DOTS

ROBERTO CHRISTEN
UMEÅ INSTITUTE OF DESIGN
EXPERIENCE PROTOTYPING, FALL 2008

Brief

The project completed during the Experience Prototyping course had as its main purpose to explore hardware prototyping. As the outcome, students were asked to create projects that challenged them to sketch in hardware at different fidelities. As special focus, *wireless* was a suggested theme, open to be interpreted literally or not.

To coincide with a Halloween party that took place at the Umeå Institute of the Design, the resulting projects were to be incorporated into the event in some fashion.

While my project, *Dots*, was presented at the Halloween party, the focus was not on wireless and the spirit behind it was more open to a life beyond the event and the Halloween theme.

As a participating student in the course, I very early made it my priority to use the course as an opportunity to become more literate in programming, using the *Processing* language as the vehicle.

Abstract

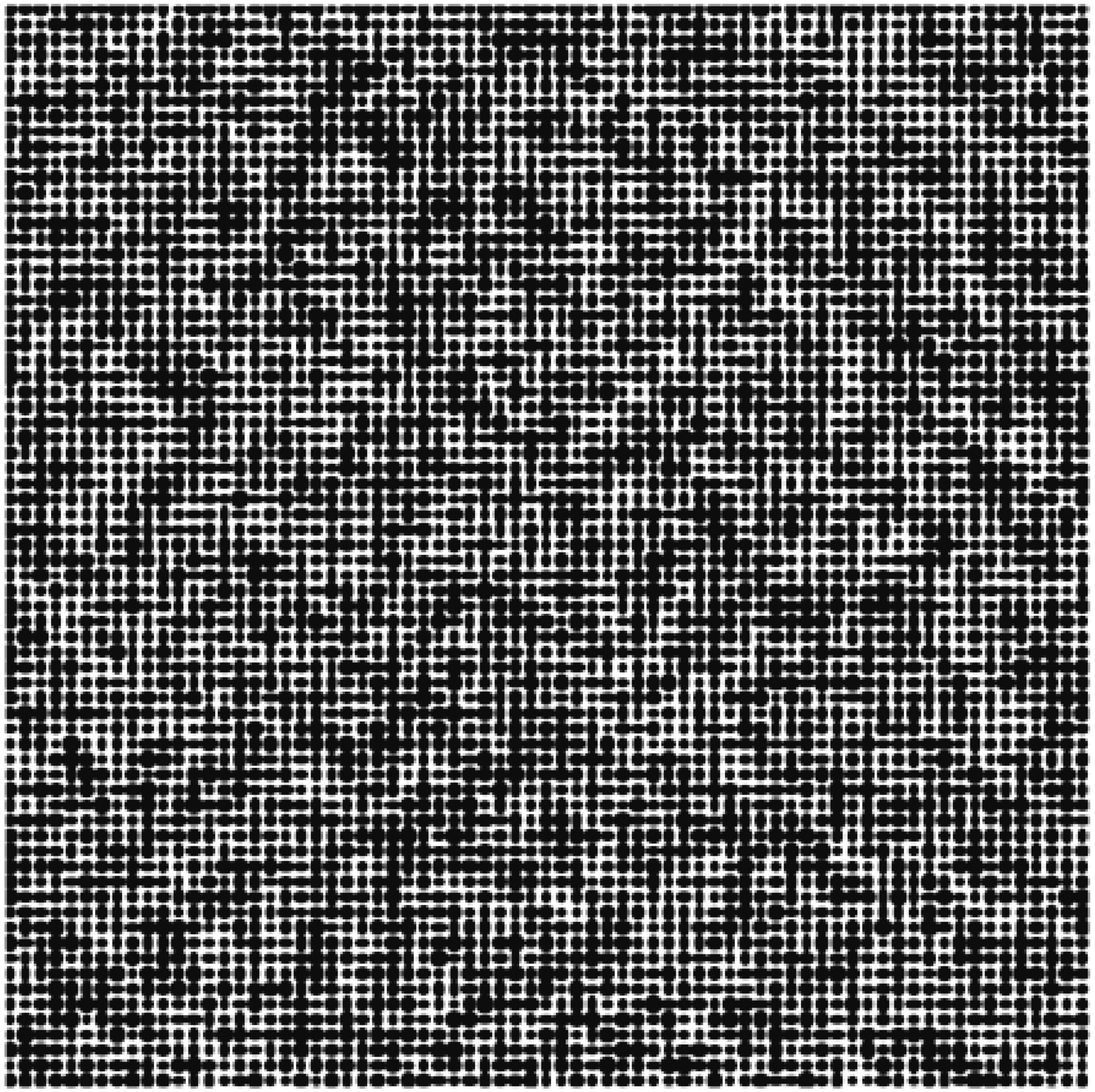
Dots is an effects controller that processes one or two video sources in real time. The inputs of the controller are on a physical device connected to a computer that is processing video. The keyboard and mouse are also used.

