Project 3

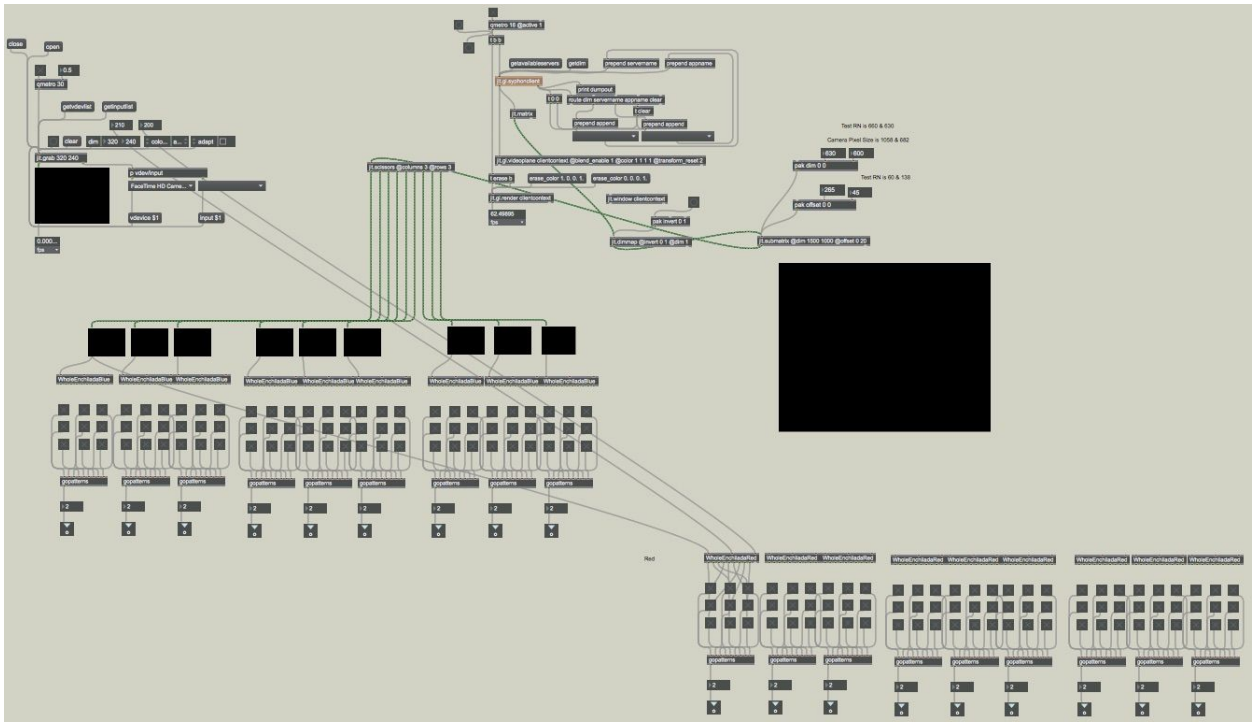
Abstract:

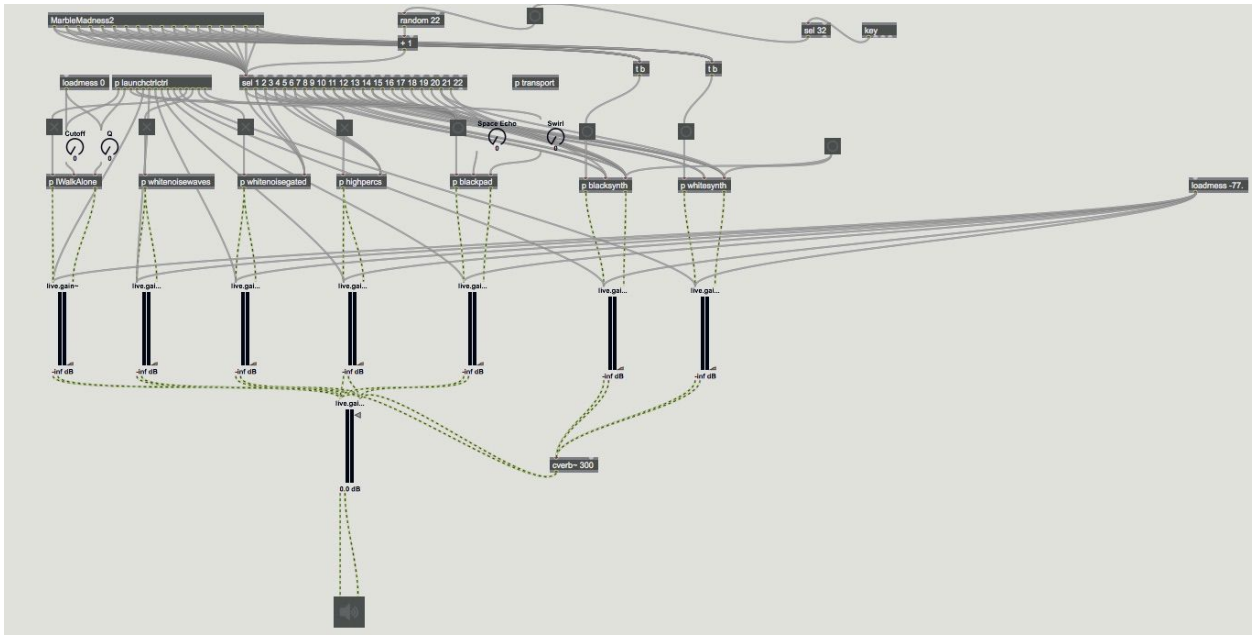
The idea behind this work was to create a piece that utilized the many possible patterns that players create when playing the board game Go. Upon creating particular shapes on the board, patterns from each player were recognized by a camera system and would trigger sound. The piece begins with a drone, and upon the creation of a pattern with a minimum of two stones, different effects and sounds would trigger. Adding on to existing patterns changed the music exponentially. A slowed game pace results in less effect and sample triggering, and the piece fades to completion.

Max Patches:

<u>High level</u>

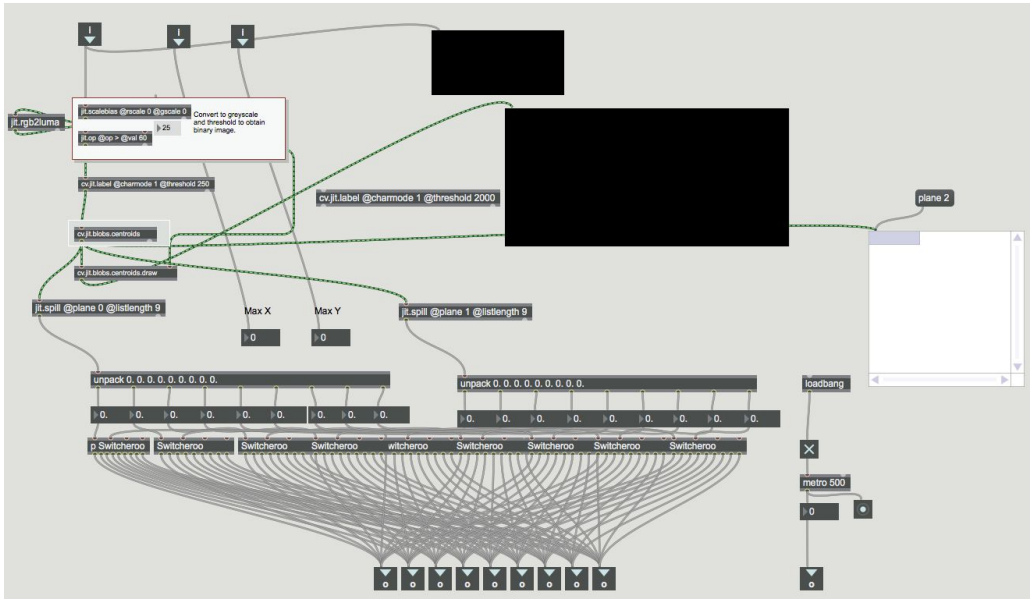
The max patch uses the syphon client to recognize the camera used. This camera inputs a video matrix into max, and offsets and scaling are used to arrange the camera's view to entirely encapsulate the 7x7 Go board. The submatrix patch is also used to do this, and feeds into a jit.scissors object which separates the image of the go board into nine 3x3 overlapping subdivisions of the board. These nine separate videos are then sent to WholeEnchilada patches which determine the pattern coming in from the video. When the pattern is determined, the gopatterns patch translates that pattern into a number which is then sent to a selector which causes a number of events to happen. These events range from effect values and sample triggerings.

