

Tagebuch Processing Patch und Ableton Projekt Nicolai Schneider

Problematik: HI-Objekt

Der Plan war zuerst, die Daten des Tablets per Max auszulesen und dort das **HI-Objekt** zu nutzen. Leider stellte sich heraus, dass es in Windows einige Einschränkungen hat und es mit dem Stifttablet nicht funktionieren würde.

Der Lösungsversuch: Das Interface in Processing zu schreiben, da ich dachte, dass dort Libraries und andere Funktionen mit mehr Kontrolle geben können.

Der erste Processing Patch


Die ersten Schritte in Processing waren natürlich kleine, da mein letzter Patch im ersten Semester geschrieben wurde. Glücklicherweise habe ich mich schnell wieder ans Programmieren gewöhnt und mich unserer Problemstellung gewidmet. Nach vielen Google-Suchen bin ich darauf gekommen, die **jpen library** zu nutzen. Diese hat nach langem Gefummel leider nicht wirklich geklappt - dann habe ich endlich die **tablet library** von **Andres Colubri** im Library-hinzufügen Dialog von Processing gefunden. Mit dieser ist es per `tablet.getX()`, `tablet.getY()`, `tablet.getPressure()` usw. sehr intuitiv die Stiftdaten zu lesen.

```
1
2
3 | Tablet tablet;
4
5 void draw()
6 {
7   println(tablet.getPressure());
8
9 }
10
11
```

Dann habe ich angefangen, die Library und Processing mehr zu verstehen (s.o.). Daraufhin ging das richtige Programmieren los und schnell wuchs das Programm und seine Funktionen.

Features und Funktionen

Bei einem Meeting hatten wir einige Features, die der Patch haben sollte, niedergeschrieben:

**Nicolai Schneider** 20:55 Uhr
Todo Nico:

- Multiple OSC Messages aus Processing schicken
- Zoom ins Bild in Processing mit Blur/Fokus Effekt
- Heatmap-Bilder versuchen
- Distanz zu Koordinaten
- Clips in Ableton abfeuern per OSC
- Menü für Auswahl von Maus und Stiftmodus

(bearbeitet)

Das war auch grundsätzlich mein Workflow: In jedem Meeting das gewünschte Featureset zu klären und dann im besten Falle bis zum nächsten Meeting alles zu implementieren. Eines dieser Features war die **getPixel()** Funktion, welche eigentlich nur von uns als „nice to have“ eingestuft wurde, da sie erst mal als schwer umzusetzen galt. Aber nach ein paar Stunden auf Google und den Processing Foren hatte ich schon die Funktion fertig, welche jetzt das Rückgrad des Patches bildet:

```
float getPixel(PImage image, String colr)
{
    //Liest das Bild mit der id "image" ein und gibt den Helligkeitswert
    //für die Farbe colr zurück.

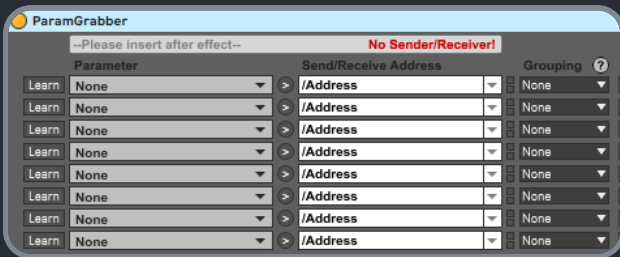
    float value = 0;
    int x = int(posX);
    int y = int(posY);
    image.loadPixels();
    loadPixels();
    int location = x + y*image.width;

    switch(colr) {
        case "r":
            value = red(image.pixels[location]);
            break;
        case "g":
            value = green(image.pixels[location]);
            break;
        case "b":
            value = blue(image.pixels[location]);
            break;
    }
    return map(value, 0, 255, 0, 1);
}
```

In dieser Woche hatte ich auch schon den ersten großen Bug: Ich habe die **OSCP5 Library** genutzt, um die OSC Implementierung zu übernehmen, und beim Senden der Stiftdaten kamen nicht alle Messages in den entsprechenden Max-Plugins in Ableton an:



