# Jitters

COMP 471 / CART 498c
Team Project Report – rev 3.2.0

Website Docs
http://www.cul-de-sac.ca/~Jitters/
http://hybrid.concordia.ca/~g_ther/jitters/

Blog
http://jitters471.blogspot.com/

Victor Yap
Sum-Tei "Olivia" Tang
Geneviève Thérien

# Table of Contents

## Introduction

Jitters is a responsive installation, designed from ideas learned through the joint COMP 471 / CART 498c course given by Dr. Sha Xin Wei, at Concordia University, fall 2006.

The idea for Jitters started during a class field trip to the EV building in late September. At the suggestion of Professor Xin Wei, we scoured the building in search of a suitable installation location when we found a large white wall near the FOFA gallery that practically shouted: “paint me”. When we walked up the stairs to get a closer look, we were astonished to find how much the stairs vibrated to our footsteps. From those two observations, we started brainstorming about a responsive installation that would show interaction with the stairs as well as attract viewers. Thus, Jitters was born.

### *Concept*

The overall concept of our project is related to the EV building itself. Since its inauguration a year and a half ago, the EV building has been visited by countless students and teachers who see it as a modern and useful study environment.

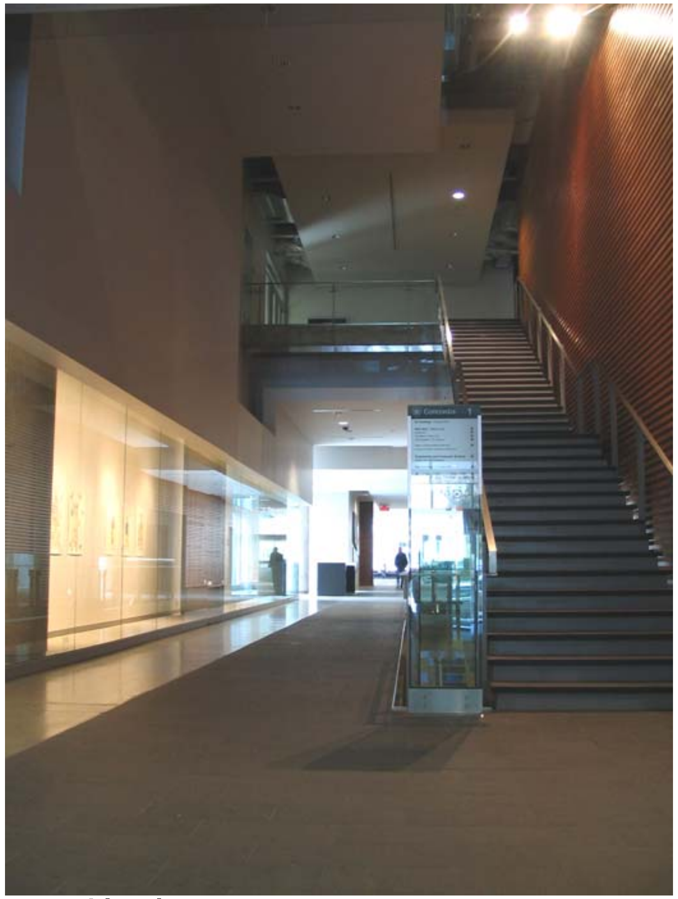
Now, the question is: how does the building perceive us? Jitters is an interpretation of this question.

