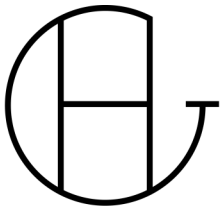


HoGent

Academiejaar 2012–2013

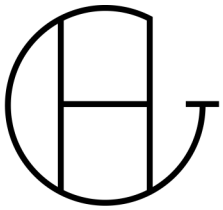
Geassocieerde faculteit Toegepaste Ingenieurswetenschappen
Valentin Vaerwyckweg 1 – 9000 Gent

Een C++-applicatie voor visualisatie en analyse van drumperformances

Masterproef voorgedragen tot het behalen van het diploma van
Master in de industriële wetenschappen: informatica

Vincent BAETEN

Promotoren: Wijnand SCHEPENS
Koen TANGHE (SampleSumo)



HoGent

Academiejaar 2012–2013

Geassocieerde faculteit Toegepaste Ingenieurswetenschappen
Valentin Vaerwyckweg 1 – 9000 Gent

Een C++-applicatie voor visualisatie en analyse van drumperformances

Masterproef voorgedragen tot het behalen van het diploma van
Master in de industriële wetenschappen: informatica

Vincent BAETEN

Promotoren: Wijnand SCHEPENS
Koen TANGHE (SampleSumo)

Voorwoord

Van jongs af aan ben ik reeds gefascineerd door muziek. Naast een fervent muziekliefhebber ben ik zelf ook muzikant en heb ik een opleiding slagwerk gevolgd aan de Stedelijke Academie voor Schone Kunsten te Beveren. Deze ervaring als drummer is erg nuttig gebleken en heeft geresulteerd in enkele interessante ideeën die hun plaats hebben gevonden in deze masterproef. Deze masterproef was voor mij dan ook een uitgelezen kans om mijn passie voor muziek te combineren met mijn studie.

De masterproef situeert zich voornamelijk in het domein van de informatica en deels in het domein van de muziek. Tijdens de literatuurstudie kwamen daarenboven enkele artikels uit de psychologie naar boven, waarin men probeert te achterhalen hoe mensen muziek ervaren en er de structuren in ontdekken. Dit uiteenlopende onderzoeksdomein heeft mijn kijk op de materie erg verbreed en zorgde ervoor dat het onderwerp steeds boeiend bleef.

Verschillende mensen hebben met raad en daad bijgedragen aan deze masterproef waarvoor zij verdienen bedankt te worden, dat doe ik dan ook van harte.

Vooreerst een oprecht woord van dank aan mijn externe promotor Koen Tanghe en zijn collega Bram de Jong om mij de kans te geven om deze masterproef te verwezenlijken. Ook een bijzonder dankwoord aan mijn interne promotor Wijnand Schepens voor zijn immer interessante inzichten in de materie.

Collega-drummer Jan Verhelst had ik graag bedankt om verschillende drumstudies in te spelen die hebben gediend als testgegevens.

Ook een woord van dank voor mijn vrienden, in het bijzonder: Maarten Wouters, Sarah Jacobs, Robbert Merlier en Thomas Stubbe. Zij konden me keer op keer motiveren en waren steeds bereid te luisteren naar nieuwe inzichten die ik had verworven bij het verwezenlijken van mijn masterproef. Zonder hen had het resultaat ongetwijfeld niet hetzelfde geweest.

Tot slot bedank ik graag mijn ouders om mij in gelijk welke omstandigheden steeds te blijven steunen.

Vincent Baeten
Gent, juni 2013

Inhoudsopgave

Abstract

Feedback is een belangrijk aspect bij het leren bespelen van een muziekinstrument. Deze masterproef onderzoekt de mogelijkheden om op een automatische manier een muzikale performance afkomstig van een elektronisch drumstel te analyseren en op basis van deze analyse de muzikant te voorzien van feedback. Hiervoor werd een applicatie ontwikkeld die een performance in MIDI-formaat kan analyseren en vervolgens zowel de performance zelf als de resultaten van de analyse op verschillende manieren kan visualiseren bij wijze van feedback. De analyse gebeurt door eerst de muzikale structuur te zoeken in de performance door middel van *beat tracking* en kwantisatie. Vervolgens worden verschillende metrieken (timing, tempo, strakheid en dynamiek) op de performance gedefiniëerd die elk een specifiek aspect van de performance analyseren. De analyse werd getest op verschillende drumstudies die door twee drummers met een verschillend niveau werden ingespeeld. De resultaten van de analyse van de metrieken timing, tempo en strakheid weerspiegelen het niveauverschil tussen de twee drummers, waaruit blijkt dat deze metrieken zinvol zijn. De resultaten van de metriek dynamiek zijn voor interpretatie vatbaar en vereisen verder onderzoek in de toekomst.

Hoofdstuk 1

Inleiding

Een muziekinstrument leren bespelen is een proces dat veel tijd en inspanning vraagt van de (aspirant-)muzikant. Een belangrijk onderdeel van dit leerproces is feedback. Om meester te worden over het instrument heeft de muzikant er nood aan om gewezen te worden op zijn fouten, maar ook op de zaken die hij goed doet (dit geldt immers voor alle leerprocessen in het leven). Deze feedback (letterlijk: terugkoppeling) zorgt er vervolgens voor dat de muzikant weet wat fout is zodat hij kan proberen om minder foute dingen te spelen, maar ook dat hij weet wat goed is zodat hij dat gedrag méér kan toepassen. Vanuit dit oogpunt blijkt het belang van een muzikale opleiding waarbij de muzikant les volgt bij een leraar. Hierbij krijgt de muzikant op regelmatige tijdstippen feedback gebaseerd op de ervaringen van de leraar (een meer ervaren muzikant), hetgeen het leerproces van de muzikant versnelt. De tijd tussen de sessies met de leraar kan de muzikant invullen door te oefenen op het bespelen van het instrument. Gedurende deze tijd is er geen mogelijkheid tot het verkrijgen van feedback.

Uit deze observatie rijst de vraag of het mogelijk is om op een automatische manier feedback te geven over de performance van de muzikant. Op deze manier zou de muzikant ook gedurende de tijd tussen de sessies met de leraar toegang hebben tot feedback, wat zijn leerproces positief beïnvloedt. In deze masterproef worden de mogelijkheden onderzocht om dit te realiseren door middel van een automatische analyse van een muzikale performance, in het bijzonder: een muzikale performance op een drumstel. Het doel is om een computerprogramma te ontwikkelen dat deze analyse kan uitvoeren om vervolgens de muzikant te voorzien van feedback. Hierbij wordt gebruik gemaakt van een elektronisch drumstel, aangezien dit de performance van de muzikant aanbiedt in een digitaal formaat (MIDI) waarin reeds heel wat nuttige informatie vervat zit.

Bij de ontwikkeling van dit programma worden enkele technologieën gebruikt die eerst en vooral nader worden toegelicht. Vervolgens blijkt de nood om de muzikale structuur te brengen in de gegevensstroom die de performance voorstelt. Dit wordt gerealiseerd door het bepalen van de posities van de beats (*beat tracking*) en door vervolgens te proberen achterhalen wat de muzikant net bedoelde te spelen door middel van kwantisatie. De verkregen structurele gegevens kunnen worden aangewend om verschillende aspecten van de performance te analyseren. Vier aspecten worden onderzocht en nader toegelicht: timing, tempo, strakheid en dynamiek. Tot slot wordt een programma ontwikkeld waarmee deze aspecten van een performance geanalyseerd kunnen worden. Het programma heeft een grafische interface met verschillende mogelijkheden voor visualisatie waarmee feedback kan worden weergegeven. Dit programma wordt ontwikkeld in C++, met behulp van de klassenbibliotheek JUCE.

Hoofdstuk 2

Gebruikte technologieën

2.1 MIDI

MIDI (*Musical Instrument Digital Interface* (Glatt, 2009)) is een technische standaard die elektronische muziekinstrumenten in staat stelt om te communiceren met elkaar, met computers of met andere gerelateerde apparaten. De standaard beschrijft een hardware-interface en een transmissieprotocol dat voor de communicatie gebruikt wordt.

2.1.1 Hardware-interface

Communicatie over MIDI gebeurt door het versturen van bytes op een seriële en asynchrone manier. Elke byte wordt voorafgegaan door een startbit en wordt beëindigd met een stopbit, zodat voor elke byte aan gegevens in totaal 10 bits verstuurd worden. De bytes worden verstuurd met een snelheid van 31.25kb/s, resulterend in een periode van 320µs per byte (inclusief start- en stopbit).

Gegevens worden verstuurd over STP[1] kabels, die aan elk uiteinde afgesloten worden met een vrouwelijke 180° 5-pins DIN connector (zie figuur 2.1). De mantel wordt aangesloten op pin 2, en beide kabels van het getwist paar respectievelijk op pin 4 en 5. Voor standaardtoepassingen worden pin 1 en 3 niet gebruikt. Sommige apparaten maken echter wel gebruik van pin 1 en 3 om bijvoorbeeld fantoomvoeding te voorzien, dus het heeft zin om ook tussen deze pinnen een verbinding te voorzien.

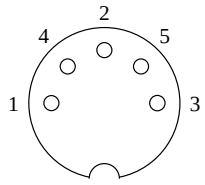
De aansluitingen op een MIDI-apparaat zijn logischerwijs de mannelijke variant van de 180° 5-pins DIN connector. Doorgaans voorziet een apparaat de aansluitingen “MIDI IN” en “MIDI OUT” om respectievelijk gegevens te ontvangen en te versturen. Vaak is er nog een derde aansluiting genaamd “MIDI THRU” aanwezig, waarlangs alle berichten die binnenkomen via “MIDI IN” terug naar buiten gestuurd worden. Dit laat toe om verschillende MIDI-apparaten aaneen te schakelen tot een ketting.

2.1.2 Transmissieprotocol

MIDI-apparaten communiceren met elkaar door het versturen van berichten (*messages*). Het MIDI-protocol beschrijft de verschillende soorten berichten en hoe deze opgebouwd zijn.

