Electronic Collage
Benjamín Dueñas Figueroa
Experience Prototyping - Tutor Camille Moussette
Masters Programme in Interaction Design
Umeå Institute of Design - Umeå University
Umeå Sweden - November 2008.

Index

Summary

The project of electronic collage was developed as a final project for the 5 week Experience Prototyping course at the Umeå Institute of Design at the Umeå University.
It is a one week project that took place at the end of the course, which objective was to be part of the Halloween Party. The course took that scenario as a real environment for the group projects.

Technology for tracking objects using a camera was the main exploration of this project, it includes the use of free ware software platforms, one used for the camera called "reactivision" combined with a programming environment called "Processing" to trigger the graphics and finally using a series of symbols called "fiducials" that can be attached to physical objects in order to be tracked.

A number of problems were faced during the development of the project, both technical, and with the coding, that are explained in the report.

The result of the final installation was successful in terms of the interaction that created on the people that used it, but still are some points that can be further developed.

Introduction

After 3 weeks of Learning to use processing, arduino and getting into the electronics in the Interaction Design Lab, the 4th week was entirely dedicated to the final project of the course.

The objective was to have something working for the Halloween Party, event that takes place every year at the Design School at the Umeå Institute of Design.

For this week project I worked individually with the intention to learn to use a new technology for object tracking using a camera and some printed targets called fiducials.

Reactable - Interactive Table to create music using reactivision and fiducials.

Fiducials & ReacTIVision

This is a new technology created in Spain, used to identify and track objects. It's main components are, the fiducials, that are predefined shapes that work similar to barcodes and that can be attached to any physical object. They are recognized by a camera by means of using a pre designed software called reactivision. You install it on a computer and it is compatible with programming software such as Processing and Pure Data.

Architecture

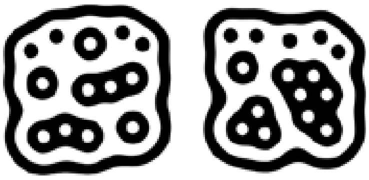
Fiducials are tracked by a computer vision algorithm that communicates with the computer with it's own protocol called TUIO. This protocol is used as a library on Processing and reads and calculates the precise position of the symbols on a 2D plane, including X, Y and rotation angle positions.

Reactivision

The reactivision application acquires images from the camera, and sends the information of the recognized patterns to Processing which interprets all the fiducial movements. Those movements can be programmed to control images, animations video and music which you can also project as a graphical interface.

Amoeba Symbols.

The design of the amoeba symbols is the most refined and robust for tracking. The current library contains 90 different symbols and it comes on a pdf file that you can print and for easily attach the symbols to an object for tracking it.



Amoeba Symbols.

The Magician

I have as a theme for the Halloween party, "The Magician". The idea came up from the chinese cookies that have a secret surprise inside.

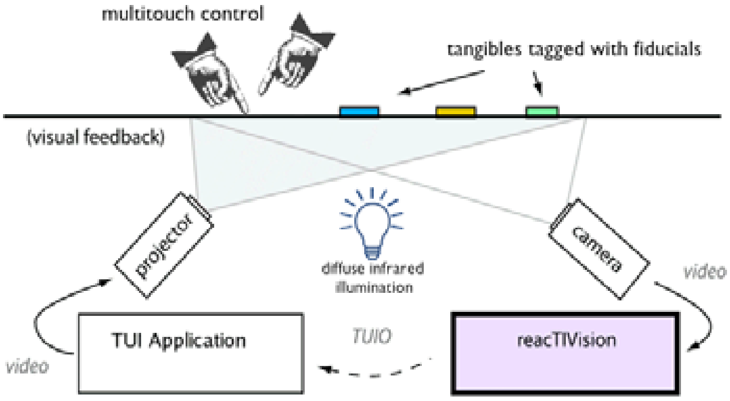
I decided to combine that idea with the fiducials by putting some secret messages or images on them and then give the cards to people for them to discover the secret messages.

