Augmented Sound

A thesis submitted to the Faculty of
The College of Imaging Arts and Sciences
in candidacy for the degree of
Master of Fine Arts

Rochester Institute of Technology
College of Imaging Arts and Sciences
School of Design
Graduate Computer Graphics Design MFA Program

Thesis Committee
Shaun Foster, Chris Jackson, Nancy Doubleday

Berio Molina February 2011

Thesis for the Master of Fine Arts Degree

Rochester Institute of Technology
College of Imaging Arts and Sciences
School of Design
Computer Graphics Design

Title: Augmented Sound
Submitted by: Berio Molina
Date: February 21, 2011

Thesis Committee Approval:

Chief Adviser: Assistant Professor Shaun Foster, Computer Graphics Design

Signature of Chief Adviser Date

Associate Adviser: Associate Professor Chris Jackson, Computer Graphics Design

Signature of Associate Adviser Date

Associate Adviser: Associate Professor Nancy Doubleday, Interactive Games & Media

Signature of Associate Adviser Date

School of Design Chairperson Approval:
Chairperson, School of Design: Patti Lachance

Signature of Chairperson Date

Abstract 1

1. Project Definition 2

1.1 Introduction 2

1.2. Inspiration 2

2. Keyworks 5

3. Research 6

3.1. Software 6

3.2. Hardware 6

3.3. Art Style 7

4. Review of Literature 8

5. Process 10

5.1. Initial Sketches 10

5.2. Hardware Implementation 13

5.3. Art Style 15

5.4. Sound Design 20

5.5. Configuration Window 20

Conclusion 24

Bibliography 25

Appendix A. Coding 26

Appendix B. Thesis proposal 37

Abstract

A Computer Vision System for doing Interactive Installations

This thesis is a computer vision system for doing interactive-creative musical installations: A portable computer vision system based on video projection and a green laser, that allows the interaction with the projection itself, with the physical space where it is projected and with multiple users.

By using a green laser as a physical user interface, people create projections of sound balls that are projected on top of objects. These balls interact with the physical characteristics of the object where they are projected, as well as with other balls that where previously created in the same space. When this interaction happens they produce sound, which varies depending on several rules that are initially set up. Depending on the complexity of this interactive relationships between the projection (balls), the space where the projection is projected (objects) and the user, it creates a new aesthetic sound and visual layer as an expansion of the object that can also be called augmented sound.

1. Project Definition

1.1. Introduction

This thesis is about transforming video projection into a musical user interface. In this case the focus of the research is not the content of the video projection but the projection itself and where it is projected. Usually, the surface where a video, game, animation, interactive work is projected is a solid plane without any kind of significant information. Normally it is a white wall, or a screen especially designed for receiving the light of the projector, but rarely is it considered as one more element that could be used for aesthetics purposes. This becomes especially true when we talk about musical interfaces. Therefore, this thesis researches about how the content of a video projection interacts with the physical space of a surface or object where it is projected, and also how users interacts with both space and content projection to make any kind of sound performance. In addition, the system is mobile and able to adapt to different physical spaces fast. So this thesis is a system for developing interactive musical installations that deal with the interaction between the projection itself, the physical space where images are projected and the user that controls what is being projected.

This thesis uses concepts related to computer vision technology like color tracking, mapping and image warping. It also use tools like firewire cameras, laser and video projectors. The software used for mapping and interpreting data from physical objects is a c++ library called Openframeworks, which is being developed by Zachary Lieberman and Theodore Watson, and which is being used by several creators for building interactive artistic works and for creating experimental creative code.

1.2. Inspiration

Where does this idea come from?

The idea for this thesis came after combining 3 projects about computer vision and interaction. The first one was the Graffiti research lab project, developed at the eyebeam lab by F.A.T (Free art and technology). The second one was the project called Artificial Dummies done by the Italian group called DOMO. The last one was the project called Funky forest by Theodore Watson.

Laser tracking by Graffiti Research Lab

What was found interesting about this project is the use of a Green Laser as user interface for interacting with the application. Moreover, the act of manipulating the Laser provokes some kind of magic that is really attractive to people. They feel that they can “touch” the surface by just pointing the green bean to it and they can interact in long distances, which makes it very useful for urban actions or installations.

The first version of the software that was written for this thesis was based on the software that GRL wrote for developing Laser tag. It was an incredible resource of code and I took several algorithms from it.

Artificial Dummies by DOMO:

The main idea that we found interesting about this project was the interaction with physical space. The video projection on the wall not only recognizes the shapes of the wall itself, but interacts with it. The animals move throughout the ston surface using steering behaviors and boids algorithms, and they identify the windows of the wall as obstacles that they avoid. The capacity of a projector to interact with the shape of the surface where it is projected is what we found interesting.

Funky Forest by Theodore Watson:

The concept that was adopted for this project is based on the idea of user interaction. This installation is all about user interaction and what is really interesting about it is how well mapped body movements are and how well they relate to virtual feedback. This body map library of physical gestures is not used in the thesis, but the concept of user interaction is quite important for drawing the thesis conceptual triangle, which vertex are: user interaction, physical user interfaces and interaction with physical spaces.

Sound is also added as an element that glues all these concepts together.

Funky Forest project	----------	User Interaction
L.A.S.E.R. Tag	----------	Physical User Interface (Green Laser)
Artificial Dummies	----------	Interaction with physical space
Thesis Project	----------	Sound + 3 listed concepts.

2. Keywords

Augmented sound

Sound balls

Laser Tag

Computer Vision

Openframeworks

Supercollider

Open Sound Control (OSC)

Mapping

Interactive art

Creative Code

3. Research

3.1. Software

Before choosing a software or library for developing the project, most of the concepts were developed in different environments to see their performance and behaviour, and also for testing to see if that environment could be the final one chosen for developing the application. These languages are actionscript 3, Processing, Openframeworks, Puredata and Supercollider.

In actionscript 3 we tested some Craig Reynolds algorithms for related group behaviors that were not implemented in the final version. This scripting language was eventually rejected because it couldn't deal with heavy processes.

Processing was used for testing object collision and also for solving other small coding problems. It was used for sketching and there were a lot of examples and implementations of ideas that were to be used in the thesis, it was also rejected because it lacked velocity to run several processes at the same time.

Openframeworks was the selected library for writing the application for the following reasons. First, It specializes in openCv and a lot of works made with this tool are about computer vision, which is the technique used for laser tracing. Second, Theodore Watson, who is one of the creators of the L.A.S.E.R. Tag project, is also a developer of Openframeworks, along with Arturo Castro and Zachary Lieberman, and finally, it is written in c++, which is a faster language than processing or as3.

PureData and Supercollider are languages used for managing audio. The difference between them is that PureDate has a graphical interface and adds one more level for programming, and also that Supercollider is focused on live coding. Both of them support OSC, the protocol chosen for comunicating with Openframeworks, and both of them have similar preformance in terms of velocity and capacity. Supercollider was chosen in the end because it doesn't use a graphical interface for programming and makes it easier to compare code with openframeworks.

3.2. Hardware

+ User interface:

At the beginning there were two candidates of user interfaces: A Wii remote + a Wii sensor bar and a green laser.

The green laser was selected for the following reasons:

1. It can be used in large distances. Whereas Wii remote has to be close to the wii sensor bar and makes it difficult to use it in public areas, where space gets wider. Since the intention of the thesis project was to be portable, creating limitation with distance was a problem.

2. Green Laser is very easy to use. While Wii remote has several buttons, green laser has one. The Thesis goal was not user interaction itself but interaction with physical space. The important point is not how user interacts or if is possible to do a wide range of things using an interface, the main goal is to interact with physical spaces so user interaction must be as easier as possible. Moreover a lot of people are already familiar with hot to turn on a laser. The main goal was to create a user-friendly application that could interact with physical space

3.3. Art Style

The whole art style is based on one shape: a circle.

This is the chosen shape for several reasons including the adaptability to several kinds of spaces, sound visualization, it is easy to create and is an uniform shape.

Adaption to spaces:

One of the important clues of the thesis is interaction with physical spaces. Physical spaces sometimes have irregular forms because of their organic anatomy. Circles are easy to interact with because of its rounded shape. A circle can adapt better to an irregular form that a rectangle for example. This is because circles don't have a vertex and it is a regular and neutral form and ideal for interacting with any kind of space.

Sound visualization:

One of the main points of the thesis is that a musical instrument expands or augment the sound of an object. Sound is vibration and a common way to visualize vibration is throughout ways forms. Ways forms are circles and it could be said that a ball is the emitting point of these ways forms. So it turned out that like balls and circles were the perfect shapes for visualizing sound.

Simplicity:

Again, the main accomplishment of this thesis was to interact with physical space and the end user to expand or augment the sound on these interactive objects. To accomplish this it was necessary to use a lot of resources from the computer. For example, when laser tracking techniques are used, while playing sound, generating visual content that is projected and when it is necessary to check the interaction of this visual content with the physical space and with itself.

A circle is a very easy shape to generate in code and it doesn't take many resources from the computer.

4. Review of Literature

It was difficult to find books that were directly related to this thesis project, because interacting with physical space using mapping techniques as well as manipulating a laser as an user interface device are relatively new and they are not well documented yet. Besides, some books and resources were really useful in order to solve concrete problems as well as help to get deeper understanding interactive concepts.

Books

Making things talk
by Tom Igoe
O'Reilly 2007
This book is about connecting several hardware and software in order to communicate them and share information. The final goal is helping on the technologic development of creative and interactive works. For instance, learning how to communicate two different softwares will help me to convert graphical data to sound data.

Processing. Creative Coding and Computational Art
by Ira Greenberg
Friendsof 2007
This book is very similar to the one written by Casey Reas and Ben Fry, that I referenced before. It is about the processing code and explains how to use it. This book was also useful to me because it allowed me to approach the same topic from a different point of view. This combination of using two similar books for a same purpose was very useful because when I didn't understand what one book said, I had another resource to look for the same information, which gave me a more wide view in some aspects.

Creative Code
by John Maeda
Thames & Hudson 2004
Creative Code is a book written by John Maeda, and it shows works that students did under the MIT Media Lab umbrella. It is a good way to see works where people use technology in a complete innovative and strange way, just because they are trying to see the aesthetic side of the technology. I think that this is the main idea I could see on this book. How technology's aesthetic can expand the creative side of the technology and make us see things from a new point of view. It helped me to expand my conceptions about technology and not be afraid to think “out of the box”.

Online Documentation

Openframeworks: Documentation

Interactive Architecture

Graffiti Research Lab

October, 2008

http://graffitiresearchlab.com/?page_id=32#video

This project is about doing generative graffiti interacting between a video projection and a physical space. This is the base from where the thesis will be built. This is the only project that I have found that combines the interaction between the projection, the physical space and the user, and because they use open technology, it is easy to implement what they have done and push it further.

Zach Lieberman and Theodore Watson

September, 2008

http://www.openframeworks.cc/documentation

This is the documentation webpage of the Openframeworks library. The Openframeworks library was created to make easier write c++ code for doinginteractive and creative works. This is the library that I am going to use to develop the multitouch system and this is the software that I am going to use for receiving physical data and convert it to a graphical environment. This online documentation of the library is my primary resource that I look when I want to something about the library. It is like the Openframework's help.

Openframeworks: Forum

Openframeworks users

September, 2008

http://www.openframeworks.cc/forum

The Openframeworks forum is the way I keep in contact with other people that are using the same technology that I am. It is also very useful for keep myself updated with the new projects that people are developing. Moreover it is an indispensable place where I can ask for help, and also the first place where I will get feedback from my works. Depending on the interaction in the forum and what the other people tell me, my thesis changed in several ways.

Vimeo

Vimeo users

September, 2008

http://www.vimeo.com

This is the place where I found most of the projects that are related to multitouch, Openframeworks, sound-visualization, Puredata, or interactive art. Moreover, this is the best way I found to track the new projects that people is doing with this technology, thanks to the rss tag syndication. For instance, I am syndicated to these three tags from Vimeo, which allow me to be informed of the new videos that people have uploaded: "openframeworks", "physicalcomputing", "tangibleinteraction".