Een MIDI-bericht bestaat uit één of meerdere bytes (zoals beschreven in §2.1.1). De eerste byte van een bericht is steeds de *statusbyte*. De andere bytes noemt men *databytes*. De statusbyte kan eenvoudig herkend worden, aangezien het de enige byte is waarvan bit #7 gelijk is aan 1. Om een stroom bytes op te delen in verschillende MIDI-berichten volstaat het dus om de bytes te zoeken

[1] Shielded Twisted Pair



Figuur 2.1: 180° 5-pins DIN aansluiting

waarvan bit #7 op 1 staat, aangezien deze het begin van het bericht aangeven. De andere bits van de statusbyte geven aan om welk soort bericht het gaat.

Soorten MIDI-berichten

De verschillende soorten MIDI-berichten kunnen worden opgesplitst in 2 categoriën: *channel* (kanaalberichten) en *system* (systeemberichten). Om dit onderscheid duidelijk te maken is het nodig om eerst het begrip *MIDI channel* te verduidelijken.

Over een MIDI-verbinding kunnen berichten verstuurd worden die afkomstig zijn van verschillende instrumenten. Om te kunnen onderscheiden van welk instrument elk bericht afkomstig is krijgt het instrument een *channel* of *kanaal* toegewezen. Afhankelijk van het kanaal waarvan een bericht afkomstig is kan een MIDI-apparaat beslissen om al dan niet te reageren op dit bericht.

Kanaalberichten zijn berichten die specifiek zijn voor een bepaald kanaal. Om het kanaal waarvoor het bericht geldt aan te duiden wordt het kanaalnummer opgenomen in de laagste nibble van de statusbyte. Een nibble kan 16 verschillende waarden aannemen, wat resulteert in 16 mogelijke kanalen. Hieruit volgt dat over 1 MIDI-verbinding 16 verschillende kanalen aangestuurd kunnen worden.

Kanaalberichten kunnen vervolgens worden opgesplitst in twee subcategorieën:

- *Voice:* Berichten die de muzikale performance beschrijven. Ze hebben vaak het grootste aandeel in een stroom MIDI-berichten.
- *Mode:* Berichten die de manier beschrijven waarop de ontvanger dient te reageren bij het ontvangen van voice berichten.

Systeemberichten zijn berichten die geldig zijn voor alle apparaten die met elkaar verbonden zijn. Ze zijn niet afhankelijk van een bepaald kanaal en hebben bijgevolg geen kanaalnummer opgenomen in de statusbyte, waardoor het laagste nibble vrijkomt voor andere doeleinden.

Ze kunnen worden opgesplitst in drie subcategoriën:

- *Common:* Berichten die bestemd zijn voor alle ontvangers in het systeem.
- *Real-time:* Berichten die eveneens bestemd zijn voor alle ontvangers. Een real-time bericht bestaat uit slechts 1 byte (de statusbyte) en mag op elk ogenblik worden uitgezonden, zelfs tussen 2 bytes die tot eenzelfde bericht behoren. Ze worden vaak gebruikt om verschillende apparaten te synchroniseren met een MIDI-klok.
- *System Exclusive (SysEx):* Deze berichten zorgen voor een zekere flexibiliteit in de MIDI-standaard. Ze laten producenten toe om hun eigen specifieke MIDI-berichten te ontwerpen. Hiervoor dient de producent een uniek nummer aan te vragen dat vervolgens opgenomen wordt in de statusbyte van het SysEx bericht. SysEx berichten zijn bijgevolg gericht op specifieke apparaten en mogen door alle andere apparaten genegeerd worden.

Voor de volledigheid worden alle berichttypes weergegeven in tabel 2.1

Tabel 2.1: De verschillende soorten MIDI-berichten

Channel	
	Voice
	Note Off
	Note On
	Aftertouch
	Control Change
	Program Change
	Channel Pressure
	Pitch Wheel Change
	Mode
	Omni Mode On
	Omni Mode Off
	Poly Mode
	Mono Mode
System	
	Common
	Midi Time Code Quarter Frame
	Song Position Pointer
	Song Select
	Tune Request
	End Of Exclusive (EOX)
	Real-Time
	Timing Clock
	Start
	Continue
	Stop
	Active Sensing
	Reset
	System Exclusive (SysEx)

Channel-voice-berichten

De berichten uit de *channel-voice* subcategorie worden gebruikt om de muzikale performance te beschrijven. Aangezien deze categorie het meest relevant is voor deze masterproef worden alle berichttypes van naderbij bekeken.

Note on

	7	6	5	4	3	2	1	0
Status	type				channel #			
	1	0	0	1				
Data 1	note number							
	0							
Data 2	velocity							
	0							

Het *note-on*-bericht geeft het begin van een noot aan. Het wordt verstuurd wanneer de muzikant een toets (of in het geval van een drumstel: een deel van het drumstel) aanslaat. De eerste databyte bevat het nummer van de noot die werd aangeslagen, vermits het nootnummer 7 bits omvat zijn er 128 verschillende nootnummers mogelijk (gaande van 0 tot 127). De tweede databyte bevat de *velocity*, dit is een maat voor de sterkte waarmee de muzikant de toets (of het deel van het drumstel) heeft aangeslagen. Ook de velocity omvat 7 bits en neemt waarden aan tussen 0 en 127.

De ontvanger van een note-on-bericht dient (wanneer hij naar hetzelfde kanaal luistert) bij ontvangst het juiste geluid laten horen dat overeenkomt met het opgegeven nootnummer. De sterkte van dit geluid dient proportioneel te zijn met de opgegeven velocity. Een note-on-bericht met een velocity gelijk aan 0 dient te worden geïnterpreteerd als een note off bericht.

Note off

	7	6	5	4	3	2	1	0
Status	type				channel #			
	1	0	0	0				
Data 1	note number							
	0							
Data 2	velocity							
	0							

Het *note-off*-bericht geeft het einde van een noot aan. Het wordt verstuurd wanneer de muzikant een toets loslaat. In het geval van een elektronisch drumstel heeft dit bericht niet veel betekenis. Ze worden dan ook zeer kort na elk note-on-bericht verstuurd. Ook hier bevat de eerste databyte het nootnummer, dat in dit geval aangeeft welke noot werd losgelaten. Het tweede databyte bevat de velocity die aangeeft hoe snel de noot werd losgelaten. Indien een instrument geen note-off-velocity ondersteunt wordt standaard een velocity van 64 ingevuld.

Bij het ontvangen van een note-off-bericht dient de ontvanger te stoppen met het spelen van het geluid dat correspondeert met het opgegeven nootnummer. Afhankelijk van de opgegeven velocity kan dit eventueel met een fade-out gebeuren.

Wanneer de *hold-pedal*-controller aan staat (dit wordt aangegeven met een *control-change*-bericht) worden alle note-off-berichten opgehouden tot een bericht ontvangen wordt dat aangeeft dat de hold-pedal werd losgelaten.

Aftertouch

	7	6	5	4	3	2	1	0
Status	type				channel #			
	1	0	1	0				
Data 1	note number							
	0							
Data 2	pressure							
	0							

Het *aftertouch*-bericht wordt verstuurd telkens de muzikant de druk wijzigt waarmee hij een reeds aangeslagen toets ingedrukt houdt. Om aan te geven over welke toets het gaat wordt in de eerste databyte het nootnummer meegegeven. De tweede databyte bevat de nieuwe druk waarmee de toets wordt ingedrukt als een waarde tussen 0 en 127. In het geval van elektronische drumstellen kunnen aftertouch-berichten gebruikt worden om het *choken*[2] van een cimbaal aan te geven, dit is echter afhankelijk van de producent van het drumstel.

Control change

	7	6	5	4	3	2	1	0
Status	type				channel #			
	1	0	1	1				
Data 1	controller number							
	0							
Data 2	controller value							
	0							

Control-change-berichten worden verstuurd telkens de waarde van een controle-element op het MIDI-apparaat. Dit kan het resultaat zijn van het duwen op een knop, het draaien aan een draaiknop, etc... Vaak gaat het om knoppen die een andere functie vervullen dan het aan- of uitzetten van noten. Bij elektronische drumstellen worden control-change-berichten vaak gebruikt om aan te geven hoe ver bepaalde pedalen (bijvoorbeeld de hi-hatpedaal) ingedrukt zijn. Bepaalde producenten gebruiken echter ook control-change-berichten om positionele informatie over een aanslag door te geven, hiervoor wordt het control-change-bericht gelijktijdig verstuurd met het note-on-bericht.

De eerste databyte bevat het nummer van de controller waarvan de waarde gewijzigd is. Ook hier zijn 128 verschillende waarden mogelijk. Een groot deel van deze nummers is gereserveerd voor vaakgebruikte functies. Er zijn echter ook enkele *general-purpose*-controllernummers, die de gebruiker in staat stellen om zelf te beslissen hoe op deze berichten gereageerd wordt. De tweede databyte bevat de nieuwe waarde van de controller.

Program change

	7	6	5	4	3	2	1	0
Status	type				channel #			
	1	1	0	0				
Data 1	program number							
	0							

Een *program-change*-bericht wordt gebruikt om aan de ontvanger duidelijk te maken dat een ander programma geladen dient te worden (op het kanaal dat vermeld wordt in de statusbyte). Dit kan op verschillende manieren geïnterpreteerd worden. Het kan betekenen dat een andere

[2]Het met de hand vastnemen van het cimbaal kort na het aanslaan ervan, om zo het geluid te dempen.

geluidsbibliotheek geladen moet worden in het geval van een instrument. In het geval dat het MIDI-apparaat gebruikt wordt om een effectprogramma aan te sturen kan het betekenen dat een andere verzameling parameters dient ingeladen te worden.

Dit bericht heeft één databyte waarin het nummer van het programma dat geladen dient te worden vermeld staat.

Channel pressure

	7	6	5	4	3	2	1	0
Status	type				channel #			
	1	1	0	1				
Data 1	pressure							
	0							

Daar waar aftertouch-berichten wijzigingen in druk registreren voor de individuele toetsen, worden *channel-pressure*-berichten gebruikt om de gemiddelde druk van alle ingedrukte toetsen op het kanaal te registreren. Telkens de druk die op een toets wordt uitgeoefend wijzigt, wordt de gemiddelde druk berekend en doorgestuurd in een channel-pressure-bericht. Hiervoor heeft het bericht één databyte dat de nieuwe gemiddelde druk van alle ingedrukte toetsen vermeldt.

