To what extent is Max/MSP a viable way of improving control over a lighting system in a Live Sound setting?

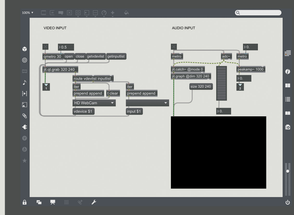
By: James Morton
U1352290

Project tutor: Dr Ian Gibson

INTRODUCTION

This aim of this project was to create a Max/MSP patch that would control stage lighting and create follow-spot effects with minimum user interaction once initialized.
The methodology of this task was to break the program down to individual sections, and prototype these seperately before connecting them together. This poster follows the program through the 4 originally planned sections, explaining how they work and relating this to initial implementation plans from report 1, where appropriate.

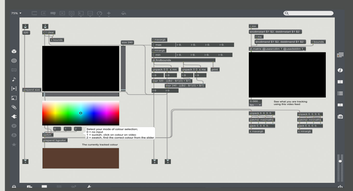
AV INPUT



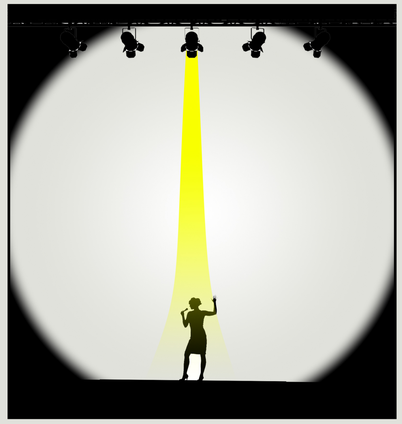
This is the first stage of the program. The video side of this patch allows you to select currently connected cameras and sends this video feed to the "tracking" patch, where you can use the video to track a colour.

The audio side of the patch was planned to be used as a way to control the brightness of the lights connected, so when the music was louder the lights would be brighter, and vice versa. In theory this works, as the 0-1 peak amp input is scaled to the required 0-255 DMX value. This works up until the sending of the DMX message, however was not tested in practice as the available lights could not be turned up past half power, so for safety I used a single number input.

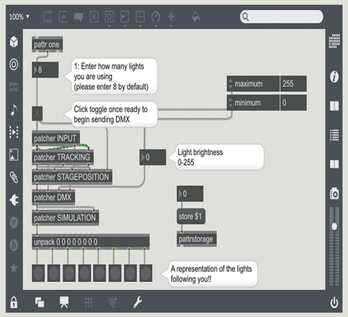
MOTION TRACKING



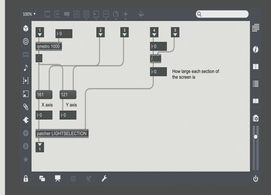
The two video screens on this patch allow you to track a selected colour. Originally only implementation of the Suckah tool (clicking on the video output from the AV patch) was planned, however through creation a Swatch (a RGB colour slider) was added for choice in how the colours could be selected. The patch uses findbounds to select a range of values to track. The middle values of these ranges (X and Y) are output to the rest of the patch.



Main Patch Window

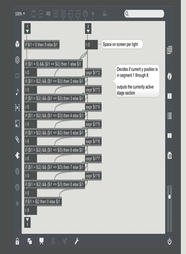


LIGHT SELECTION

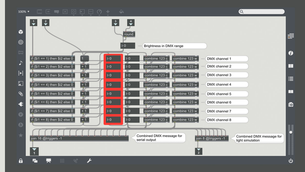


STAGEPOSITION (left) takes the X axis value (R-L) and updates it every second. It also creates the size of the segments of the stage by taking the width of the video (320) and dividing by how many lights the user has defined to use, 320 / 8 makes each segment '40' wide.

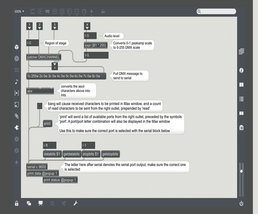
LIGHTSELECTION (right) tells the program which segment the tracked object is currently in by comparing the X axis value by bands in increments of the segment widths.



DMX COMMUNICATION

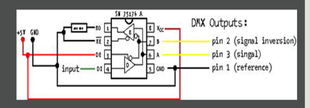


DMXCHANNEL (left) matches the currently active video segment with the selected light brightness. This creates a message of 8 DMX channels, along with a DMX value for each channel, so altogether a 16 character long message. This gets output to the main DMX patch.

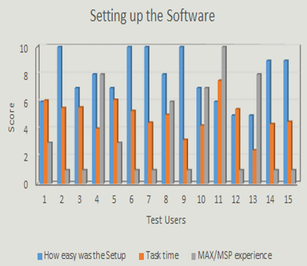


DMX (right) takes these 16 ascii characters and converts them to ints, so that the serial output can send them to the teensy DMX shield connected to a USB port. The DMX message is in the form "1c 0w ... " because that is how the pre-written code *SerialToDmx* operates.

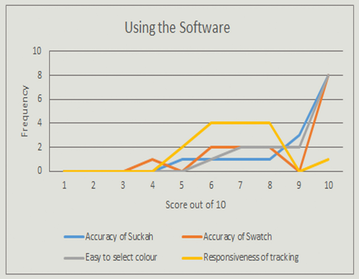
The program was connected to DMX using a Teensy 3.6 board connected to a shield using the circuit diagram to the right. This transceiver based circuit takes the 0-5v power from the Teensy board and scales it to -2.5v to +2.5v for powering DMX transmission. Pins 2 and 3 are phase inversions of the data being sent from the MAX patch. These three outputs are connected to the first 3 pins of a 5-pin XLR socket.



RESULTS AND CONCLUSIONS



To test the software, subjects were asked to complete a task using the program. This consisted of setting up the program ready to begin sending DMX messages to lights. Following the instructions, the average time for the task to be completed was 5:10:11 (minutes). This is shown against how experienced the users described themselves with using Max/MSP, and the spread of results for this and how easy they found the task show that knowledge of this is not a pre-requisite for using the program.



The task involved using the different methods of selecting colour to track. Users were asked to rate their experience using the Suckah and Swatch, along with the colour selection in general and the responsiveness of the system. In general, how responsive users found the system was reliant on the environment the testing was in, which sometimes was not perfect. However these results show that users found the colour selection to be accurate within their environment.

In conclusion, the program successfully performs what it intended to do. With further refinement, the system could be improved to fucntion better under different surroundings. At the moment the conditions shown in the accompanying video are perfect, a well lit room with a plain background to provide contrast.

However, test feedback was positive, with users giving an average enjoyment score of 8.4/10.

In addition, when asked if they thought they program had real life applications, 9/15 completely agreed that it did. Here is a representation of what they thought it could be used for, which correlate with the idea I had for the project originally: