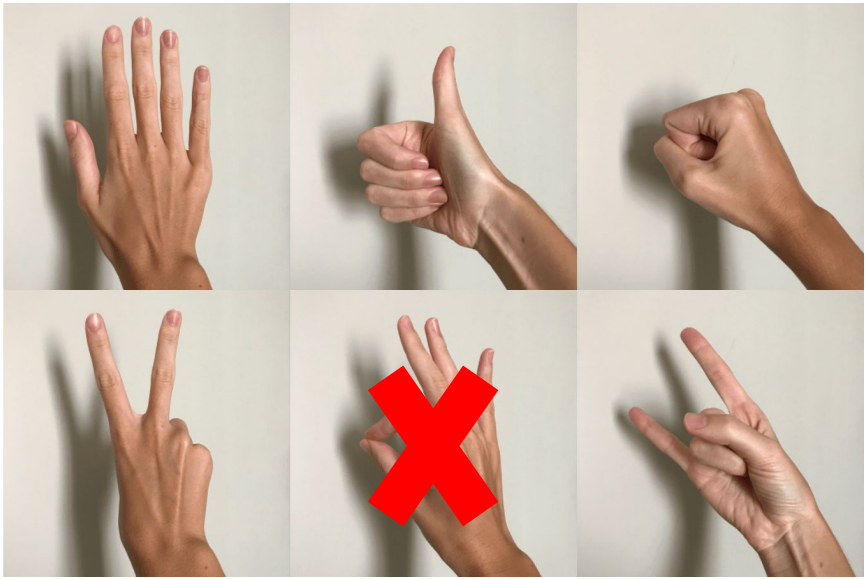
Hand Pose and Instrumental Control

Tiankai Li, Lujie Wang, Ruby Zhang

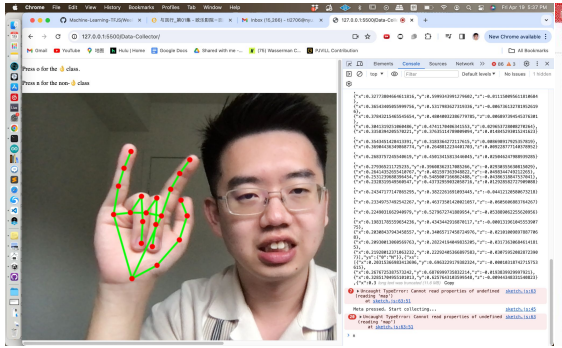
Introduction

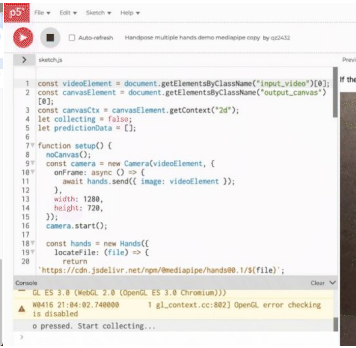


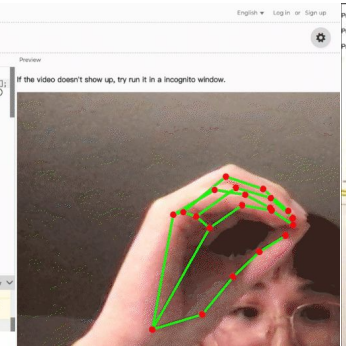
Label O: Trigger Hand Pose: Hand pose 👌.
Instrument control gets initiated and music starts to play 🎵

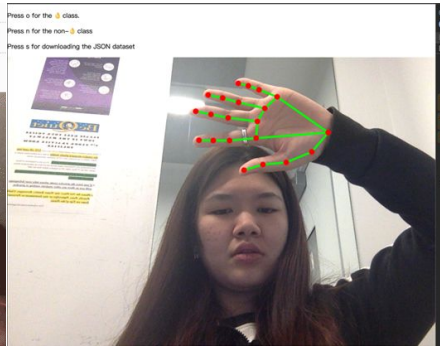
Label N: Other random hand poses other than 👌.
The Instrument gets paused.