Pitch wheel change

	7	6	5	4	3	2	1	0
Status	type				channel #			
	1	1	1	0				
Data 1	value							
	0							
Data 2	value							
	0							

Een *pitch-wheel-change*-bericht wordt verstuurd wanneer het toonhoogtewiel van het instrument bewogen wordt. Hiermee kan de toonhoogte van de noten die momenteel klinken worden verhoogd of verlaagd.

De nieuwe waarde van het wiel wordt meegegeven in de 2 volgende databytes. Deze worden gecombineerd tot een 14-bit waarde, waarbij de 7 bits van het eerste databyte de minst significante bits van het 14-bit resultaat zijn en de 7 bits van het tweede databyte de meest significante.

Wanneer het wiel in de rustpositie staat dient de waarde 0x2000[3] te worden teruggegeven, en mag geen toonhoogteverandering optreden. Verder zou een een increment van 1 moeten resulteren in een toonhoogtewijziging van 1 cent[4].

2.1.3 Standard MIDI File

Om MIDI-gegevens op te slaan werd het *Standard MIDI File* (afgekort SMF) formaat ontwikkeld.

De MIDI-berichten die opgeslagen dienen te worden, worden gegroepeerd in verschillende sporen (*tracks*). Meestal wordt voor elk verschillend instrument, of voor elke verschillende stem een apart spoor gebruikt.

Om ervoor te zorgen dat de berichten in de juiste volgorde en op het juiste tijdstip kunnen worden afgespeeld (of doorgestuurd naar een ander MIDI-apparaat) krijgt elk bericht een tijdsaanduiding (*timestamp*) toegewezen.

[3] De prefix 0x geeft aan dat dit een hexadecimaal getal is.
[4] 100 cent is gelijk aan een halve toon.

Naast de MIDI-berichten en hun tijdsaanduidingen kan een MIDI-spoor nog andere gegevens bevatten (zogenaamde *metagegevens*). Deze metagegevens laten toe om onder andere het tempo, de signatuur, de toonaard, het gebruikte instrument of informatie over de componist op te geven.

2.2 JUCE

JUCE (*Jules' Utility Class Extensions* (Storer, 2013)) is een zeer uitgebreide klassenbibliotheek voor de programmeertaal *C++* (Stroustrup, 2008). Ze is ontworpen met het oog op platformonafhankelijkheid en voorziet onder meer in een grafische toolkit, daarnaast is er ook ondersteuning voor audio, MIDI, threading, audio plugins, netwerken, cryptografie, XML en nog veel meer. De klassenbibliotheek is opgebouwd uit verschillende modules die elk een deel van de functionaliteit voorzien. Een ontwikkelaar dient enkel die modules te laden die de functionaliteit bevatten die hij nodig heeft.

Wat opvalt is dat de ondersteuning voor audio, MIDI en audioplugins zeer uitgebreid is. Dit is te wijten aan het feit dat JUCE oorspronkelijk ontwikkeld is als deel van de DAW[5] *Tracktion*, maar later hiervan is afgesplitst als alleenstaande bibliotheek. Deze uitgebreide ondersteuning maakt JUCE uitermate geschikt voor het gebruik bij deze masterproef.

2.2.1 Grafische toolkit

Voor het opbouwen van GUI's[6] beschikt JUCE over een heel arsenaal klassen waarvan de functionaliteit sterk gelijkt op die van andere toolkits zoals bv. *Java Swing*. Om ervoor te zorgen dat een JUCE applicatie er op elk platform identiek uit ziet voorziet JUCE zelf in alle *widgets*[7] en is het niet mogelijk om de widgets die het platform standaard aanbiedt te gebruiken.

De hoofdbouwsteen van een GUI is de klasse *Component*, elk onderdeel van de GUI is hiervan afgeleid. Een Component kan gebruikt worden als container van andere Component-objecten. Met de lidfunctie `addAndMakeVisible (Component *child, int zOrder=-1)` kan een Component op het Component-object dat de functie oproept geplaatst worden. Aangezien er gewerkt wordt met dynamisch aangemaakte objecten is het belangrijk deze ook te verwijderen van zodra ze niet meer gebruikt worden. Hiervoor volstaat het om de lidfunctie `deleteAllChildren()` op te roepen in de destructor. Het wordt echter aangeraden om hiervoor gebruik te maken van klassen die instaan voor het automatisch beheer van de levensloop van dynamische aangemaakte objecten, zoals bijvoorbeeld de klasse *ScopedPointer* die een object automatisch verwijdert van zodra het *out of scope* gaat.

Een Component wordt op het scherm getekend door middel van de lidfunctie `paint(Graphics& g)`. Bij de implementatie van deze functie kan op de Component getekend worden door middel van de referentie naar het *Graphics* object. Telkens een gebeurtenis plaatsvindt die ervoor zorgt dat de Component opnieuw op het scherm getekend moet worden dient de lidfunctie `repaint()` opgeroepen te worden.

Om te reageren op invoer van de gebruiker is er een systeem van berichtenzenders (*message broadcasters*) en luisteraars (*listeners*). Wanneer een component wijzigt van toestand zendt deze een bericht naar alle geregistreerde luisteraars, deze kunnen dan zelf beslissen wat ze met dit bericht doen. Het registreren (en verwijderen) van een luisteraar bij een zender kan met de lidfuncties `addListener (Listener* listener)` en `removeListener (Listener* listener)`. Het maken van een luisteraar gebeurt door een klasse te laten overerven van de klasse *Zender::Listener* (hierbij is Zender de klasse van zender-objecten naar wiens berichten geluisterd wordt) en de nodige puur virtuele callback functie(s) te implementeren.

[5] Digital Audio Workstation
[6] Graphical User Interface
[7] Herbruikbare elementen waaruit de GUI is opgebouwd, bv. knoppen, tekstvelden,...

2.2.2 MIDI

JUCE voorziet uitgebreide ondersteuning voor het werken met MIDI-gegevens. Een greep uit het aanbod:

- Functionaliteit voor MIDI in- en uitvoer die het mogelijk maakt om MIDI-apparaten aan te spreken.
- Verschillende klassen om MIDI-*streams* te synchroniseren met audio-*streams*
- Een handige *wrapper*-klasse voor MIDI-berichten.
- Een klasse om een reeks MIDI-berichten te bundelen tot een sequentie, en die hiervoor verschillende nuttige operaties voorziet.
- De mogelijkheid om bestanden te lezen en te schrijven in het Standard MIDI File (zie §2.1.3) formaat.

In wat volgt worden de belangrijkste zaken besproken die gebruikt werden bij deze masterproef.

De *MidiMessage* klasse

De *MidiMessage* klasse is een *wrapper*-klasse voor MIDI-berichten. Achter de schermen slaat de klasse het oorspronkelijke MIDI-bericht op, terwijl ze voor de gebruiker een handige interface voorziet om het bericht te inspecteren en aan te passen.

Vooreerst beschikt de klasse over lidfuncties waarmee kan gecontroleerd worden of een bericht van een bepaald type is. Ze hebben het formaat `isX()`, waarbij `X` een berichttype is (bijvoorbeeld `isNoteOn()` of `isAftertouch()`, maar ook `isMetaEvent()` (zie §2.1.3)). De waarde die wordt teruggegeven is een *boolean* die aangeeft of het bericht van het gevraagde type is of niet.

Vervolgens zijn er *getters* en *setters* waarmee verschillende waarden uit het MIDI-bericht opgevraagd en aangepast kunnen worden. Voorzichtigheid is geboden bij het opvragen van waarden die specifiek zijn voor een bepaald berichttype. In het geval dat het bericht niet van het verwachtte type is zal de teruggegeven waarde geen steek houden. Het is dus zinvol om steeds eerst te controleren of het bericht wel van het juiste type is alvorens er waarden van op te vragen. Wanneer een setter van een berichtobject wordt aangeroepen voor een waarde die niet bestaat in het bepaalde bericht, gebeurt niks.

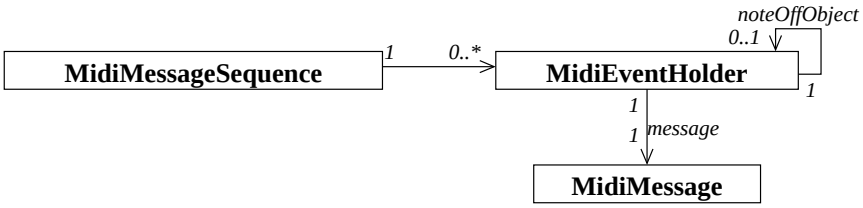
Verder heeft de klasse statische operaties waarmee bepaalde berichten aangemaakt kunnen worden. Zo een operatie geeft telkens een nieuw object terug van de MidiMessage klasse met het gevraagde berichttype en de gewenste eigenschappen.

De *MidiMessageSequence* klasse

De *MidiMessageSequence* klasse bundelt een verzameling MIDI-berichten (met tijdsaanduiding) en metadataberichten (zie §2.1.3) tot een sequentie.

Achter de schermen heeft een MidiMessageSequence een dynamische tabel waarin de berichten opgeslagen worden, nadat ze verpakt worden in *MidiEventHolder*-objecten. Een MidiEventHolder-object bevat het MidiMessage-object zelf en een pointer naar een ander MidiEventHolder-object. Deze pointer (genaamd `noteOffObject`) wijst, in het geval dat het oorspronkelijke bericht van het type note-on was, naar het MidiEventHolder-object waarin zich het corresponderende note-off-bericht bevindt. In alle andere gevallen is dit een *nullpointer*. Figuur 2.2 toont hiervan een verduidelijkend klassendiagram.