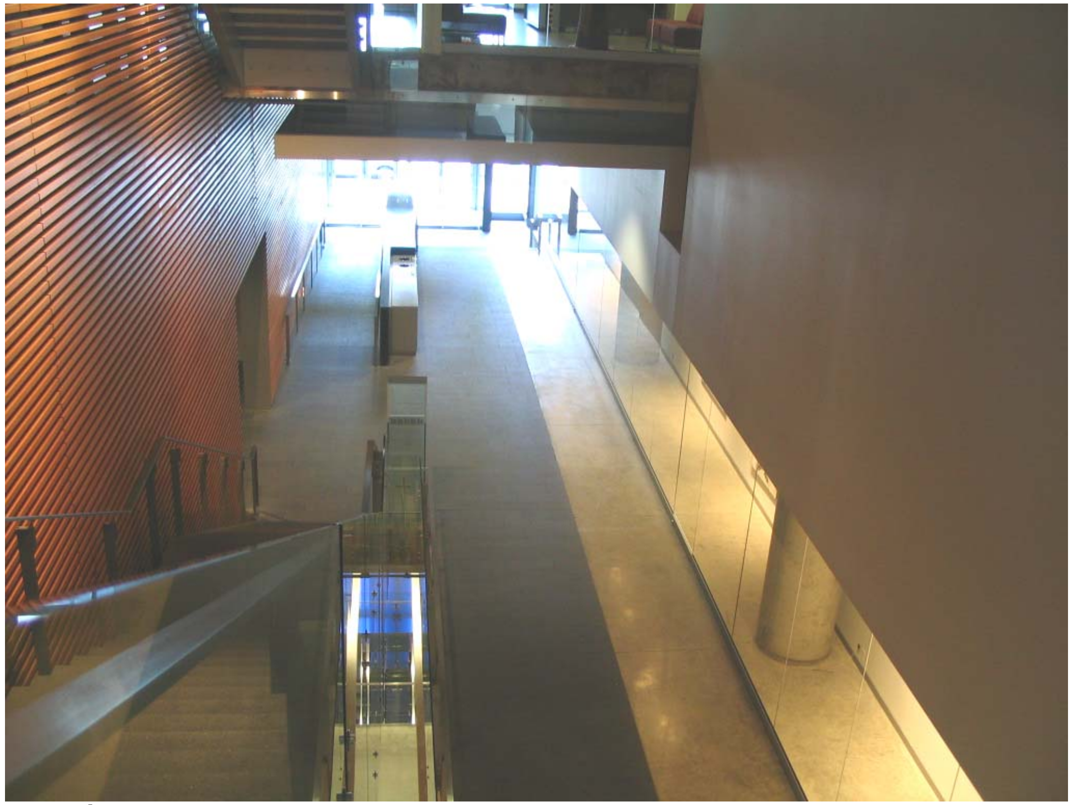
## Location

The FOFA Gallery Vitrines corridor on the 1st floor of the EV building is the targeted location for Jitters. This space was chosen because it features many of the qualities needed for an audio/visual installation.

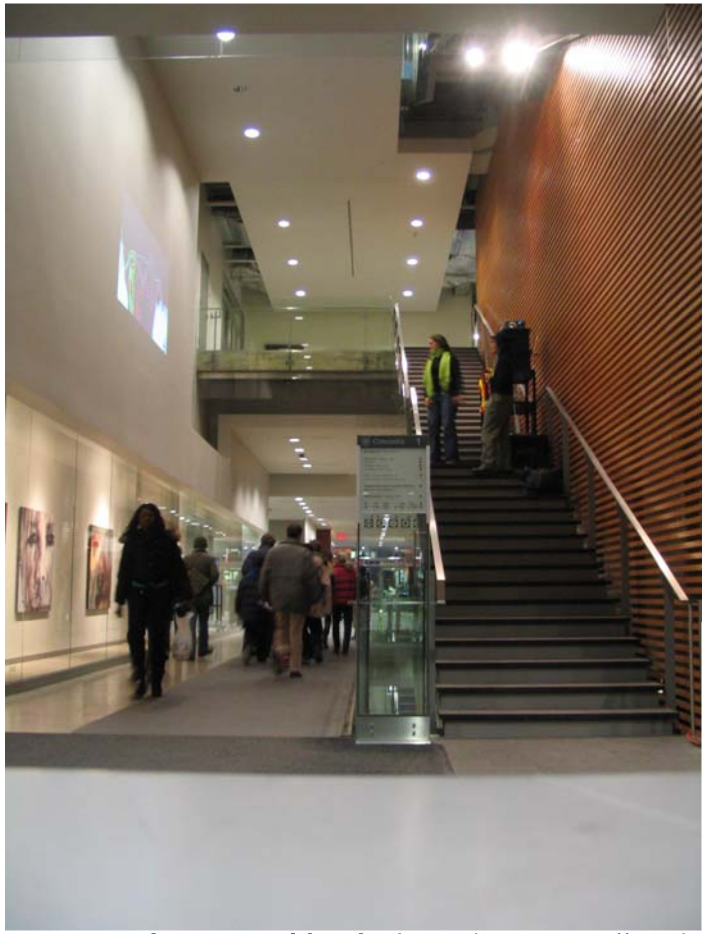
- High traffic.
- White wall for projection, above.
- Good lighting conditions.
- Good support infrastructures (to install projector, speakers, etc).
- The stairs vibrate to movement, an interesting quality to use.

