue with the shadow represented using a texture.
3.5 Networking

Since LivingShadows wants to be a way to connect people and installations across the world, we then focused our attention on trying to develop the networking part of the project.

We planned to create a Client–Server architecture with an authoritative server which: handles the connections with the clients, plays the scene and then sends back to the clients the resulting data. In this way we have the clients running a unity application which reads the data coming from the 3D sensor and send them to the server. The server then mixes together the data from each of the clients obtaining as a result a texture with all the shadows coming from the different clients. At this point the server displays the texture on its own scene and it processes the interaction with 3D virtual objects (in our example a 3D sphere). Then, the server sends to the clients the texture that the main camera generates. This texture corresponds to the scene the server displayed including the shadow and the interaction with 3D objects. Finally the clients receives the server scene in form of data, they convert the data into a texture and display it.

Figure 3.13: Client-Server Architecture.
3.5.1 Client

The client application is composed by two scenes: Menú and Client.

Menú (Fig. 4.11) is the first scene that is displayed and it allows the passer-by to insert data such as the port number and the ip of the server to connect.

![Client Menú](image)

Figure 3.14: Client Menú.
The **Client** scene is instead responsible for the communication process with the server. A flow chart schema of the client is reported below.

![Client Flow-Chart](image)

**Figure 3.15: Client flow-chart.**

At the beginning, the Client starts a connection with the server (1 Connect), which ip has been specified in the Menú scene, then it handles the data acquisition process from the 3D sensor and its transfer to the server. Once the server has processed the data, it sends the same amount of data back to the client which will convert them into a texture in such a way to display it (2 UpdateTexture). When the client quits the application (3 OnApplicationQuit), it sends a byte array to the server to inform it.
3.5.2 Server

For the server, we realized two applications: ServerConnectionHandler responsible for handling the connections with the clients and ServerInteractionHandler for handling the interaction with the 3D virtual objects.

**ServerConnectionHandler** is composed by two scenes: Menú and ServerConnection.

**Menú** allows to insert the ip and the port number for the server and the maximum number of clients it can handle.

![InteractiveSphereMenu](image)

*Figure 3.16: Server Menú.*
ServerConnection handles the clients connections and synchronize the communications.

Figure 3.17: ServerConnection flow-chart.
When we launch the application ServerConnectionHandler, it provides a call to the other application ServerInteractionHandler that will be automatically launched. ServerConnectionHandler then starts a loop in which it waits until one or more clients engage a connection. The connections are handled using threads (3 Client Thread): one thread per connection (or one thread per client). The communications between server, threads and clients requires a form of synchronization. We reached this synchronization taking advantage of the shared memory by the threads and the main server program. We declared four Boolean arrays: receiveArray, sendArray, resetArray and readArray.

- **readArray** is used to give the permission to the threads to read the data coming from the client connected to the server;
- **receiveArray** is used to declare if the data coming from the client have been ridded;
- **sendArray** is used to give the permission to the thread of sending data back to the client;
- **resetArray** is used to establish if the shared memory between the threads can be reset;

During the first phase (1 Server Connect) before each connection is handled generating a client thread, another thread called Synchronizer is generated. Synchronizer (2 Synchronizer Thread) is encarged for checking the values of the boolean arrays, we introduced before, and for modifying them to ensure a correct synchronization between the various generated threads. In this way, synchronizer ensure that the data coming from different clients: data corresponding to a passer-by A from one client and data corresponding to a passer-by B from another client; are correctly fused together. Moreover the Synchronizer also handles a connection with the second application "ServerInteractionHandler" and communicates with this one via socket. The synchronizer sends to ServerInteractionHandler the "fused data" then ServerInteractionHandler processes them and sends the same amount of data back to synchronizer which manages to tell to the other threads they can send those data back to the clients.
ServerInteractionHandler handles the interaction between the shadow and the 3D objects.