Naast de verwachte operaties om de sequentie te doorlopen en om berichten toe te voegen of te verwijderen, beschikt de MidiMessageSequence klasse over enkele interessante andere operaties, waaronder:



Figuur 2.2: Klassendiagram van een MidiMessageSequence

- `int getNextIndexAtTime(double timeStamp)`: Geeft de index terug van het eerste bericht dat op of na de opgegeven tijdsaanduiding valt. Indien de opgegeven tijdsaanduiding groter is dan de tijdsaanduiding van het laatste bericht wordt het aantal aanwezige berichten teruggegeven.
- `void updateMatchedPairs()`: Zorgt ervoor dat alle note-on-berichten in hun MidiEventHolder een verwijzing hebben naar het juiste note-off-bericht.
- `void addTimeToMessages(double deltaTime)`: Schuift alle berichten in de sequentie op over de opgegeven tijd.

Lezen en schrijven van bestanden in het Standard Midi File formaat

In- en uitvoer van en naar bestanden in het Standard Midi File formaat is mogelijk met de klasse *MidiFile*. Een MidiFile-object bevat alle informatie die ook in een SMF-bestand kan worden teruggevonden.

Zo is het mogelijk om een MIDI-spoor toe te voegen met de operatie `addTrack()`, die als argument een referentie naar de MidiMessageSequence die als spoor moet toegevoegd worden neemt. Het opvragen van de verschillende sporen kan met de operatie `getTrack()`, waarbij het nummer van het spoor als argument wordt meegegeven, en vervolgens de sequentie van het spoor wordt teruggegeven in de vorm van een (pointer naar een) MidiMessageSequence-object.

Verder zijn er operaties om de gebruikte *MIDI Time Code* (MTC) in te stellen. Voor de gebruiksvriendelijkheid is er echter ook een operatie `convertTimestampTicksToSeconds()` die alle tijdsaanduidingen omzet van MTC naar seconden.

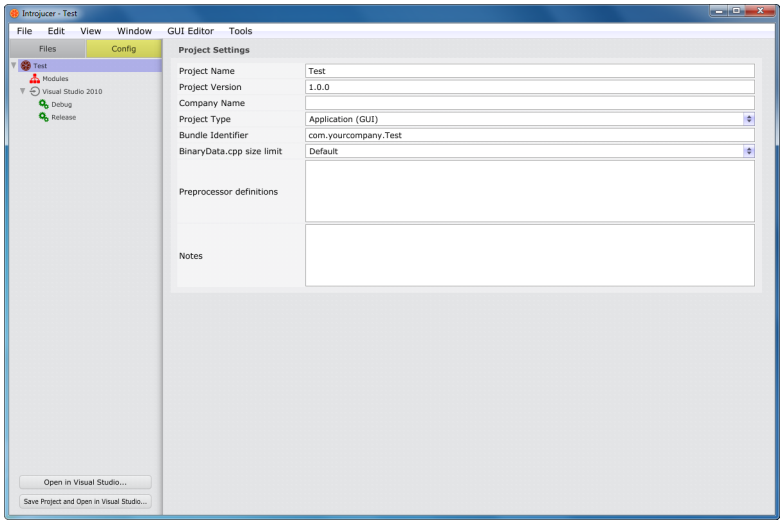
Het inlezen of wegschrijven van een MidiFile-object kan met de operaties `readFrom()` en `writeTo()`, die respectievelijk een *InputStream* en een *OutputStream*-object als argument nemen. Door voor deze streams *FileInputStream* en *FileOutputStream* objecten te gebruiken kan eenvoudig worden gelezen en geschreven van en naar een bestand in het SMF-formaat.

2.2.3 XML

JUCE voorziet de basisfunctionaliteit om XML-bestanden te manipuleren in de vorm van twee klassen: *XmlDocument* en *XmlElement*.

De *XmlDocument* klasse voorziet de parser die tekst kan parsen naar een op DOM[8] gebaseerde boomstructuur. Het is mogelijk om een object aan te maken van de klasse om er vervolgens de operatie `getDocumentElement()` van aan te roepen, die een pointer naar de wortel van de boomstructuur teruggeeft. Het is echter eenvoudiger om de statische lidfunctie `parse()` te gebruiken die dezelfde wortel teruggeeft. Deze statische lidfunctie kan aangeroepen worden met een File object als argument, of met een tekststring die de XML tekst bevat. Tijdens het parsen kan ook informatie uit een DTD worden gelezen. Dit heeft enkel nut wanneer er bijkomende entiteiten gebruikt worden in de XML-tekst. Validatie is niet voorzien.

[8] Document Object Model.



Figuur 2.3: Het hoofdscherm van The Introjucer

De boomstructuur wordt opgebouwd uit objecten van de klasse *XmlElement*. Zoals de naam doet vermoeden stelt een object van deze klasse een XML element voor. Bijgevolg zijn er operaties aanwezig om de verschillende kindelementen op te vragen of toe te voegen. Het opvragen van de kindelementen kan in de volgorde dat ze aanwezig zijn, of op naam. Verder zijn er verschillende operaties aanwezig om waarden van attributen in te stellen of op te vragen. Deze operaties kunnen de attribuutwaarden interpreteren als tekststring, maar ook als enkele elementaire datatypes, zodat extra controles overbodig worden.

Een boomstructuur bestaande uit XmlElement-objecten kan worden weggeschreven in XML-tekstformaat door op het wortelelement de operatie `writeToFile()` of `writeToStream()` aan te roepen.

2.2.4 The Introjucer

The Introjucer (zie figuur 2.3) is een handige tool die meegeleverd wordt bij JUCE. Hij staat in voor het beheer en de configuratie van JUCE-projecten.

The Introjucer maakt het mogelijk om een nieuw project aan te maken, broncodebestanden toe te voegen aan dit project, het project te configureren en vervolgens buildconfiguraties te genereren voor verschillende platformen. Hierdoor hoeft de gebruiker de moeizame configuratie niet manueel uit te voeren voor elk platform waarop hij de applicatie wilt uitbrengen.

De belangrijkste taak van the Introjucer is het beheren van de verschillende JUCE-modules die in een project gebruikt worden. De gebruiker kan aanvinken welke modules hij wenst te gebruiken en op welke manier deze gelinkt moeten worden aan het project. The Introjucer zal vervolgens de nodige broncodebestanden genereren en toevoegen aan het project, zodat de gebruiker slechts 1 headerbestand hoeft te *includen* om gebruik te kunnen maken van de JUCE klassen.

Verder bevat the Introjucer nog enkele handige tools:

- Een *code editor* waarmee de broncodebestanden van een project aangepast kunnen worden (let op: compileren/builden van het project is niet mogelijk met the Introjucer, maar dient te gebeuren met de specifieke compilers waarvoor de buildconfiguraties gegenereerd zijn).
- Een tool om GUI-componenten te ontwerpen. De GUI-componenten die in deze masterproef ontwikkeld werden zijn echter manueel geschreven.

Hoofdstuk 3

Analyse

De analyse van de performance gebeurt door de performance uit verschillende hoeken te bekijken. Vanuit elk oogpunt kunnen andere gegevens worden afgeleid die een beeld schetsen van hoe goed de muzikant voor dat specifieke aspect presteert. Het uit verschillende hoeken naar de performance kijken wordt gerealiseerd door het definiëren van verschillende metrieken op de performance.

3.1 Metrieken

De verschillende metrieken kunnen als volgt beschreven worden:
Het *tempo* is een maat voor de snelheid waarop een muziekstuk gespeeld wordt. Het wordt vaak uitgedrukt in beats per tijdseenheid (meestal beats per minuut of bpm). Deze grootheid krijgt meer betekenis in de muzikale zin wanneer de nootwaarde van de beat gekend is. Het is zinvol om het verloop van het tempo in de tijd te analyseren. Hieruit kan afgeleid worden wanneer de muzikant versnelt of vertraagt.
Timing wordt gedefiniëerd als de afstand (in de tijd) van de aanslagen die de muzikant speelt tot de gekwantiseerde waarde van deze aanslagen. De gekwantiseerde waarde van de aanslag is het tijdstip waarop deze noot door een machine gespeeld zou worden, waarbij deze noot perfect op een beat of op een punt van een gehele onderverdeling van een beat valt.
De *strakheid* van een performance wordt gemeten aan de hand van de spreiding van verschillende aanslagen die dezelfde gekwantiseerde waarde hebben. Een lage mate van strakheid gaat bijgevolg steeds samen met een slechte timing. Omgekeerd is dit echter niet zeker, daarom is het zinvol om beide metrieken te onderzoeken.
Dynamiek is het bereik van de sterkte waarmee gespeeld wordt. De dynamiek wordt gemeten door de spreiding van de aanslagsterkte. Deze spreiding kan apart berekend worden voor de verschillende delen van het drumstel.

Uit deze beschrijvingen volgt de nood aan twee belangrijke operaties: het bepalen van de beats en het kwantiseren van de performance. Deze operaties hangen zeer nauw samen. Zonder een idee te hebben van de posities van de beats is het namelijk onmogelijk om een performance correct te kwantiseren. De hoofdstukken 4 en 5 respectievelijk behandelen deze onderwerpen in meer detail.

Een meer uitgebreide beschrijving van de metrieken komt aan bod in hoofdstuk 6.

Hoofdstuk 4

Beat tracking

Beat tracking is het proces waarbij de posities van de beats in een performance bepaald worden. Dit is een taak die door de meeste mensen eenvoudig kan uitgevoerd worden (bijvoorbeeld: met de voet meetikken met de muziek). Voor een machine is dit echter minder triviaal. Het feit dat ook mensen zonder muzikale voorkennis deze taak kunnen uitvoeren doet echter vermoeden dat dit ook voor een machine geen onoverkomelijke taak is.