Data Collection

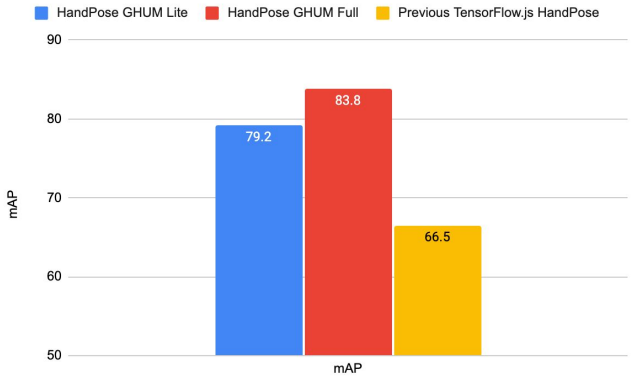








This data collector is based on TensorFlow hand pose detection model: https://blog.tensorflow.org/2021/11/3D-handpose.html. The model has a high performance and low latency. 😄



Hand model evaluation on American Sign Language dataset

Data Collection

How does it work?

- **Eventlistener:** “keydown” eventlistener listens to the user’s key and put it as our y-train label. Auto saves JSON dataset when “s” key is pressed.
- **Timer:** During the “timeout”, data from TensorFlow Handpose model is appended to predictionData array.
- **JSON.stringify:** This function deep copies predictionData array and put it into JSON data format. We later transfer **JSON into numpy array** for our dataset.

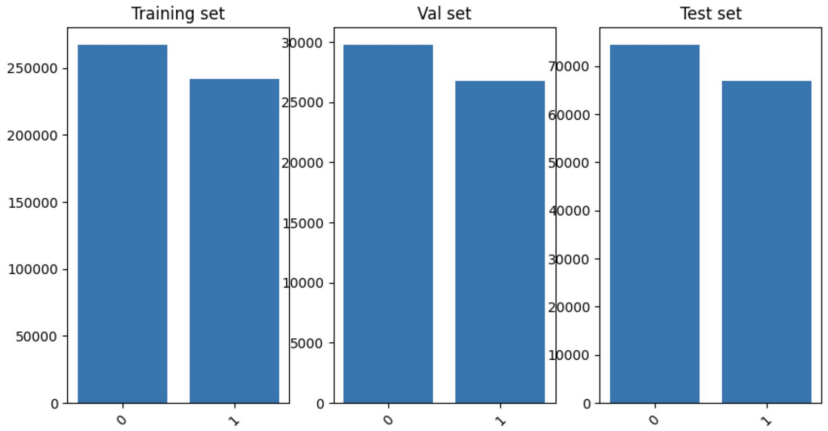
```
document.addEventListener('keydown', keyPressed);

function keyPressed(event) {
  if (event.key !== 's') {
    if (!collecting) {
      key = event.key;
      collecting = true;
      console.log(`${event.key} pressed. Start collecting...`);

      setTimeout(() => {
        console.log('Stopping collection...');
        collecting = false;
      }, 5000);
    }
  } else if (event.key === 's') {
    const labeledData = {
      data: predictionData
    };
    const blob = new Blob([JSON.stringify(labeledData)], { type: 'application/json' });
    const url = URL.createObjectURL(blob);

    const a = document.createElement('a');
    a.href = url;
    a.download = 'labeledData.json';
    document.body.appendChild(a);
    a.click();

    document.body.removeChild(a);
    URL.revokeObjectURL(url);

    console.log('Data saved and download initiated.');

  }
}
```

Code: https://github.com/RubyQianru/Hand-Pose-and-Music-Control/tree/main/Data-Collector