The purpose of this project was to serve as an introduction to programming with *Processing*,using hardware inputs and outputs.

Dots degrades the captured video by pixelating and limiting its color depth. A series of hardware knobs control various parameters that either further distort the image or add new artifacts on top of it. A single fader allows toggling between two video channels.

Dots uses a PhidgetLCD Interface Kit that includes 8 analog inputs (of which only 6 are currently used). The keyboard and mouse/trackpad serve as an additional input to control a base layer of visuals: the grid of dots that give this project its name.

It also accepts an audio input that adds yet another grid of dots that responds to that signal. That grid of dots can be controlled using a Monome 40h (further described in this report).



Goals of the Project

- To learn basic programming using *Processing*
- To build something tangible
- To develop something with enough visual quality that it draws interest and generates curiosity
- To find a way to use the outcome of the project in a performance or event

Expected Outcomes

- To create a functioning interactive artifact that can amuse and entertain the operator
- To entertain and distract a crowd in an environment surrounded by other visual and aural stimuli

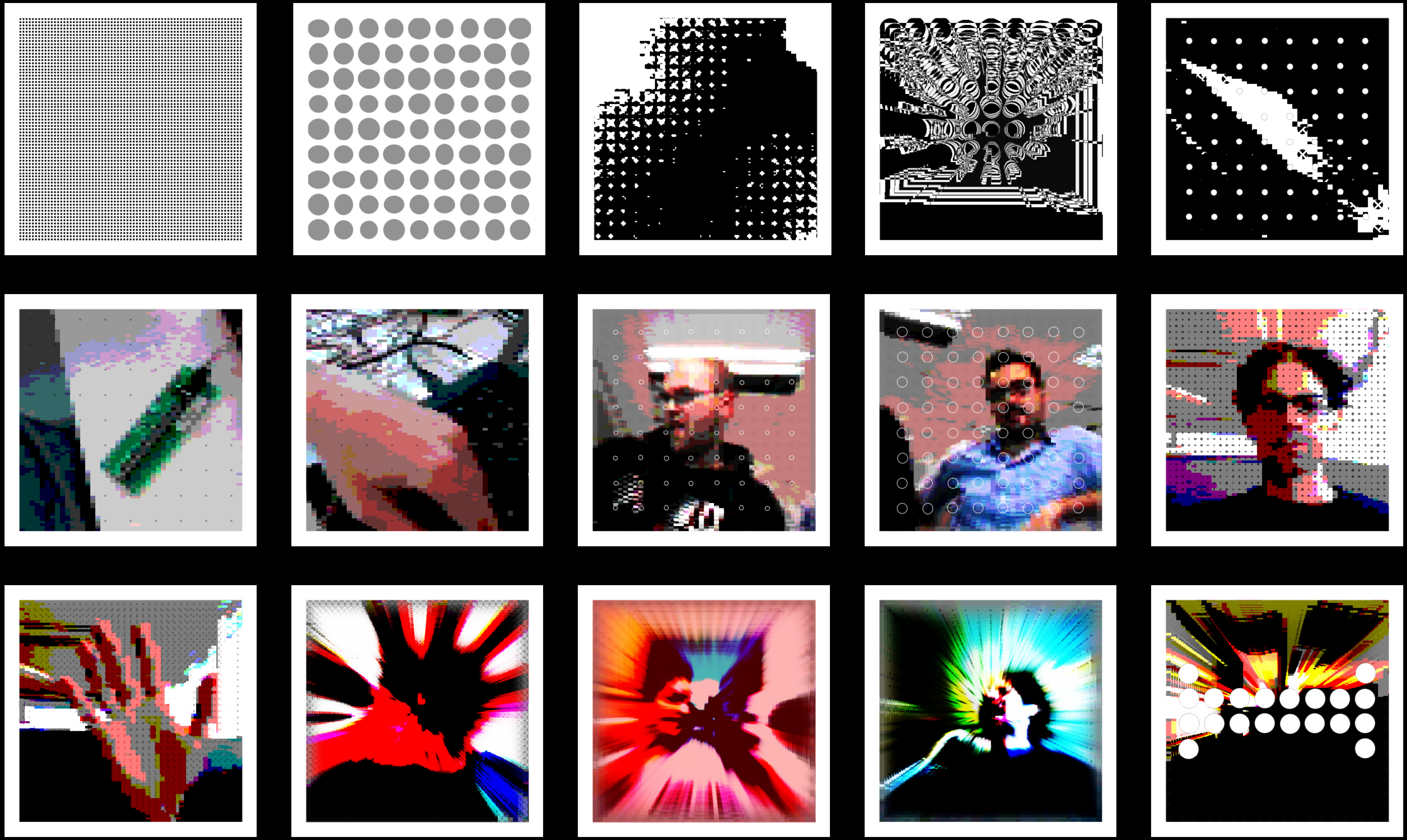
Actual Outcomes

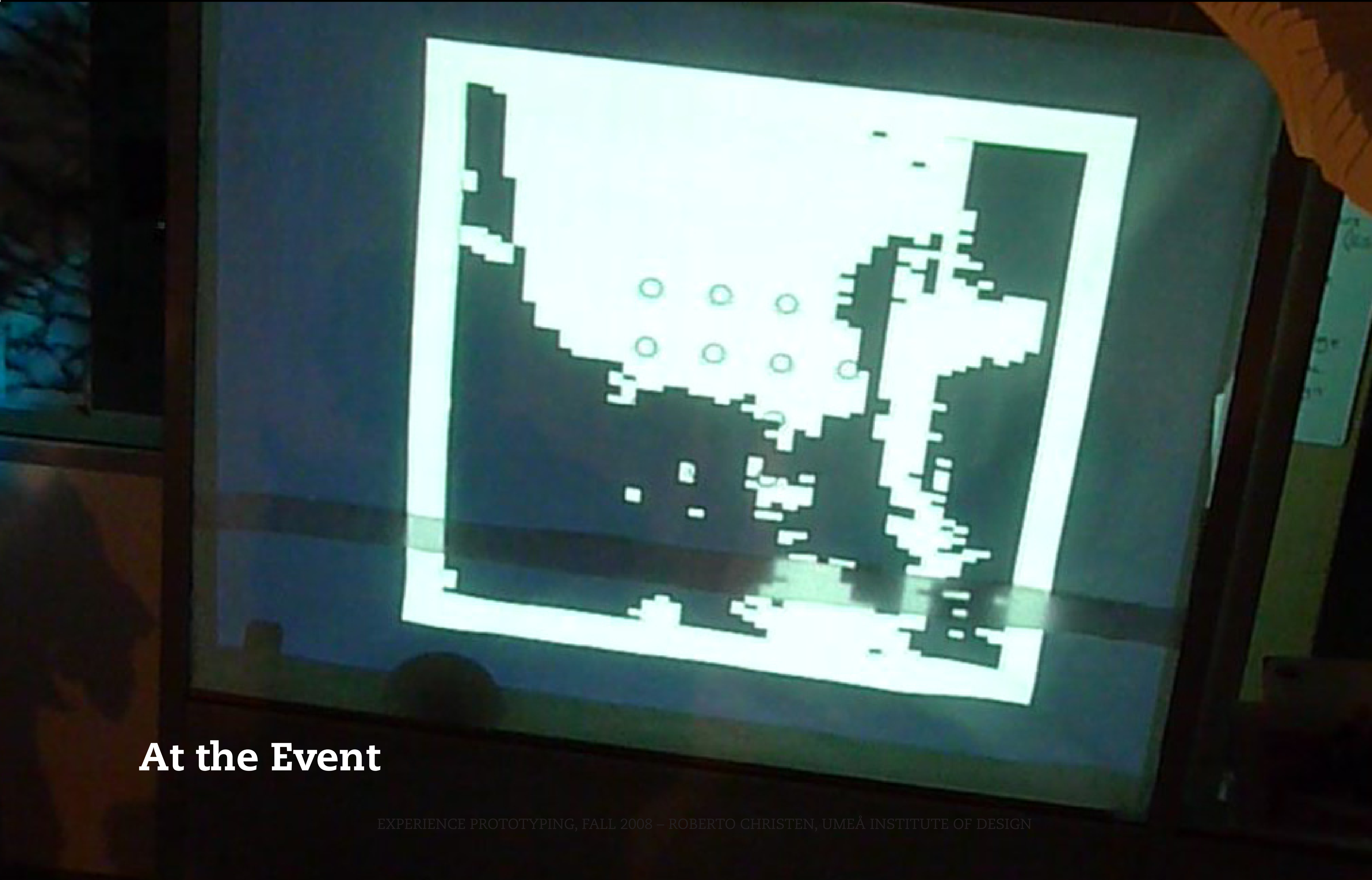
- Created an image projection tool that was abstract and ambiguous in intent
- Created a still image generator that can produce a wide variety of distorted effects using a video camera live capture as its source