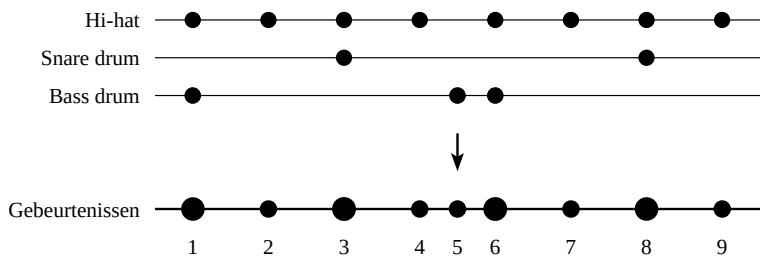
In dit hoofdstuk wordt het algoritme besproken dat voorgesteld wordt in Dixon (2001), met enkele aanpassingen die nuttig gebleken zijn tijdens de implementatie van het algoritme. Dit algoritme is in staat om de beatposities te bepalen uit een gegeven performance in een symbolisch formaat zoals MIDI. Het algoritme is oorspronkelijk ontworpen om een performance offline te verwerken, hier wordt echter ook een mogelijkheid voorgesteld om het in real-time te gebruiken. Het algoritme bestaat uit 2 fasen: tempo-inductie en beat tracking. In de tempo-inductiefase worden verschillende hypothesen opgesteld betreffende het tempo, op basis van de gegevens binnen een bepaalde tijdspanne vanaf het begin van de performance. Deze tempohypothesen worden vervolgens gebruikt in de beat trackingfase, waar de posities van de verschillende beats bepaald worden.

4.1 Ritmische gebeurtenissen

De meeste ritmische informatie zit vervat in de tijdstippen van de verschillende aanslagen (de *onset* tijdstippen). Wanneer meerdere aanslagen zeer dicht in elkaars nabijheid liggen worden zij waargenomen alsof ze op hetzelfde tijdstip plaatsvinden (dit is meestal ook hetgeen de muzikant bedoelt te spelen). Het heeft dus zin om deze aanslagen te groeperen in zogenaamde *ritmische gebeurtenissen*, zoals geïllustreerd in figuur 4.1.

Een ritmische gebeurtenis wordt gekenmerkt door een *onset tijdstip* en een waarde die de muzikale aanwezigheid van de gebeurtenis in de performance voorstelt. Het onset tijdstip van een gebeurtenis is gelijk aan het gemiddelde tijdtip van alle aanslagen die er in vervat zijn. De muzikale aanwezigheid is moeilijker te definiëren, deze kan beïnvloed worden door het aantal aanslagen in een gebeurtenis, de aanslagsterkte van de verschillende aanslagen, de verschillende delen van het drumstel die werden aangeslagen, enz... Voorlopig wordt de muzikale aanwezigheid van een gebeurtenis gedefiniëerd als de som van de aanslagsterkten van alle aanslagen die in de gebeurtenis vervat zijn. Toegepast op het voorbeeld uit figuur 4.1 krijgen de gebeurtenissen 1, 3, 6 en 8 een hogere muzikale aanwezigheid dan de gebeurtenissen 2, 4, 5, 7 en 9. Deze gebeurtenissen vallen dan ook samen met de beats, hetgeen later zinvol zal blijken.

De aanslagen worden verzameld in ritmische gebeurtenissen tijdens het doorlopen van de lijst van aanslagen in de volgorde dat ze gespeeld werden, zoals beschreven in algoritme 1.



Figuur 4.1: Groeperen van aanslagen in ritmische gebeurtenissen

Algoritme 1 Verzamelen van aanslagen in ritmische gebeurtenissen

Input: Een gerangschikte rij onset tijdstippen $O = \{O_1, O_2, \ldots, O_n\}$
Output: Een gerangschikte rij ritmische gebeurtenissen $E = \{E_1, E_2, \ldots\}$
E_i is zelf een verzameling van onset tijdstippen die tot de ritmische gebeurtenis behoort
$E_i.eersteOnset$ is het kleinste onset tijdstip dat tot de ritmische gebeurtenis behoort
$E.laatste$ is de laatste ritmische gebeurtenis in de verzameling E
Methode:
$eventWidth \leftarrow$ De maximale tijd tussen twee aanslagen in een ritmische gebeurtenis
$E \leftarrow \emptyset$
Maak een nieuwe ritmische gebeurtenis $E_0 = \{O_1\}$
$E \leftarrow E \cup \{E_0\}$
for i **from** 2 **to** n **do**
  $E_m \leftarrow E.laatste$
  if $O_i - E_m.eersteOnset \leq eventWidth$ **then**
    $E_m \leftarrow E_m \cup \{O_i\}$
  else
    Maak een nieuwe ritmische gebeurtenis $E_j = \{O_i\}$
    $E \leftarrow E \cup \{E_j\}$
  end if
end for
return E

4.2 Tempo-inductie

Tijdens de tempo-inductiefase worden verschillende hypothesen opgesteld met betrekking tot het tempo. Deze hypothesen hebben als doel om de *beattracker* op gang te zetten met het correcte initiële tempo. Daarom worden ze opgesteld op basis van de gegevens die zich binnen een bepaalde tijdspanne vanaf het begin van de performance bevinden. Naar deze tijdspanne wordt verder verwezen als het initiële venster.

De tempohypothesen komen tot stand door het analyseren van de interonsetintervallen van de verschillende ritmische gebeurtenissen. Een *interonsetinterval* wordt hier gedefiniëerd als de tijd tussen twee (niet noodzakelijk aangrenzende[1]) aanslagen (hier: ritmische gebeurtenissen). Aangezien het aantal interonsetintervallen kwadratisch is in het aantal ritmische gebeurtenissen beschouwen we hier enkel de intervallen tussen 50ms en 2s, dit zijn daarnaast de intervallen die het meest relevant zijn met betrekking tot het ritme (Handel geciteerd door Dixon, 2001). Deze interonsetintervallen worden gegroepeerd in clusters van gelijkaardige intervallen. Meerbepaald behoort een interonsetinterval tot een cluster wanneer het verschil tussen het interval zelf en het gemiddelde van alle intervallen die tot de cluster behoren, kleiner is dan een bepaalde waarde. Deze waarde is de *clusterbreedte*.

Het clusteren van interonsetintervallen gebeurt incrementeel en kan daarom gebeuren tijdens het opstellen van de ritmische gebeurtenissen (telkens een nieuwe gebeurtenis aangemaakt wordt is het zeker dat er geen aanslagen meer zullen worden toegevoegd aan de vorige gebeurtenis en kunnen de interonsetintervallen van de vorige gebeurtenis met alle voorafgaande gebeurtenissen berekend worden). Daarnaast zorgt het feit dat het clusteren incrementeel gebeurt er ook voor dat het gemiddelde van de intervallen in een cluster steeds verandert telkens wanneer een interval wordt toegevoegd. Hoe deze clusters gevormd worden kan gezien worden in algoritme 2 Dit kan resulteren in clusters die naar elkaar toebewegen, en zo in elkaars clusterbreedte terechtkomen. Na het initiële vormen van de clusters worden deze daarom vervolgens overlopen, waarbij clusters die binnen elkaars clusterbreedte liggen samengevoegd worden. Het samenvoegen van de clusters gebeurt volgens algoritme 3.

Algoritme 2 Het vormen van clusters van interonsetintervallen

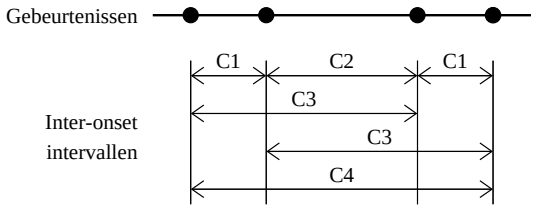
Input: Een gerangschikte rij ritmische gebeurtenissen $E = \{E_1, E_2, \ldots\}$
$E_i.onset = \frac{\sum_n O_n \in E_i}{|E_i|}$
Output: Een verzameling clusters $C = \{C_1, C_2, \ldots\}$
$IOI_{i,j}$ is het interonsetinterval van gebeurtenissen E_i en E_j
C_k is een verzameling van interonsetintervallen
$C_k.interval = \frac{\sum_{i,j} \{IOI_{i,j} \in C_k\}}{|C_k|}$
Methode:

```
for all E_i in E do
  for all E_j waarvoor E_j.onset < E_i.onset en 50ms ≤ E_i.onset − E_j.onset ≤ 2s do
    Zoek de cluster C_k zodat |IOI_{i,j} − C_k.interval| ≤ clusterBreedte
    if C_k bestaat then
      C_k ← C_k ∪ {IOI_{i,j}}
    else
      Maak een nieuwe cluster C_k = {IOI_{i,j}}
      C ← C ∪ {C_k}
    end if
  end for
end for
return C
```

[1] Oorspronkelijk werd een interonsetinterval gedefiniëerd als de tijd tussen twee aangrenzende aanslagen.

Algoritme 3 Het samenvoegen van clusters

Input: Een rij clusters $C = \{C_1, C_2, \ldots\}$, gerangschikt volgens $C_k.interval$
Output: De rij clusters C zonder clusters die binnen elkaars clusterbreedte vallen
Methode:
for all C_k **in** C **do**
for all C_m waarvoor $C_m.interval > C_k.interval$
en $C_m.interval - C_k.interval \leq clusterBreedte$ **do**
$C_k \leftarrow C_k \cup C_m$
$C \leftarrow C \setminus \{C_m\}$
end for
end for



Figuur 4.2: Clusteren van interonsetintervallen

Eens de clusters gevormd zijn en gelijkaardige clusters samengevoegd zijn wordt voor elke cluster een score berekend die weergeeft hoeveel ritmische informatie deze bevat en in welke mate hij bijgevolg relevant is voor het tempo van de performance. Een eerste bepalende factor die bijdraagt aan de score is het aantal intervallen dat een cluster bevat. Een cluster die veel intervallen bevat heeft een hoge waarschijnlijkheid om relevant te zijn voor het tempo. De andere factor kan het best geïllustreerd worden aan de hand van het volgende idee: stel dat een bepaalde cluster het tempo van de performance voorstelt, dan zijn er waarschijnlijk ook clusters aanwezig waarbij hun interval de helft, een vierde, een achtste, of een andere gehele verhouding tot het interval van deze cluster is. Deze factor wordt voorgesteld als $f(d)$, met d de gehele verhouding tussen de intervallen van twee clusters, en wordt als volgt gedefiniëerd:

$$f(d) = \begin{cases} 6 - d, & 1 \leq d \leq 4 \\ 1, & 5 \leq d \leq 8 \\ 0, & alle\ andere\ gevallen \end{cases}$$

Hoe deze factoren vervolgens gecombineerd worden tot een globale score voor een cluster is te zien in algoritme 4.