## *Location Photos*

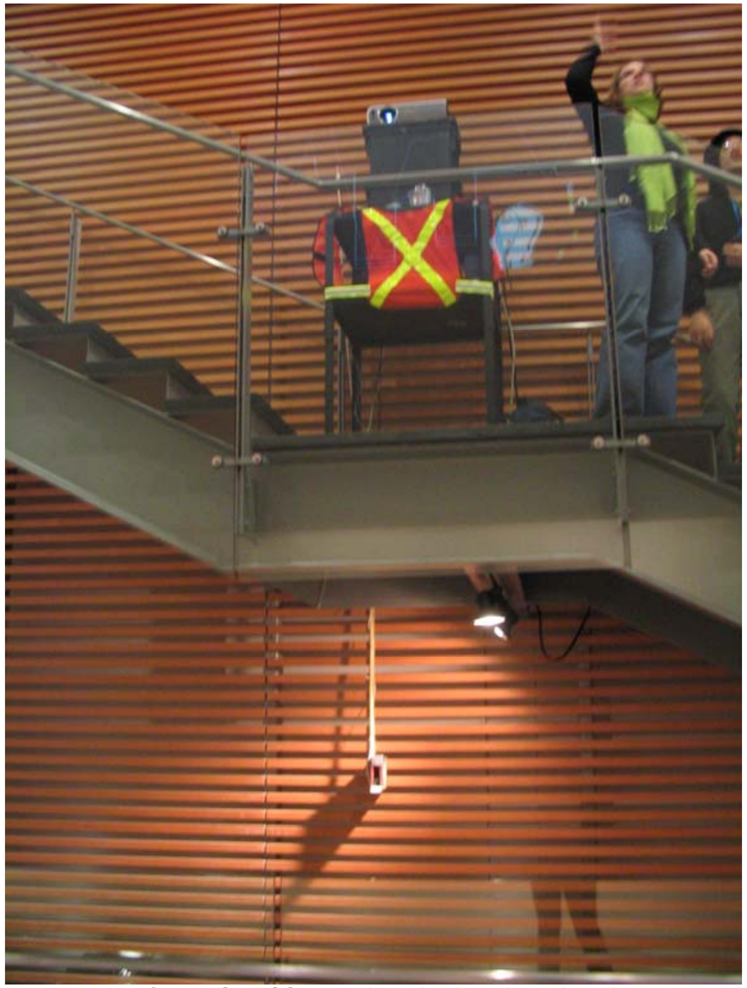

**Ground-level**


**From the Top**

## *Setup Photos*



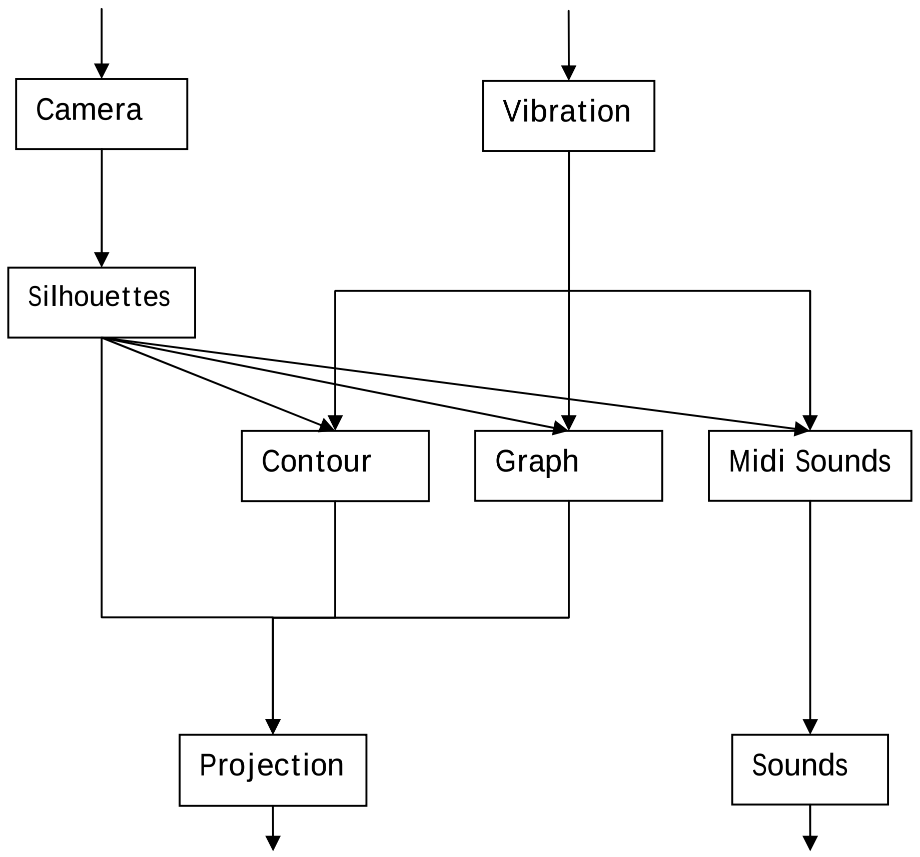
**Setup seen from ground-level.** The projector as well as the iMac rest on the cart on the stairs while the iSight dangles below it.


**Setup seen from the side.** Notice the box hanging down from the stairs? That’s where the iSight is hiding.

## Setup Equipment

- Mac Mini with Max/MSP 4.6 +Jitter 1.6
- Mouse and keyboard
- Projector
- 2 Channel Speaker Set
- iSight
- Cart and boxes to elevate projector above heads of average participants walking down the stairs.
- Extension cables (power, video)
- Power bar
- Contact microphone
- Gaffe tape to tape down the extension cables
- Bamboo sticks to hold iSight
- Granola bar box to house the iSight to create stability and protect from fall