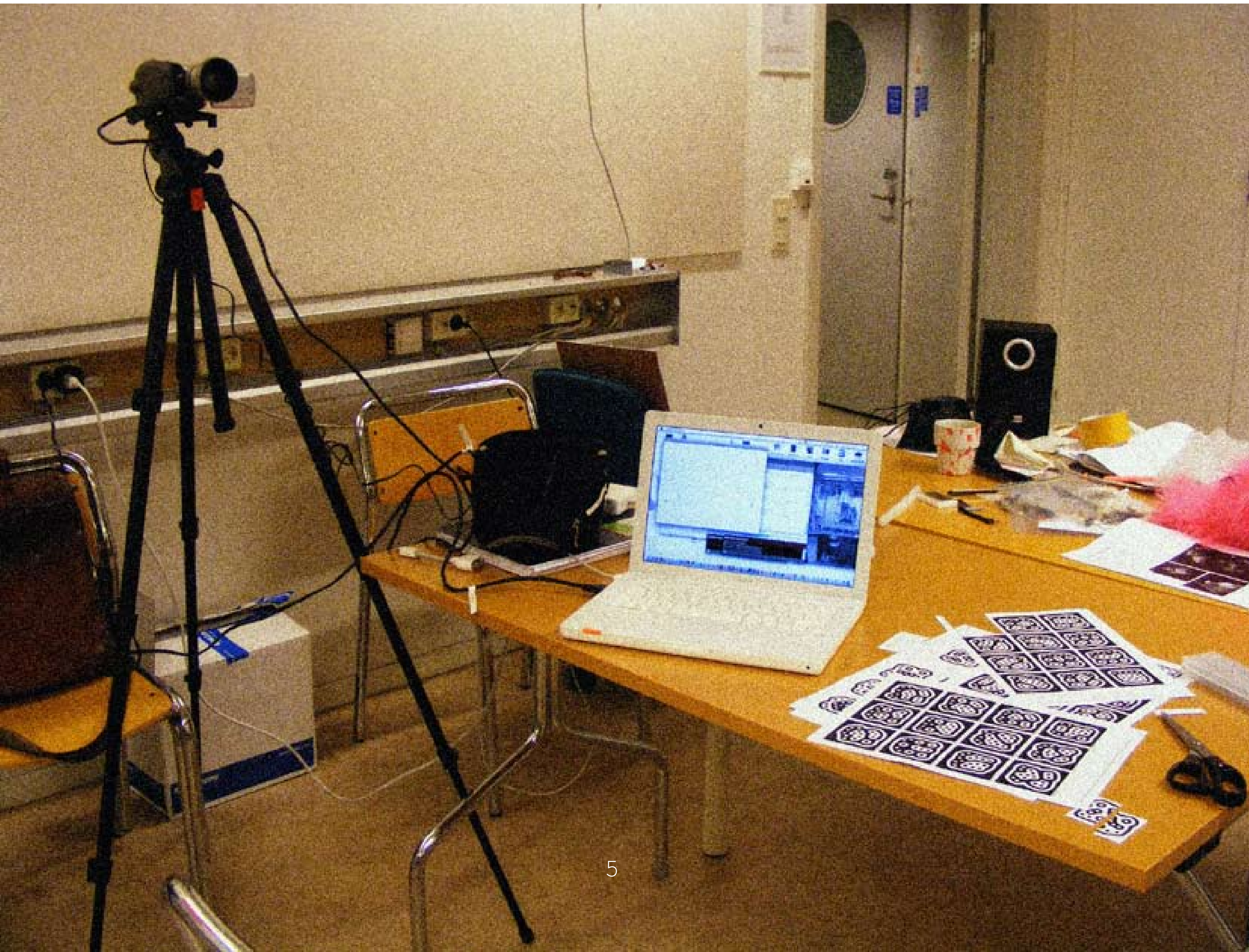
The Splitting Fiducial

At the beginning of the project I thought it was a good idea to split the fiducials into 2 parts, then give them all to the people in the party, so when they find the other half, a hidden message would appear on the screen, but then I realized that it could be hard for people in the party to find the other half and I could do the same just by triggering a reaction with code, for 2 different fiducials, so I ended up using the fiducials as complete cards.

Ingredients

In most of the cases, reactivision is used as in the diagram, reading and projecting at the same surface that is usually the table where the symbols are located. This configuration face some issues with the illumination, because for the projection you need an opaque surface but for the camera to recognize the fiducials, the more transparent the surface, the better.
In the reactivision web page recommend to put an infrared light in between the projector and the camera to avoid interference with the illumination for projecting and reading.