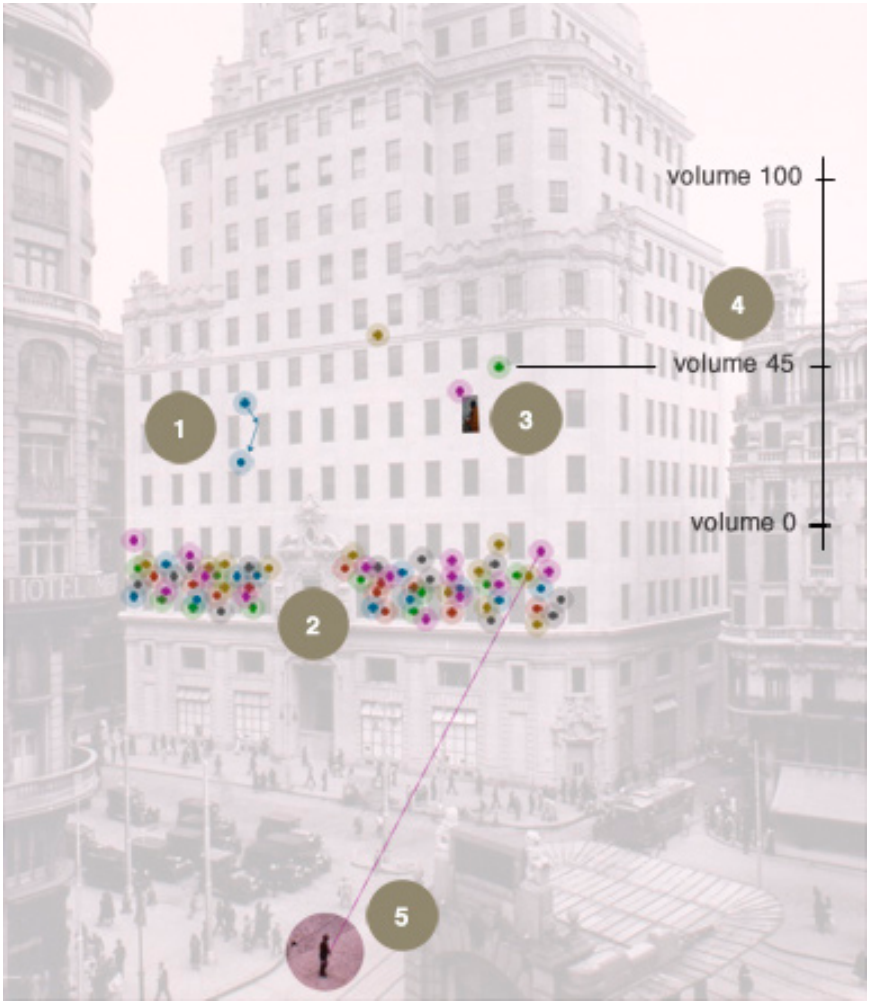
5. Process

The working process for this thesis could be divided into four parts: Initial sketches, hardware implementation, art styles, sound design, graphical user interface and configuration. First, there are going to be presented some sketches that will show what the initial ideas were, how they would look land possible concept solutions to achieve the goals stated in theses sketches. Second, some problems and solutions related to the hardware used are going to be pointed out. Third, it is going to be explained how different art styles are applied to different elements, and finally, the user interface is going to be analyzed as well as the configuration window of the application.

5.1. Initial Sketches:

The initial idea was to build an interactive and musical installation that could interact with a physical space. Using video projection we see which content would change depending on the interaction with physical surface where it would be projected.

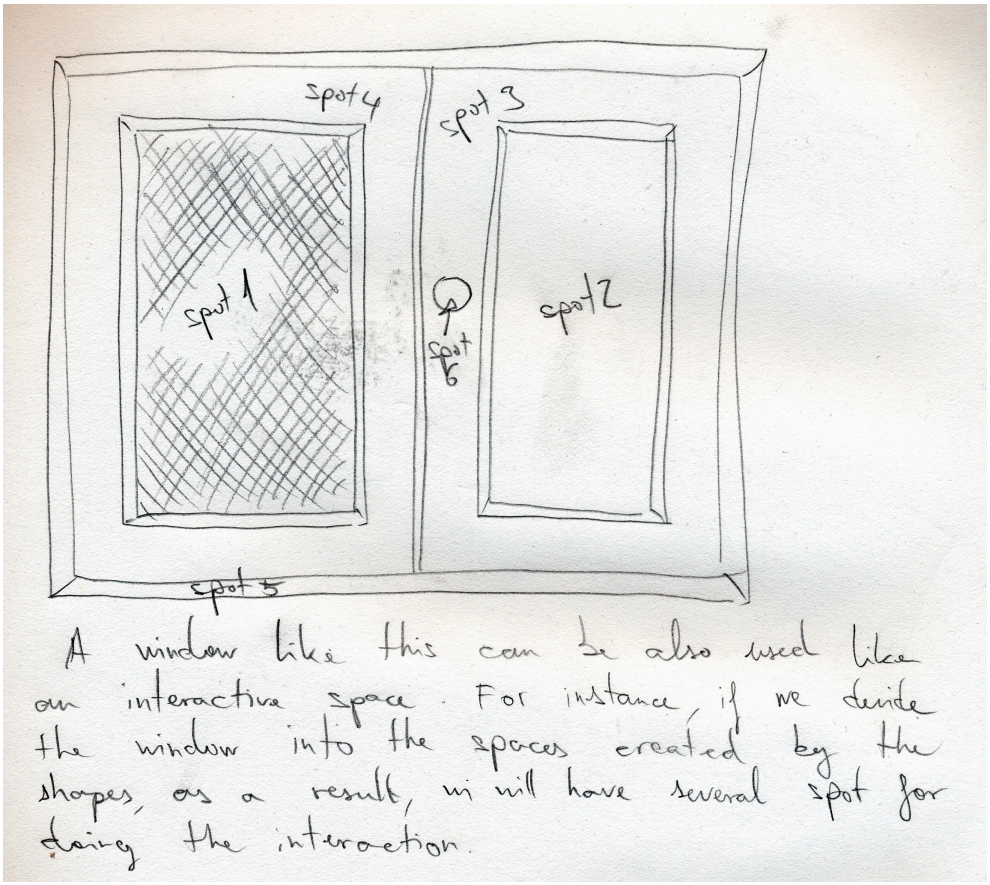
Here is a first sketch that shows how it would look:



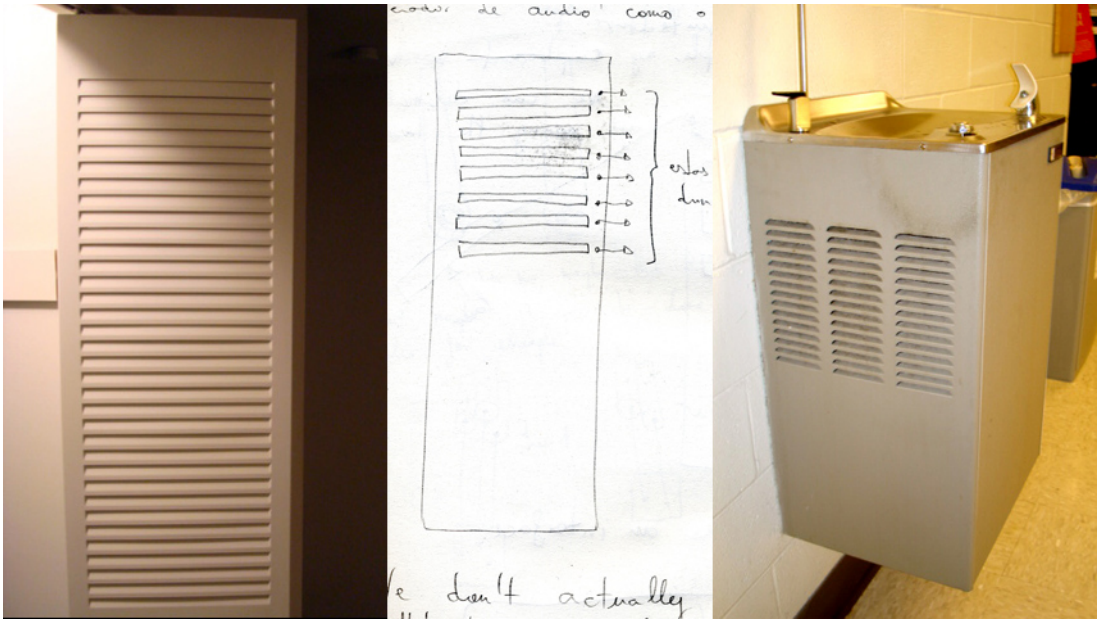
The basic concept of the thesis can be found looking at the figure. There is user interaction with a physical interface(a man pointing to the building with a laser), interaction with a physical space (balls that bounce with windows and images that appear inside those window), and sound (the balls emit sound).

This project was too ambitious so the next step was to narrow it down to make it more affordable. In order to make it possible we applied the same concepts to smaller objects. Taking pictures of them made it possible to imagine how the interaction could be applied.

Here are two examples. A window and a fountain.

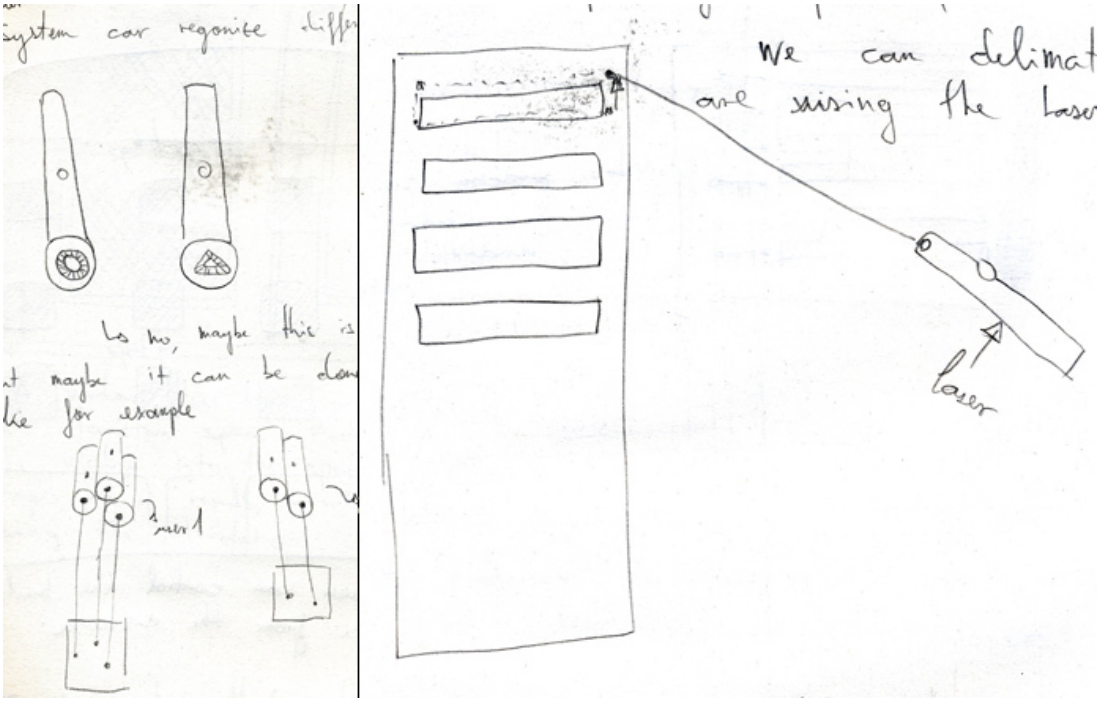


Here we can see how the different parts of the window (spot1, spot2, spot3 ...) could be used as different independent interaction spaces.



Here we can see how the holes of one side of the fountain could be used as interactive spaces. In this case it could work as a musical sequencer if we think of each one as a musical track. Balls bouncing inside each one of the spaces could be seen as instruments inside each musical track.