Possible Next Steps

- Compile a collection of stills that shows the breadth of possible styles that can be generated with *Dots*
- Build a case for the set of inputs so it can be reused as a controller for other purposes
- Improve performance and responsiveness
- Attempt to use the device during another event with better testing of placement and lighting

Sample Generated Images





At the Event

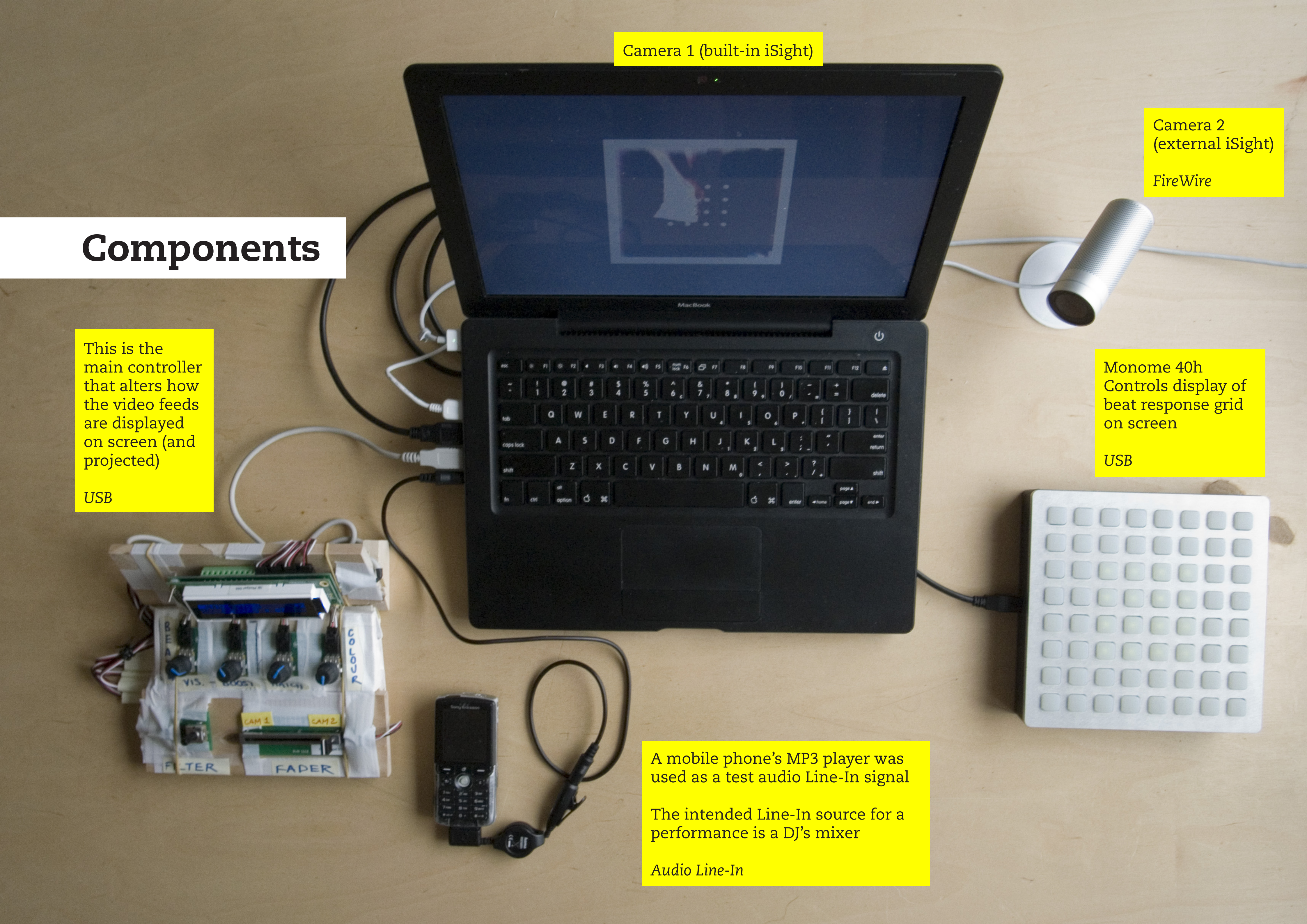
Inputs to *Dots*

- Cameras (1 or 2)
- Controller (composed of 5 knobs and 1 slider attached to a PhidgetLCD board)
- Monome 40h
- Audio Line-In
- Computer keyboard and mouse/trackpad

Outputs from *Dots*

- Screen and projector
- Monome LEDs
- LCD on Controller (PhidgetLCD Kit)

Components



Camera 1 (built-in iSight)

Camera 2
(external iSight)

FireWire

This is the main controller that alters how the video feeds are displayed on screen (and projected)