In figuur 4.2 wordt een voorbeeld gegeven van hoe dit in zijn werk gaat. Op de bovenste horizontale lijn worden de ritmische gebeurtenissen in functie van de tijd weergegeven. De pijlen daaronder stellen de interonsetintervallen voor. Bij elk interval wordt vermeld tot welke cluster het behoort. In dit voorbeeld worden 4 clusters gevormd, waarvoor de respectievelijke score als volgt berekend wordt:

$$\begin{aligned} C_1.score &= 2 \cdot 2 \cdot f(1) = 20 \\ C_2.score &= 2 \cdot f(2) + 2 \cdot 1 \cdot f(1) = 18 \\ C_3.score &= 2 \cdot f(3) + 2 \cdot 2 \cdot f(1) = 26 \\ C_4.score &= 2 \cdot f(4) + 1 \cdot f(2) + 2 \cdot 1 \cdot f(1) = 22 \end{aligned} \tag{4.1}$$

Algoritme 4 Toekennen van scores aan clusters

Input: Een rij clusters $C = \{C_1, C_2, \ldots\}$, gerangschikt volgens dalend $C_k.interval$
Output: De rij clusters C waarbij elke cluster een score $C.score$ heeft
Methode:

```
for all C_k in C do
  C_k.score ← |C_k|
  for all C_m met C_m.interval ≤ C_k.interval do
    // C_m begint met waarde C_k
    for i from 1 to 8 do
      if |C_k.interval − i · C_m.interval| ≤ clusterBreedte then
        C_k.score = C_k.score + f(i) · |C_m|
      end if
    end for
  end for
end for
```

4.3 Beat tracking

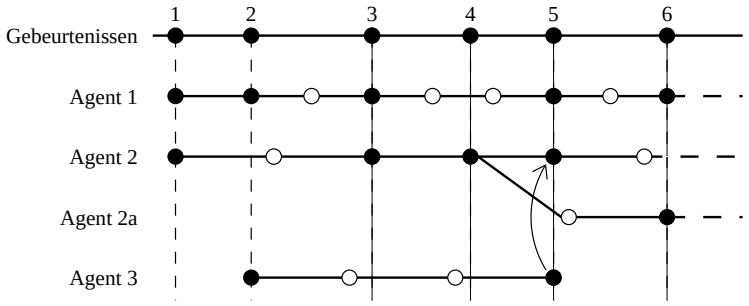
De clusters uit de tempo-inductiefase met de hoogste score geven een idee van het tempo van de performance. Ze vertellen echter niks over de posities van de beats of over het verloop van het tempo doorheen de performance. Wanneer deze clusters berekend worden op de gegevens uit het initiële venster, zijn de clusters met de hoogste score wel een goed vertrekpunt voor de beat trackingfase.

Tijdens de beat trackingfase worden verschillende hypothesen over het tempo en de positie van de beats (de *beat fase*) opgesteld. Deze hypothesen komen tot stand door het gebruik van zogenaamde *beat tracking agents*. Een beat tracking agent verwerkt ritmische gebeurtenissen en probeert op basis daarvan de posities van de beats te bepalen. Hiervoor beschikt de agent over een geschiedenis en een toestand. De *geschiedenis* bevat de posities van de beats die de agent tot dusver bepaald heeft. De *toestand* van de agent bestaat uit een huidige veronderstelling van het tempo en een voorspelling van de volgende beat. Tijdens het verwerken van ritmische gebeurtenissen kan een agent zijn veronderstelling van het tempo aanpassen naargelang hoe goed zijn voorspelling van de beat overeenkomt met de gebeurtenissen. Dit stelt de agents in staat om geleidelijke tempowijzigingen te kunnen blijven volgen.

Aangezien er meerdere agents gebruikt worden die elk een verschillende hypothese over het tempo en de fase van de beat hebben, heeft het zin om aan deze agents een score toe te kennen. Deze score moet aangeven in welke mate de voorspellingen van de agent overeenstemmen met de ritmische gebeurtenissen, en in welke mate de verwerkte gebeurtenissen gerelateerd zijn aan het tempo. De score hangt bijgevolg af van de afstand (in de tijd) van de voorspellingen van de agents tot de gebeurtenissen en van de muzikale aanwezigheid van de gebeurtenissen. De uiteindelijke beatposities zijn deze die zich in de geschiedenis bevinden van de agent die, na het verwerken van alle ritmische gebeurtenissen, de hoogste score heeft.

4.3.1 Het beat trackingalgoritme

Zoals reeds vermeld vertrekt de beat trackingfase van de resultaten van de tempo-inductiefase. Wanneer het tempo niet gekend is worden de hoogst scorende clusters uit de tempo-inductiefase gebruikt als tempohypotheses. Wanneer het tempo op voorhand wel gekend is kan dit gebruikt worden als tempohypothese, of kunnen de clusters uit de tempo-inductiefase die slechts weinig afwijken van het gegeven tempo als hypotheses gebruikt worden. Aangezien ook de positie van de eerste beat niet gekend is wordt ook hier gewerkt met een initiëel venster van gebeurtenissen om het algoritme te initialiseren. De eerste beat wordt verwacht in dit venster te liggen. Het is zinvol om hiervoor hetzelfde venster te gebruiken als in de tempo-inductiefase. Wanneer de eerste beat



Figuur 4.3: Levensloop van beat tracking agents

in dit venster valt zal het initiële tempo hoogstwaarschijnlijk ook afgeleid kunnen worden uit de gegevens in dit venster.

Om het algoritme te initialiseren wordt vervolgens voor elke ritmische gebeurtenis uit het initiële venster een groep beat tracking agents aangemaakt, meerbepaald: één agent voor elke tempohypothese. Deze agents nemen het tijdstip van de gebeurtenis op in hun geschiedenis, initialiseren hun score met de waarde van de muzikale aanwezigheid van de gebeurtenis en stellen hun tempo in op de tempohypothese waarvoor hij werd aangemaakt. Het is nu zeer waarschijnlijk dat er onder alle agents die zijn aangemaakt, zich een agent bevindt die begint met de juiste fase en het juiste tempo. Indien dit niet het geval is zijn er toch voldoende agents aanwezig met verschillende fases en tempi, zodat de kans groot is dat een van deze agents zich na enig verloop van tijd aanpast aan de correcte fase en het correcte tempo.

Figuur 4.3 illustreert hoe de agents worden aangemaakt. Op de bovenste horizontale lijn worden de ritmische gebeurtenissen in functie van de tijd getoond. De lijnen daaronder geven weer hoe de beat tracking agents de gebeurtenissen verwerken. Een gekleurd bolletje geeft weer dat het voorspelde beat tijdstip van een agent overeenkomt met een ritmische gebeurtenis, een cirkeltje stelt een voorspeld beat tijdstip voor dat niet overeenkomt met een ritmische gebeurtenis (of juister: een geïnterpoleerd beat tijdstip, maar daarover later meer). Stel dat de tempo-inductiefase 2 tempi (een sneller en een trager tempo) als resultaat heeft en dat het initiële venster van de beat tracking fase gebeurtenissen 1 en 2 omvat, dan worden voor beide tempi agents aangemaakt voor beide gebeurtenissen. Agent 1 uit de figuur is de agent met het snelle tempo die overeenkomt met de eerste gebeurtenis. Normaalgezien zou er ook een agent aangemaakt worden met het snel tempo voor gebeurtenis 2, maar aangezien de voorspelling van agent 1 overeenkomt met gebeurtenis 2 zou deze nieuwe agent dezelfde toestand hebben als agent 1. Dit wil zeggen dat beide agents in de toekomst dezelfde voorspellingen zouden maken, waardoor het aanmaken van deze agent overbodig wordt. Voor het tragere tempo worden respectievelijk agent 2 en agent 3 aangemaakt voor gebeurtenissen 1 en 2.

Na het aanmaken van de agents voor alle events in het initiële venster kan het eigelijke beat tracking beginnen. Hierbij worden alle ritmische gebeurenissen in behandeld in de volgorde dat ze gespeeld werden. Voor elke beat tracking agent wordt gekeken of de gebeurtenis die in behandeling is op een beat valt, volgens de huidige staat van de agent. Hiervoor beschikt de agent over twee vensters rond de huidige voorspelling van de volgende beat, die bepalen of de gebeurtenis als nieuwe beat aanvaard wordt of niet. Wanneer een gebeurtenis binnen het binnenste venster rond de voorspelling van een agent valt, wordt het tijdstip van deze gebeurtenis aanvaard als beat tijdstip. Wanneer een gebeurtenis binnen het buitenste venster valt, wordt het tijdstip van de gebeurtenis beschouwd als een mogelijk beat tijdstip. Het binnenste venster heeft een vaste grootte die hetzelfde is aan weerskanten van de voorspelling en wordt hier ingesteld op 40ms. Het buitenste venster heeft een variabele grootte die afhankelijk is van het tempo van de agent en is daarenboven asymmetrisch.

Hier wordt de grootte van het buitenste venster ingesteld op 20% van het inter-beat interval[2] vóór de voorspelling en 40% van het interbeatinterval ná de voorspelling. Deze asymmetrie weerspiegelt het feit dat vertragingen tijdens een performance vaak veel expressiever zijn dan versnellingen.