Data Preprocessing

Dataset

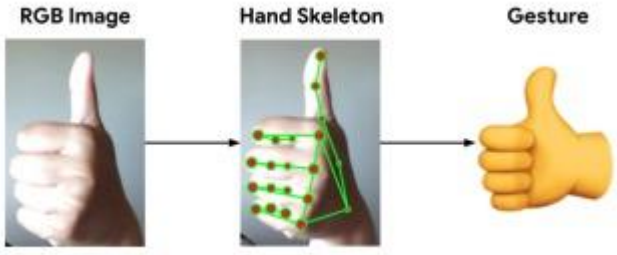
- 7 JSON files combined with a total of 600,000+ data samples.
- The dataset is preprocessed and split into **training set, validation set, and test set**.

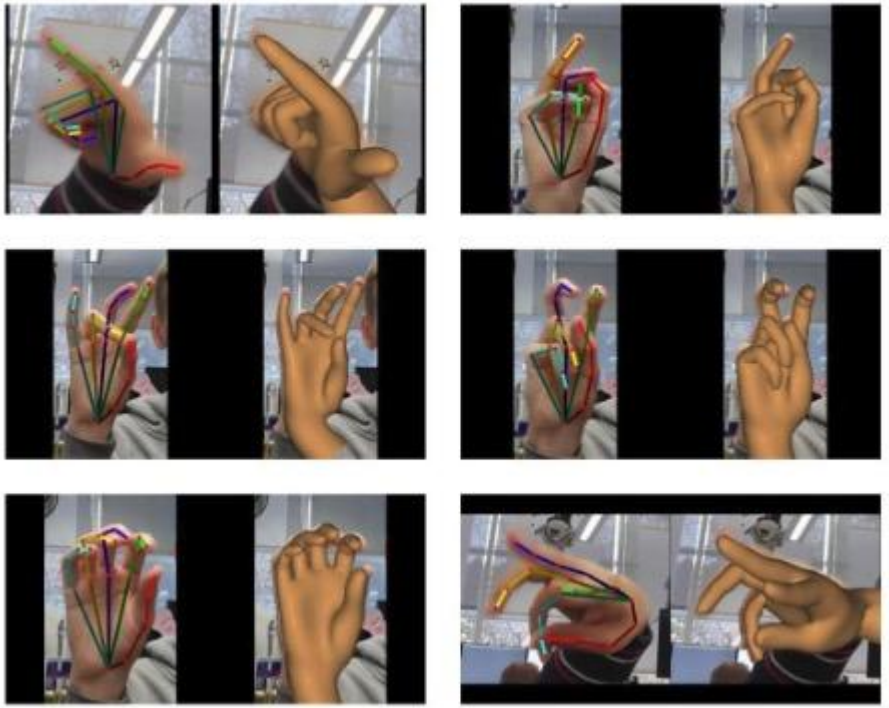



```python
def load_data(file_paths):

    all_x_data = []
    all_y_data = []

    for file_path in file_paths:
        with open(file_path, "r") as file:
            json_data = json.load(file)

        x_data = [sample['xs'] for sample in json_data['data']]
        x_data = [[coord for dict_ in entry for coord in (dict_['x'], dict_['y'], dict_['z'])] for entry in x_data]
        all_x_data.extend(x_data)

        y_data = []
        for sample in json_data['data']:
            label = sample['ys']['0']
            if label == 'o':
                y_data.append(1)
            elif label == 'n':
                y_data.append(0)
        all_y_data.extend(y_data)

    all_x_data = np.array(all_x_data)
    all_y_data = np.array(all_y_data)

    return all_x_data, all_y_data
```

Machine Training

- Using TensorFlow.js with MediaPipe Hand Pose Detection model to capture user hand gestures in real time.
- Identify and track hand positions and postures in videos, capturing 21 * 3 (x, y, z) key points of the hand.
- Process and standardize captured gesture data for music control.