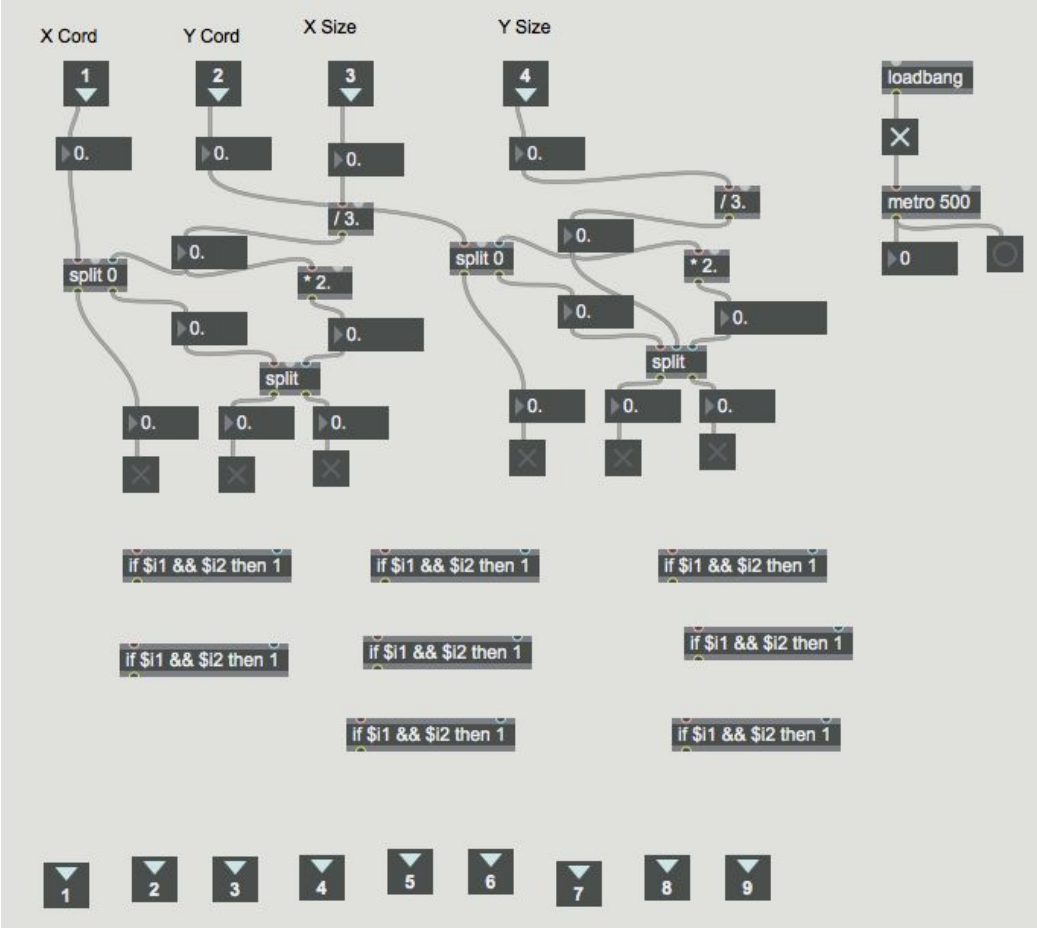




WholeEnchilada

This patch takes in video data for a 3x3 section of the Go board. The patch scales the video matrix to only include either red or blue, to match the red and blue marbles being used. This way when the video is transformed to greyscale, the marble retains significantly more light than anything else on the board. This allows for the cv.jit.centroids object to detect the marbles very well. The x and y coordinates of each blob detected is then passed into the Switcheroo patch. The Switcheroo subpatch determines if each x or y element passed in fit in which third of the video matrix being inputted from the split camera feed. This determines a toggle output that correlates to the position of the marble on the board. This is done for each color and for each subsection of the board. These toggles are all passed into the Gopatterns patch.