USB

Monome 40h
Controls display of beat response grid on screen

USB

A mobile phone's MP3 player was used as a test audio Line-In signal

The intended Line-In source for a performance is a DJ's mixer

Audio Line-In

Controller

A. Status LCD
Two lines show
current state of the input
being manipulated

B. Beat Visualization
OFF and 5 levels

C. Visualization Booster
Range from -3 to +3,
controlling the diameter
of audio generated dots

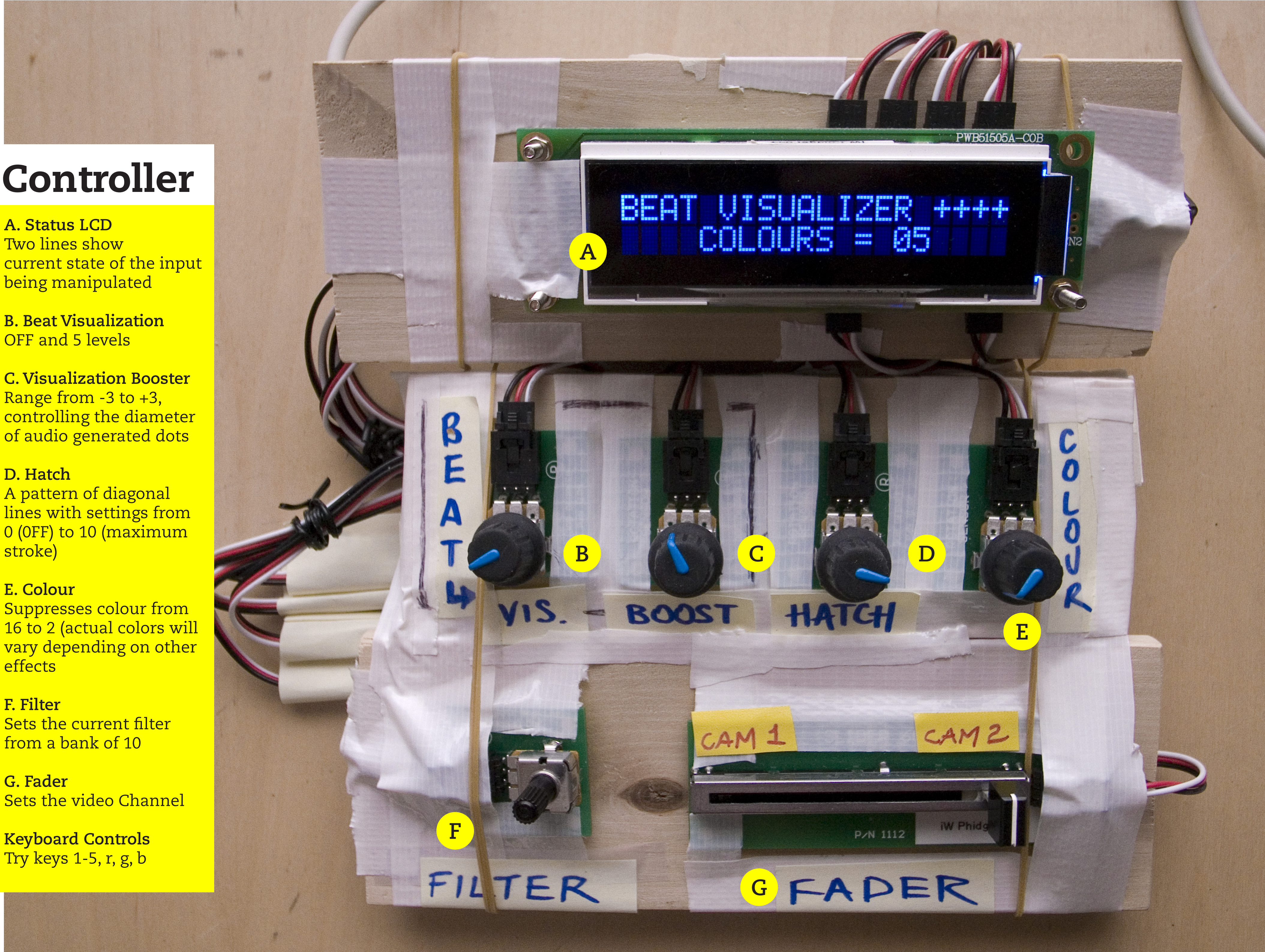
D. Hatch
A pattern of diagonal
lines with settings from
0 (0FF) to 10 (maximum
stroke)

E. Colour
Suppresses colour from
16 to 2 (actual colors will
vary depending on other
effects

F. Filter
Sets the current filter
from a bank of 10

G. Fader
Sets the video Channel

Keyboard Controls
Try keys 1-5, r, g, b



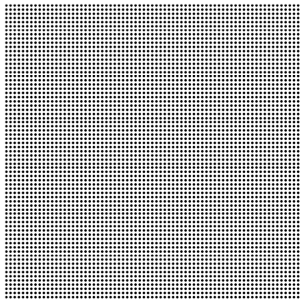
Process and Challenges

A. IDEA GENERATION

1. The Basics
Dots was born out of a desire to learn programming and experiment.

It started as a simple loop that drew a grid of dots on the screen.

It then became a user controllable grid of dots, where mouse position affected the darkness of the dots and key presses affected their size.



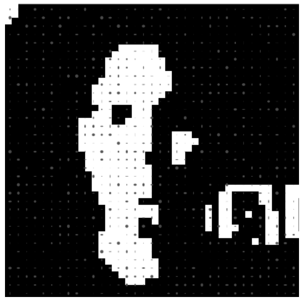
I discovered randomness, and applied it to the diameter of the dots, giving them some movement as they are re-drawn in every loop. I made that degree of randomness user-controllable through mouse movement.