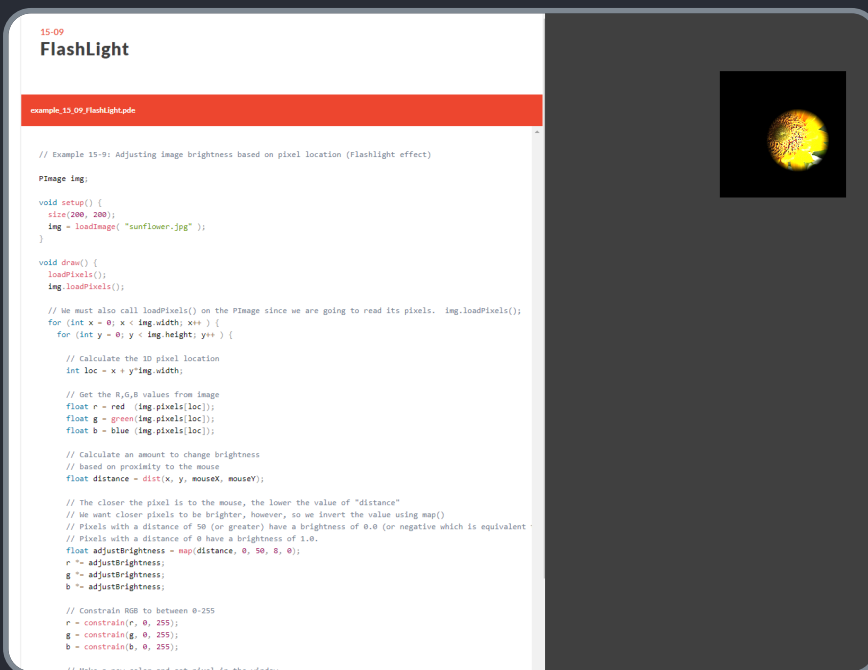
Dieses Plugin gehört zu den MAX for Live Plugins. Dort ging dann meine leider recht lange Suche nach der Ursache des Problems los. Nach sehr vielen Iterationen von der **sendOSC()** Funktion habe ich probiert, in Ableton schlicht einen anderen OSC-Empfänger zu nutzen und siehe da: das Problem wurde gelöst:



Der **Livegrabber** von **showsync.com**. Scheinbar scheint dieses Plugin besser mit der hohen Frequenz der aus Processing kommenden Daten klar zu kommen. Zum debuggen habe ich meistens ein einfaches Ableton Projekt aufgemacht und Parameter im Utility Effekt per OSC gesteuert.

Bildbearbeitung

Da wir noch ein visuelles Feedback für den ändernden Stiftedruck brauchten, wollte ich eine Art Taschenlampen-Effekt programmieren. Auch dort bin ich in ein Google-Processing-Forum-Rabbithole gestiegen und mit einer einfachen Suche nach „**Flashlight Processing**“ zum Ziel gekommen:



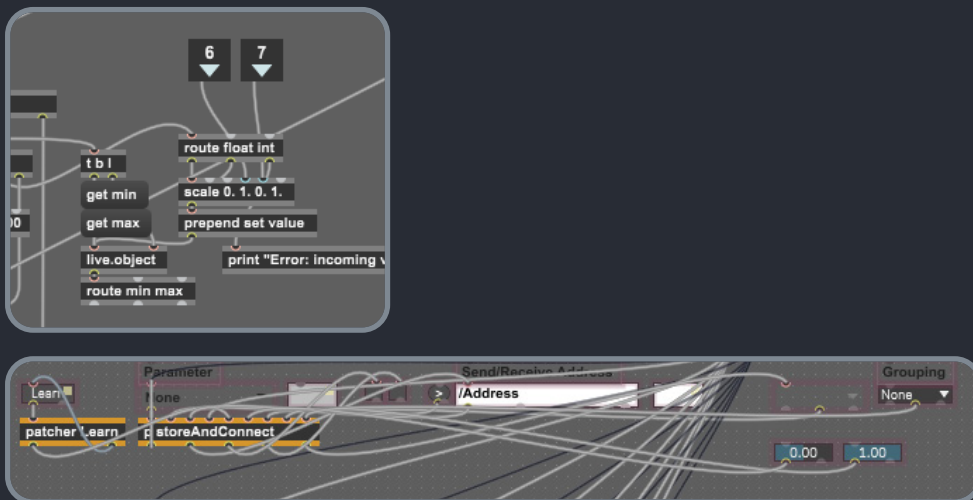
Einem Demoprojekt im Processing Forum, welcher genau einen solchen Effekt mit der **getPixel()** Methode realisiert. Dieser ist auch ein wenig angepasst in unserem Code zu finden.

Modifizierte MAX Devices

Beim weiteren Entwickeln der Technologien für Spannbettlaken hat das Problem der Skalierung der Werte mir die ganze Zeit schon aufgelauert: Wenn ich mit `tablet.pressure()` zB. ein Gain von $-\infty$ dB bis 0dB steuern will, erwartet der Regler in Ableton Werte von -1 bis 0 als floats. Wenn ich jedoch einen Stereo-Balance Regler komplett aussteuern möchte, wären es Werte von -1 bis 1.

Um also ein angepasstes Sounddesign zu realisieren müsste ich (zumindest nach dem Stand des Patches zu der Zeit) für jeden Regler eine eigens skalierte OSC-Message senden. Da der Weg über OSC, also UDP, ein serielles Netzwerkprotokoll ist, wäre dieser Weg aufgrund der Auslastung dieser Leitung sehr unpraktikabel. Logischer scheint es zu sein, die empfangenen OSC-Werte beim Zuweisen auf die Parameter in Ableton zu skalieren. Leider haben die Livegrabber Plugins diese Funktionalität nicht.