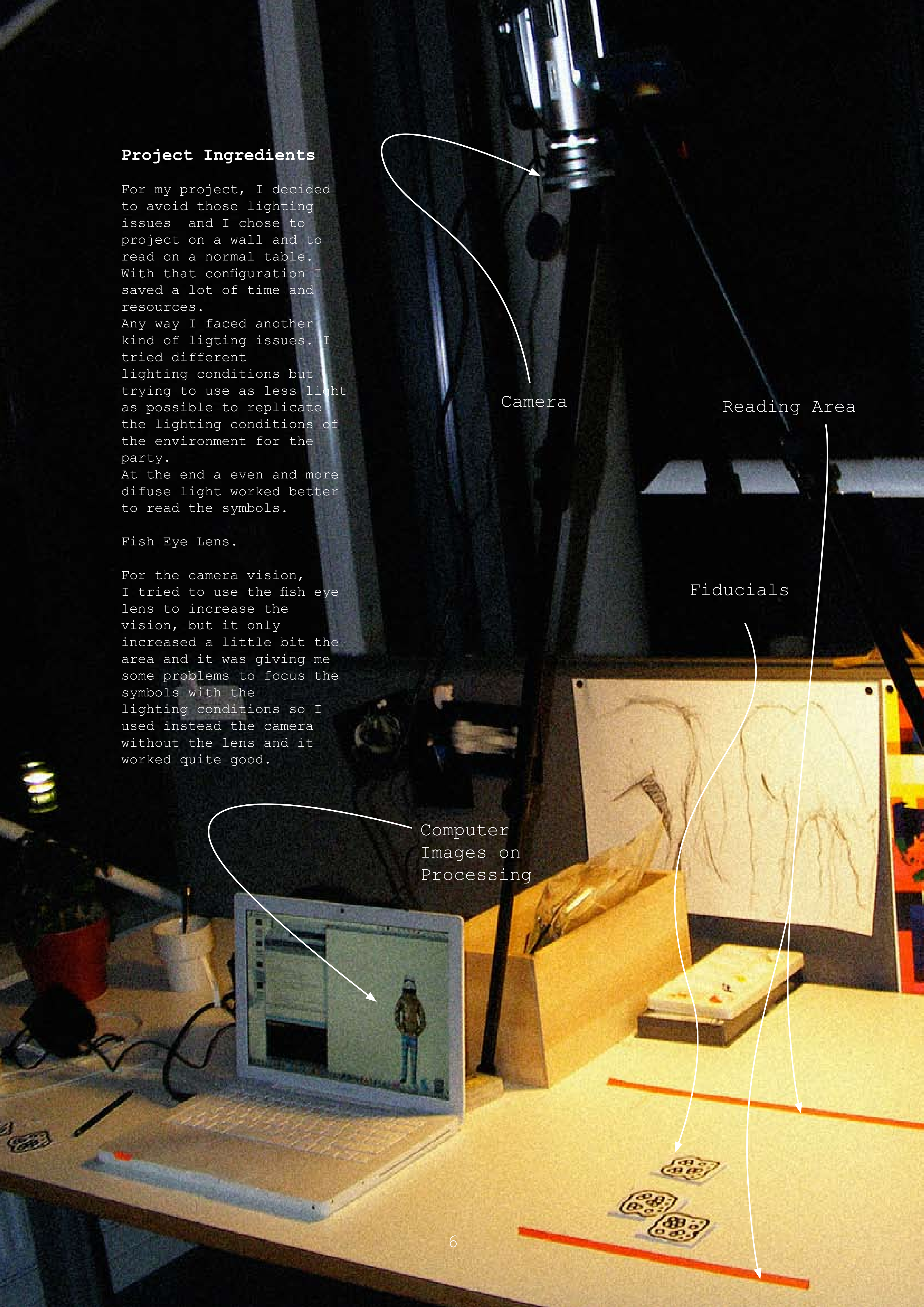


Project Ingredients

For my project, I decided to avoid those lighting issues and I chose to project on a wall and to read on a normal table. With that configuration I saved a lot of time and resources.
Any way I faced another kind of ligting issues. I tried different lighting conditions but trying to use as less light as possible to replicate the lighting conditions of the environment for the party.
At the end a even and more difuse light worked better to read the symbols.

Fish Eye Lens.

For the camera vision, I tried to use the fish eye lens to increase the vision, but it only increased a little bit the area and it was giving me some problems to focus the symbols with the lighting conditions so I used instead the camera without the lens and it worked quite good.



Processing & Reactvision

I started installing the TUIO libraries for processing to be able to recognize the symbols with the camera.

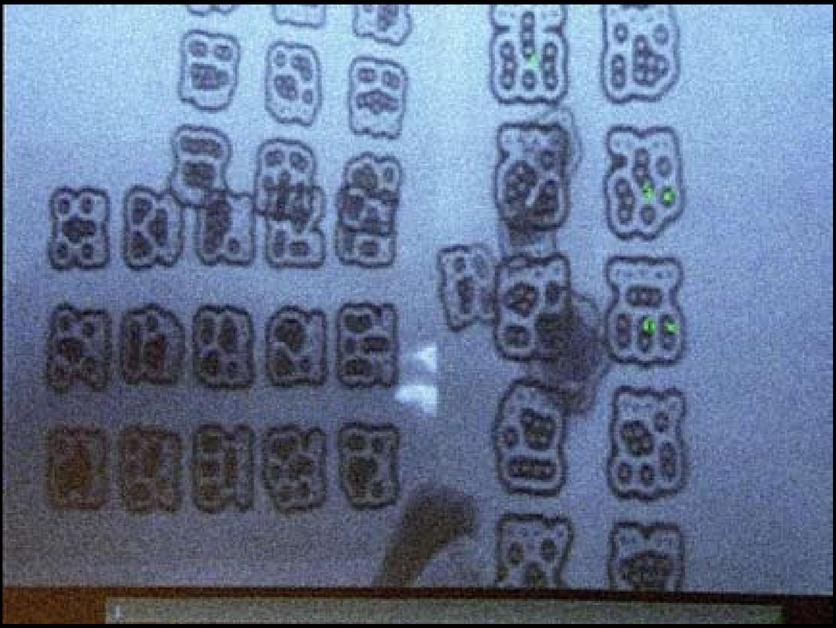
I had to download the software “ReacTIVsion”, the Tuio library for Processing, and print the symbols in order to have something working the first day.

I am using for this project the 0148 Processing version, and the only thing to do is to put the Tuio folder inside the folder called “Libraries” on the Processing application, for the last version you have to create a new folder called “libraries” where all the processing projects are saved.

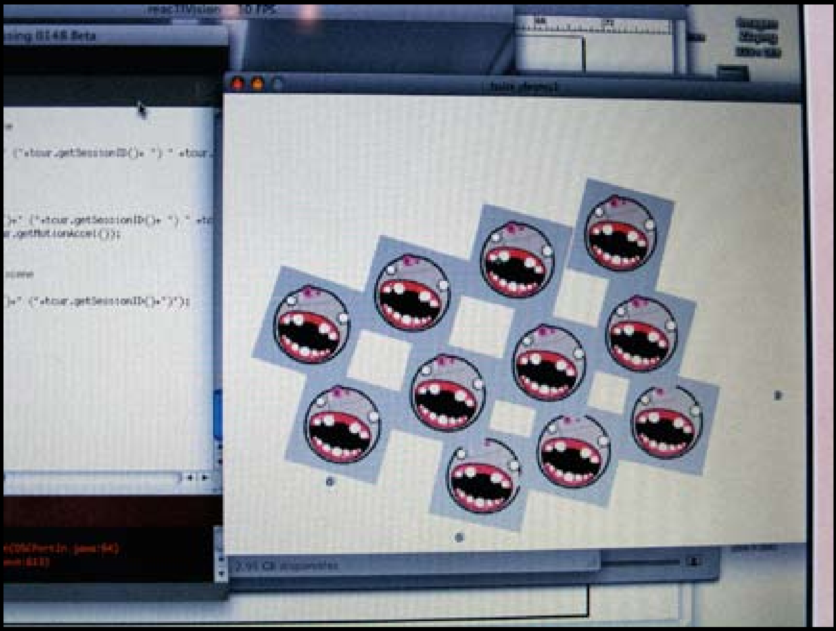
Starting

After finishing those steps, and having all the necessary equipment, I just run the tuio_demo file for processing and I replace the default graphic with a jpeg image, as a result I had was this image repeated for all the symbols on the screen.