Neural Network Structure

- The first example model is a Keras sequential model object representing the built model. The model architecture consists of **four Dense layers**: the first dense layer of size 64 with relu activation, the second dense layer of size 32 with relu activation, the third dense layer of size 2 with relu activation, followed by an output layer with a single unit and sigmoid activation function. The model is compiled with the **binary cross-entropy loss function**, an adam optimizer, and the accuracy metric.

- The second example model has the same structure but only three dense layers of size 64, 4, and the same output layer.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                4096

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 2)                 66

 dense_3 (Dense)             (None, 1)                 3

=================================================================
Total params: 6245 (24.39 KB)
Trainable params: 6245 (24.39 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 64)                4096

 dense_5 (Dense)             (None, 10)                650

 dense_6 (Dense)             (None, 1)                 11

=================================================================
Total params: 4757 (18.58 KB)
Trainable params: 4757 (18.58 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Machine Performance

- Model Performance: Two example models both perform an **accuracy of approximately 88%** with a 96% and 99% rate of recalling a target hand pose.
- Model demo and model report is based on DL4M Homework I.

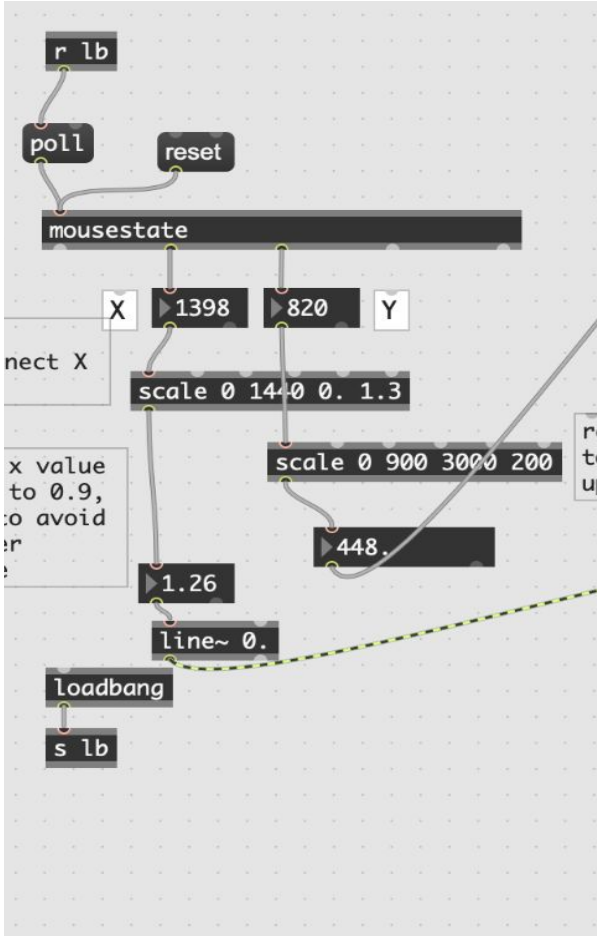
```
This handpose is the target handpose.
1/1 [==============================] - 0s 34ms/step
The model says this handpose is the target handpose.
```

```
This handpose is not the target handpose.
1/1 [==============================] - 0s 23ms/step
The model says this handpose is not the target handpose.
```

```
4414/4414 [==============================] - 3s 679us/step
                    precision    recall  f1-score   support

    Target Handpose       0.84      0.96      0.90     74302
Not Target Handpose       0.95      0.80      0.87     66944

           accuracy                           0.88    141246
          macro avg       0.89      0.88      0.88    141246
       weighted avg       0.89      0.88      0.88    141246
```