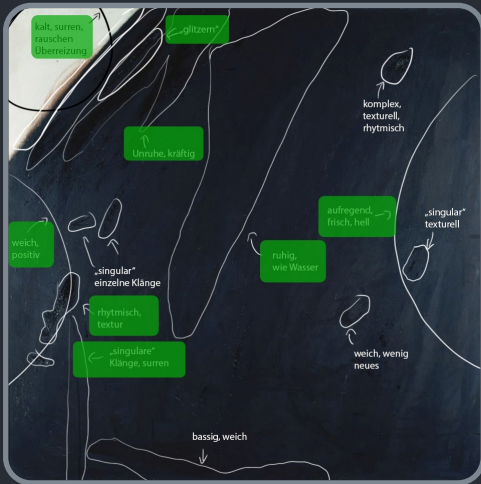
Unsere Lösung war es schlichtweg, die Plugins um diese Funktion zu erweitern:



Durch diese Änderung konnten wir also bequem die Werte entsprechend skalieren. Beim Sounddesign haben wir später auch noch den ParaGlider gepatcht – eine Version vom ParameterGrabber, welcher auf Zahlen von -20.000 bis 20.000 skalieren kann, da manche Regler Werte außerhalb von -1 und 1 benötigen.

Die Online-Sounddesign Sessions

Vor dem Sounddesign hatte jeder „auf Verdacht“ Sounds mit Recordern aufgenommen, um diese später im Projekt verwenden zu können. Parallel dazu habe ich eine grobe **Sounddesign Karte** angelegt, wo ich bestimmte Sounds oder Emotionen im Bild erwarten würde:



Anhand dieser Karte und unseren Sounds haben wir dann in Zusammenarbeit jede Woche 1,5 Mal die Woche unser Sounddesign gemacht.

Bugs & Performance Optimierungen

Während dem Sounddesign-Prozess sind wir selbstverständlich auf Probleme der Technik gestoßen: Wir haben an Ableton so viele OSC Messages geschickt, dass diese entweder sehr stark gestört waren oder teilweise einfach gar nicht mehr ankamen. Außerdem ist der Patch abgestürzt, wenn man außerhalb des Fensters mit gedrücktem Stift gekommen ist.

Das Problem der OSC Messages haben wir mit einer **Klasse** gelöst, welche die zu sendenden Werte in einen **Buffer** schreibt und diese nur an Ableton weiter gibt, wenn der Buffer nicht aus gleichen Elementen besteht:

```
//Map2 | FloatList size=5 [ 0.47843137, 0.43137255, 0.45882353, 0.43529412, 0.43137255 ]  
//Map17 | FloatList size=5 [ 0.5529412, 0.47843137, 0.45882353, 0.43529412, 0.43137255 ]  
/Y | FloatList size=5 [ 0.5345793, 0.5203238, 0.50583833, 0.49206233, 0.47828823 ]  
//Map1 | FloatList size=5 [ 0.43137255, 0.4509804, 0.47058824, 0.47843137, 0.4862745 ]  
//Map17 | FloatList size=5 [ 0.47843137, 0.45882353, 0.43529412, 0.43137255, 0.43137255 ]  
/Y | FloatList size=5 [ 0.5203238, 0.50583833, 0.49206233, 0.47828823, 0.46324414 ]  
//Map1 | FloatList size=5 [ 0.4509804, 0.47058824, 0.47843137, 0.4862745, 0.4862745 ]  
//Map17 | FloatList size=5 [ 0.45882353, 0.43529412, 0.43137255, 0.43137255, 0.4509804 ]  
/Y | FloatList size=5 [ 0.50583833, 0.49206233, 0.47828823, 0.46324414, 0.44998038 ]  
//Map1 | FloatList size=5 [ 0.47058824, 0.47843137, 0.4862745, 0.4862745, 0.41568628 ]  
//Map17 | FloatList size=5 [ 0.43529412, 0.43137255, 0.43137255, 0.4509804, 0.5294118 ]  
/Y | FloatList size=5 [ 0.49206233, 0.47828823, 0.46324414, 0.44998038, 0.44244584 ]  
//Map1 | FloatList size=5 [ 0.47843137, 0.4862745, 0.4862745, 0.41568628, 0.26666668 ]  
//Map17 | FloatList size=5 [ 0.43137255, 0.43137255, 0.4509804, 0.5294118, 0.5686275 ]  
/Y | FloatList size=5 [ 0.47828823, 0.46324414, 0.44998038, 0.44244584, 0.44004378 ]  
//Map1 | FloatList size=5 [ 0.4862745, 0.4862745, 0.41568628, 0.26666668, 0.15686275 ]  
//Map17 | FloatList size=5 [ 0.43137255, 0.4509804, 0.5294118, 0.5686275, 0.54599807 ]
```

Für das Problem des „ArrayOutOfBounds“ habe ich für das Lesen der Pixel die **image.get()** Funktion verwendet, welche per default 0,0,0 ausgibt und damit keine Fehlermeldung mehr schickt, wenn man außerhalb des Fensters ist.