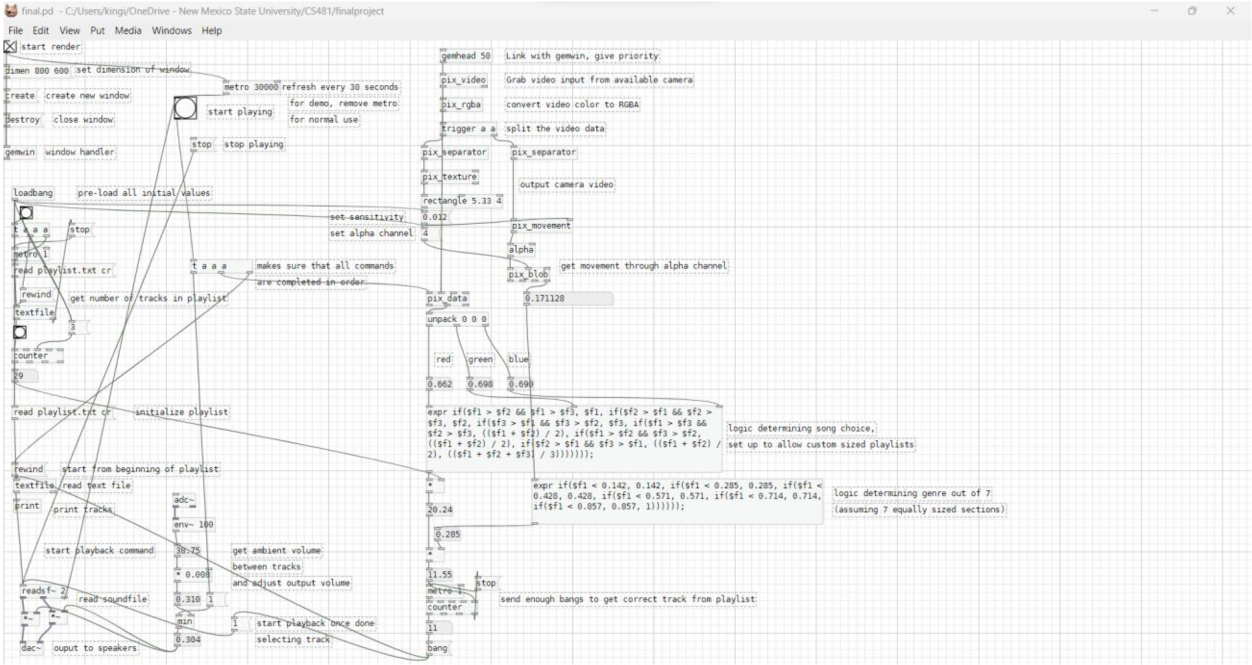
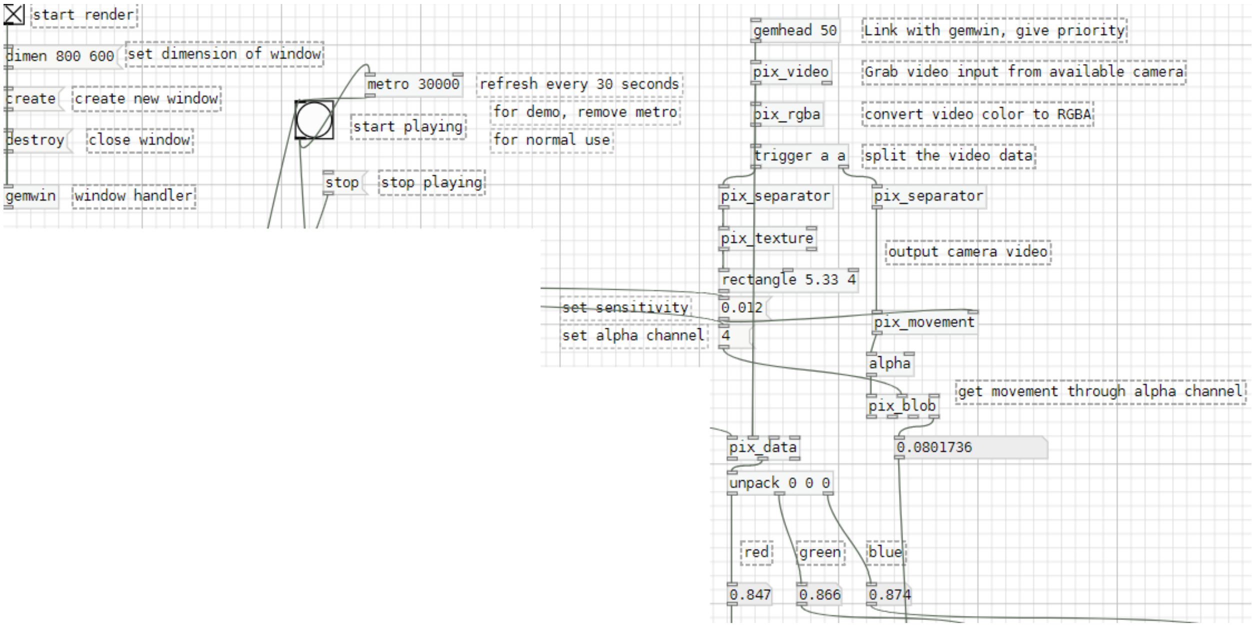


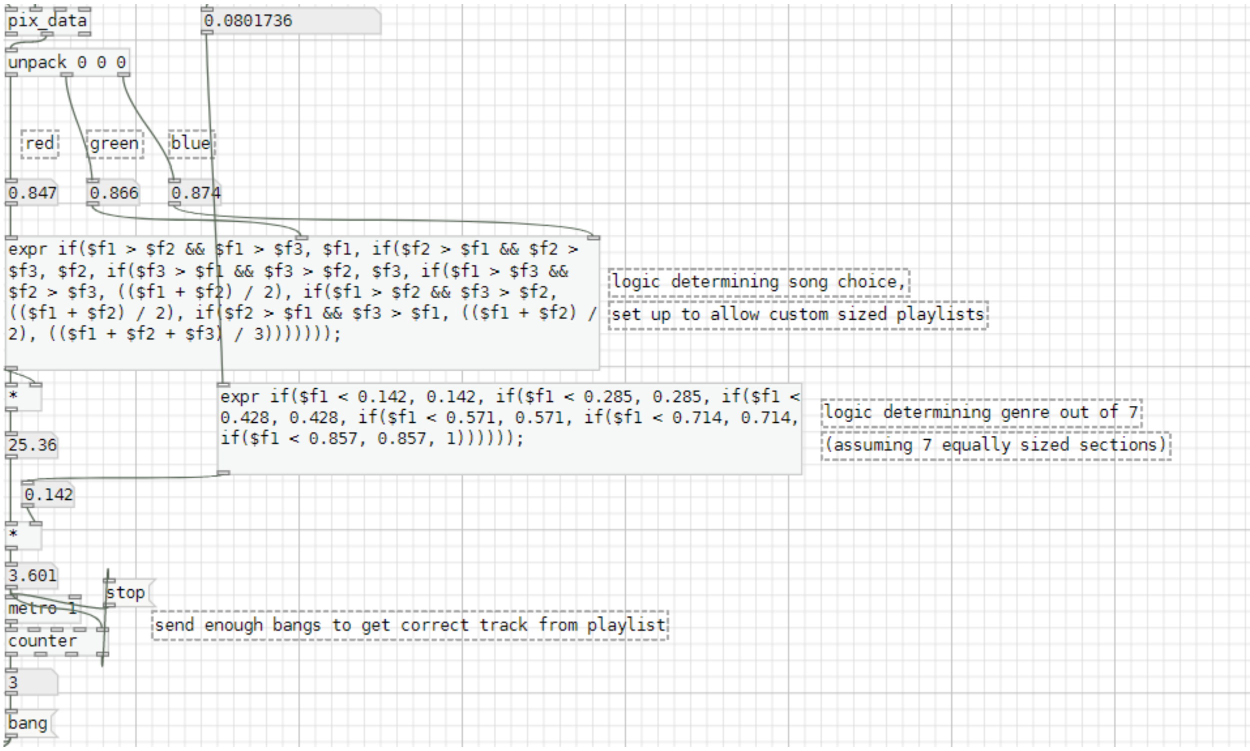
```
expr if($f1 > $f2 && $f1 > $f3, $f1, if($f2 > $f1 && $f2 >
$f3, $f2, if($f3 > $f1 && $f3 > $f2, $f3, if($f1 > $f3 &&
$f2 > $f3, (($f1 + $f2) / 2), if($f1 > $f2 && $f3 > $f2,
(($f1 + $f2) / 2), if($f2 > $f1 && $f3 > $f1, (($f1 + $f2) /
2), (($f1 + $f2 + $f3) / 3)))))));
```

```
expr if($f1 < 0.142, 0.142, if($f1 < 0.285, 0.285, if($f1 <
0.428, 0.428, if($f1 < 0.571, 0.571, if($f1 < 0.714, 0.714,
if($f1 < 0.857, 0.857, 1))))));
```

Our program takes in a video and microphone input, plays a track from a variable list of music using the color and motion data from the camera, and adjusts the volume of the music to match the room's volume.



First, we'll go over the video rendering and data retrieval from it. The user creates the window and starts the render using gemwin, that calls the gemhead command, connecting to pix_video, which is the camera, pix_rgba converts the video to a usable color format RGBA, which gets sent to the RGB retrieval. The RGB values are left as decimal values between 0 and 1 to represent the intensity of each color. The video is split into the output for the user to see and the processing for movement through the alpha channel. The pix_blob gets the movement through the alpha channel, where the sensitivity is set to 0.012 by default, but can be changed if needed.



```
expr if($f1 > $f2 && $f1 > $f3, $f1, if($f2 > $f1 && $f2 >
$f3, $f2, if($f3 > $f1 && $f3 > $f2, $f3, if($f1 > $f3 &&
$f2 > $f3, (($f1 + $f2) / 2), if($f1 > $f2 && $f3 > $f2,
(($f1 + $f2) / 2), if($f2 > $f1 && $f3 > $f1, (($f1 + $f2) /
2), (($f1 + $f2 + $f3) / 3)))))));
```

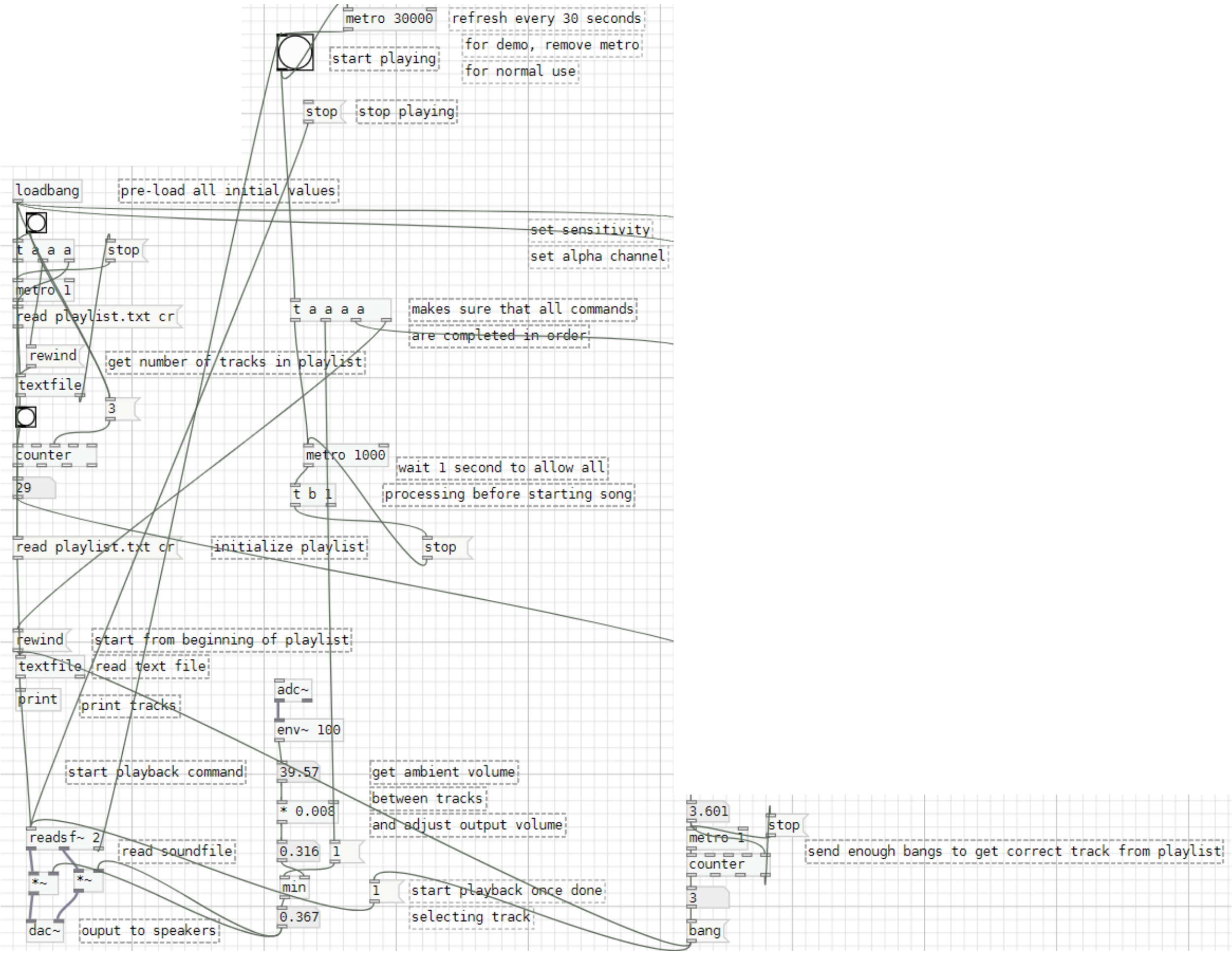
logic determining song choice,
set up to allow custom sized playlists

```
expr if($f1 < 0.142, 0.142, if($f1 < 0.285, 0.285, if($f1 <
0.428, 0.428, if($f1 < 0.571, 0.571, if($f1 < 0.714, 0.714,
if($f1 < 0.857, 0.857, 1)))))));
```

logic determining genre out of 7
(assuming 7 equally sized sections)

send enough bangs to get correct track from playlist

From the video output, we go into the logic for song selection. The RGB values get put into a nested if statement that checks to see what the color is of the room, this is based on 7 different color ranges, that value is multiplied by the size of the playlist. That value is then multiplied by 1 of 7 values from 1/7 – 1, incremented by 1/7 retrieved from the motion data, to select a genre from the playlist. This then gets entered as the max value of a counter, so that we send the appropriate number of bangs to the song selection.



Finally, this last section includes a loadbang that initializes all of the values to start the program including the start value for the playlist's size counter to account for counter reset bangs and because the textfile command starts at 1, we read in the playlist, and set up the alpha channel/sensitivity for motion processing so the user does not need to do that manually. The user will press the bang to start playing music, that bang also gets triggered every 30 seconds for the demo, but at the end of the song normally. That bang connects to a trigger that rewinds the reader for the playlist text file to the top, gets the color and motion data, and gets the room's volume in that order. The already processed video data from earlier gets sent to read the text file by reading it line-by-line until the appropriate song it reached. During the ~1 second that processing takes, volume is checked and adjusted. Once the command, "open 'song_path'", is used, we then play the song and once the song completes, a new bang is sent to repeat the process.