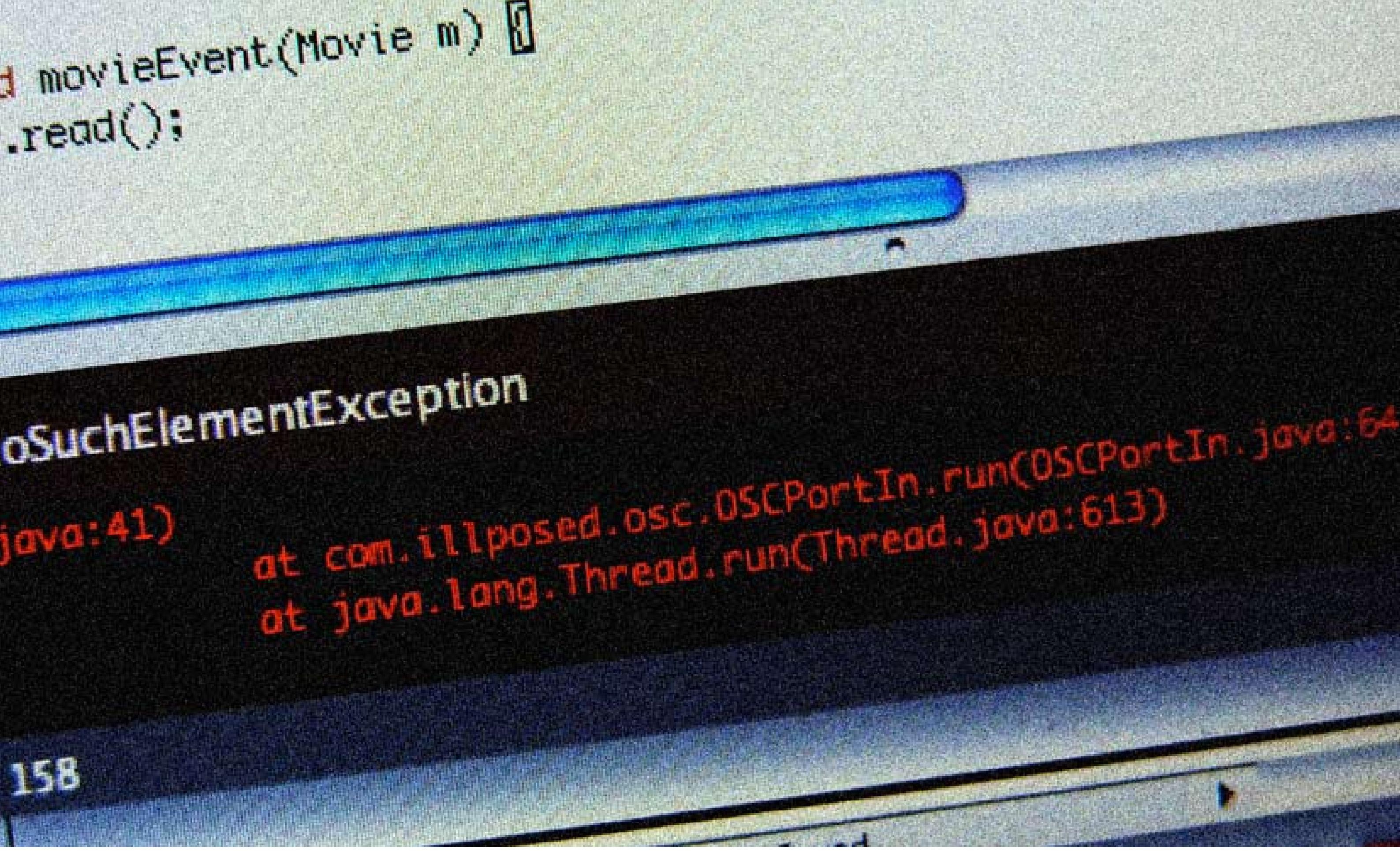
I experimented also with small “mov” videos and it worked very good, but I chose to work only with images since I did not have the time to make a different video for each fiducial.



Video screen from the camera, recognizing the symbols.



Repeated Image triggered by a printout with fiducials.



```
d movieEvent(Movie m) {
.read();

oSuchElementException
java:41)
        at com.illposed.osc.OSCPortIn.run(OSCPortIn.java:64
        at java.lang.Thread.run(Thread.java:613)

158
```

Code for Processing

```
//-Add by Benjamin-2008
// with this code you import all the images from 0 until the number 200
PImage[] images = new PImage[200];
```

```
  void setup()
  {
```

```
//-Add by Benjamin-2008
// I changed the size of the file to fit into my screen (i mac 1280 x 800)
//other way you have to include a function for fullscreen
```

```
  size(1280, 800);
  noStroke();
  fill(0);
```

```
//-Add by Benjamin-2008
// The framerate is set in 12, it was on 24 for a smooth transition of the moving
//images, but it was to much for the computer and I think it was generating errors
//So I keep it in 12 frames p/s.
```

```
  frameRate (12);
```

```
//-Add by Benjamin-2008
// This is a very important part of the code, where you preload all the images
//that you will use on the collage, it is very good because it makes quite robust
// the recognition of the images, other way generates errors all the time.
```

```
  for (int k = 0; k < 200 ; k++){
   images[k] = loadImage(“storm.png”);
  }
```

```
//-Add by Benjamin-2008
//With this code you manage all the files that correspond to each fiducial
//you see that you call an image first - images[2] inside the bracket is
// the fiducial id “2”. So whenever the cammera recognizes the number 2 it is loaded
//the image on the instruction = loadImage(“gato.png”); .
//I’m usign png images to get rid of the background of the images.

//-Add by Benjamin-2008
//This are all the images I’m using.
```

```
   images[0] = loadImage(“storm.png”);
   images[1] = loadImage(“pants.png”);
   images[2] = loadImage(“gato.png”);
   images[3] = loadImage(“C3PO1.png”);
   images[4] = loadImage(“cala.png”);
   images[5] = loadImage(“lata.png”);
   images[6] = loadImage(“gold_jacket.png”);
   images[7] = loadImage(“gold_jacket.png”);
   images[29] = loadImage(“alas.png”);
   images[30] = loadImage(“soldado.png”);
   images[31] = loadImage(“flores.png”);
```

Troubleshooting with Code.

The programme was working with the same image repeated on the projection, and after that I started placing different images on the different ids for different fiducials. For that step, I used ***png*** format for the images to have transparency on them while the images overlap each other in the screen.