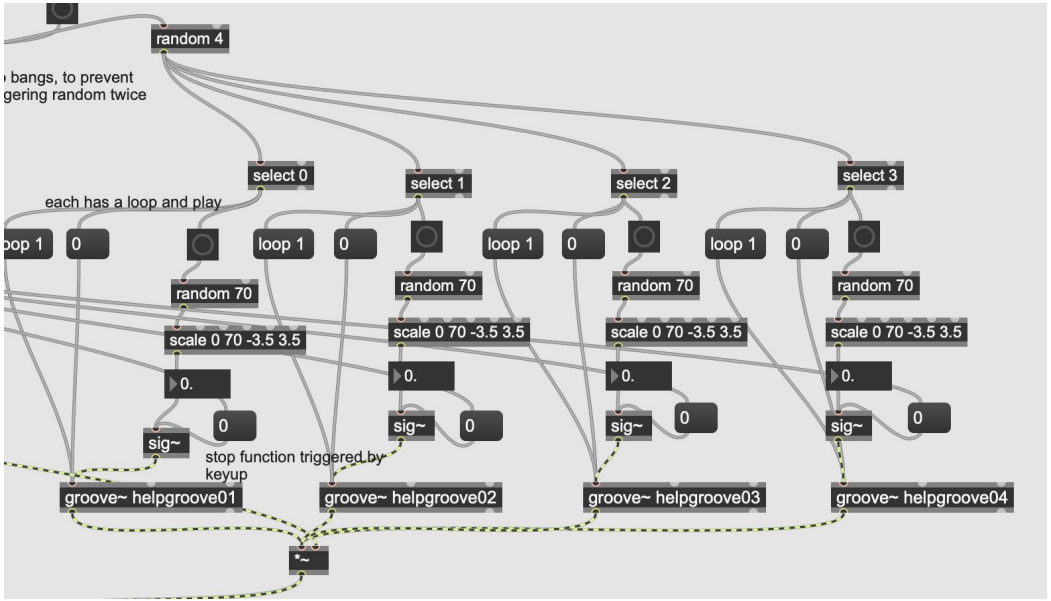
Instrument Design: Tiankai Li

- Similar to theremin, the Max Patch would allow users to control the pitch of the sound by changing hand positions of the camera, and controlling the volume by sliding your hand left and right.
- The Max patch will take coordinates from the hand gesture capturing model tensorflow js file in real time



Timbre Control

- Not only pitch, but also the wavetable can be controlled by the patch; using deep learning model to recognize the hand gesture of the sound, in order to change the type of sound the instrument is going to trigger