ServerInteractionHandler runs the same scene of a client with the addition of the code relative to the 3D virtual objects and interactions. The scene uses the "fused data" coming from ServerConnectionHandler to process the interaction between the virtual objects and the shadows which are actually represented by that fused data. The shadows and the virtual objects are then displayed. At this point using RenderTexture\(^2\) the scene is captured and sent to ServerConnectionHandler which will send those data back to the clients.

\(^2\) Render Textures are special types of Textures that are created and updated at runtime. To use them, you first create a new Render Texture and designate one of your Cameras to render into it. Then you can use the Render Texture in a Material just like a regular Texture. Definition by unity3d.com
Chapter 4

Applications

We realized some applications to test our work, in particular: one for testing the collision detection between passer-by’s shadow and virtual objects, another one to highlight the responsiveness of our prototype and the last one to test the networking part.

4.1 Burst Them All

This application allows to test the performances of the used hardware accordingly with our implementation of the collision detection algorithm. It consist in a sort of mini-game that requires the passer-by to intercept, with his shadow, some spheres causing their explosion. These spheres are falling from the top of the scene to the ground and the goal of the passer-by is to stop them before they hit the ground. The player starts with 3 lifes which are decreased each time the passer-by fails to intercept a sphere which reaches the ground.

As we said, we implemented this application just to test the performances of the platform and the algorithm we implemented but it can actually be a funny game especially for kids that may need to run from one side of the projection to the other to reach the falling spheres and intercept them (it depends on the dimension of the projection and the distance of the passer-by from the 3D sensor).
4.2 Bouncing Ball

This application highlights the power of this project. As Myron W. Krueger et al. (1985) report while describing VIDEOPLACE: "When people see their image displayed with a graphic object, they feel a universal and irresistible desire to reach out and touch it". It is this irresistible desire that we want our passers-by to experience. A typical scenario for this application is the one where a passer-by is walking through the platform and he notices a sphere on the wall. While he is crossing the platform, he notices his shadow interacting with the sphere causing its movement then he may start to play with the sphere exploring the virtual environment. This application is really effective and it is surprising how immediately the passers-by respond to the stimuli of kicking or touching the ball.
Figure 4.2: Bouncing Ball.
4.3 Client-Server

Our implementation of the Client-Server architecture allows two clients to share the same virtual environment. Two passers-by from two different platforms can for instance meet each other on this virtual environment and start playing with virtual objects such as the bouncing ball we introduced before.

Figure 4.3: Client-Server with two clients.

4.4 Lag

We have been able to test our applications and verify that, if we run the applications only on one installation avoiding the networking, the applications result to be quite responsive and smoothed. However, introducing the concept of networking, as we did, it can lead to have some kind of lag during the execution of the applications. The lag consists in a delay that may occur because the application requires data that, for some reasons, are not yet available.
In our applications, some causes that may lead to the presence of delay are described below:

1. Time the 3D sensor needs to acquire data.
2. Time to send those data to the server through the Internet.
3. Time the server needs to handle the synchronization between clients.
4. Time the server needs to process the scene with fused data and virtual objects.
5. Time required to send the data back to the client.

For what concerns the time the 3D sensor needs to acquire data from the real world (1) this parameter is fixed based on the sensor that we want to use. In our case with the XBOX Kinect it is 30 frames per second.

The time to send data from the client to the server (2) depends on the amount of data we need to send and on the network bandwidth we have available to send them. Considering a texture of 160x120 pixel the corresponding number of bytes is 75Kb while increasing the texture to 320x240 the number of bytes is 300Kb and for a texture of 640x480 the number of bytes is 1200Kb or 1.171875Mb. The bandwidth, however, depends from our Internet Service Provider (ISP) but it is generally measured in megabit per second Mbit/s. Moreover speaking about different part of the world, the communication speed depends on how these places are connected to each other. Then the server location may play a main role in maintaining rapid communications.

The time the server needs to handle the synchronization between clients (3) is influenced by the time the clients need to transfer data to the server and by the time that the server has to wait to receive every data coming from all the connected clients.