During this process I was having some errors with the code, and the programme was crashing every 2 minutes, this problem was due to the large amount of information processed by the computer at the same time but I could solve the problem by pre downloading all the images to processing before loading them with the symbols.

I am writing in this report most of the code I used, but I am skipping some parts to refer only to the code I added into the tuio_demo.

```
//-Add by Benjamin-2008
//With the next code commented is loading a video instead of the images, you can try

//-Add by Benjamin-2008
// myMovie = new Movie(this, “pixels2.mov”);
//  myMovie.loop();

}

// within the draw method we retrieve an array of TuioObject and TuioCursor
// from the TuioClient and then loop over both lists to draw the graphical feedback.

void draw()
{

  background(255);
  textFont(font,18*scale_factor);
  float obj_size = object_size*scale_factor;
  float cur_size = cursor_size*scale_factor;

  TuioObject[] tuioObjectList = tuioClient.getTuioObjects();
  for (int i=0;i<tuioObjectList.length;i++) {
     TuioObject tobj = tuioObjectList[i];
     stroke(0);
     fill(0);
     pushMatrix();
     translate(tobj.getScreenX(width),tobj.getScreenY(height));
     rotate(tobj.getAngle());
//-Add by Benjamin-2008
//In this part you decide what to draw on the fiducials.
//I left this pieces of code if you want to try, having a processing graphic or
//an image, or a video, now it is working only the one for the images, the first
//commented code is for a square drawn in processing and the 3rd one is for the video

//-Add by Benjamin-2008
//1 - Every fiducial loads a blacks square
//rect(-obj_size/2,-obj_size/2,obj_size,obj_size);

//-Add by Benjamin-2008
//2 - A different image is attached to every fiducial - this is the one working.
image(images[tobj.getFiducialID()], obj_size/2, obj_size/2);

//-Add by Benjamin-2008
//3 - A video is loaded with all the fiducials.
//image(myMovie, obj_size, obj_size);

    popMatrix();
     fill(255);
     text(“”+tobj.getFiducialID(), tobj.getScreenX(width), tobj.getScreenY(height));
     println(tobj.getFiducialID());
   }


// these callback methods are called whenever a TUIO event occurs

// called when an object is added to the scene
void addTuioObject(TuioObject tobj) {
  println(“add object “+tobj.getFiducialID()+” (“+tobj.getSessionID()+”) “+tobj.getX()+” “+tobj.
getY()+” “+tobj.getAngle());

//-Add by Benjamin-2008
// I didn’t use this code for the exhibition, but it could be really nice to use it!!.
// With the next condition you can make two fiducials to do something whenever
// they are being detected, in the condition is expressed that every time
// you the cammera reads the fiducial 7 then is changing or loading another image
// on the fiducial 4.

//-Add by Benjamin-2008

//  if(tobj.getFiducialID() == 7){
//   images[4] = loadImage(“0over.jpg”);
//  }

}

// called when an object is removed from the scene

void removeTuioObject(TuioObject tobj) {
  println(“remove object “+tobj.getFiducialID()+” (“+tobj.getSessionID()+”)”);

//-Add by Benjamin-2008
//With the next code (condition) you are telling the programm that every time the fiducial
//is removed it returns to the previous image.

//-Add by Benjamin-2008

//  if(tobj.getFiducialID() == 7){
//   images[4] = loadImage(“cala.png”);
//  }
}
```

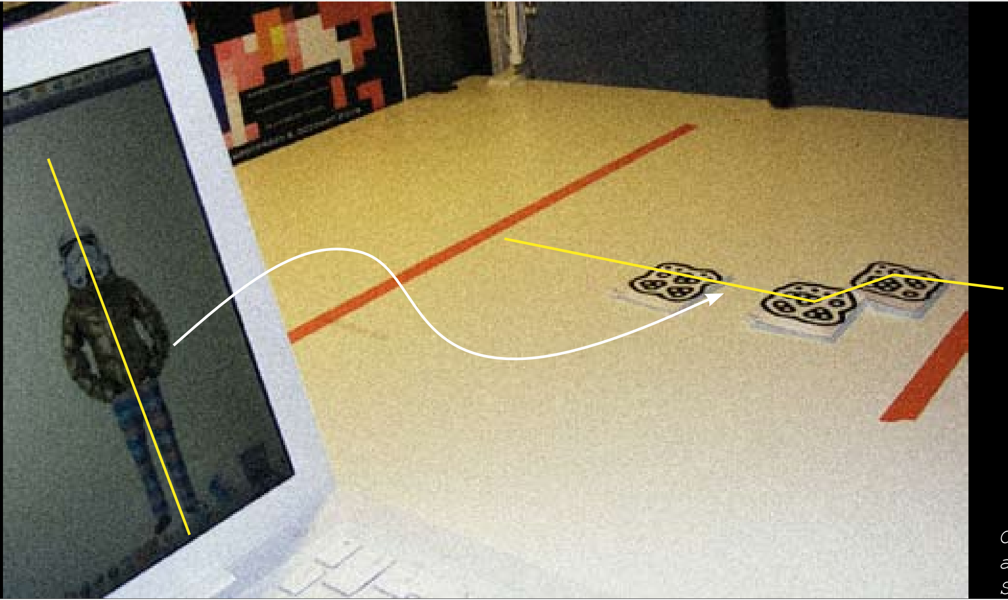
```
// called when an object is moved

void updateTuioObject (TuioObject tobj) {
//println(“update object “+tobj.getFiducialID()+” (“+tobj.getSessionID()+”) “+tobj.
getX()+” “+tobj.getY()+” “+tobj.getAngle() +” “+tobj.getMotionSpeed()+” “+tobj.getRota-
tionSpeed()+” “+tobj.getMotionAccel()+” “+tobj.getRotationAccel());
}
```

```
// called when a cursor is added to the scene

void addTuioCursor(TuioCursor tcur) {
  println(“add cursor “+tcur.getFingerID()+” (“+tcur.getSessionID()+ “) “ +tcur.getX()+”
“+tcur.getY());
}
```

```
// called when a cursor is moved

void updateTuioCursor (TuioCursor tcur) {
  //println(“update cursor “+tcur.getFingerID()+” (“+tcur.getSessionID()+ “) “ +tcur.
getX()+” “+tcur.getY()+” “+tcur.getMotionSpeed()+” “+tcur.getMotionAccel());
}
```

```
// called when a cursor is removed from the scene
void removeTuioCursor(TuioCursor tcur) {
  //println(“remove cursor “+tcur.getFingerID()+” (“+tcur.getSessionID()+”)”);
}
```

```
// called after each message bundle
// representing the end of an image frame
void refresh(long timestamp) {
  //redraw();
}
```