The same fountain



The image on the right shows how the laser could be used for limiting spaces. By drawing the shape of a given space, it is possible to limit an interaction. In this case, we can see that the area is a

rectangle that is being drawn by the laser, and we can also see that there are four points that could make it possible to move and re-locate the vertex of the rectangle.

The image on the left shows the possibility of using several lasers, so the interaction can be applied for several users. The two lasers on the top project have different shapes, a circle and a triangle, making it possible for the computer to identify them and give them a unique ID, so they could be tracked.

The two laser drawings from the bottom follow the same concept, but instead of using different shapes it tries to identify them by building several clusters of lasers with a different number of them, so depending on the number of lasers in the cluster, they can be identified.

5.2. Hardware implementation

The special hardware used for this thesis, besides computer, are Green Laser, video-projector and a firewire camera.

Green Laser:

The one used for this thesis is a green laser manufactured by wicked lasers with a maximum output power of 40mW. The color of the laser is important because color green is easiest to track by the computer. Moreover, its power is important because the brighter and intense the bean is, the easier it is to track, because the algorithm that deals with color detection use HSV color model (Hue, Saturation and Value-Brightness), making a significant difference in the Brightness value and will allow us to separate this color from the rest.

Video Projector:

The projector used is Epson EMP-X5 3LCD Projector, because it is cheap and is bright. It would be necessary to use a much brighter projector if the interaction is used outside.

Camera firewire:

The camera used is Unibrain Fire-i digital camera, which is a really good camera for computer vision experiments because it is firewire and works really good for streaming and because it can be manually configured so you can change the white balance, brighness, velocity, etc... This is very important because having the possibility of manipulating the color values used by the camera will allow the addition of a primary filter for isolating the green color before arriving to the software processes.

It would be better to use a camera that can handle more frames per second, in order to follow the fast movements made by the laser, but at the time this thesis was being developed, their price was quite high.

5.3. Art Styles

Art design can be divided in two big blocks. The first one corresponds to the configuration screen, and is going to be described in detail in point 5.5. The second one corresponds to the projection screen and it holds the whole interactive and generative graphics.

As it was pointed out before, all graphics are balls, because they can adapt and interact easily with all kind of shapes that spaces could have due to their regular form, because balls, points and circles represent vibration and helps to visualize sound and because they are actually very easy to generate and computers don't require a great amount of resources.

These balls are grouped in two families: the color family and the behavior family.

The color family determines the sound of the balls, so depending of the color of the ball it will generates a determined sound.

Sounds are grouped by frequency. Each one of the frequency group corresponds to a given color. This relationship goes as follows:

Yellow ball	----------	High frequencies
Pink ball	----------	Mid-high frequencies
Blue ball	----------	Mid-low frequencies
Purple ball	----------	Low frequencies

The reason why sound is classified by frequencies is because each one has to occupy a place in the audible space the same way a color occupies a place in the color spectrum. Actually they both are frequencies.

The Behavior family determines how the balls interact with the space they belong and with other balls.

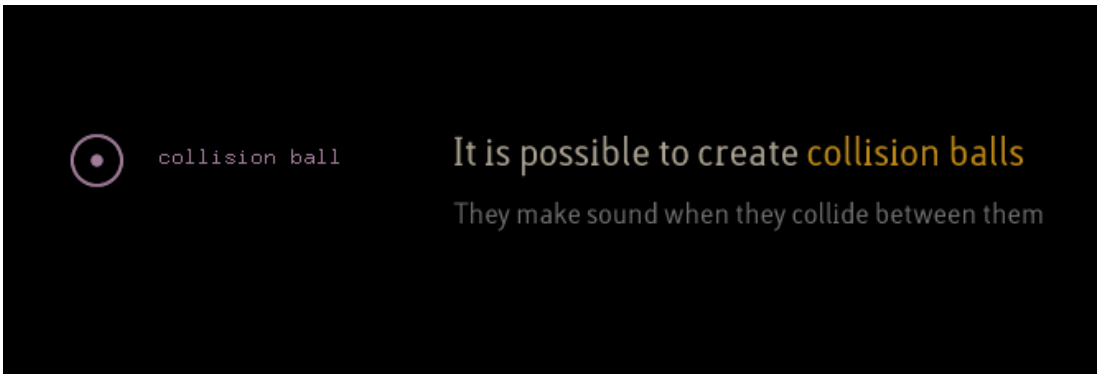
They are classified in four type of balls: Regular Balls, Collision Balls, Mass Balls and Friction Balls.

Regular Balls:

These balls are generated by default. They move in a constant velocity and they bounce when they find the limits of the space they belong. Each time they bounce they play the assigned sound related to the color that was previously chosen. Their color has a low alpha percent so it is possible to see through them.

Their shape is the simplest one. Just a colored dot.

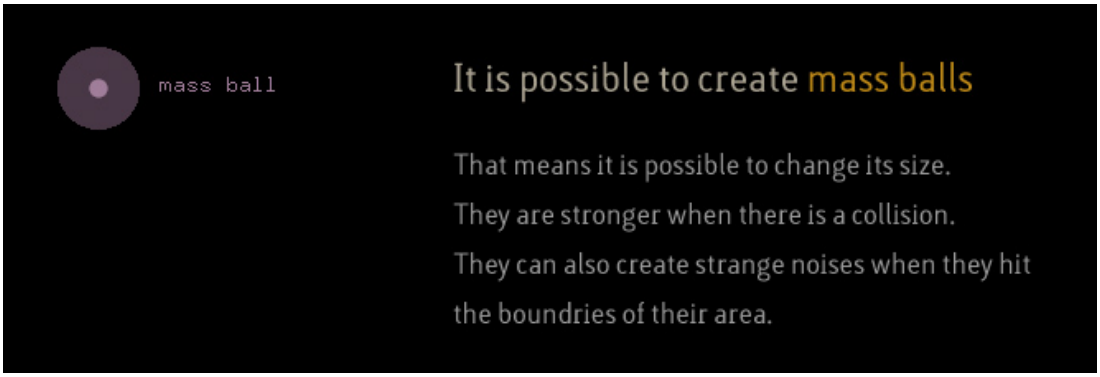
Collision Balls:



These balls add a new behavior to the Regular ones, which is that they can collide into each other. Each time they collide or bounce within the limits of their habitat (space) they emit sound. When there are a lot of balls it produces a special sound, similar to the one generated by granular synthesis.

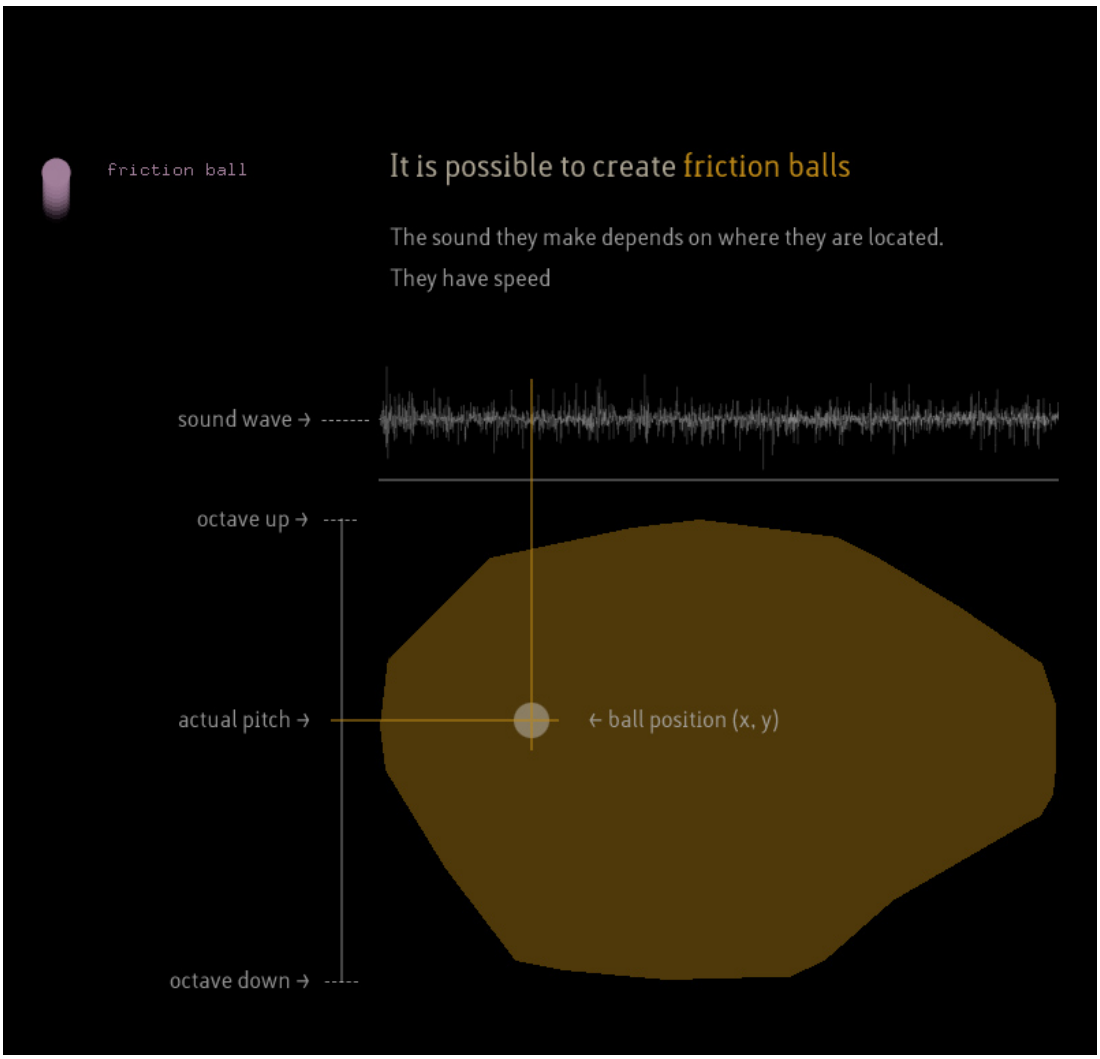
They are completely opaque and when they collide they produce repetitive waves that fade out and become bigger while they disappear. This represents how sound propagates when there is a collision, similar to when, for example, a stone is thrown into a river.

Mass Balls:



Mass balls can be big or small and it is possible to change their size while they are being created. If they are also a collision ball they are stronger when they collide with others balls that are smaller in size. If they are bigger that the space they belong, they produce strange and random noises, which produces some interesting textures. The sound they produce has a high level volume. They move slowly.