The time the server needs to process the scene with the fused data, from the clients, and virtual objects (4) depends on the time required to apply the fused data (passers-by’s shadows) into the scene and process the collisions with virtual objects.

Finally the time required to send the data back to the clients (5) depends once again on the amount of data and the network bandwidth we have available. In our case the amount of data it’s the same amount we send to the server.

In our opinion, the critical part of the communication consists in the delay required for the server to manage the synchronization between clients and also the transfer of data between clients and server. Then a good location for the server must be individuated and some improvements may be done on the way we synchronize clients.
and server. Moreover a better implementation of the collision detection algorithm, to detect and handle interactions between virtual objects and shadows, may simplify the server operations reducing part of the delay generated by the server. We are confident that, with these improvements, the applications may run across the internet in a more smoothed way.

Another option could be to completely change the Client-Server architecture to a Peer-to-peer architecture. This way avoids the use of a centralized server to favour the distribution of work to each node that will act as both supplier and consumer of resources. A Peer-to-peer network may reduce the delays caused by the use of a centralized server but will require a re-adaptation and re-implementation of all the networking part and of the applications that use the networking.
Chapter 5

Related Works

Here we introduce projects that we found particularly relevant for our research.

5.1 Culture Walls

Culture Walls: Echoing Sounds of Distant Worlds Across Planet Earth — A Proposal", Kristinn R. Thórisson (2000). In this project the main goal is to bring in contact different locations of the world using walls — instruments that are commonly used to divide. These walls integrate technologies that allow to capture sounds from one wall in a country A and reproduce them in another country B. Together the Walls create a way to connect societies and people from "different worlds" using modulated audio ambience.

Figure 5.1: CultureWalls.
Differently to CultureWall, in our project we want to use the concept of "shadow" to create the communication instead of using sounds and we want to provide the possibility to interact, with the shadows, in a shared virtual environment. Moreover while CultureWall has never been implemented, we already have a prototype of LivingShadows running in Reykjavík University.

### 5.2 ShadowTalk

A project that involves the use of shadows is "ShadowTalk" by Hannes and Joey Chang (1997). ShadowTalk was one of the components of the Literary Salon, a café environment which aims to encourage the interaction between the guests. ShadowTalk covers an important role in trying to make easier the interaction between guests that may do not know each other. You had just to send a shadow to another user to start a conversation: this could be a helpful way for people that are too shy to introduce themselves, in person, to another guest. The project was based on the projection of computer rendered shadow-like characters on a wall between two tables. Each shadow could be controlled from the table by using a trackball. The user could type on a keyboard and read from a display utterances that the shadow then delivers along with an appropriate body language. The acting done by the shadows contained communicative gestures, facial expression, and subtle backchannel listener feedback to simulate a real environment in which two people were talking up close.

![Draft of the ShadowTalk installation.](image)

ShadowTalk is quite different from a face to face style talk and an internet style talk session. It allowed a compromise between these two common social interactions.
Especially it didn’t require the use of internet that is why, differently from our project, it’s not conceived to keep in touch people across the network. Moreover as we mentioned, ShadowTalk made use of computer rendered shadow-like characters while in LivingShadows the shadows come directly from the users giving them the possibility to easily identify a correlation with the shadows.

5.3 Scott Snibbe’s art works

We found also interesting the work of Scott Snibbe, a media artist and entrepreneur which produced several art works based on shadows and projections. In Scott Snibbe (2003) he describes the origin of shadow plays, probably China over 2000 years ago, continuing with a description on how they spread into the world. He concludes introducing his art works: "Compliant" (2002) and "Impression" (2003).

![Figure 5.3: Compliant](image1)

Bodies distort a soft rectangle of light.

![Figure 5.4: Impression](image2)

A screen that holds an imprint, like clay.