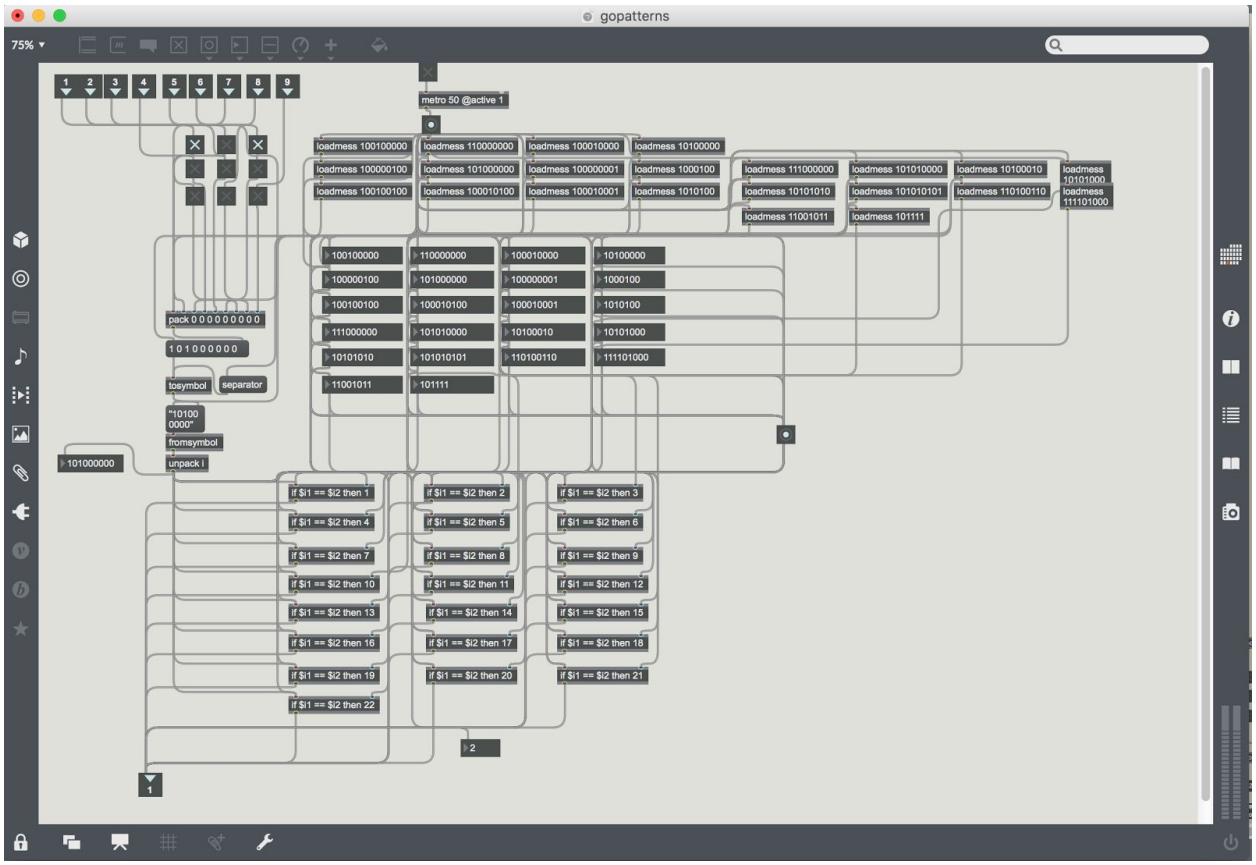
Gopatterns

This patch served to parse the incoming position data into meaningful positions which matched the patterns. To accomplish this each grid was assigned a binary value and compared to a list 9 units long. For example, if a piece was in the top first position along with the third, this would trigger a toggle in our abstracted grid and output a 101000000. A simple conditional statement compared this to a known pattern value and assigned a number to the positioning, in this example; a 2. The end result was transferred over to the next patch to be used as input triggered the sound samples:

Sound Design

The sound design system recieved input from the jitter patch and assigned different values to trigger different effects and samples. If the same numbers happened multiple times it had an additive effect. Every time a piece was played (or a number was given to the sound design

patch) a random sequence from a synth was triggered. I had two synths that were made only using max objects and a third drone that was used with the Poli.amxd object.



Ableton Live Samples:

The Go Board:

This 7x7 Go board is made of ½-inch thick Birch wood plywood, finished with an Espresso colored oil stain, and features a hand-carved grid with notches on intersections for our green and red marble pieces. Green and red were chosen for high contrast with the board and any potential surroundings (it should be clear we made extensive use of image processing, and it's ultimately quite finicky). Unrelated to our in-class performance, I'd just like to mention that I designed the board with a hollow space in-between the bottom and the top such that "graveyard" components could be added and stored as board drawers in the future.

Team Functionality:

Gladstone Butler- Sound Design

Jack Kasbeer- Physical Components

Nick Pourazima- Max patches: input handling, grid splicing, position/pattern matching.

Garrett Osborne- Max patches: pattern recognition, camera angling