B. REFINE AND BUILD

2. Finding a Source of Visuals
A grid of dots is hardly enough to create interest.

I considered taking a turn towards typography, where jittery dots could become pixels that build up letterforms.
I discarded this idea when I found that video input could be more interesting and diverse.

Because my laptop computer has a built-in video web-cam (as many others do), it was easy to start doing tests with video.

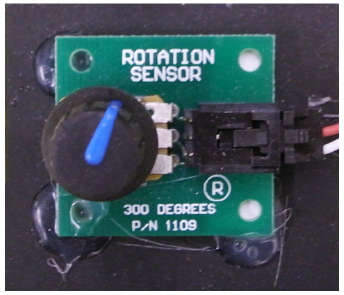


I kept the dots, and quickly became interested in highly-pixelated images, hoping that the dots and video together would create a lo-fi composite that could be interesting.

Aspect ratio, size and frame rate all became issues to decide on. My motivation was still exploration, so I chose to keep things crude so I could continue to add more elements with less fear of overtaxing the computer's processor.

3. Manipulating the Source
I didn't want to do all the interaction and manipulation using the mouse and keyboard, so it was time to find a way to control the dots (and other effects that I was experimenting with) using a physical device.

I chose a PhidgetLCD kit and started to attach knobs.



I also discovered the `filter` and `blend` functions in *Processing* that let me try various effects on the captured video.

Among the parameters I wanted to control was the degree of pixelation of the video.

To achieve the pixelated effect, I simply captured video at a very small size (60 x 90 pixels), and then scaled it up:

```
scale (10, 5)
```

10 is the X(horizontal) value, and 5 is the Y(vertical) value.

The are uneven intentionally, since the "pixels" in the composite image are wide, an intentional decision that yields a more interesting visual.

I experienced failure as I tried to make the resolution of the video controllable using a knob. It did not respond by turning a size parameter into a variable. I tried techniques around this (scaling the video after collected at a higher resolution), but I never achieved the intended effect and moved on without it.

4. Expanding the Inputs
I continued to tinker with

more knobs, and it occurred to me that I should try additional controls. A fader was a sensible choice, and immediately the idea of switching between two channels came to mind.

For this to happen I needed a second source of live video.

I needed another camera in addition to the built-in webcam. I decided to try my luck with an external out-of-production iSight.

I successfully got 2 cameras working, but I had to find a away to toggle between them.

Layering two videos over each other with transparency didn't work, so I settled for a "window-blind" effect when transitioning from one video source to another.

I used *Processing*'s `copy` command to copy strips of one video on top of the other:

```
for
(int i = 0; i < 90; i = i + 2)
{
videoOff.copy(videoOn, 0, i, 60,
             1, 0, i, 60, 1);
}
```

5. Accepting Audio
Having a grasp on the visuals and being content with their appearance begged for a new challenge.

In a performance where music is a main draw, synchronizing visuals and audio is an natural pairing. It was time to attempt this pairing with my the existing visuals or new ones.

I used the minim java sound library to allow processing to read an incoming audio signal.

```
import ddf.minim.*;
import ddf.minim.analysis.*;
```

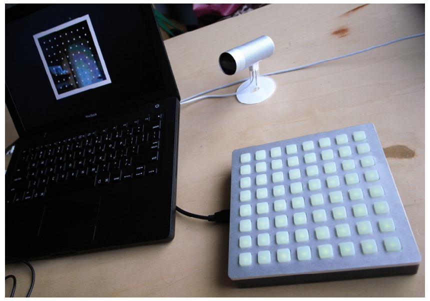
I discovered the `beatDetect` function and used it to trigger a new grid of dots on top of the existing visuals.

6. Attaching a Monome
By now *Dots* had acceptable visuals and simple audio synchronization that made it feel more complete.

Given the base grid I was working with, the new grid of dots generated by the audio ended up an 8x8 grid.

I knew that this is exactly the size of the LED grid on a Monome device (a customizable controller for any use a person can imagine and has the ability to build).