Bij het behandelen van een gebeurtenis door een agent wordt eerst gekeken of de gebeurtenis voorbij het buitenste venster rond de voorspelling van de agent valt. Indien dit het geval is wordt de voorspelling van de agent aangepast door er een zo klein mogelijk geheel veelvoud van zijn interbeatinterval bij op te tellen, zodanig dat het uiterste punt van het buitenste venster voorbij het tijdstip van de gebeurtenis valt. Het is belangrijk om hierbij te vermelden dat enkel de voorspelling van de agent wordt aangepast, niet de geschiedenis. Een invariant tijdens het behandelen van de ritmische gebeurtenissen is dat het laatste tijdstip in de geschiedenis van een agent steeds het tijdstip is van de laatste gebeurtenis die door de agent aanvaard werd als beat tijdstip. Vervolgens zijn er 3 gevallen mogelijk, deze worden ook geïllustreerd in figuur 4.3. Het eenvoudigste geval vindt plaats wanneer de gebeurtenis buiten beide vensters rond de voorspelling van de agent valt. In dit geval wordt de gebeurtenis genegeerd door de agent. In het voorbeeld wordt gebeurtenis 2 genegeerd door agent 2, gebeurtenis 3 door agent 3, gebeurtenis 4 door agents 1 en 3 en gebeurtenis 6 door agent 2. Het tweede geval treedt op wanneer de gebeurtenis binnen het binnenste venster rond de voorspelling valt. Dit wil zeggen dat het tijdstip van de gebeurtenis door de agent aanvaard wordt als beat tijdstip. Voorbeelden van deze situatie zijn gebeurtenissen 2, 3, 5 en 6 voor agent 1, gebeurtenissen 3 en 4 voor agent 2 en gebeurtenis 5 voor agent 3. De geschiedenis van de agent wordt aangevuld met beat tijdstippen die berekend worden door telkens het interbeatinterval op te tellen bij het laatste tijdstip uit de geschiedenis, tot vlak voor de laatste voorspelling. Een alternatief is om het interval vanaf het laatste beat tijdstip op te delen in gelijke intervallen om zo de gemiste beats te interpoleren. In de figuur worden deze voorspelde tijdstippen die niet overeenkomen met een gebeurtenis weergegeven als holle cirkels. Vervolgens wordt het tijdstip van de gebeurtenis toegevoegd aan de geschiedenis, wordt het tempo van de agent aangepast naargelang de afstand van de gebeurtenis tot de voorspelling en wordt tenslotte de voorspelling van de agent aangepast door de nieuwe waarde van het interbeatinterval bij het tijdstip van de laatste beat op te tellen. Wanneer een gebeurtenis overeenkomt met de voorspelling van een agent moet ook de score van de agent aangepast worden. Hiervoor wordt de muzikale aanwezigheid van de gebeurtenis, vermenigvuldigd met een factor die afhankelijk is van de de positie van de gebeurtenis in het venster, opgeteld bij de vorige score van de agent. Het laatste en meest ingewikkelde geval vindt plaats wanneer de gebeurtenis buiten het binnenste venster, maar binnen het buitenste venster rond de voorspelling van de agent valt. In dit geval is het niet zeker maar wel mogelijk dat de gebeurtenis samenvalt met een beat. Uitgaande van de veronderstelling dat de gebeurtenis samenvalt met een beat wordt ze door de agent aanvaard en worden dezelfde operaties uitgevoerd als in het tweede geval. Aangezien de mogelijkheid echter ook bestaat dat de gebeurtenis niet samenvalt met een beat wordt ook een nieuwe agent aangemaakt. Deze nieuwe agent is een exacte kopie van de agent waarbij de gebeurtenis aanvaard werd, met als enige verschil dat de gebeurtenis bij deze agent niet aanvaard wordt. Op deze manier worden beide mogelijkheden overwogen en krijgen beide agents de kans om een score te ontwikkelen. De juiste beslissing zal uiteindelijk blijken uit de agent die na verloop van tijd de hoogste score heeft. Dit laatste geval treedt in het voorbeeld op voor agent 2 bij gebeurtenis 5, hier wordt de gebeurtenis als beat tijdstip aanvaard door agent 2, maar aangezien deze in het buitenste venster valt wordt agent 2a aangemaakt waarbij de gebeurtenis niet als beat tijdstip aanvaard wordt.

De score en het interbeatinterval van een agent worden aangepast afhankelijk van hoe goed de voorspelling van de agent overeenstemt met de gebeurtenis die aanvaard wordt. Om het nieuwe interbeatinterval B_{nieuw} te berekenen wordt een parameter s ingevoerd die de gevoeligheid van de beat tracker voorstelt. De gevoeligheidsparameter s kan waarden aannemen uit het interval $[0, 1]$.Stel B_{oud} gelijk aan het huidige beatinterval, $A.voorspelling$ gelijk aan de voorspelling van de beat tracker en $E.onset$ gelijk aan het tijdstip van de gebeurtenis die aanvaard wordt, dan

[2]De inverse van het tempo: $\frac{60}{tempo}\frac{\text{s}}{\text{beat}}$

wordt het nieuwe interbeatinterval gegeven door vergelijking 4.2.

$$B_{nieuw} = B_{oud} + s \cdot (E.onset - A.voorspelling) \tag{4.2}$$

De parameter s geeft bijgevolg aan hoe snel de agents zich aanpassen aan een nieuw tempo. Wanneer s gelijk is aan 1 zullen de agents zich onmiddellijk aanpassen wanneer het tempo wijzigt. Een agent zal echter ook veronderstellen dat het tempo gewijzigd is wanneer er een ritmische gebeurtenis plaatsvindt die eigenlijk niet op een beat valt, maar die wel binnen een van de vensters rond de voorspelling van de agent valt. Als s gelijk is aan 0 zullen de agents zich helemaal niet aanpassen en zullen tempowijzigingen bijgevolg ook niet gevolgd worden door de beat tracker.

Wanneer een gebeurtenis aanvaard wordt door een agent, wordt bij zijn score een fractie van de muzikale aanwezigheid van de gebeurtenis geteld. Deze fractie is afhankelijk van de positie van de gebeurtenis in het buitenste venster van de agent. Stel dat $E.aanwezigheid$ de aanwezigheid van de gebeurtenis voorstelt en dat W_{buiten} de grootte is van het buitenste venster (in seconden), aan de kant waar de gebeurtenis ligt ten opzichte van de voorspelling, dan wordt de score S gegeven door vergelijking 4.3.

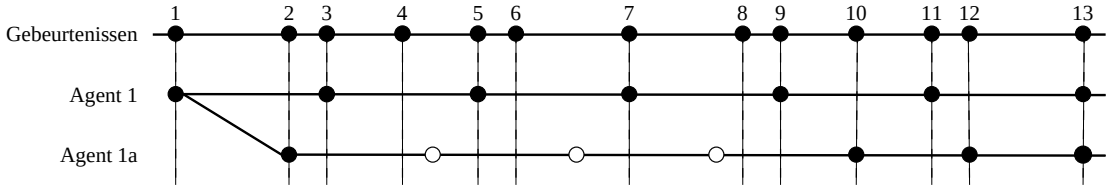
$$S = S + \left(1 - \frac{|E.onset - A.voorspelling|}{2 \cdot W_{buiten}}\right) \cdot E.aanwezigheid \tag{4.3}$$

Voor elke gebeurtenis die als beat tijdstip aanvaard wordt Op deze manier krijgen de agents een score die afhankelijk is van:

- het aantal gebeurtenissen dat overeenkomt met voorspelde beats,
- de correctheid van de voorspellingen en
- de muzikale aanwezigheid van de gebeurtenissen die op de beats vallen.

Aangezien er tijdens het behandelen van gebeurtenissen nieuwe agents aangemaakt worden telkens een gebeurtenis in het buitenste venster rond de voorspelling van een agent valt kan het aantal agents snel oplopen. Hierdoor is er nood aan een mechanisme om het aantal agents in de hand te houden. Een eerste eenvoudige optimalisatie kan gebeuren door agents die voor een bepaald aantal opeenvolgende beats geen overeenkomende gebeurtenissen gevonden hebben te markeren als *timed out* en vervolgens te verwijderen. De kans dat deze agents een juiste hypothese over de posities van de de beats hebben is immers klein. Een andere belangrijke optimalisatie is het verwijderen van *duplicaten*. Twee agents zijn elkaars duplicaat wanneer ze hetzelfde tempo hebben en dezelfde voorspelling van de volgende beat. Aangezien hun toestand gelijk is zullen deze agents in de toekomst dezelfde beats voorspellen. Bijgevolg mag een van de twee duplicaten verwijderd worden. Er is echter wel voorzichtigheid geboden bij het verwijderen van een duplicaat. Beide agents hebben een verschillende geschiedenis (anders waren ze immers vroeger reeds herkend als duplicaten) en dus ook een verschillende score. Aangezien de score berekend word op basis van de prestaties van de agent in het verleden, kan de agent met de laagste score zonder problemen verwijderd worden. Van deze situatie is agent 3 uit figuur 4.3 een voorbeeld: bij gebeurtenis 5 heeft deze hetzelfde tempo en dezelfde voorspelling van de volgende beat als agent 2, en aangezien agent 2 reeds meer correcte beats voorspeld heeft en dus een hogere score heeft kan agent 3 verwijderd worden.