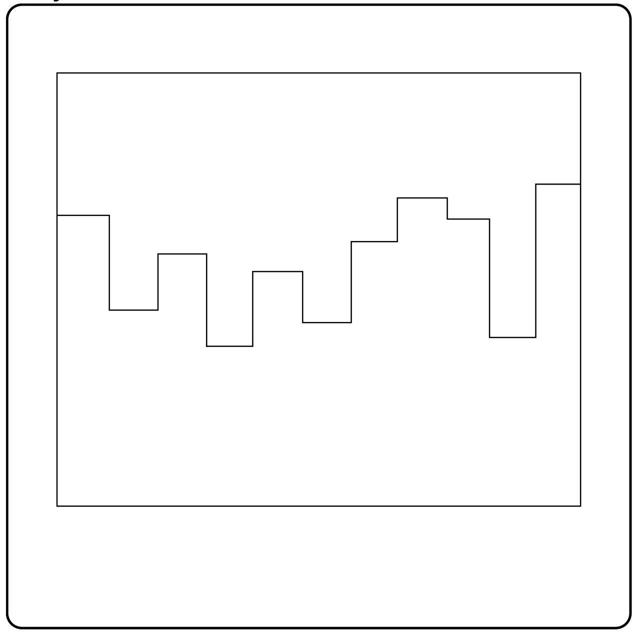
## Software Architecture

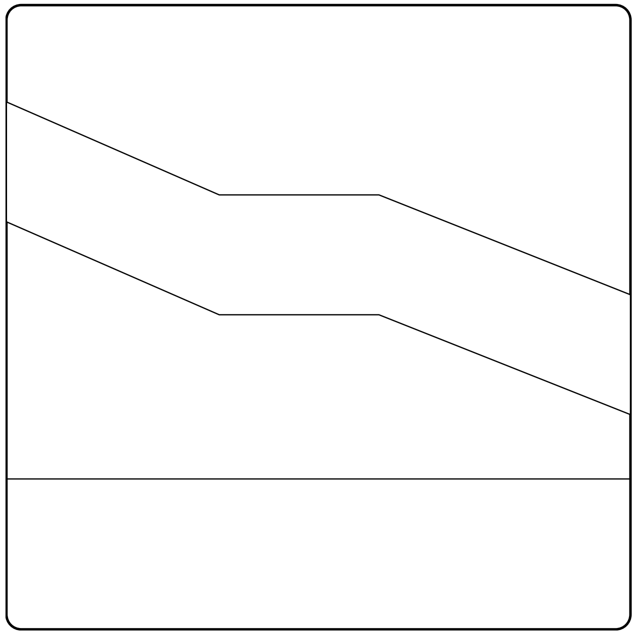


## Use Case Scenarios

### *Example Case A. Minimal*

Projection view / Installation view

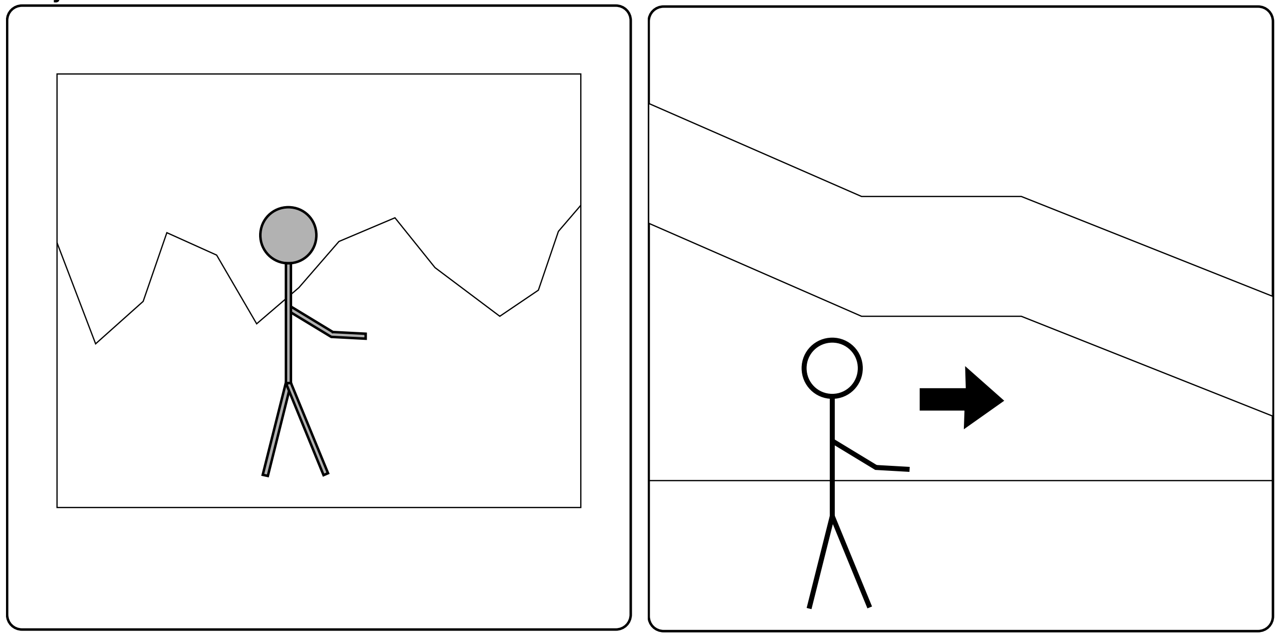




- No activity is detected in the corridor or in the stairs.
- Audio and video are simple.

