29-10-2014 Datastrukturer

# Presets

Presets kan have 1...x (?) sensorkonfigurationer.

**Array**.

## Preset

**Vector** af keys som er referencer til sensorkonfigurationer. Typedef eller lille klasse.

# Sensorkonfigurationer

**Map**.

## Sensorkonfiguration

*Rettelse! Man kan GODT have flere end 16 sensorkonfigurationer på ét preset (fx kan et gyroskop have 2 konfigurationer). Ved forkerte setupts bruges default-værdier.*

**Klasse**

Nødvendigt med ’signalIndex’ fordi flere forskellige sensorkonfigurationer måske kan lægge sig til ét signal.

Fx en sensor styrer intensitet, en anden styrer volumen. Hvis den skal ændre volumen på det samme signal, er det vigtigt at de referer til det samme signal.

**Navn**

string name;

**Valg af sensor**

int sensorID;

**signal Index**

int signalIndex;

**MIDI-parameter fra en liste af MIDI-parametre.**

MIDIParameter midiParam; //evt int hexkode – implementeres som enum

**Mapping Scheme fra mapping scheme liste.**

MappingScheme mScheme; // vi ved ikke så meget om denne

**Reference til lydpakke-object.**

lydPakke \*lyd; // pointer til object der indeholder en sti til lydfilen

+ get og set metoder

# Sample bibliotek

**Map**. Et sæt af keys og items.

## Lydpakke

**(Klasse med min. én member, altså stien)**

Kun én lydpakke til én sensorkonfiguration. Men det er ikke et must at en sensorkonfiguration skal have en lydpakke, da en sensorkonfiguration kan påvirke lydpakken på en anden sensor. Én lydpakke er én fil – evt. som et **objekt** med en sti hver, for det bliver irriterende at gemme på stier i sensorkonfigurationer frem for navn på et objekt!

# Lagring af datastrukturer?

Evt. XML – til næste sprint.