### Mapping Scheme

#### Design

Mapping Scheme klassens har til funktion af løse to opgaver

1. Lagre brugerindstillinger for hvordan data fra en given sensor omdannes til et MidiSignal
2. At syntetisere et MidiSignal ud fra en given sensor måling.

##### Klassediagram

Herunder ses et klassediagram for MappingScheme og dets lagringsstructs[[1]](#footnote-1) samt for MidiSignal, som MappingScheme også benytter i sin map() –funktion. Bemærk at der for klassediagrammet herunder er udeladt set- og get-metoder.



##### Funktionsbeskrivelser

|  |  |
| --- | --- |
| **Funktion** | map(int data, MidiSignal & signal): void |
| **Parametre** | int data: sensordata i range 0-127 MidSignal & signal: en reference til et givent MidiSignal, der skal ændres |
| **Returværdi** | Ingen |
| **Beskrivelse** | Funktionen tilretter MidiSignalet, der refereres til fra parametren signal på baggrund af parametren ”data” samt de lagrede brugerindstillinger. |

|  |  |
| --- | --- |
| **Funktioner** | mapKey(int data, MidiSignal & signal): void mapVelocity(int data, MidiSignal & signal): void mapCCAbs(int data, MidiSignal & signal): void mapCCRel(int data, MidiSignal & signal): void |
| **Parametre** | int data: sensordata i range 0-127 MidSignal & signal: en reference til et givent MidiSignal, der skal ændres |
| **Returværdi** | Ingen |
| **Beskrivelse** | Funktionen tilretter MidiSignalet, der refereres til fra parametren signal på baggrund af parametren ”data” samt de lagrede brugerindstillinger. Disse funktioner kaldes fra funktionen map() afhængig af hvilken param\_, der er indstillet i MappingSchemet. |

|  |  |
| --- | --- |
| **Funktion** | quantizeDiatonic(int &): void |
| **Parametre** | int & dataIn |
| **Returværdi** | Ingen |
| **Beskrivelse** | Hjælpe funktion, der kvantiserer parametren dataIn, så den til slut genererede tone passer i den af brugeren valgte key\_.scale\_. |

|  |  |
| --- | --- |
| **Funktion** | noteStringToInt(string): int |
| **Parametre** | string note |
| **Returværdi** | int |
| **Beskrivelse** | Hjælpefunktion, der konverterer den af brugeren valgte grundtone key\_.root\_ til en tilsvarende int. |

#### Implementering

Herunder foreligger beskrivelser af implementeringsfremgangen for klassens funktioner.

Se bilag MappingScheme.h og MappingScheme.c for den endelige MappingScheme implementering.

##### map

map() har til ansvar at kalde den en underfunktion jf. den param\_, brugeren har indstillet.

##### mapKey

Programflowet i mapKey er som følger:

1. Tilret data: Systemet kan generere toner fra oktaverne -2 til 9 = 12 oktaver. I hver oktav er der 12 toner. Dette giver mulighed for at generer i alt 120 forskellige toner. Derfor benyttes kun sensorData i intervallet 3-122.
2. ”Vend” toner hvis bruger har valgt direction\_ ”falling”.
3. Forskyd data jf. valgt grundtone. Dette er en forberedelse til funktionen quantizeDiatonic, der sørger for at den valgte skala kvantiseres som ønsket.
4. Kald quantizeDiatonic.
5. Forskyd data tilbage.
6. Modificerer signal referencen. Der først tjekkes om tonen er ny inden den sættes. Er den ny, slukkes først den gamle. Dermed kræves to gennemløb med samme modtagne data før tonen ændres.

##### quantizeDiatonic

Dur- eller mol-skalaer hører begge under kategoriseringen diatonisk.

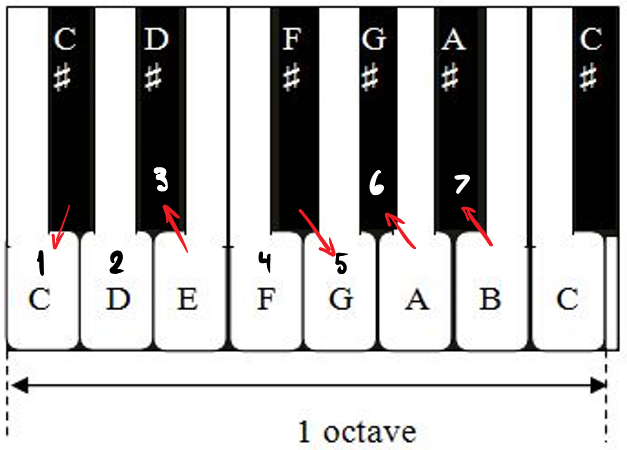
Når det indkomne data (som indeholder alle oktavens toner) skal kvantiseres, skiftes de toner, der ikke ligger i skalaen op- eller ned til en tone i skalaen. Kvantiseringen er designet således, at de mest brugte toner favoriseres: Grundtone (1), terts (3), kvint (5).

Dur kvantiseringen foregår som på illustrationen af en oktav herunder:



Påskrevet er durskalaens trin (1-7) for en c-dur skala samt hvilken vej det ønskes at tonerne uden for skala skal kvantiseres til (røde pile).

Mol kvantiseringen er illustreret herunder:



Funktionen er opbygget af først en registrering af hvilken tone i oktaven, data er, og dernæst en kvantisering vha. switch case sætninger.

##### mapVelocity

Programflowet i mapVelocity er som følger:

1. Tjek om data er valid (0-127).
2. Hvis signal før var NOTEOFF og data overstiger den af brugeren satte lowerThreshold, sættes den kommandoen til NOTEON.
3. Hvis signal før var NOTEON og data overstiger lowerThreshold, sættes kommandoen til AFTERTOUCH.
4. Hvis signal før var NOTEON og data er lavere end lowerThreshold, sættes kommandoen til NOTEOFF.

##### mapCCAbs

Programflowet i mapCCAbs er som følger:

1. Validér data (0-127).
2. Skalér data jf. af brugeren indstillede minVal\_ og maxVal\_
3. Sæt signal

##### mapCCRel

Herunder ses et flowchart for mapCCRel



#### Test

1. Som specificeret i afsnittet ”Krav til Mapping Scheme” [↑](#footnote-ref-1)