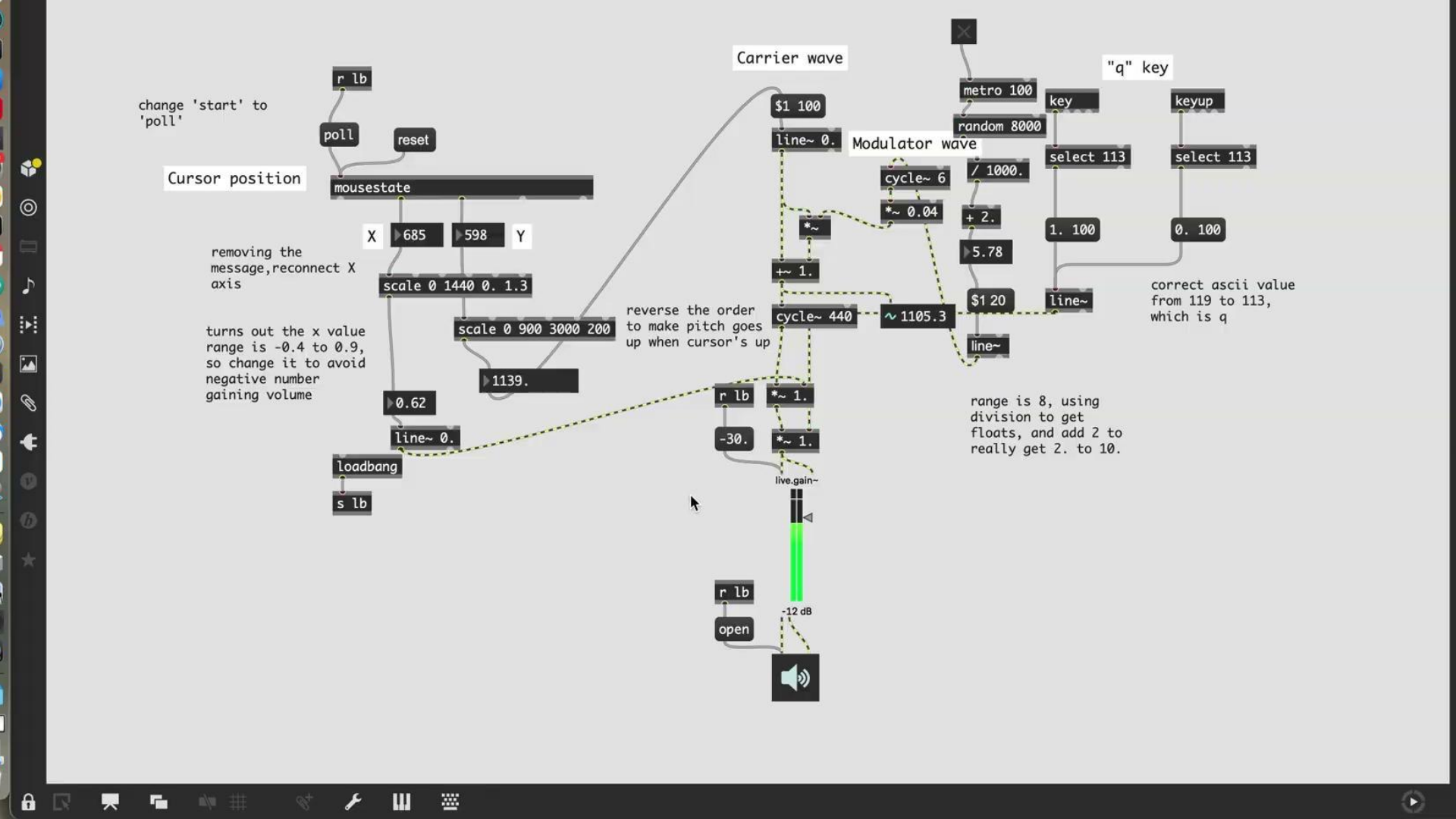


Realtime Data Transmission

- Max Msp internally support .js javascript code, using external compiler to edit the code and have it run within Max environment. Using this function, the coordinate matrix and hand gesture will be imported real time into Max Msp environment to control the instrument
- Create a server environment using Websocket that receives our model output (prediction result, x, y axis of the hand pose) and transmit the data to outlets in MAX MSP.

Javascript Patcher Scripting in Max/MSP
https://www.youtube.com/watch?v=uQBaZdqpnio

```
1  function bang() {
2      outlet(0, "Bang received at " + new Date().toLocaleTimeString());
3  }
4
5  function randomInt(max) {
6      outlet(0, Math.floor(Math.random() * Math.floor(max)));
7  }
```


change 'start' to 'poll'
r lb
poll
reset
Cursor position
mousestate
X
685
598
Y
removing the message,reconnect X axis
scale 0 1440 0. 1.3
turns out the x value range is -0.4 to 0.9, so change it to avoid negative number gaining volume
scale 0 900 3000 200
1139.
0.62
line~ 0.
loadbang
s lb
Carrier wave
$1 100
line~ 0.
reverse the order to make pitch goes up when cursor's up
cycle~ 440
*~
+~ 1.
r lb
-30.
*~ 1.
*~ 1.
live.gain~
-12 dB
r lb
open
Modulator wave
cycle~ 6
*~ 0.04
/ 1000.
+ 2.
5.78
~1105.3
$1 20
line~
line~
range is 8, using division to get floats, and add 2 to really get 2. to 10.
"q" key
metro 100
random 8000
key
keyup
select 113
select 113
1. 100
0. 100
correct ascii value from 119 to 113, which is q

Thank you!