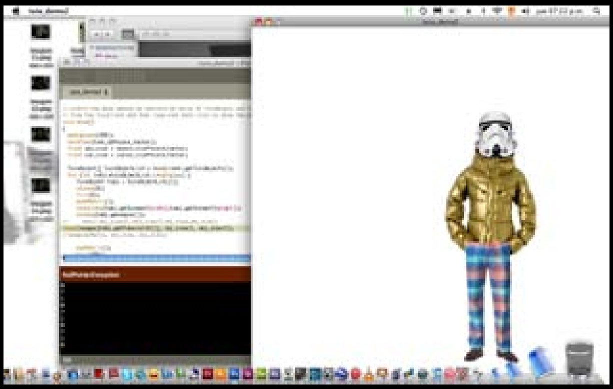
Object Alignment and Correspondence with the Screen.

Images Alignment.

During the testing, I also faced the problem of the alignment of the images on the screen, as you can see on the picture above, the aligned set of figures not necessarily corresponds to the same alignment of fiducials.

I could not solve that problem due to time restrictions, but I found that if you split an image directly in photoshop from the original file and then attaching those parts to different symbols, they can match more easily while using them on processing.

What I did not find is how to match the center of an image to the center of each symbol.



Fiducials glued on a recycled cardboard.

Marks on the floor and on the table for the projection.

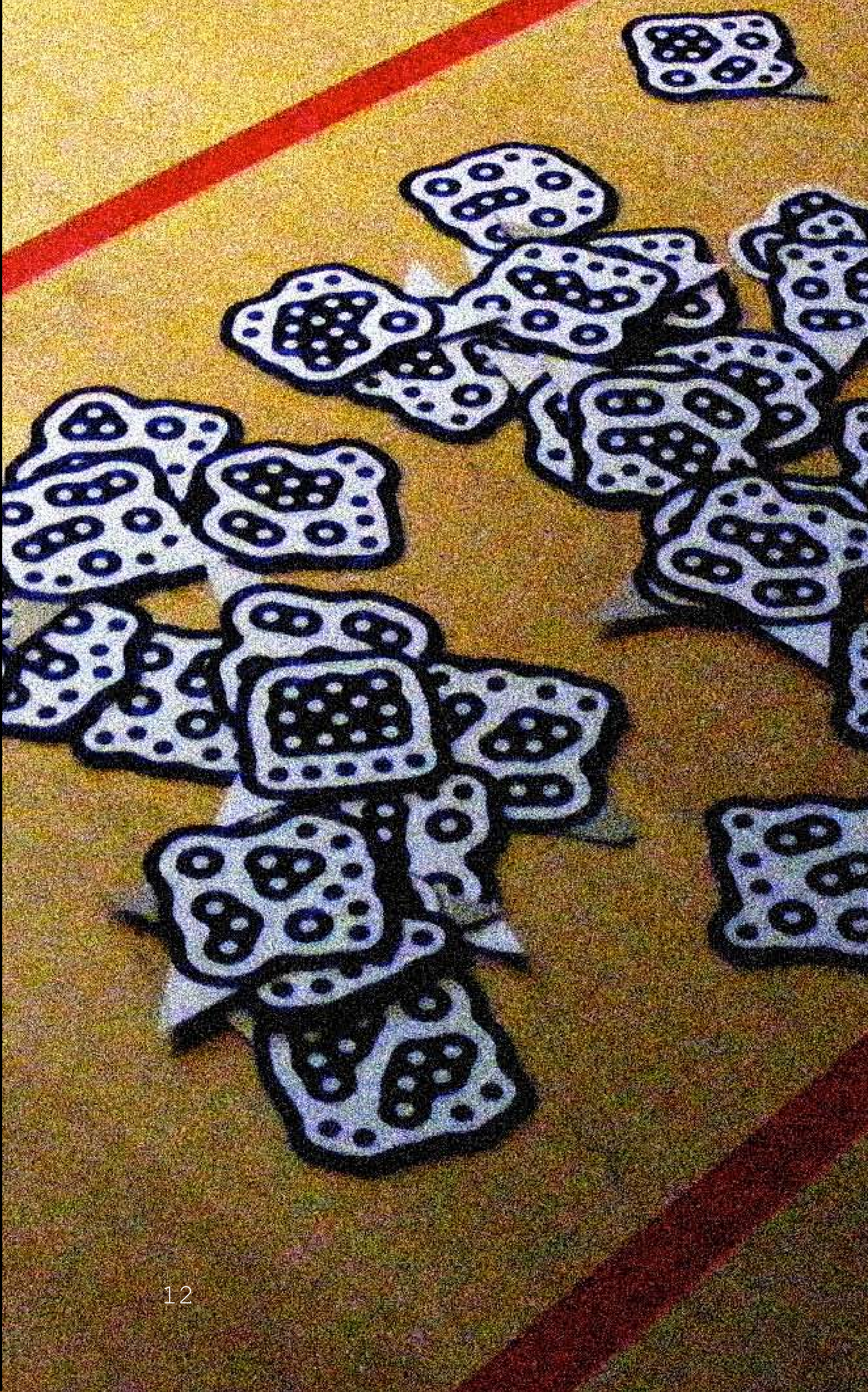
The Reading Area.