In Molly Polk (2005) the author created a catalog where she presents some art works by Scott Snibbe for the Boston Cyberarts Festival (2005). In this catalog are present art works such as: "Deep Walls" (2003), "Visceral Cinema: Chien" (2005), "Shadow Bag" (2005) and also "Compliant" (2002). All these art works allow the user to interact, somehow, with the installation and play with their own representation.
5.4 Myron Krueger and Videoplace

A large contribution in the fields of virtual environment and augmented reality has been given by Myron Krueger. Myron Krueger (born 1942) is an American computer artist who developed early interactive works. He is also considered to be one of the first researchers in virtual and augmented reality.

In Krueger M. W. (1977) the author reported: "man-machine interaction is usually limited to a seated man poking at a machine with his fingers...I was dissatisfied with such a restricted dialogue and embarked on research exploring more interesting ways for men and machines to relate". Krueger developed several responsive environment technologies that evolved and led to his most famous work, "VIDEOPLACE" and his most cited book Artificial Reality II Myron W. Krueger (1991) VIDEOPLACE, Myron Krueger et al.(1985), wanted to be a project that placed the user in the middle of an artificial reality able to respond to the users actions and movements without the necessity (for the user) to use any kind of input device.

VIDEOPLACE is considered a milestone in the field of Virtual Reality and for this reason it is on permanent display at the State Museum of Natural History located at the University of Connecticut.
We can consider our project a re-implementation of VIDEOPLACE with the use of updated hardware and software.
5.5 Passages

Passages is a project by the Human Connectedness research group in the Media Lab Europe by Joëlle Bitton, Céline Coutrix and Stefan Agamanolis. It consists of a media space—a system that integrates multiple types of media to connect distant places and groups of people—which allows passers-by to engage in communication with other passers-by from a different city. The passers-by can approach the interaction surface as close as they wish, then the passers-by silhouette is displayed requiring the passers-by to overlaid their silhouette to be able to see each other. With Passages, they aim to explore the possibilities and outcomes of being more emotionally and physically engaged in a media space: by incorporating elements of a private space in a public urban space; by enabling a heightened sense of proximity and intimacy; by using the body and its movement as an interface; by connecting strangers from different places and cultures. These topics are further discussed in Joëlle Bitton (2008).

This project, and its documentation provided by Céline Coutrix (2004) inspired us on how to manage the transfer of shadows across the network even if in our project we have also to deal with virtual objects and the interaction between them and the passers-by’s shadow. Another significant difference with LivingShadows is the proximity factor. In Passages the proximity plays a main role in fact the passers-by are able to reach the displaying surface so close that they can touch it and this may give them the sensation of being physically closer to each other of what they really are, while in our project the proximity is limited by the real passers-by’s shadow.
interference with the projection which we discussed before. Stefan Agamanolis produced, in the Media Lab Europe, other interesting works based on virtual reality and telepresence, some of them are described in Stefan Agamanolis (2003).

5.6 OneSpace

Referring to the networking part of our project, we found interesting OneSpace a video conferencing system by David Ledo et al. (2013) where the video images of two different spaces are merged into a single shared, depth-corrected video. This project is essentially different from LivingShadows because we aim to place the users in a virtual environment that we previously create while in OneSpace they place the users in a merged environment which may give them the impression of looking through a mirror which reflects more objects and people than the ones that are physically there. They also provided the possibility to add effects to the reflection such as: background changes; shadow representation of the users; a trace effect where ghostly trails of people’s motions are overlaid atop one another, and the reproduction of recorded scenes.

![OneSpace](image)

Figure 5.11: OneSpace: A video conferencing system that creates a depth mirror.
Initially, in our project we considered the possibility to share not only the passer-bys’s silhouette but also the shape of objects that may be located between the 3D sensor and the display surface reflecting then, in the virtual world, part of the real world. But for practical reasons we decided to work, for now, only with the passers-by’s silhouette.

Figure 5.12: LivingShadows: Passer-by silhouette, virtual object sphere and real object chair.
Chapter 6

Conclusions

We started this project analysing some hardware components that form our prototype, especially we had the possibility to study the range of 3D sensors now on the market, their characteristics and their capabilities.