***Example Case B. Median***

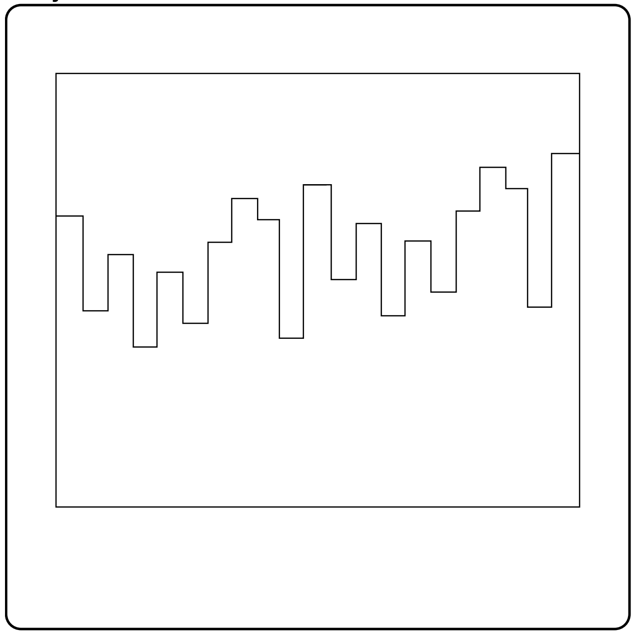
Projection view / Installation view

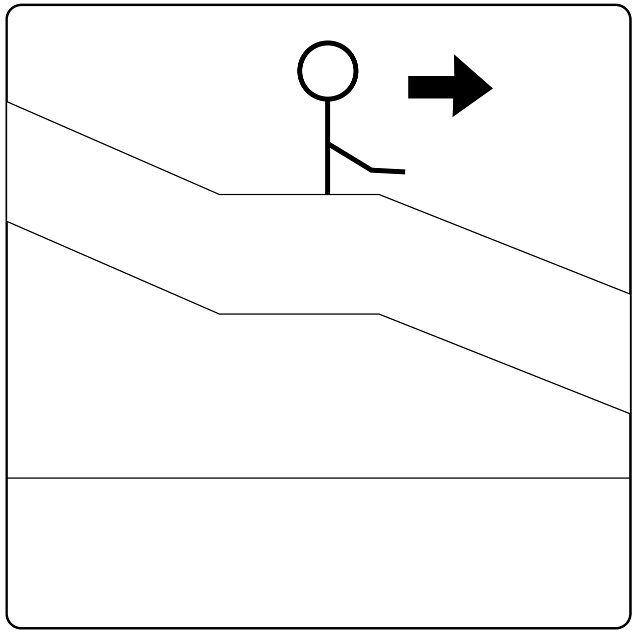


- Activity is detected in the corridor.
- The projection change to an outline and a fill of the activity.
- The graph change to a spiky format.
- Audio changes but keeps playing.

***Example Case C. Median***

Projection view / Installation view

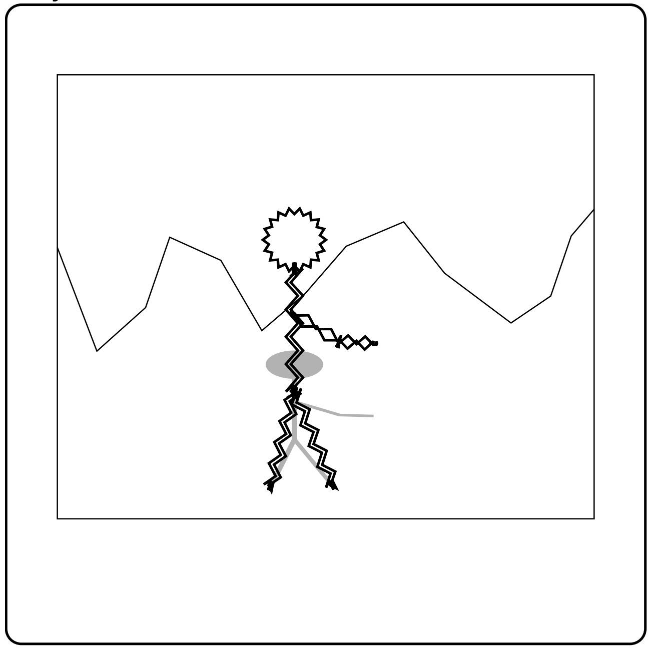


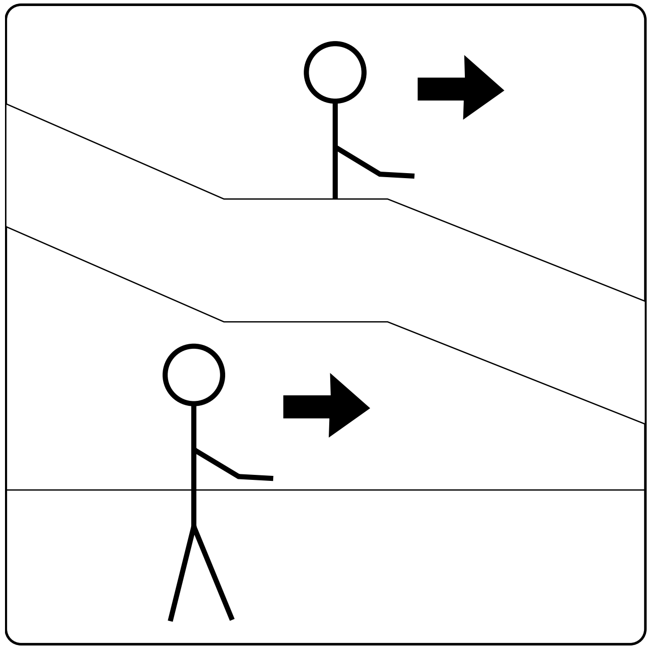


- Activity is detected on the stairs.
- The number of points projected change.
- Audio stay constant.

***Example Case D. Maximum***

Projection view / Installation View





- Activity is detected both in the corridor and on the stairs.
- The projection change to display a jitter contour of the activity detected in the corridor as well as a squash fill of the activity detected.
- The graph change to a spiky format.
- Audio changes again but keeps playing. Audio becomes more frenetic as number of people in front of camera increases.

## Max Patch Overview

The Max/MSP patch used consists of 5 overall parts: Main, Graph, Silhouettes, Outlines, MIDI Instruments. We think of the project as those components and troubleshooting in each of them is generally independent of the others. There are also some utility patches, which handle things like input and instanced number creation. Many of our patches use *patcher attributes* as a novel way to send parameters using just 1 inlet, or as inline presets using the '@' character.