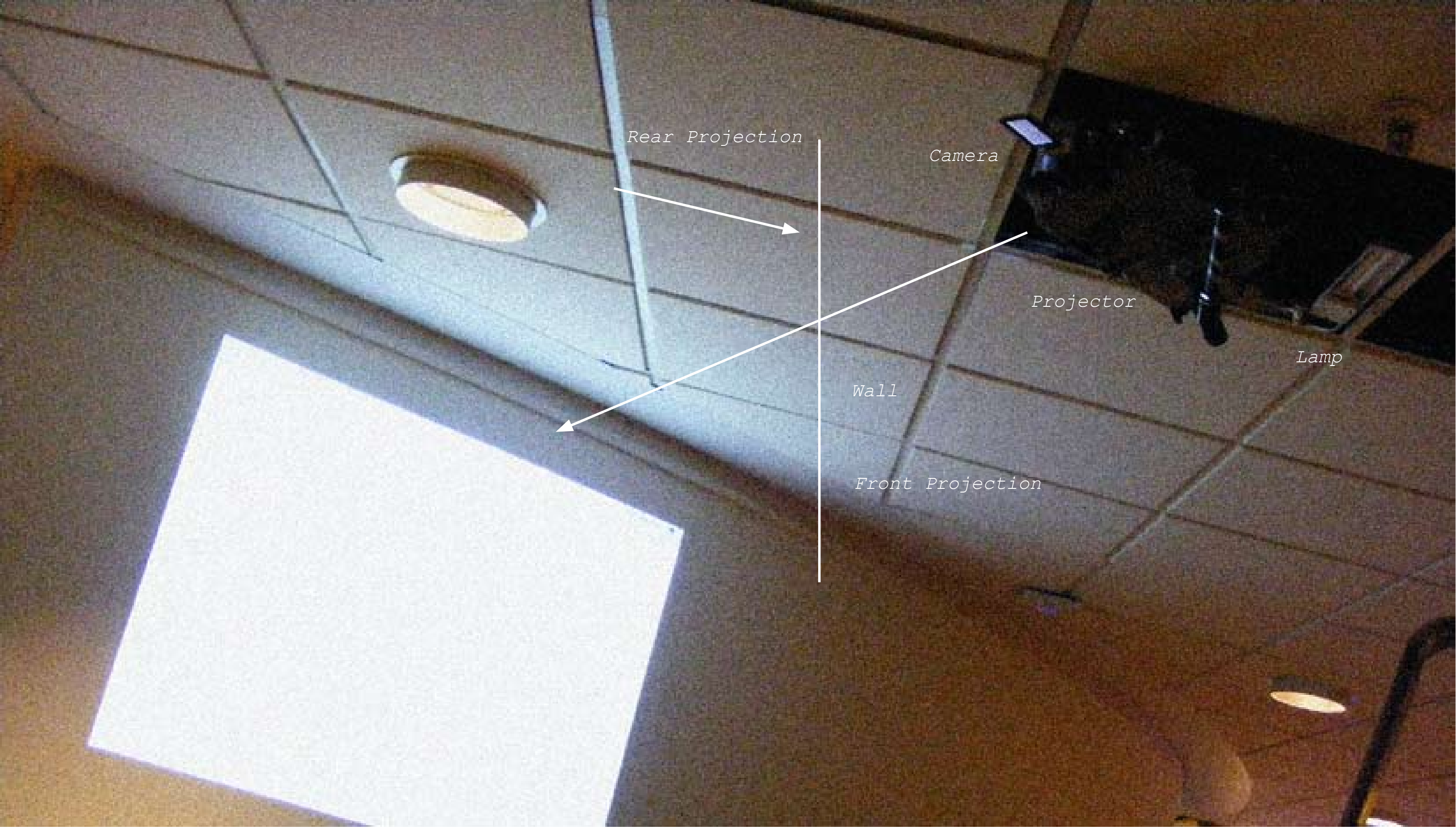
For the Halloween party I set a normal table on a place below the camera which was hanging from the ceiling. I marked the reading zone for the fiducials as I notice it was important for the people to know in which part of the table they could expect something to appear.

I put some red electric tape to mark that zone. This marks also helped me to identify in the camera how much I had to zoom in to the reading area.

The cards.

For the cards I selected the number of symbols I wanted to use in the party and I glued them on recycled cardboard, then I cut them and I left a little triangle on one corner for people to move them and avoid covering them with the hand.





The projection.

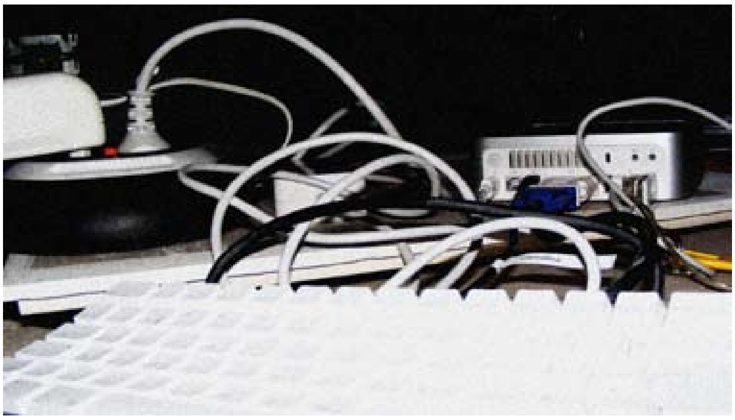
I chose to place the projector on the ceiling for security reasons. I used two mechanical “manfroto” arms to hang up the projector and the camera from one tube that was part of the building installation on the ceiling.

That configuration was quite convenient and fast to prepare, and the arms gave me a lot of possibilities for many positions.

Fiducial Movements.

The fiducial movements on the table did not match exactly with the movements of the images on the screen.
This problem is related with the position and orientation of the camera and the projector.