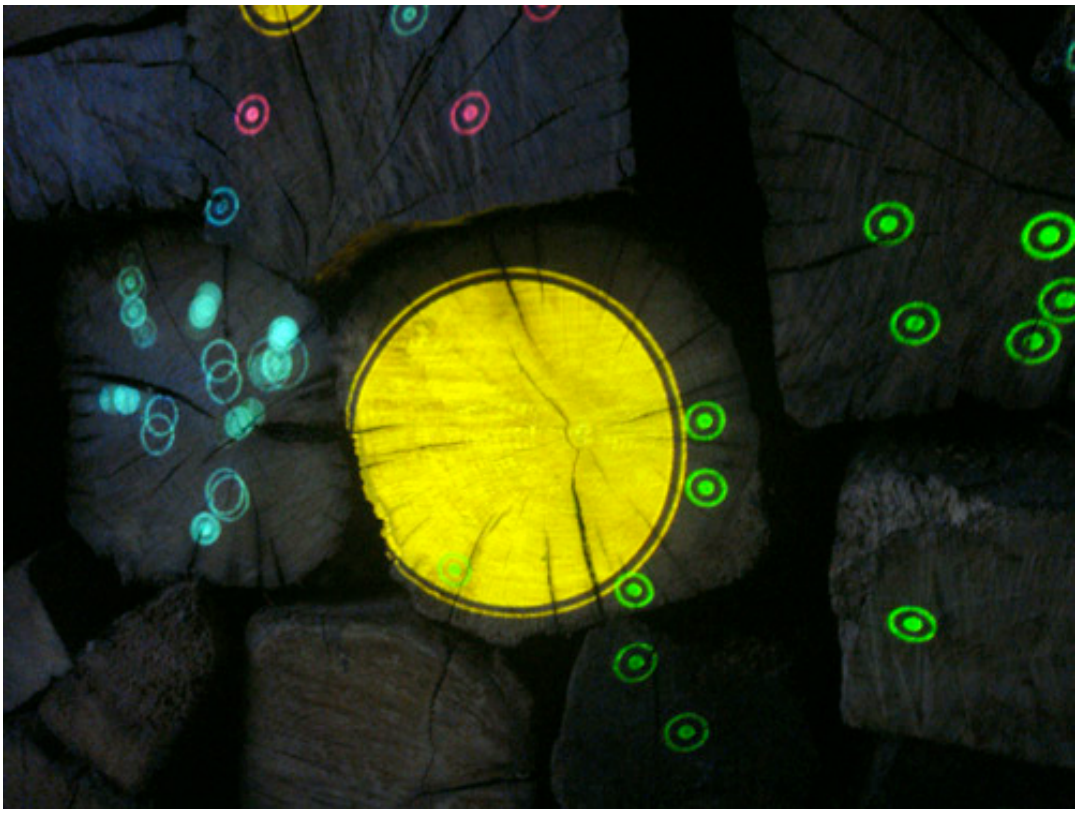
Friction Balls:

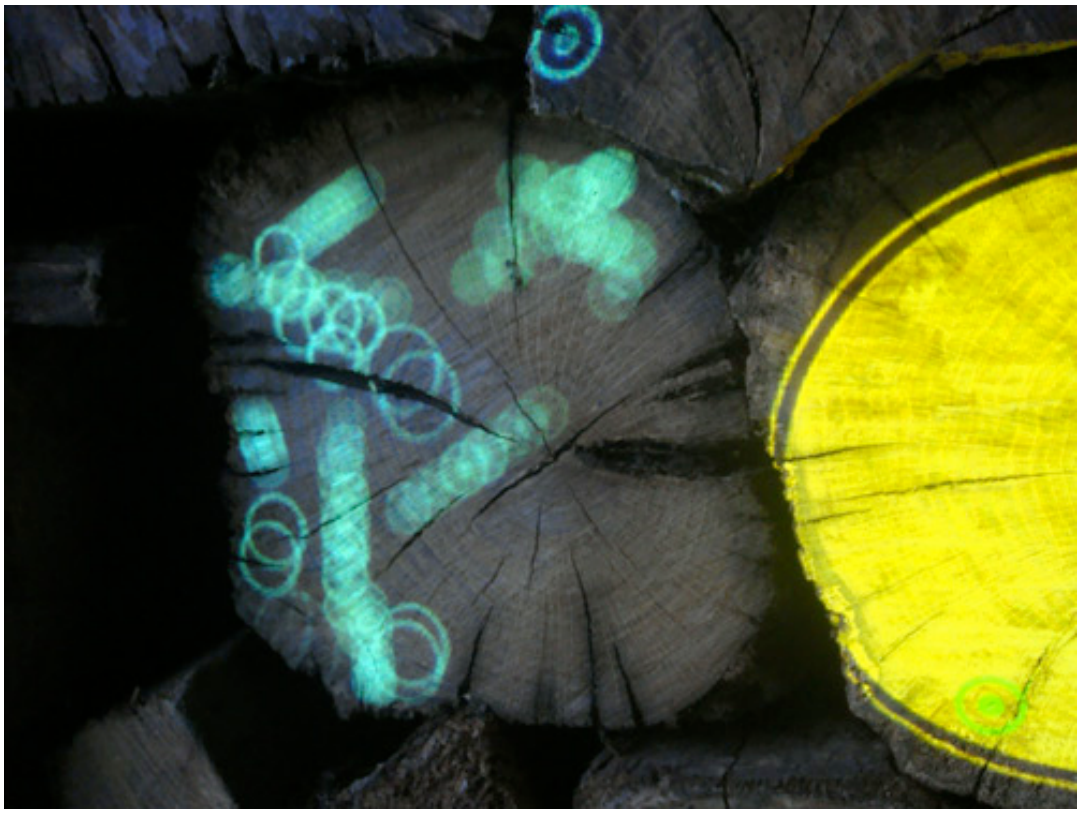


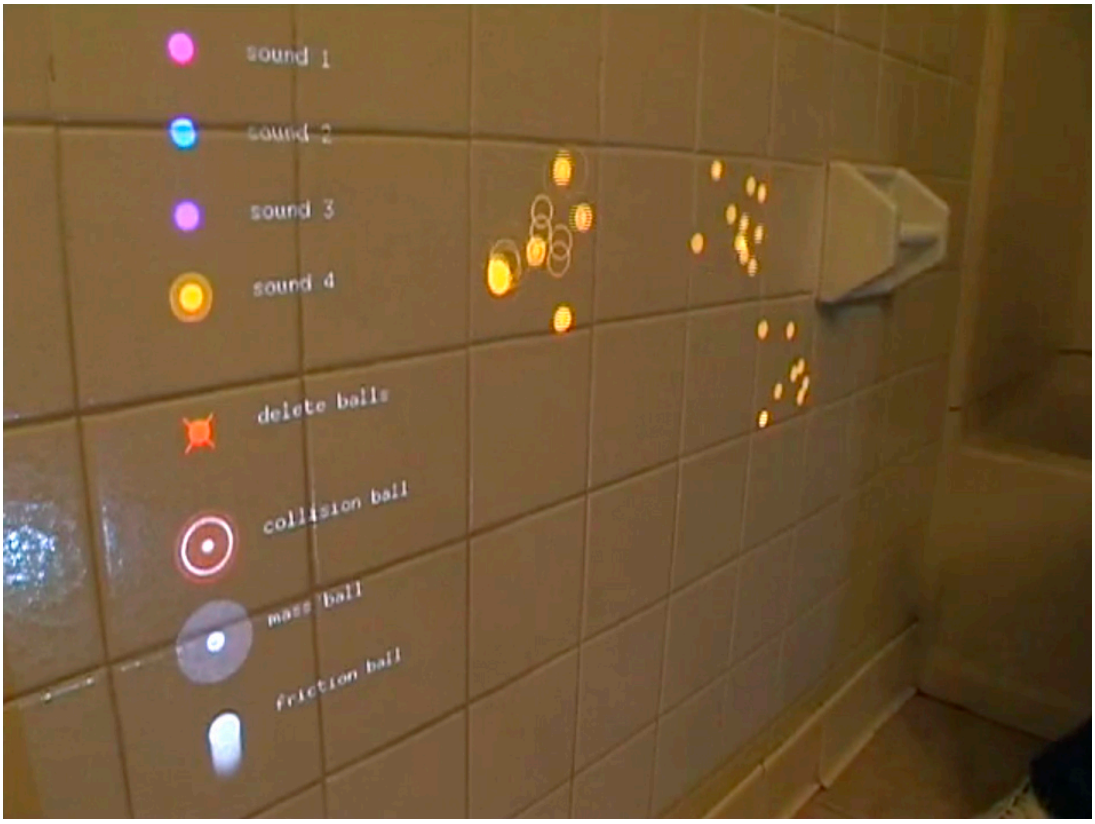
Friction balls are different for two reasons:

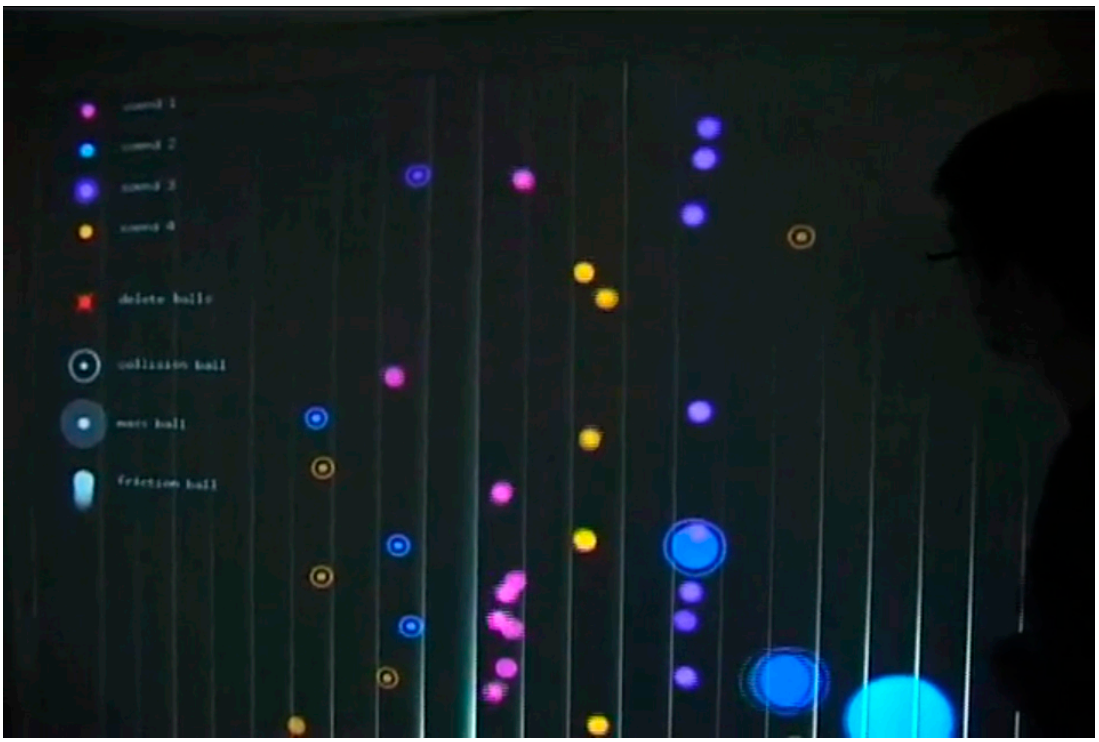
First, they produce a new kind of sound that is continuous and that is modified depending on where the ball is located in the space and its velocity. If the ball has a low "y" coordinate, its frequency is higher and vice-verse. Also, the x coordinate of the ball indicates where the head player of the sound is located. In relation with velocity, if the ball goes fast the sound also plays fast and if the ball goes slow the velocity of the sound will also decrease.

This ball has a tail that indicates velocity. If the tail is large this means that the ball goes fast and if the tail is short this indicates that it goes slow.






sound 1
sound 2
sound 3
sound 4
delete balls
collision ball
mass ball
friction ball



5.4. Sound Design

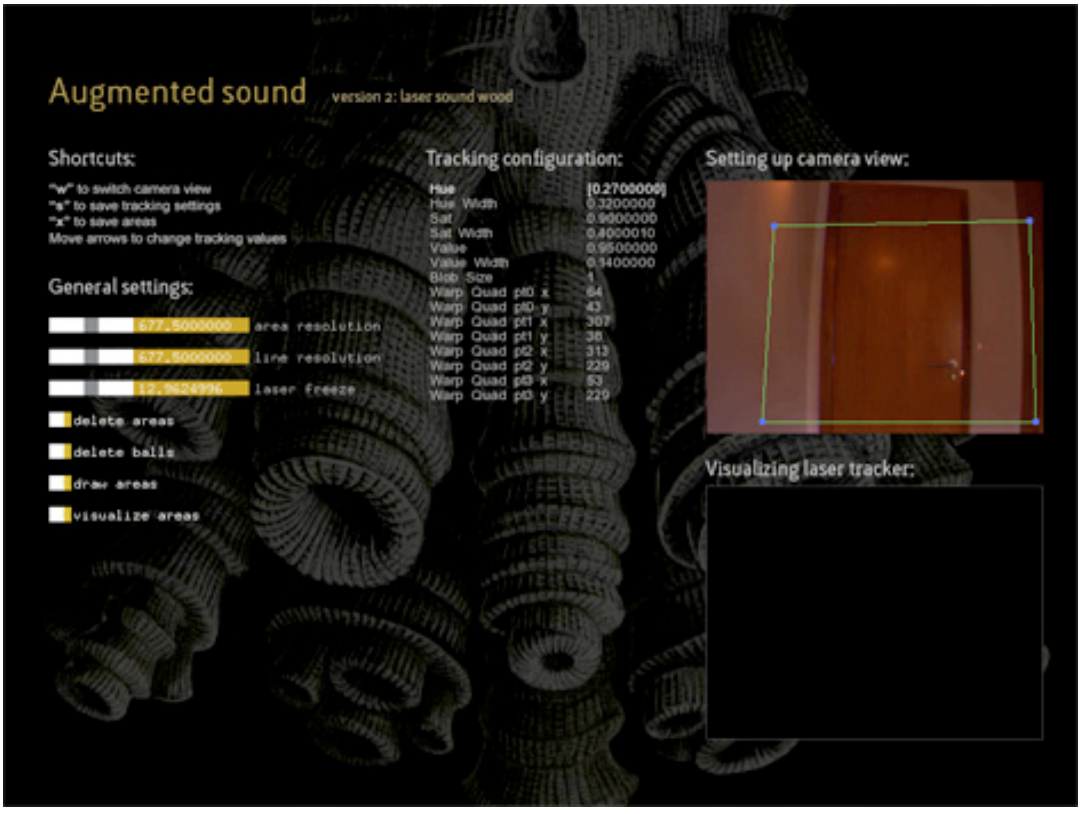
Sound Design follows one big concept that is to expand and augment the sound generated by the object where the projection is projected. In other words, imagine that there is a glass bottle. This bottle emits a sound when it is snapped with a fork, for instance. Now imagine that instead of using a fork we use the thesis balls, and imagine that when they hit the bottle, they then emit a sound. However, the sound that they emit is not the sound of the fork knocking on the glass, but it is the sound of a virtual ball hitting a physical bottle. So the sound should augment the natural sound of the bottle according to the behaviors that we previously defined.

This idea of expanding physical sound by adding a virtual behavior related to the physical aspects of the objects that emits the “real” sound is called: Augmented Sound.

So, all the sound that can be heard while interacting with the installation are filed recordings of the real object where the projection is projected. For example, if we are projection on top of a curtain, all sounds that are generated by balls interacting with it, are live manipulations made by those balls from previous recordings.

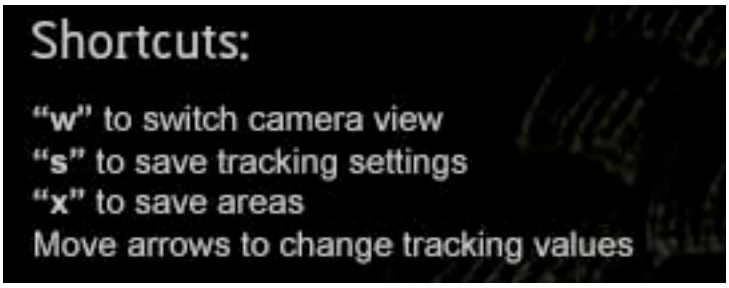
5.5. Configuration Window.

In order to setup the whole interaction, a configuration window was designed where parameters like, HSV values of laser tracking, coordinates of the warper vertex, etc... can be adjusted.



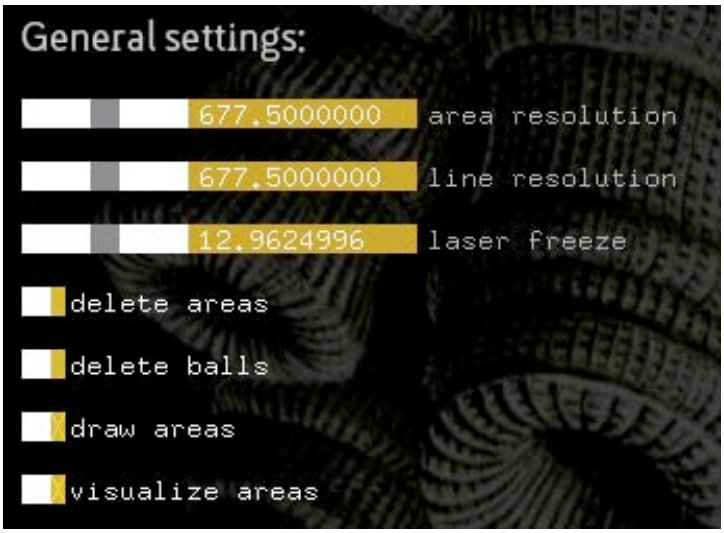
It is divided in 5 sections.

1. Shortcuts



The first section called shortcuts, show us what keys of the keyboard have special actions:
"w" key switches between the camera view, which makes possible to see the video that is being taken from the camera and the laser tracking view, which makes possible to see the blob generated by the laser tracking algorithms.
"s" key is used to save laser tracking configuration. It means that it saves the HSV values and the warping rectangle vertex points.
"x" key is used to save areas position and dimensions (they are going to be explained in more detail later on)
"arrow keys" are used to navigate throughout HSV values in order to change them.
"d" key is used to delete areas (to do that, it is necessary to point the laser to an area and press "d").
"m" key is used to move areas (the laser must be pointed to one area, and then the “m” key should be pressed. Then it is possible to move the area while moving the laser).

2. General Settings



This section of the configuration window is used to change some general configuration settings.

"Area resolution" is used to increase or decrease the number of points of the area. When there is an organic shape, for instance a leaf, it would be interesting to increase resolution (that will add more points), but when there is a less organic shape, for instance a rectangle, it would be interesting to decrease resolution = less points.

"Line resolution" is used to make a line that is drawn when areas are created to look smoother. It is not really important.

"Laser freeze" is a timeout used to freeze the laser tracking blob. It is useful when laser tracking is not good enough due to light conditions, laser low battery.

"delete areas" is a button that deletes all areas when pressed.

"delete balls" is a button that deletes all balls when pressed.

"draw areas" is a switch-toggle button that allows drawing areas during configuration stage.

"visualize areas" is a toggle button that allows to visualize the areas and change their shape.

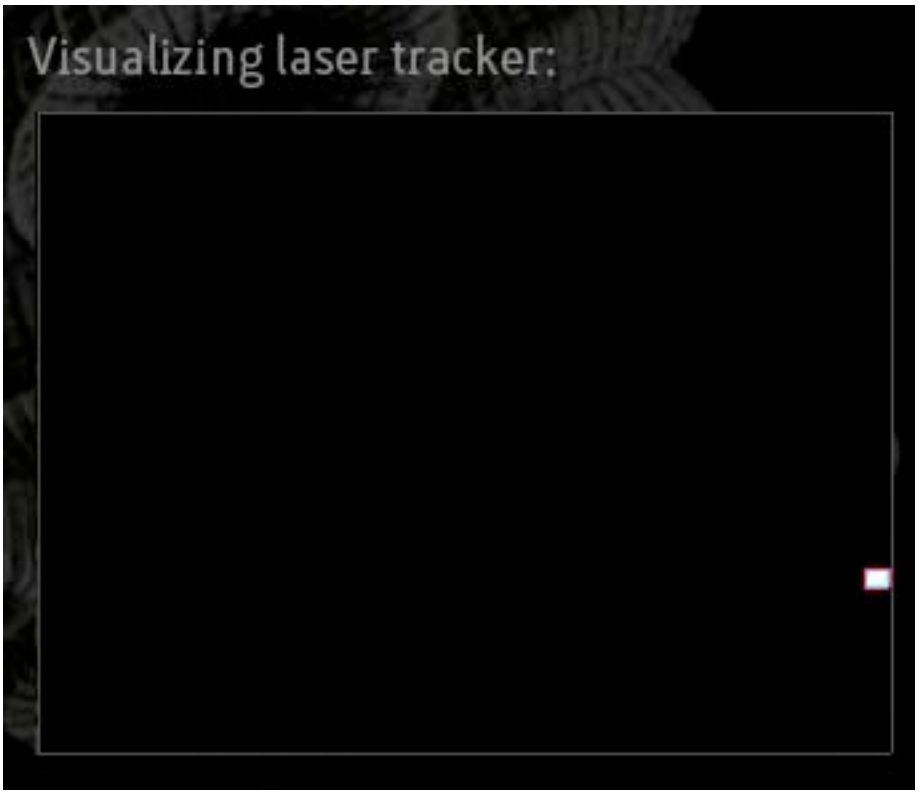
3. Tracking configuration

Here is where we change the HSV values for tracking the laser. Moving the arrow keys up, down, left and right these values can be changed. Moreover here is where the x an y coordinates of the warp rectangle of the right can be visualized.

4. Setting up camera view:

This section is used to warp camera frame. In order to change the warping rectangle it is necessary to drag its vertex by using a mouse. Vertex is drawn as blue circles.

5. Visualizing laser tracking:



Here is where we visualize the laser. It is possible to switch between the camera view and the tracking view by pressing the "w" key.

6. Conclusion

This thesis is about creating an interactive system that allows people to manipulate sound by interacting with the physical characteristics of objects without touching them. Technically it plays with video projections that interact with the shapes of the objects where they are projected and their boundaries, as well as with the content of the projection itself.

At a very early stage of this project, I was planning to do something much bigger, especially because I wanted to develop the whole project for interacting with building facades, with a lot more functions for user to interact with. Then I soon realized I was too ambitious. I went in another direction and I started to pay attention to little things and little objects. That was the moment when the whole thesis concept took form and the idea of expanding sound aesthetic reality of objects using just their shape appeared. I drew some sketches to imagine how that could be done and I tried to solve all the problems stated in the sketches.

That was where I had the biggest conceptual breakthrough for this thesis. I had some clear ideas about what I wanted to do and I had some detailed concept sketches but I didn't know how to bring the concept to life. It took me a long time to realize how to track the green color of a laser, and how to memorize shapes that were drawn by the laser, for instance, but I tried to do exactly what was presented in those initial sketches. I didn't know anything about programming in c++, and I barely knew about object oriented programming. My background was fine arts so it was an enormous challenge to achieve these goals.

When I started to think about this thesis, as a CGD-major student, my idea was that it should not necessarily be related to programming. I thought it was a great opportunity to delve deep into interactive sound related interactive graphics and I thought it was the ideal opportunity to improve my programming skills. It worked out really well and in the process of researching for this thesis I learned a lot in the field of interactive multimedia, to the extent that it is critical knowledge for the job that I have today. Moreover i have created and maintain relationships with members of the community related to Openframeworks, that I maintain nowadays.

During my thesis presentation at the Imagine RIT Exhibit, the interactive project was manipulated for a relatively large audience and they all enjoyed playing and interacting with it. They liked how the balls bounced with the boundaries of the objects to make sound, which was the primary objective of the project.

This thesis project was also an exhibit at Sonar Festival and Vigotransforma Festival in Spain. In both cases the project was used and projected on buildings facades, bringing to life the original concept which this thesis was based upon, and it came out well.