### *Main » Jitters.Main*

`Jitters.Main` is the master patch, like the `main()` function of the common programming languages such as C/C++, Java and the like. It is our "just press start" patch, allowing the system to be opened by anyone and run with minimal efforts. The start-up routine generally involves:

1. Selecting input (live feed or recorded media).
2. Clicking 'start'.
3. Tweaking the effects multipliers.
   - … for the Silhouettes + Squish
   - … for the Outlines + Jitters
4. Pressing 'esc' to go fullscreen.
5. Optionally, pressing 'spacebar' to capture the mean used in background subtraction; whenever necessary.
6. Optionally, clicking 'record' within Jitters.Main to make a recording of input and output.

***Graph » Jitters.Graph + Steps.class (Java)***

This sub-component generates the graph line for the background. Steps.class, the Java extension, generates a cycling list of floating-point values between -0.5 and 0.5 (each point moves up and down independently of the rest). These values are then sent to *`jit.gl.graph`*, which plotted a graph across -1.0 and 1.0 of the x-axis.

There are 4 attributes which could be adjusted to create a variety of graphs and behaviours: mode, count, speed, scale.

| | |
|---|---|
| Mode | » 0 = step function graph. 1 = linear function graph. |
| Count | » the number of points used to generate the graph. |
| Speed | » scales the velocity of each point. |
| Scale | » scales the maximum and minimum values of each point. |
| | » negative values give a “jittering graph” kind of effect (due to double negatives) |

***Silhouettes » Jitters.Squish [tweak at start]***

The *`Jitters.Squish`* sub-patch controls the squishing effect on the background-subtracted silhouettes of people who walk in front of the camera. It does a quick and cheap sound input analysis and squishes the silhouettes when vibrations are detected. The silhouettes will gradually be stretched to normal, over time. It usually takes a start-up tweak on the multiplier and fill (stretch) rate to achieve the right balance. For the presentation, a multiplier of 15 and a fill of 10 worked well.

***Outlines » Jitters.Sobel + Jitters.Jitters [tweak at start]***

*`Jitters.Sobel`* generates the Sobel-filtered outlines, as well as the RGB “snapshots”. It is extremely simple, yet highly effective. The snapshots are made by cycling though color planes using *`jit.unpack`* and *`jit.pack`* once per second (an empty plane selection is also possible).

The outlines are then sent through *`Jitters.Jitters`* to be affected by an 8-point jit.repos, based on the sound input.

***MIDI Instruments » Jitters.MIDI***

*`Jitters.MIDI`* is a generic MIDI generating patch. It produces and plays notes of random length, for whatever instrument of choice. The patch is given a variety of presets that are selected based on the *`cv.jit.mass`* of the background subtracted silhouettes. In theory, if the mass could stay constant long enough, a random melody would play… but since people tend to move quickly across the screen, the most common sound is a jumble of preset selections because each time a preset is selected, it immediately outputs its first note. All the same, if enough time was given to just stand still in front of the camera with a large mass, a rare dimension of the sound system could be heard. Three *`Jitters.MIDI`* instances are used to give depth in the sounds.

## Technical Interests

### Vibration Detection

Our project uses the stairs' vibration as a means to change the projected video. There exist two ways of detecting the vibration: using an accelerometer or using a microphone. Based on research and availability of materials, we have decided to use the latter and have built a contact microphone.

The use of a microphone to detect the vibrations comes from the fact that both sounds and vibrations are waves that propagate through a material. This method has the advantage of using a normal input for Max/MSP+Jitter since the program is design to work both with video and with sounds. The difficulty of this approach is that the ambient noise of the room will contaminate our input.

Based on tests we have performed, we have determined that the best location to put the contact microphone is on the railing of the stairs using a magnet to anchor it.

*» Vibration, http://www.ndt-ed.org/EducationResources/HighSchool/Sound/vibration.htm*
*» From Vibration to Sound, http://www.iit.edu/~smile/ph8813.html*
*» A Beginner's Guide to Accelerometers, http://www.dimensionengineering.com/accelerometers.htm*
*» Physique Ondes, optique et physique moderne, ERPI, 1999, Harris Benson and colleagues*

## Contact Microphone

A contact microphone is essentially a piezo element affixed to a surface with some pressure, connected to an audio input of a system. Pressure waves through the contacted medium will then generate voltage changes when the piezo vibrates, making a useful sound input. The piezo element is largely insensitive to general vibrations in the air, if placed properly. However, if loud noises are made through the air in close proximity to the piezo, a signal may be picked up weakly by the piezo itself, but more significantly as vibrations on the contact surface.

Instructions on how to construct a low-cost contact microphone have been documented by "Erinys" at http://home.earthlink.net/~erinys/contactmic.html and have been referenced by many as a prime contact microphone kit.

Essentially, it comes to:

1. Purchase a piezo buzzer.
2. Break apart piezo buzzer to harvest piezo element.
3. Solder piezo element leads to wires, and wires to a plug, or jack.
4. Attach piezo element to things and record.

The project used a magnet to affix the contact microphone to a steel railing on the staircase.

## Mean Background Subtraction

Background subtraction exhibits problems when the background to be subtracted is created from a single-frame snapshot of the scene, due to the heavy noise in most consumer-level digital cameras. The solution is to do background subtraction on the averaged background, its mean over time. By taking the average image, the noise from any input will already be accounted for when doing the subtraction. The resulting image will have a more uniform distribution within the light and dark areas. Passing such a result through a threshold yields an optimal low-frequency (i.e.: much less noisy) binary image. Without using the mean to smooth out the background noise, background subtraction would result in a high-frequency binary image. Essentially, the mean acts like a low-pass filter.