My goal became to map the audio dots on *Dots* to the LED grid on the Monome. Pressing an LED button on the Monome would thus toggle the corresponding dot on the screen.



Using the monomic library;

```
import jklabs.monomic.*;
```

and with some coaching, I used a 2D-array to map all buttons to the visuals generated by audio.

C. CONCEPT TESTED

7. The Party
Due to the continued development of the concept up until the day of the party, there wasn't a proper testing session before the event itself, so improvisation was key.

The challenges at the party were:

- Lighting
- Colour reproduction by Projector
- Placement of equipment

Lighting was poor, and the camera had to be set to nightvision to be able to capture movement of the crowd during the party.

Colours produced by the

projector in use were washed out and very dull.

Since i built several effects that could be used on the composite image, I found the one that worked the best in the environment: a black and white high-contrast filter. The drawback of this filter was that the image became totally abstract and more reminiscent of a 2D barcode than any kind of live footage.

Placement of the setup was convenient in that equipment was protected and away from the crowd, but tricky in that access for the operator(me) was also limited. At the party interaction was sporadic and quite static. I expect this system to work better when placed within easy reach of the operator.

Human operation of the controller can create more interesting results that can be in tune with what a DJ is playing and how the crowd is behaving.

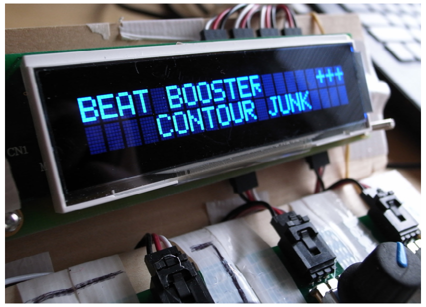
The “start it and leave it” approach makes is repetitious.

D. PACKAGING IT

8. Follow Through
After the party, there was still a lot to be solved.

The principal task I laid out for myself was to use the LCD screen of the Phidget controller to relay the values of the knobs being manipulated in a meaningful way. Up until and through the party, that LCD screen was unused.

Once I knew how to pass strings to the LCD, I had to decide how best to use the 2 rows of text available, since I had more inputs than rows.



The most sensible answer for me was to assign the bottom row to more static parameters that would be most useful to have always visible, and the top row to more dynamic controls that manipulate smaller details. In both cases, when a control was changed, that control’s value would take over whichever row it was assigned to and would stay visible until another control was used.

I created further distinction between the two rows by center-aligning the bottom row, making it easier to make sense of the display.

Once the LCD hurdle was passed, I wanted to package all knobs, the fader, and the Phidget board and LCD into a single unit. Up until then, it was a mess of uncontained wires and knobs.

The result of the packaging is show earlier in this document (Controller picture with labels).

While clearly a sketch, this new packaged version of the controller makes it much easier to use and carry.

Yet another loose end was the generation of still images. I figured how to export images, name them sequentially, and finally build a system to prevent overwriting, so each new session would continue the image count where the last session left off.

E. NEXT STEPS

9. Possible Plans
A lot remains to be done and improved – I hope to be able to explore a few of these ideas:

- Create a collection of pixelated portraits
- Improve behaviour of the Phidget LCD screen
- Expand the number of inputs to a total of 8 (it is now 6)
- Build a proper casing for the Phidget board + LCD and the bundle of inputs
- Add physical push-buttons mapped to keyboard shortcuts
- Extend *Dots* by allowing the use of video clips or stills in addition to captured video
- Expand the library of visual effects
- Improve response to audio
- Build animated sprites to for Monome LED grid

Ending Thoughts

The Experience Prototyping course was a wonderful opportunity to get acquainted both with basic programming concepts as well as exploring the use of physical controls.

I plan to continue my learning by further developing *Dots* and hope to create a generic controller based on the PhidgetLCD kit that can be used for other purposes.

Credits

Roberto Christen, November 2008

I wrote *Dots* as part of the Experience Prototyping course at the Interaction Design MA Programme at the Umeå Institute of Design in northern Sweden.

Direction, assistance and supervision by Camille Moussette.