We moved on studying the software with which we managed the data coming from the 3D sensor. We focused our attention on Unity3D, OpenNI and the Zigfu library using this software to develop some applications. The applications we developed for the LivingShadows prototype, aim to place passers-by at the center of a virtual environment giving him the possibility to engage interactions with virtual objects or creatures and passers-by from other part of the world taking advantage of an internet connection to link different platforms together across the network. All this required a study on how to manage the passers-by’s representations and their interactions with the virtual objects inside the projected virtual environment, which leaded to the decision of leaving the passers-by’s representations in the form of texture and going to operate on a pixel level. Moreover the networking part required also a previous study about the client-server architecture and its implementation in C# inside Unity3D. After these studies we have been able to realize some applications which enable the passers-by to interact with virtual objects using their "shadows" and provide the possibility to connect more platforms together in such a way to share the virtual environment with passers-by walking through platforms in different locations.

Finally our work contributed to improve the LivingShadows project forming the bases for future developments and leaving a considerable documentation for future developers.
6.1 Future Works

Other applications must be implemented to increase the range of interaction available for the passers-by, and creatures need to be designed to populate the virtual environment. Like the creature that Páll Arinbjarnar and Katrin Olina - two other collaborators - were working on: an animated spider that can interact with the passer-by, for example climbing the shadow from the bottom to the top or jumping on the shadow and performing some particular animations. Another possible application could use Artificial Intelligence to enable a virtual shadow to present, to passers-by, news of the day coming from an online newspaper or some information which they may be interested in.

We also envisioned a future application which may be used as a Turing Test\(^1\) (Alan Turing 1950) were one of the shadows is actually controlled by the computer running an Artificial Intelligence. The test would see a real passer-by, which generates a shadow, interacting with the platform with another shadow controlled by the computer. If the real passer-by would not be able to say whether the other shadow is controlled by a real passer-by or by an artificial intelligence we may say, somehow, that our implementation passes the Turing Test.

The networking part of the project must be improved in such a way to be able to transmit the data across the network without a significant delay. At this purpose, we may apply some decompression techniques to decrease the amount of data to transfer. Moreover the technique we used for handling the connections and the synchronization between more clients may be improved and may be implemented outside Unity3D avoiding to run another unity application that may be slower than a simple windows application. In particular Unity3D is not thread safe and during the implementation phase we had some problem trying to figure out how we could handle all the connections, that is why we ended up dividing the connections part from the display of the scene and interaction with 3D objects.

We may also change the Client-Server architecture to a Peer-to-peer one and try to understand if this solution may be better.

Before these improvements we must set up a website reserved for the LivingShadows project and start to form a community which can interact and share ideas. On the website we may also provide the possibility to upload applications so the developers can share them and maybe rank the best ones. Some applications can also be tested using the Unity Web Player feature which allows to run an unity application

---

\(^1\) The Turing test is a test of a machine’s ability to exhibit intelligent behaviour equivalent to, or indistinguishable from, that of a human.
on a web browser, in this case we need only to share a link to the application. We finally hope that Reykjavík University and the University of Camerino as well as other universities may collaborate on the project. Setting up other hubs would be instrumental in taking the project to the next level and paving the way for a global collaboration with numerous sites across the globe. A practical example of our project’s use may be as an attractor for some particular spots or as a way for people to enjoy a waiting area.

The several fields of study that the project incorporates make it suitable for students interested in computer graphics, computer programming, networking, artificial intelligence, game design and others.
References

Stevenson, R. L. (1895). Strange Case of Dr. Jekyll and Mr. Hyde (Vol. 7). Translated from the german BY R. F. C. HULL. Scribner
Hannes Högni Vilhjálmsson and Joey Chang (1997). ShadowTalk. MIT.
Joëlle Bitton (2008), Flirting Across a Distance: How a screen creates intimacy with the shadow, Ambidextrous, Fall 2008, pp. 32 - 33.
Appendix A

3D Sensor

The 3D sensors analyzed are provided with: Color sensor, IR Emitter, IR Depth Sensor and Microphone Array. Only the Kinect has the Tilt Motor which allows to rotate the sensor down to $-27^\circ$ or up to $+27^\circ$.