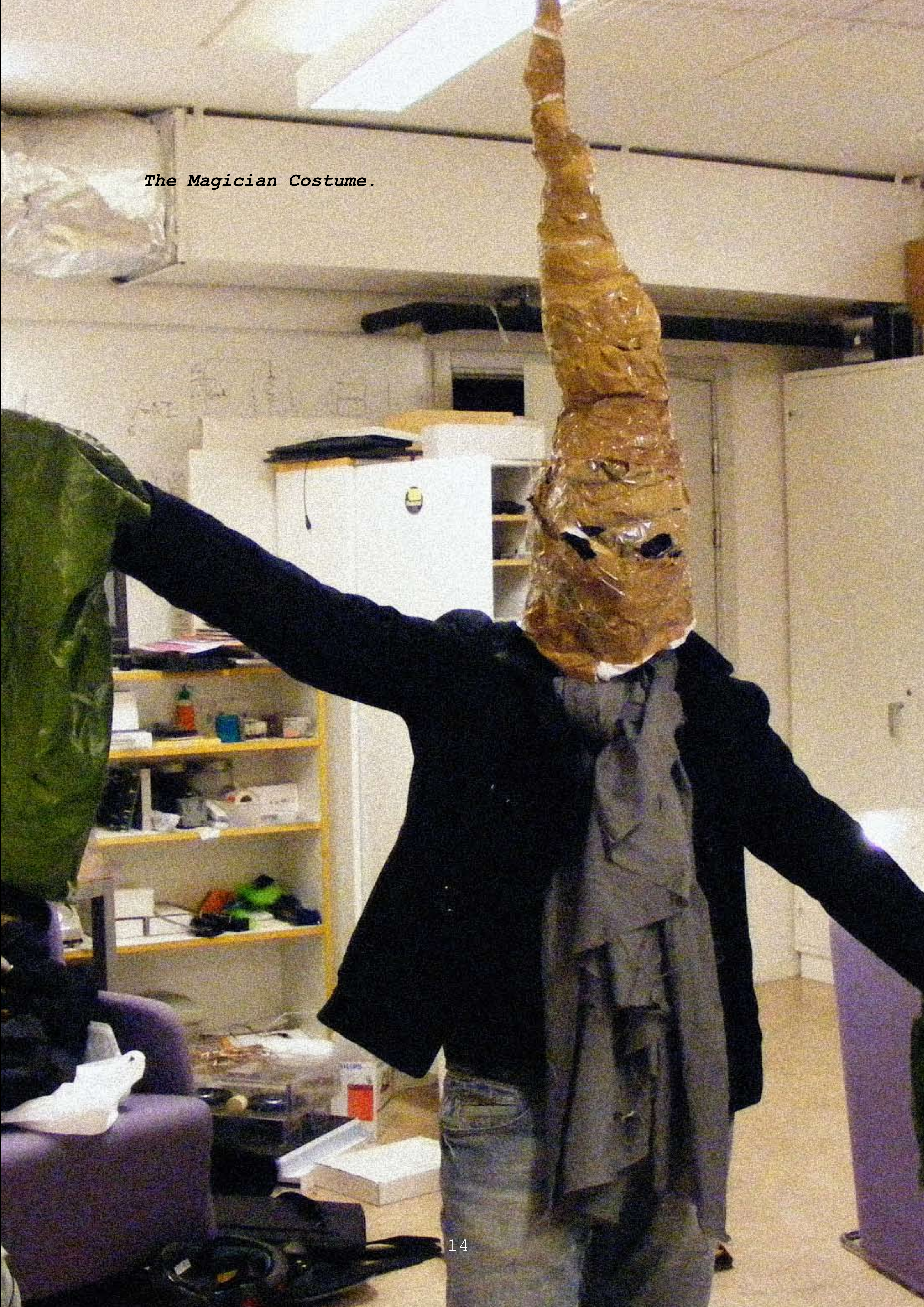
I could get either the right movements going to the left and right or going up and down, but the only way to obtain both movements correctly, was projecting in rear mode, which made the text on some images appear inverted, so I decided to project in front mode.



Computer, keyboard, mouse and all the necessary equipment was on the ceiling.

The set up on the ceiling, lamp, projector and camera.

The Magician Costume.



The Party

Collages

Conclusions

Fiducial technology for tracking objects worked very good, still under restricted light conditions.
The projection on the white wall was lighting the surrounded space and it was enough for the camera to recognize the symbols on the table.

The installation was running during the whole party for about 5 hours without crashing or any other problem related with the software performance.

Further Considerations

For many people was difficult to move the images and make them match exactly with the other ones, because the images on the screen were constantly disappearing as they were covered by people's hands or not recognized correctly by the camera due to the symbol position and the overlapping with other symbols or cards, nevertheless having the printouts on cardboard made them handy, cheap and destruction proof for the party environment.

Interaction

The interaction with people was incredible successfully since they were amazed with the fact that they could control images with the cards on the table, some others who spent a little more time could form some figures and other even created games from it, like dedicating to other people the image inside a fiducial.

Technology

I found this technology very handy and easy to use for anyone that want to explore a tracking technology, all the software and tools are on a free ware platform and are available on the internet.

The technical equipment as the camera and the projector are also pretty common or easy to get also with a little more investment and the rest of the materials can be so easy to get from the shelf.

This is just the result of a week project and I can see many possibilities to enhance any other project like this by just investing a little more time.

References

Web

http://processing.org/

http://processing.org/discourse/yabb_beta/YaBB.cgi?board=Integrate;action=display;num=1203528337

http://mtg.upf.edu/reactable/?software

http://mtg.upf.edu/reactable/

http://www.designinginteractions.com/book

http://dev.processing.org/bugs/show_bug.cgi?id=961

Books

Moggridge, Bill - **Designing Interactions, 2006.**

Igoe, Tom - **Making Things Talk, 2007.**

Articles

Bencina, R. & Kaltenbrunner, M. "reacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction", 2007.