7. Bibliography

Maeda, J. (2004). *Creative Code*. CA: Thames & Hudson

O`Sullivan, D., Igoe, T. (2004). *Physical computing*. CA: Thomson Course Technology

Greenberg, I. (2007). *Processing. Creative Coding and Computational Art.* CA: Friendsof

Reas, C., Fry, B. (2007). *Processing. A programming handbook for visual designers and artists*. CA: The MIT Press

William Flake, G. (2000). *The Computational Beauty of Nature*. CA: The MIT Press

Fry, B. (2007). Visualizing data. CA: O'Reilly

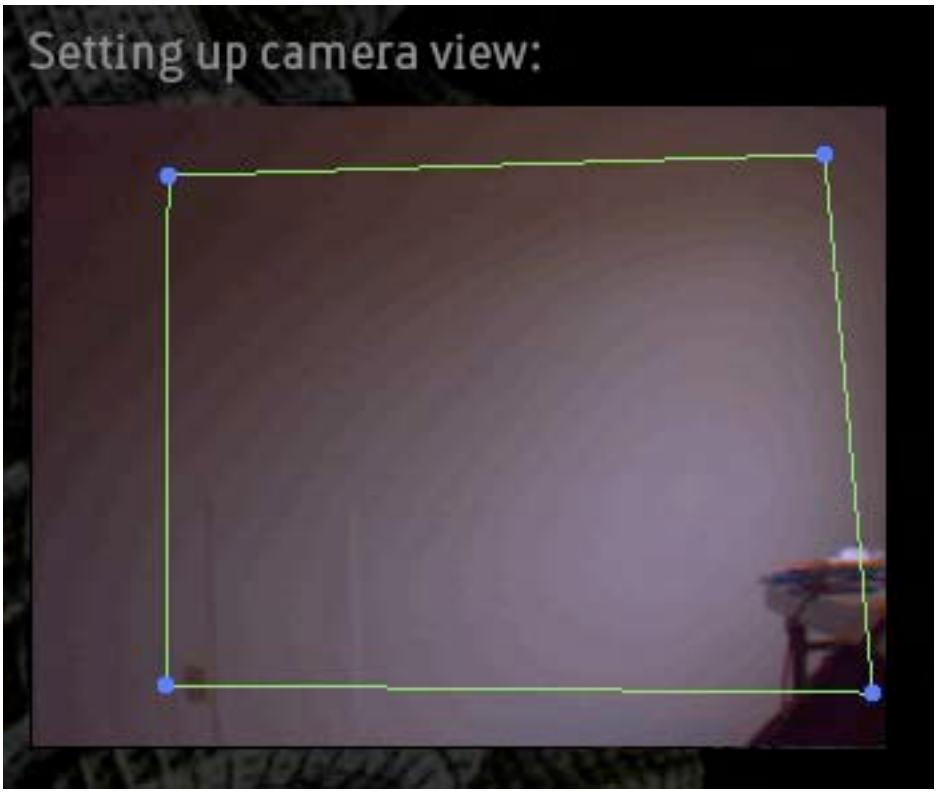
Igoe, T. (2007). *Making things talk*. CA: O'Reilly

Appendix A. Coding

Openframeworks

Warping Camera View:

In order to match the projection with the physical space and the image that records the camera it is necessary to move the boundaries of the camera frame to match the boundaries of the frame that is projected. In order to achieve this we created a user interface by using four points that draw a rectangle, making it possible to adapt the frame of the image that is recorded by the camera to match the projection.




```
1.  // HSV
2.  void laserTracking::mousePressed(int x, int y, int button){
3.      //this is not the best way
4.      activePointIn       = -1;
5.
6.      float smallestDist  = 999999;
7.    float clickRadius = 10;
8.
9.    for (int j = 0; j < 4; j++){
10.           ofPoint inputPt;
11.           inputPt.x = srcPositions[j].x + topLeftPos.x;
```

```
            inputPt.y = srcPositions[j].y + topLeftPos.y;
            inputPt.z = 0;
            float len = sqrt( (inputPt.x - x) * (inputPt.x - x) +
                        (inputPt.y - y) * (inputPt.y - y));
            if (len < clickRadius && len < smallestDist){
                   activePointIn  = j;
                   smallestDist = len;
            }
   }
}

void laserTracking::mouseDragged(int x, int y, int button){
    if (activePointIn > -1){
        srcPositions[activePointIn].x = x - topLeftPos.x;
        srcPositions[activePointIn].y = y - topLeftPos.y;
   }
}
```

Laser tracking:

For doing laser tracking, the aplication first has to convert the incoming image from the camera to HSV values. Then it looks pixel by pixel to determine if anyone of them is between the maximum and minimum HSV values of green color, that where setup previously. If they are inside this range they are converted to white color, and if they are not, they are converted to black. Then the dilate function is called to make the blob (a group of pixels of the same color that are neighbors) bigger and finally findCountour function is called to find the blob, that will be the pointer for the lasers:

```
// HSV
colorImgHSV = colorImg;
colorImgHSV.convertRgbToHsv();
colorImgHSV.convertToGrayscalePlanarImages(hueImg, satImg, valImg);

// Perform tracking calculations
unsigned char * huePixels = hueImg.getPixels();
unsigned char * satPixels = satImg.getPixels();
unsigned char * valPixels = valImg.getPixels();
int nPixels = widthInput * heightInput;

// Look every pixel in order to find the one between min and max HSV values
previously setup
for (int i = 0; i < nPixels; i++){
    if ((huePixels[i] >= minHue && huePixels[i] <= maxHue) &&
        (satPixels[i] >= minSat && satPixels[i] <= maxSat) &&
        (valPixels[i] >= minVal && valPixels[i] <= maxVal)){
        colorTrackedPixels[i] = 255;
    } else {
        colorTrackedPixels[i] = 0;
```

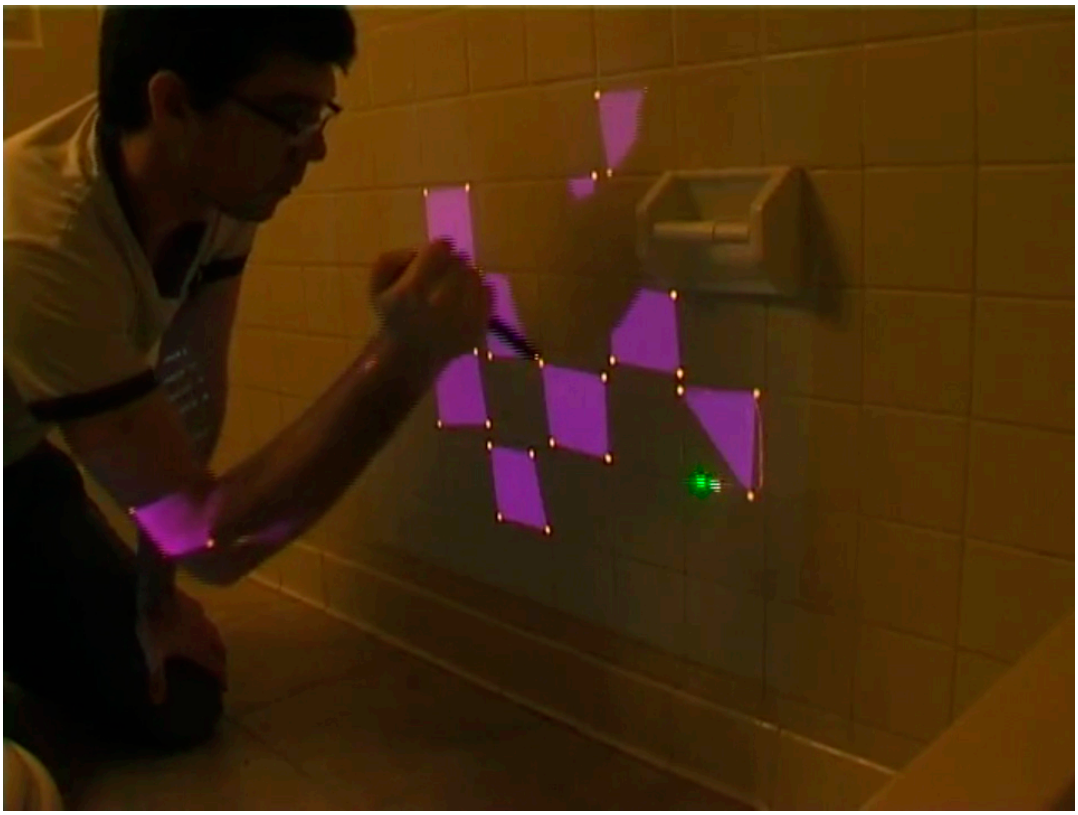
```
    }
}

trackedTexture.setFromPixels(colorTrackedPixels, widthInput, heightInput);
// dilate white pixels, it will make the blob bigger.
trackedTexture.dilate();
trackedTexture.dilate();
warpedTrackedTexture.warpIntoMe(trackedTexture, srcPositions, dstPositions);

// ******Blobs
// find contours which are between the size of 30 pixels and 1/3 the w*h pixels.
contourFinder.findContours(warpedTrackedTexture, blobSize,
    (widthInput*heightInput)/3, 30, false);
```

Interacion areas:

The mapping effect is done by drawing the areas that are going to be interaction spots. By pointing the laser and moving it along the boundaries of the space limited by its shape, the program records points that later draw the shape. Once these areas are drawn and set, we can adjust them by moving their points.



```
//----- Function that creates areas. If we are already creating an area, we add points to that area, if not, we create a new area and add it to the areaList vector
void imageProjection::createArea(){
  if(letCreateArea){
      if(checkLaser){
          if(creatingArea){
              areaList[areaOnCreation].build(laserPosition);
          }else{
              areaList.push_back(area());
              areaOnCreation = areaList.size()-1;
              areaList[areaOnCreation].create(laserPosition, areaResolution);
              creatingArea = true;
          }
      }else{
          if(creatingArea){
              areaList[areaOnCreation].finishBuilding();
          }
          creatingArea = false;
      }
  }
}
```

```
//----- Function that creates one area, located at area class. It sets an initial point at laser coordinates and a resolution value that is how often it creates a point.
void area::create(ofPoint _lPos, float _ar){
  setAreaVertex(_lPos);
  setLineVertex(_lPos);
  setAreaTimer = ofGetElapsedTimeMillis();
  setLineTimer = ofGetElapsedTimeMillis();

  areaResolution = _ar;
}
```

```
//----- Function that builds an area. It adds new points to the area by calling the function setAreaVertex
void area::build(ofPoint _lPos){
  if((ofGetElapsedTimeMillis() - setAreaTimer) > areaResolution){
           setAreaVertex(_lPos);
           setAreaTimer = ofGetElapsedTimeMillis();
  }
  // Set line vertex sequenced.
  if((ofGetElapsedTimeMillis() - setLineTimer) > LINERESOLUTION){
           setLineVertex(_lPos);
           setLineTimer = ofGetElapsedTimeMillis();
  }
  drawCreationLine(_lPos);
```

```
13. }
```

```
1.  //----- Function that add points to an area
2.  void area::setAreaVertex(ofPoint _lPos){
3.
4.    // We add a new point to the Area's vertex array (type vector).
5.    areaVertexList.push_back(_lPos);
6.    // We set the max and min coordinates of the area
7.    setSurroundingBox(_lPos);
8.  }
```

Bounce balls:

Bounce, friction and mass balls all extend ball class. Bounce balls are the more complex of them because we have to deal with non orthogonal collision. To resolve this, we followed the code that Ira Greenberg wrote on Processing: Creative Coding and Computational Art (Foundation)

```
1.  // Loop through all the balls
2.  for(int i=0; i<ballList.size(); i++){
3.      // check if the current ball is a bounce ball
4.      if(ballList[i].getBounce()){
5.          // create another loop to have more balls to compare with
6.          for(int j=i+1; j<ballList.size(); j++){
7.              if( ballList[j].getBounce() && (ballList[i].getIsMoving() || ballList
    [j].getIsMoving()) ){
8.                  // we call the functions that manage all the collision work.
9.                  ballList[i].checkObjectCollision(ballList[j]);
10.                 ballList[j] = ballList[i].getCollBall();
11.
12.                 if(ballList[i].getKillMe() && ballList[i].getDissapeared()){
13.                     ballList[i].killSynth();
14.                     ballList.erase(ballList.begin()+i);
15.                 }
16.             }
17.         }
18.     }
19. }
```

```
1.  // Main collision function
2.  void ball::checkObjectCollision(ball oBall){
3.
4.      if(creationFinished && oBall.creationFinished){
5.          ofVec2f bVect;
6.          bVect.set(oBall.x - x, oBall.y - y);
7.          float bVectMag = sqrt(bVect.x * bVect.x + bVect.y * bVect.y);
8.
9.          if(bVectMag < r + oBall.r && !dissapear && !oBall.getDissapeared()){
```

```
10.
11.             if(collided){
12.                 dissapear = true;
13.             }
14.             if(oBall.getCollided()){
15.                 oBall.setDissapeared(true);
16.             }
17.
18.             if(!collided && !dissapear && !oBall.getCollided() && !
    oBall.getDissapeared()){
19.
20.                 // We manage audio and visualization staff. We create a wave ball
    to draw and see expansive waves that surrender the ball
21.                 if(ofGetElapsedTimeMillis() - extressTime > 300){
22.                     setPan();
23.                     sendOscMessage("collision");
24.                     oBall.setPan();
25.                     oBall.sendOscMessage("collision");
26.                     waveList.push_back(wave());
27.                     waveList[waveList.size()-1].setup(x, y, r, cYr, cYg, cYb);
28.                     oBall.addWave(oBall.x, oBall.y, oBall.r, oBall.getRcolor(),
    oBall.getGcolor(), oBall.getBcolor());
29.                     //dissapear = true;
30.                     extressTime = ofGetElapsedTimeMillis();
31.                 }
32.
33.                 // Math that deal with collision
34.                 float theta  = atan2(bVect.y, bVect.x);
35.                 float sine = sin(theta);
36.                 float cosine = cos(theta);
37.
38.                 ball bTemp0;
39.                 ball bTemp1;
40.                 bTemp1.x  = cosine * bVect.x + sine * bVect.y;
41.                 bTemp1.y  = cosine * bVect.y - sine * bVect.x;
42.
43.                 // rotate Temporary velocities
44.                 ofVec2f vTemp0;
45.                 ofVec2f vTemp1;
46.                 vTemp0.x  = cosine * v.x + sine * v.y;
47.                 vTemp0.y  = cosine * v.y - sine * v.x;
48.                 vTemp1.x  = cosine * oBall.v.x + sine * oBall.v.y;
49.                 vTemp1.y  = cosine * oBall.v.y - sine * oBall.v.x;
50.
51.                 ofVec2f vFinal0;
52.                 ofVec2f vFinal1;
53.                 // final rotated velocity for b[0]
54.                 vFinal0.x = ((m - oBall.m) * vTemp0.x + 2 * oBall.m * vTemp1.x)
```

```
                / (m + oBall.m);
                vFinal0.y = vTemp0.y;
                // final rotated velocity for b[0]
                vFinal1.x = ((oBall.m - m) * vTemp1.x + 2 * m * vTemp0.x)
                / (m + oBall.m);
                vFinal1.y = vTemp1.y;

                // hack to avoid clumping
                bTemp0.x += vFinal0.x;
                bTemp1.x += vFinal1.x;

                // rotate balls
                ball bFinal0;
                ball bFinal1;
                bFinal0.x = cosine * bTemp0.x - sine * bTemp0.y;
                bFinal0.y = cosine * bTemp0.y + sine * bTemp0.x;
                bFinal1.x = cosine * bTemp1.x - sine * bTemp1.y;
                bFinal1.y = cosine * bTemp1.y + sine * bTemp1.x;

                // update balls to screen position
                oBall.x = x + bFinal1.x;
                oBall.y = y + bFinal1.y;
                x = x + bFinal0.x;
                y = y + bFinal0.y;

                // update velocities
                v.x = cosine * vFinal0.x - sine * vFinal0.y;
                v.y = cosine * vFinal0.y + sine * vFinal0.x;
                oBall.v.x = cosine * vFinal1.x - sine * vFinal1.y;
                oBall.v.y = cosine * vFinal1.y + sine * vFinal1.x;
            }

            collided = true;
            oBall.setCollided(true);

        }else if(collided){
            collided = false;
        }else if(oBall.getCollided()){
            oBall.setCollided(false);
        }
    }
    ballCollList[0] = oBall;
}
```

OSC communication:

OSC is the protocol used for communicating the application that deals with graphics and interaction (Openframeworks) with the application that deals with audio (Supercollider). Openframeworks works as the sender and Supercollider as the receiver (although they both send and receive).

Sender: Openframeworks

OSC messages are sent by the balls when they are created, when they collide between them or with other balls, when they are of type “friction ” and they move through the screen and they die. they use the function sendOscMessage() that is located in the ball class.

```
1.  // send regular OSC message
2.  void ball::sendOscMessage(string _type, string _action){
3.      if(_type == "collision"){
4.          ofxOscMessage m;
5.          m.setAddress("/collision");
6.          m.addStringArg("true");
7.          m.addIntArg(sample);
8.          m.addFloatArg(vol+0.4);
9.          m.addFloatArg(pan);
10.         senderOsc.sendMessage(m);
11.     }
12.     if(_type == "friction"){
13.         ofxOscMessage w;
14.         w.setAddress("/friction");
15.         w.addStringArg(_action);        // parameter 1
16.         w.addIntArg(myId);              // parameter 2
17.         w.addIntArg(mySoundId);         // parameter 3
18.         w.addIntArg(sample);            // parameter 4
19.         w.addFloatArg(vol);             // parameter 5
20.         w.addFloatArg(pan);             // parameter 6
21.         w.addFloatArg(position);        // parameter 7
22.         w.addFloatArg(pitch);           // parameter 8
23.         senderOsc.sendMessage(w);
24.     }
25.     if(_type == "kill"){
26.         ofxOscMessage o;
27.         o.setAddress("/kill");
28.         o.addStringArg("true");          // parameter 1
29.         o.addIntArg(myId);               // parameter 2
30.         o.addIntArg(mySoundId);          // parameter 3
31.         senderOsc.sendMessage(o);
32.     }
33. }
```

```
1.  // send OSC message for deleting all balls
2.  void imageProjection::sendDeleteAllSoundsOscMsg(){
```

```
    ofxOscMessage killAllMsg;
    killAllMsg.setAddress("/killall");
    killAllMsg.addStringArg("true");            // 1
    sender.sendMessage(killAllMsg);
}
```

```
// Receive OSC message
void imageProjection::receiveOsc(){
    while(receiver.hasWaitingMessages()){
        ofxOscMessage m;
        receiver.getNextMessage(&m);
        if(m.getAddress()=="/soundId"){
            int ballId = m.getArgAsInt32(0);
            int soundId = m.getArgAsInt32(1);
            ballList[ballId].setSoundId
(soundId);
        }
    }
}
```

Receiver: Supercollider

```
//SETUP
(
// read sound files into buffers
m = Array.with(Buffer.read(s, "sounds/friction01.aiff"), Buffer.read(s, "sounds/
friction_02.aiff"), Buffer.read(s, "sounds/friction_03.aiff"), Buffer.read(s,
"sounds/friction_04.wav"));

n = Array.with(Buffer.read(s, "sounds/01/01.aif"), Buffer.read(s, "sounds/
02/02.aif"), Buffer.read(s, "sounds/03/03.aif"), Buffer.read(s, "sounds/
04/04.aif"));
b = Buffer.read(s, "sounds/04/01");

// Synths List
l = List.new;

// OSC
// Send Port and setup
y = NetAddr.new("127.0.0.1", 12345);

// Receive Port and setup
z = NetAddr.new("127.0.0.1", 57120);

// *** Play buffer ** S N A P S **
```

```
SynthDef(\playBuf,{ arg out = 0, bufnum, amp=0.7, pan=0.0;
  var signal, env;
  env = EnvGen.ar(Env.perc, doneAction:2); // envelope
  signal = PlayBuf.ar(1, bufnum, BufRateScale.kr(bufnum), doneAction:2) * env;
  signal = Pan2.ar(signal, pan) * amp;
  Out.ar(out, signal);
}).load(s);

// *** Play buffer ** W A R P S **
// pointer between 0 an 1
// pitch between 0.5 and 2.0 (octave down - octave up)
SynthDef(\warp, {arg out = 0, bufnum, amp=0.7, pan=0.0, pointer = 0.1, pitch =
1.0, gate = 1;
  var signal, env;
  env = EnvGen.ar(Env.asr(0.001, 2, 3, -3), gate);
  signal = Warp1.ar(1, bufnum, pointer, pitch, 0.1, -1, 8, 0.15, 1.0);
  signal = Pan2.ar(signal, pan) * amp * env;
  Out.ar(out, signal);
}).send(s);

// Receive OSC and play ** S N A P ** the sound
// /collision msg[]:
// msg[1] = true / false
// msg[2] = sample
// msg[3] = vol
// msg[4] = pan
// msg[4].postln;
o = OSCresponder.new(nil, "/collision", { |time, resp, msg|
  if(msg[1] == 'true'){
        x = Synth(\playBuf, [\bufnum, n[msg[2]].bufnum, \amp, msg[3], \pan, msg
[4]]);
  };
}).add;

// Receive OSC and ** KILL ** the synth
// /kill msg[]:
// msg[1] = true / false
// msg[2] = ballId
// msg[3] = soundId
q = OSCresponder.new(nil, "/kill", { |time, resp, msg|
  if(msg[1] == 'true'){
        l[msg[3]].free;
  };
}).add;

// Receive OSC and ** KILL ALL ** the synths
// /killall msg[]:
// msg[1] = true / false
```

```
r = OSCresponder.new(nil, "/killall", { |time, resp, msg|
  if(msg[1] == 'true'){
        l.do({ arg item, i; item.free; });
        l.clear;
  };
}).add;

// Receive OSC and ** W A R P ** the sound
// /friction msg[]:
// msg[1] = 0 / 1 / 2 (starts / set / stops)
// msg[2] = ballId
// msg[3] = soundId
// msg[4] = sample
// msg[5] = vol
// msg[6] = pan
// msg[7] = pointer
// msg[8] = pitch
p = OSCresponder.new(nil, "/friction", { |time, resp, msg|
  if(msg[1] == 'play'){
        w = Synth(\warp, [\bufnum, m[msg[4]].bufnum, \ballid, msg[2], \soundId,
  msg[3], \amp, msg[5], \pan, msg[6], \pointer, msg[7], \pitch, msg[8]]);
        l.add(w);
        // send "/soundId": ballId, soundId
        y.sendMsg("/soundId", msg[2], l.size-1);
  };
  if(msg[1] == 'update'){
        l[msg[3]].set(\pan, msg[6], \pointer, msg[7], \pitch, msg[8], \gate, 1);
  };
  if(msg[1] == 'stop'){
        l[msg[3]].set(\gate, 0);
  };
}).add;

// Background bass sound
SynthDef(\playBufMonoLoop, {| out = 0, bufnum = 0, rate = 1 |
  var scaledRate, player;
  scaledRate = rate * BufRateScale.kr(bufnum);
  player = PlayBuf.ar(1, bufnum, scaledRate, loop: 1, doneAction:2);
  Out.ar(out, player)
}).play(s, [\out, 0, \bufnum, b.bufnum, \rate, 1]);
)
```

Appendix B. Thesis Proposal

Thesis Proposal

A computer vision system for doing interactive-creative musical installations: A portable computer vision system based on video projection that allows the interaction with the projection itself, with the physical space where it is projected and with multiple users.

Abstract

The thesis is about building a computer vision prototype for allowing interaction using a video projection in order to find new ways for playing sounds. The interaction is done between the projection and the physical surface where it is projected, the user and the content of the projection. It is done using the c++ Openframeworks library, a video projector, an infrared light, puredata or max/msp for manipulating sound and some kind of physical interface controller like Laser or wii. The areas that are going to be applied in this thesis are interactive installation, physical computing, GUI, computer vision, creative coding and new interfaces for musical expression.

Problem Statement

Basically, what this thesis is achieving is transforming the video projection into a musical user interface. We are not talking here about the content of the video projection but the projection itself. Usually, the surface where a video, game, animation, interactive work, etc is projected is a solid plane without any kind of significant information. Normally it is a white wall, or a screen specially designed for receiving the light of the projector, but rarely the space where the images are projected is considered as one more elements that could be used with aesthetics purposes. This becomes especially true when we talk about musical interfaces. Therefore, this thesis will be researching about how the content of a video projection will interact with the physical space of the surface or object where this content is projected, and also how users can interact with both space and content projection for making music. In addition, the system must be mobile and able to adapt to different physical spaces fast, giving the idea of an autonomous adaptable system. For instance, imagine that somebody is projecting and interactive animation about several balls moving through the screen. Imagine that this animation in being projected on the facade of a building, and each time any ball hits a window, it change its direction and make sound. Also imagine that people from the street can create more balls just pointing with their hand where they want to create the balls, or thatpeople can use the windows like buttons. This is basically what this thesis is about, a system for developing interactive musical installations that will deal with the interaction between the projection itself, the physical space where images are projected and the people that is controlling what is projected.