***The arithmetic mean:***

$$\overline{m} = \frac{1}{n}\sum_{t=1}^{n} m_t$$

$$r = i - \overline{m}$$

Where $r$ is the result matrix, and
$i$ is the input matrix.

***The mean background subtraction, applied frame by frame:***

$$r = i - \overline{m}$$

Where $r$ is the result matrix, and
$i$ is the input matrix.

***The threshold (example, based on 8-bit images):***

$$r'(x,y)=\begin{cases}0, r(x,y)\le k\\ 255, r(x,y)>k\end{cases}$$

Where $x, y$ are matrix indexes, and

$k$ is the threshold value.

Note that the result may be inversed, depending on what is needed downstream.

*» Background Subtraction Techniques, McIvor, http://www.mcs.csuhayward.edu/~tebo/Classes/6825/ivcnz00.pdf*
*» Background Subtraction for Moving Objects, Kamath, August 2005, http://www.llnl.gov/casc/sapphire/background/background.html*
*» Background Sub Using Color & Gradient Information, Vong, http://www.llnl.gov/casc/sapphire/background/background.html*

## Color Image to Binary Image

A large part of our project is related to the movement of the people passing in front of the camera. We determine size to decide on the output of sound. More importantly, we analyze the contour of the participants in front of the camera to create the final output video of the installation.

As such, since color is not necessary for our project and potentially detrimental to our analysis, our first step once a frame is captured is to grayscale the image.

Once a background subtraction has been performed on the image, we go a step further and completely binarize the image.

*» Asif, Conversion of a Color Image to a Binary Image, Coder Source.NET, April 18 2005 http://www.codersource.net/csharp_color_image_to_binary.aspx*
*» MIL 8.0 Guide, Jan 2006, http://www.matrox.com/imaging/products/mil/milguide.pdf*

## Filtering

Despite our best efforts with finding the “best” threshold value, it is inevitable that noise mar our video capture. Since aesthetics is an integral part of this project and clean lines are a must, filtering becomes a necessity.

To clean up the silhouettes of the participants in front of the camera, we must convolve the image using a Gaussian mask. Using such filtering techniques, the silhouettes are cleaner and less ridden with holes.

To clean up the outline of the participants in front of the camera, we perform a close operation on the binary image.  A close operation consists of the basic morphological operations of erosion followed by dilation.

*» Noise filtering, Wikipedia, November 006http://en.wikipedia.org/wiki/Noise_reduction#Gaussian_filters*

## Edge Detection

Once the image is binarized, we will use an edge detection filter to find the smooth contour of the participants. This operation will enable the Jitters.Sobel component to leave shadow outline for the final video of this installation.

Since the Sobel, Prewitt, Cross and Canny filters come with Jitter or cv.jit, we will most likely use one or a combination of the filters to perform the edge detection. As of now, we prefer using the Sobel filter as it gives thicker lines and takes less processing power than the Canny filter.

Sobel filters compute edges by calculating an approximation of the gradient of the image intensity function. The result therefore shows how "abruptly" or "smoothly" the image changes at that point, and therefore how likely it is that that part of the image represents an *edge.*  Sobel uses 2 3X3 kernels to detect horizontal and vertical changes in the images.

In the following, **A** represents the source image and $\mathbf{G_x}$ and $\mathbf{G_y}$ are the resulting images, which at each point contain the vertical and horizontal derivative approximations.

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, using:

$$\mathbf{G} = \sqrt{\mathbf{G_x}^2 + \mathbf{G_y}^2}$$

As a result, regions in the image where there is an abrupt shift in color will appear while consistent color patches will not be visible.

*» Canny, Wikipedia, October 2006, http://en.wikipedia.org/wiki/Canny*
*» Edge Detection, November 2006, http://en.wikipedia.org/wiki/Edge_detection*
*» Sobel, November 2006, http://en.wikipedia.org/wiki/Edge_detection*
*» Sobel filter, David Blythe, August 1999, http://www.opengl.org/resources/code/samples/sig99/advanced99/notes/node258.html*

## Matrix repositioning

The matrix repositioning technique uses a function to reposition each pixel of a matrix individually.

For a **matrix I[i, j],** we have a reposition **function f(x, y)**. Where each element of the m**atrix I** will be reposition using the **function f**. This will result in the **matrix I'**. The size of the **matrix I** and **I'** are m by n.

**I'[i, j] = f(I[x, y])**

where: x goes from 0 to m
y goes from 0 to n

When using this technique, a decision must be taken for a position in **matrix I'** that doesn't exist in **matrix I**. Those positions can either be assigned a constant color or assigned a color based on **matrix I**. That is, they can repeat the last color that was used in the same column and/or row of **matrix I** as if the matrix was circular.

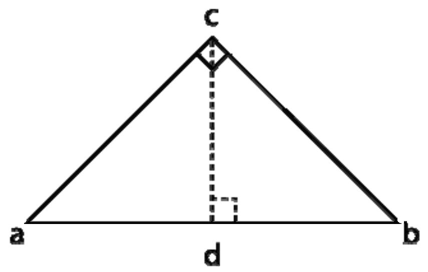
*» JitterTutorial16: Tutorial 39: Spatial Mapping*

### *OpenGL*

OpenGL will be use in Jitters to create the final output of the video for the installation. It can either be used to create a 3D model of the edge that are distort or to directly distort a 2D frame of the video.