A.1 Depth Camera

While the Color sensor is a common camera used to acquire RGB images, the IR Emitter and the IR Depth Sensor compose a Depth camera. A Depth camera, uses infrared and offer several advantages over traditional intensity sensors, working in low light levels, giving a calibrated scale estimate, being color and texture invariant, and resolving silhouette ambiguities in pose. It also greatly simplify the task of background subtraction.
In more details, the 3D sensor projects a pseudo random pattern of light (Speckles) into the 3D scene, using a diffuser and diffractive element of IR light. IR Depth Sensor observes the scene. The distortion of light pattern allows computing the 3D structure. This coding has to be unique per position in order to recognize each point in the pattern.

The depth data are calculated with the triangulation of each speckle between the projected pattern and the observed pattern. Each point has its correspondence speckle. Having a calibrated speckle pattern the 3D sensor computes the 3D map of the beginning frame then computes the x-direction speckle shift to renew the 3D map. Calibration is carried out the time of manufacture. A set of reference images, for the first computation, were taken at different locations then stored in the memory.
The speckles size and shape vary on distance and orientation with regard to the sensor. Kinect uses 3 different sizes of speckles for 3 different regions of distances.

- **First Region**: Allows to obtain a high accurate depth surface for near objects aprox. (0.8 – 1.2 m)
- **Second Region**: Allows to obtain medium accurate depth surface aprox. (1.2 – 2.0 m).
- **Third Region**: Allows to obtain a low accurate depth surface in far objects aprox. (2.0 – 3.5 m).

Data from the depth sensor builds a depth map, which is used to map both moving and fixed objects. In particular, users can be identified by body tracking that is done via a skeleton tracking system which assigns joint data to the motion information captured.

The application of a skeleton tracking algorithm allows to extract the information about the users. Combining this algorithm with background subtraction, we have an image with only the user’s silhouette.

![Figure A.4: Extrapolated user’s silhouette](image)

For who wants to learn more about skeleton tracking algorithms we can add that, unfortunately both Microsoft and OpenNI don’t provide any possibility to access to the source code of the skeleton tracking algorithm limiting the developers to use higher function calls to access skeleton data. However, an opensource skeleton tracking algorithm has been released by IGALIA and it is called Skeltrack so you may start from that one.
Appendix B

Noise Reduction

Several kind of noise can be identified in the data generated by the infrared. The main perceived noise is the one that produce a flicker of the image caused by the light interferences. Another one is caused by the fact that infrared can be reflected on almost any surface that reflects normal light, then we may have noise caused by their reflection and propagation in some other directions. If they are reflected and propagated somewhere else, they cannot be detected by the 3D sensor and this translates in white holes inside the image. This must be considered during the installation of our prototype, speaking in terms of reflective materials that may be in front of the 3D sensor.

Moreover the 3D sensor can detect some shadows generated by the fact that the IR emitter and the IR reader are located in a slightly different position. But, if we use the user image that is generated after the application of skeleton tracking and background subtraction, we probably will not have the depth shadow noise because the algorithms applied avoid this problem.
Figure B.1: Infrared Reflection caused by a reflective object.

In the picture we can see how the passer-by image is disturbed by the IR reflection caused in this case on purpose with the passer-by holding a mirror while the flicker of the image can be seen just launching an example application.

NOTE: probably with the new generation of 3D sensor (in this case Kinect) we will not have anymore light interferences because the new sensor will use a new technique called active IR\(^1\). This means that we may not have the

\(^1\) The all–new active–IR capabilities allow the new sensor to work in nearly any lighting condition and, in essence, give businesses access to a new fourth sensor: audio, depth, color and now active IR. This will offer developers better built-in recognition capabilities in different real–world settings independent of the lighting conditions including the sensor’s ability to recognize facial features, hand position, and more. Source: Kinect for Windows Blog http://blogs.msdn.com