For solving this problem, there will be used the concepts under the computer vision technology using infrared cameras, laser, video projectors and maybe other kind of physical interfaces like the wii remote. The software that will be used for mapping and interpreting the data from the physical space will be a c++ library called Openframeworks that is being developed by Zachary Lieberman and

Theodore Watson, and that is being used for several creators for building interactive artistic works and for doing experimental creative code. Moreover, because this library is on its begging phase of development, it will be also an additional goal to contribute to extend its possibilities.
This thesis idea came after combine 3 projects about computer vision and interaction. The first one was the Graffiti research lab project, developed at the eyebeam lab by F.A.T (Free art and technology), and using the Openframeworks library. This project is about doing graffiti in public spaces using a laser and a video projector. The main idea that was interesting for the thesis is how people interact with the projector and what kind of physical interface they use. Moreover it is interesting the software that they use and because it is open, how easy is work with it to push it further. The second one was the project called Artificial Dummies done by the Italian group called DOMO. This project is about a projection of some animations that interact with facades of buildings. The idea that was taken from this project is the interaction between the projection and the physical space
upon which it is projected. The last one was the project called Funky forest by Theodore Watson, where people interact with a video projection. The idea taken from this project is the interaction between the projection and people.
After doing some research, I found that the Graffiti research lab has done some investigations with the combinations of these interactions (physical space, video projector and people) that they applied in the project called Interactive Architecture. This is the reason I would like to push these ideas further, using this technology for also creating music and for playing not only with architectural spaces but also with other kind of physical surfaces.

Scope
The areas that are going to be applied in this thesis are interactive installation, physical computing, GUI, computer vision, creative coding and new interfaces for musical expression. I am going to deal with tools for doing physical interfaces like using laser for controlling graphics or maybe other kind of remote controls like wii. It will also be important dealing withGUI concepts in order to design new interfaces for interacting with the physical spaces. The technology for taking data from the physical space is going to be computer vision, using infrared or regular cameras and concrete computer vision techniques like detecting bobs. Because this is going to be a kind of musical interface, it is interactive, uses new technology and force people to play music on a new different way. It can also be consider that this thesis project is also dealing with concepts from NIME (new interfaces for musical expression).

Literature Survey

Books

Making things talk
by Tom Igoe
O'Reilly 2007
This book is about connecting several hardware and software in order to communicate them and share information. The final goal is helping on the technologic development of creative and interactive works. For instance, learning how to communicate two different softwares will help me to convert graphical data to sound data.

Physical computing
by Dan O`Sullivan and Tom Igoe
Thomson Course Technology 2004
This book had a special meaning for the independent artistic community because it put together the Do It Yourself (DIY) way of thinking and the new technology related to hardware. After reading this book a lot of people interested on art, began to develop interactive projects. So, this book talks about building physical interfaces to establish an interaction between machine and human. It was very useful to me in order to gain some basic knowledge about electronics.

Visualizing data
by Ben Fry
O'Reilly 2007
This book is about taking data from some kind of source and how to transform this data into something more readable for the user. This book is very useful for learning how to parse raw data, like for instance xml, csv or tsv files that are standards and everybody use them, in order to create interactive graphics for manage this data. It uses processing language (Ben Fry is one of the guys who is developing processing). It is also a very useful resource for building graphical user interfaces like buttons, maps,networks, graphs, etc... It will help me to learn how to read data from a source and convert it to something more. Because during the development of my thesis I will be dealing with different ways of visualize the same data, the knowledge I will gain from this book will become essential.

Processing. A programming handbook for visual designers and artists
by Casey Reas and Ben Fry
The MIT Press 2007
This is a manual for the programming language called Processing. It is written by the people who develop the Processing library, Ben Fry and Casey Reas. This book is not only a huge tutorial about

how to make computer graphics using code, but also it is a good reference book for learning concepts like movement, distance, time, physics, biology, color,... from the point of view of the math. Moreover, it is a good way for learning the basics of writing code. I have already read this book, and it was very useful to me because with this book I learned how to write code and its basic concepts.

Processing. Creative Coding and Computational Art
by Ira Greenberg
Friendsof 2007
This book is very similar to the one written by Casey Reas and Ben Fry, that I referenced before. It is about the processing code and explains how to use it. This book was also useful to me because it allowed me to approach the same topic from a different point of view. This combination of using two similar books for a same purpose was very useful because when I didn't understand what one book said, I had another resource to look for the same information, which gave me a more wide view in some aspects.

Creative Code
by John Maeda
Thames & Hudson 2004
Creative Code is a book written by John Maeda, and it shows works that students did under the MIT Media Lab umbrella. It is a good way to see works where people use technology in a complete innovative and strange way, just because they are trying to see the aesthetic side of the technology. I think that this is the main idea I could see on this book. How technology's aesthetic can expand the creative side of the technology and make us see things from a new point of view. It helped me to expand my conceptions about technology and don't be afraid on thinking out of the box.

Online Documentation

Openframeworks: Documentation
Interactive Architecture
Graffiti Research Lab
October, 2008
http://graffitiresearchlab.com/?page_id=32#video
This project is about doing generative graffiti interacting between a video projection and a physical space. This is the base from where the thesis will be built. This is the only project that I have found that combines the interaction between the projection, the physical space and the user, and because they use open technology, it is easy to implement what they have done and push it further.

Artificial.Dummies: A.I. enhanced graffiti

TODO

October, 2008

http://www.todo.to.it/projects.php?id=28&

The aim of this project is extend the idea of graffiti interacting with the physical surface where images are projected. They use kind of artificial intelligence that is just some flocking behaviors that interact with the physical space. This is the project that was the main inspiration for doing this thesis, because when I realized that projections could interact with the surfaces where they were projected, a whole new world appeared in front of my eyes.

Funky forest

Theodore Watson

October, 2008

http://muonics.net/site_docs/work.php?id=41

This is an interactive installation where the walls of a room are complete covered with video projections. Therefore, the people who enter the room interact with the projections, giving the sensation that they are interacting inside a new projected environment. I use this project as a reference on how the video projections can interact with people.

Zach Lieberman and Theodore Watson

September, 2008

http://www.openframeworks.cc/documentation

This is the documentation webpage of the Openframeworks library. The Openframeworks library was created to make easier write c++ code for doinginteractive and creative works. This is the library that I am going to use to develop the multitouch system and this is the software that I am going to use for receiving physical data and convert it to a graphical environment. This online documentation of the library is my primary resource that I look when I want to something about the library. It is like the Openframework's help.

Openframeworks: Forum

Openframeworks users

September, 2008

http://www.openframeworks.cc/forum

The Openframeworks forum is the way I keep in contact with other people that are using the same technology that I am. It is also very useful for keep myself updated with the new projects that people are developing. Moreover it is an indispensable place where I can ask for help, and also the first place where I will get feedback from my works. Depending on the interaction in the forum and what the other people tell me, my thesis change in several ways.

Touchkit

Nortd

September, 2008

http://touchkit.nortd.com/

Touchkit is a project developed by Nortd, a research and development studio from Austria and New York City. Working along with the Eyebeam lab in New York they have done a multitouch kit using the Openframeworks library. This multitouch system is open hardware and software, which means that everybody can make one by them. That is why this project was so interesting for my thesis, because it allowed me to build a multitouch system prototype for experimenting by my own and without spending too much money.

Vimeo

Vimeo users

September, 2008

http://www.vimeo.com

The Vimeo webpage is a social site where users can upload and share videos that were created by them. This is the place where I found most of the projects that are related to multitouch, Openframeworks, sound-visualization, Puredata, or interactive art. Moreover, this is the best way I found to track the new projects that people is doing with this technology, thanks to the rss tag syndication. For instance, I am syndicated to these three tags from Vimeo, which allow me to be informed of the new videos that

people have uploaded, and that are related to my thesis project: http://www.vimeo.com/tag:openframeworks http://www.vimeo.com/tag:physicalcomputing http://www.vimeo.com/tag:tangibleinteraction

Project Description / Methodology

The methodology used for doing this thesis will consist basically in dividing the process in two big parts. The first one will be oriented to develop a very basic physical-hardware prototype, and the second one will be focused on building the software that will make things happen.

The first step will be building the physical prototype. For doing that it will be necessary setting up the video projector and the infrared cameras as well as make it interact with the physical space and the user physical controller interface. The resulting data of these interactions must be sent to audio software in order of making sound. Therefore, it will be necessary to write a very basic program for communicating all these parts together and for testing if everything is working fine.

The second step will be developing a more advanced program that could extend the possibilities of interacting with the space and the user. It has to be portable, so it can be adapted rapidly to different spaces, it also has to communicate well with other audio software and it has to have a computer vision implementation. For doing all of this I am going to use the Openframeworks environment, which is a c++ library specially designed for developing creative interactive and experimental projects. Also I am going to use the OSC protocol to communicate Openframeworks with Puredata, because this last one is open audio software that is specially designed for doing interactive and generative sound. For the computer vision part I am going to use the OpenCV addon for Openframeworks that works with the technology of "finding bobs" that is useful for tracking things on an image, like for instance tracking the color from a video.

Because this thesis is based on other open projects that are being developed, it will not be necessary to start the development from the beginning. So basically what I am going to do is continue the work that have been done by other people. That means that I am going to use the code and hardware development done by projects like the Graffiti Research Lab or Theodore Watson as the base starting point.

Limitations

The limitations that can be found while this thesis is developed are related to the controller user interface, the characteristics of the surface where images are projected andhow complex is the sound interaction. With regard to the user controller, the Graffiti Research Lab has been using a Laser because they just need to track the position of the pointer, but we will also need to take other data from the user interaction, like kind of click events or some more analog information. For that reason maybe it is necessary to develop one more controller or simply add some buttons to the Laser. It will also be necessary the possibility of identify different lasers, so the system can interact with several independent subjects at the same time. One way to solve this problem could be use laser of different colors.

The surface where the images are projected can also be considered a limitation. But for this thesis it will be consider like another aesthetic element, that must affect how the interaction happen.

Another limitation is the sound interaction. There will be not possible doing complex sound compositions because the user interaction will not be so accurate. However, this is not the main purpose of this project, which is more an interactive art installation than a musical tool.

Marketing plan

Because this is an experimental project, once the prototype is finished and showed to the audience, it will be sent to several new media festivals like NIME(New Interfaces for Musical Expression festival), Sonar, Ars electronica, etc... Moreover, because I will use some kind of technology that is beginning to be developed (Openframeworks is still on a pre-relase version (0.05)), I know some institutions that could be interested on showing some work built with these tools, like medialab-prado in Spain, Eyebeam in NYC, or MTG-UPF in Spain. Actually they were doing some workshops using these tools, and because no too much people know how to use them, it could be a good change to show these projects on such important places like the ones listed above.

Budget

Video projector	$ 600.00
Infrared camera	$ 154.00
Lasers	$ 50.00
Sound system	$ 550.00
Total	$ 1354.00

Software and hardware requirements

+ Macintosh G5.
+ 500 MB free disk space and 128 MB main memo.
+ Openframeworks (free software).
+ Xcode (apple development software).
+ Puredata (free software).
+ Video projector (minimum resolution: 1024 x 768)
+ IR camera (Fire-i Board Camera B/W).
+ 4 laser pointers of different colors
+ A portable table

Target audience

The target audience are musicians, artists, anybody interested in interactive art, multimedia art, performace art, sound art, music and new technologies. Specially, this thesis is related to people who have a special sensibility for free art and culture.