Both paths could be explored for their feasibility (such as finding the CPU power needed), since the rendering of each frame should be almost instantaneous. More research and testing for that part of the project will be needed to determine the most efficient way of accomplish our desired visual output.

Mathematically, the only significant portion here was the calculation of the geometric coordinates needed for the camera. This was needed since the default camera position is probably defaulted to work using an aspect ratio of 4:3, which was not our case. Below, points a b c lay on the x-axis, and point c, on the z-axis. Using simple geometric knowledge, we get the following measures:



$$\overline{ab} = 1$$
$$\overline{ad} = 0.5$$
$$\overline{cd} = \overline{ad}$$
$$\therefore \overline{cd} = 0.5$$

*» Obj. Deformation…, http://www.csc.calpoly.edu/~zwood/teaching/csc471/finalproj02/morning/afaruque/*
*» Image Manipulation using OpenGL, http://cg.cs.uni-bonn.de/project-pages/imogl/*
*» Jitter OpenGL Documentation, http://www.cycling74.com*

## Member Roles

***Shared Tasks***

- Design

***Victor Yap***

- Technical research and construction of contact microphone.
- Patch: Jitters.graph, Jitters.midi, Jitters.Main, steps.java
- Patch: Main editor of the patch
- Installation setup
- Blog creator
- Documentation: original editor, max patch and conclusion

***Olivia Tang***

- Technical research mainly on background subtraction and filtering
- Patch: Jitters.bgsubtract.live, Jitters.sobel, cv.jit.mass
- Documentation: Photographer, second editor, introduction, location, selected parts of technical interests

***Geneviève Thérien***

- Technical research mainly on Jit.repos
- Patch: Jitters.squish, Jitters.jitters
- Installation setup
- Website creator
- Documentation: Use case scenarios, software architecture, matrix reposition, video editing.

## Conclusions

### *How well it worked*

Without a doubt, Jitters has been a very successful project. As a team, we have learned a lot about the many things involved in producing a real-time video installation. There were many techniques taught during the lectures, which we were able to experiment with and learn more about in our researches.

Ultimately, a lot of the choices made were made based on the principles of responsiveness. The key for us was to create a system, which covered both the artistic and technical factors of responsiveness. A responsive piece, artistically, is one in which the images and sounds for the project change in direct response to the viewer. Technically, those changes should happen very quickly; in our case, the goal was to maintain a frame rate of 30fps or better. While testing, we had to concede about 10-15fps, in the end, using the older dual-processor G5 systems, but when using a brand new Mac Mini, the extra processing speed allowed us to achieve our target frame rate.

### *Reactions from viewers*

Jitters certainly caught a lot of attention during its limited presentation time. Many people just passing-by seemed pleasantly amused once they figured out that it was responsive/interactive. It appeared, however, that most people used the word “interactive” to describe the system that they were playing with. This probably means that responsive art is still something to be developed within the art world, as a category encompassing interactive art.

### ***Changes that could be made***

Although the project was a huge success, there are a few items that we believe could be changed to improve it even more.  These include:

- Having a smaller platform which can be mounted to the wall to make more space on the staircase; for the projector, computer, and speakers,
- Having a better mount for the camera which is easier to make level. When too many people were stomping on the stairs, the camera would shift slightly and disturb the process of background subtraction. This in turn would make noise artefacts appear, forcing us to reset the mean.

### ***Challenges that were present***

Many of the challenges to this project were in the early development stages, in finding solutions for everything we wanted done.  A big time consumer was getting piezo elements for our custom-made contact microphone, because few shops actually had them in any form (commonly, they're encased piezo speakers). Eventually, a source was found in Laval, Maddison's.  It was also interesting to try to prototype our installation before we had approval from the Dean, with Security and Healthy & Safety personnel wondering about things.

## Appendix A: Special Thanks

- **Dr. Xin Wei** (Project would not exist without him)
- **Chris Salter** (Help with contact microphone)
- **Bill Vorn** (Help with contact microphone)
- **Michael Fortin** (Help with Friday setup)
- **Micheal Longford (Chair of Design and Computation Arts) and Wolf Krol (Associate Dean of Fine Arts)** (permissions and approval of installations)
- **Nick Gauthier** (Thank you for getting us our equipment!)
- **FOFA Gallery Vitrines** (Credit for Bonnie Baxter's Baphomet)

## Appendix B: Milestones

| | | |
|---|---|---|
| Week 1 | - Draft Design Basis | |
| | - Proposal Writing | |
| Week 2 | - Secure Location | |
| | - Secure Standard Equipment | |
| | - Secure Construction Materials | |
| Week 4 | - Complete Design Basis | |
| | - Complete Code Sections | |
| | | - Sound |
| | | - Images |
| | | - Response |
| Week 5 | - Prototype Construction | |
| | - Prototype Test Demo | |
| Week 6 | - Revisions and Additions | |
| | | - Revised / Extra Features |
| | | - Final Design Documentation; based from Proposal |
| Week 7 | - Presentation Construction | |
| | | - Presentation Demo |
| Week 8 | - Final Report Documentation | |