Zelfs met deze twee optimalisaties kan het aantal agents onhandelbaar toenemen. Een drastische maatregel zou zijn om een beperking op te leggen aan het aantal agents. Het ligt dan voor de hand om enkel de agents met de hoogste score over te houden. Hoewel deze maatregel zeer drastisch lijkt, is er een positief gevolg aan gebonden. Vooraleer dit duidelijk gemaakt kan worden dient eerst het begrip *ontaarde agent* gedefinieerd te worden. Het kan gebeuren dat een agent een tempo heeft dat net iets hoger ligt dan het correcte tempo. De beats die door zo'n agent voorspeld worden zullen normaalgezien slechts zelden samenvallen met ritmische gebeurtenissen, waardoor de agent uiteindelijk gemarkeerd zal worden als timed out en bijgevolg zal worden



Figuur 4.4: Het ontstaan van een ontaarde agent

verwijderd. Wanneer de agent er echter telkens in slaagt om, vlak voor hij gemarkeerd wordt als timed out, een beat te voorspellen die samenvalt met een gebeurtenis, of wanneer er net veel gebeurtenissen plaatsvinden die nog net in het buitenste venster vóór de voorspelde beat vallen, zal hij er toch in slagen om nog in leven gehouden te worden, zij het dan wel met een lage score. Als zo'n agent lang genoeg kan blijven overleven kan het gebeuren dat zijn tempo convergeert naar een hoger tempo dat overeenkomt met een vaak voorkomend interonsetinterval in de performance. Hierdoor zal hij tegen dit hogere tempo vaak beats voorspellen die overeenkomen met ritmische gebeurtenissen. Aangezien het tempo van deze agent hoger ligt dan het werkelijke tempo van de performance zal hij meer beats voorspellen die overeenkomen met gebeurtenissen, dan een agent die wél aan het juiste tempo de performance volgt. Bijgevolg bestaat de kans dat deze agent een hogere score ontwikkelt dan een agent die de juiste beats voorspelt. Zo'n agent wordt een ontaarde agent genoemd. Figuur 4.4 illustreert een extreem geval van een ontaarde agent. Agent 1 is een agent die de correcte beats voorspelt. Gebeurtenis 2 valt echter in het buitenste venster van agent 1, agent 1 zal deze gebeurtenis niet aanvaarden als beat tijdstip, maar zal agent 1a afsplitsen waarbij de gebeurtenis wel als beat tijdstip aanvaard wordt. Agent 1a krijgt hierdoor een iets sneller tempo dan agent 1, zijn voorspellingen vallen niet meer samen met ritmische gebeurtenissen tot gebeurtenis 10. Met het snellere tempo zal agent 1a vervolgens gebeurtenissen 12 en 13 aanvaarden als beatposities. Deze gebeurtenissen volgen elkaar sneller op dan de werkelijke beats, waardoor agent 1a sneller een hoge score opbouwt indien zijn voorspelde beats in de toekomst blijven samenvallen met gebeurtenissen. Aangezien ontaarde agents tijdens hun convergentieperiode een lagere score hebben, zullen deze geëlimineerd worden door een limiet te zetten op het aantal agents. Zo krijgen ze de kans niet om zich te stabiliseren en onterecht een hoge score te ontwikkelen.

Het is belangrijk om deze optimalisaties te implementeren in de volgorde waarin ze hier beschreven staan. Het verwijderen van duplicaten is een dure operatie en gebeurt dus best op een verzameling agents die zo klein mogelijk is. Daarom worden eerst de agents verwijderd die als timed out gemarkeerd zijn. Deze operatie kan al geïmplementeerd worden tijdens het overlopen van alle agents wanneer een ritmische gebeurtenis behandeld wordt. Eens alle agents overlopen zijn, zijn alle agents die timed out zijn reeds verwijderd en kunnen de duplicaten verwijderd worden. Als laatste operatie wordt de verzameling pas gelimiteerd tot de agents met de hoogste score. Stel dat dit vroeger gebeurt, dan bestaat de kans dat er zich duplicaten en/of agents die als timed out gemarkeerd zijn bevinden in de overblijvende verzameling. Door deze operatie als laatste uit te voeren blijven enkel agents over die nog een zinvolle bijdrage kunnen leveren en die de grootste kans hebben om een hoge score te ontwikkelen.

Verder dient opgemerkt te worden dat het limiteren van het aantal agents niet mag gebeuren bij het behandelen van de gebeurtenissen in het initiële venster. Hier moeten alle agents nog de kans krijgen om zich te ontwikkelen. Wanneer hier het aantal agents reeds beperkt zou worden, bestaat de kans dat enkel de agents die voor de eerste gebeurtenissen werden aangemaakt overblijven. Dit terwijl het mogelijk is dat de eerste beat pas op een latere gebeurtenis uit het initiële venster valt. Het verwijderen van agents die timed out zijn en van duplicaten kan hier echter zonder problemen gebeuren.

Tenslotte is het volledige beat trackingalgoritme uitgeschreven in algoritme 5.

Algoritme 5 Het volledige beat trackingalgoritme

Input: Een gerangschikte rij ritmische gebeurtenissen $E = \{E_1, E_2, \ldots\}$
Een verzameling $C = \{C_0, C_1, \ldots C_n\}$ die de n clusters met de hoogste score bevat
Output: Een gerangschikte rij van beat tijdstippen
Methode:
$A \leftarrow \emptyset$ // De verzameling beat tracking agents
// Initialisatie
for all C_i **in** C **do**
for all E_j **in** E, met $E_j.onset - E_1.onset \leq initiëelVenster$ **do**
Maak een nieuwe agent A_k
$A_k.beatInterval \leftarrow C_i.interval$
$A_k.voorspelling \leftarrow E_j.onset + C_i.interval$
$A_k.score \leftarrow E_j.aanwezigheid$
$A_k.geschiedenis = \{E_j.onset\}$
$A \leftarrow A \cup \{A_k\}$
end for
end for
// Beat tracking
for all E_i **in** E **do**
$A_{nieuw} \leftarrow \emptyset$ // De verzameling van nieuw aangemaakte agents
for all A_j **in** A **do**
while $E_i.onset > A_j.voospelling + W_{buiten,rechts}$ **do**
$A_j.voospelling \leftarrow A_j.voorspelling + A_j.beatInterval$
end while
if $A_j.voospelling - W_{buiten,links} \leq E_i.onset \leq A_j.voospelling + W_{buiten,rechts}$ **then**
if $|A_j.voorspelling - E_i.onset| > W_{binnen}$ **then**
Maak een nieuwe agent $A_k \leftarrow A_j$
$A_{nieuw} \leftarrow A_{nieuw} \cup \{A_k\}$
end if
while $A_j.geschiedenis.laatste + A_j.beatInterval < A_j.voospelling$ **do**
$A_j.geschiedenis \leftarrow A_j.geschiedenis \cup \{A_j.geschiedenis.laatste + A_j.beatInterval\}$
end while
$A_j.geschiedenis \leftarrow A_j.geschidenis \cup \{E_i.onset\}$
$A_j.beatInterval \leftarrow A_j.beatInterval + s \cdot (E_i.onset - A_j.voorspelling)$
$A_j.voorspelling \leftarrow E_i.onset + A_j.beatInterval$
$A_j.score \leftarrow A_j.score + \left(1 - \frac{|E_i.onset - A.voorspelling|}{2 \cdot W_{buiten}}\right) \cdot E_i.aanwezigheid$
else
$gemisteBeats \leftarrow \frac{A_j.voorspelling + W_{buiten,rechts} - A_j.geschiedenis.laatste}{A_j.beatInterval}$
if $gemisteBeats \geq timeOut$ **then**
$A \leftarrow A \setminus \{A_j\}$
end if
end if
end for
$A \leftarrow A \cup A_{nieuw}$
Verwijder duplicaten onder de agents
if $E_i.onset - E_1.onset > initiëelVenster$ **then**
Beperk het aantal agents tot de m agents met de hoogste score
end if
end for

4.4 Real-time beat tracking

Het feit dat de ritmische gebeurtenissen sequentieel behandeld worden geeft aanleiding tot een *real-time* benadering van het beat tracking probleem. Het algoritme dat hier besproken wordt is een zeer intuïtieve benadering die toch vaak goede resultaten oplevert, er zijn echter wel situaties waarin het resultaat inconsitenties kan vertonen. Er wordt ook een oplossing voorgesteld die in de toekomst verder onderzocht kan worden.

Het verzamelen van aanslagen in ritmische gebeurtenissen gebeurt tijdens het behandelen van de aanslagen. Telkens een nieuwe ritmische gebeurtenis wordt aangemaakt is het zeker dat de vorige ritmische gebeurtenis niet meer zal veranderen in de toekomst en kan deze bijgevolg verwerkt worden. Op deze manier kunnen de clusters van interonsetintervallen reeds berekend worden: telkens een nieuwe gebeurtenis wordt aangemaakt kunnen de interonsetintervallen tussen de vorige gebeurtenis en alle voorgaande gebeurtenissen berekend worden en aan de juiste cluster worden toegevoegd. Dit dient uiteraard enkel te gebeuren voor de gebeurtenissen die zich in het initiële venster bevinden. Aangezien de informatie in verband met het tempo wordt afgeleid uit de gegevens die zich in het initiële venster bevinden, kunnen er gedurende deze periode nog geen beats voorspeld worden. Wanneer een benadering van het tempo op voorhand wordt opgegeven kan wel een tijdelijke agent opgestart worden met dit tempo en met de eerste ritmische gebeurtenis als fase. Deze agent kan reeds beats voorspellen voor de periode van het initiële venster.

Van zodra de eerste gebeurtenis voorbij het initiële venster behandeld wordt, kunnen de clusters van interonsetintervallen samengevoegd worden, en kan voor elke cluster een score berekend worden. Vervolgens worden er beat tracking agents aangemaakt voor de hoogst scorende clusters, voor elke ritmische gebeurtenis uit het initiële venster, net zoals bij de offline versie van het algoritme. De gebeurtenissen uit het initiële venster worden nu ook een tweede keer behandeld, waarbij voor elke gebeurtenis alle beat tracking agents overlopen worden om ze de kans te geven om de gebeurtenis als beat tijdstip te aanvaarden.

Vanaf hier verloopt het real-time algoritme volledig analoog aan het offline algoritme. Het enige waarmee rekening gehouden moet worden is dat de juiste voorspelling van de volgende beat op het juiste moment naar buiten gebracht wordt. Het ligt voor de hand om steeds de voorspelling van de agent met de hoogste score naar buiten te brengen. Voorzichtigheid is hierbij echter geboden. De mogelijkheid bestaat dat een andere agent net de hoogste score heeft gekregen en dat deze agent dezelfde beat opnieuw voorspelt. Wanneer dit het geval is zou de uitvoer 2 beats die onnatuurlijk dicht tegen elkaar liggen bevatten. De regel is om de voorspelling van de agent met de hoogste score enkel naar buiten te brengen indien deze voorbij het buitenste venster rondom de vorige voorspelling die naar buiten gebracht werd ligt. Verder kan tijdens het behandelen van de ritmische gebeurtenissen steeds de geschiedenis van de agent met de hoogste score opgevraagd worden voor het geval dat informatie over de eerdere beats gewenst is.

Deze benadering kan in veel gevallen een bevredigend resultaat opleveren. Er treden echter inconsistenties op wanneer meerdere agents met een geheel verschillende hypothese van de beat elk een hoge score hebben, waarbij hun score slechts weinig verschilt. Stel dat een agent momenteel de hoogste score heeft en een andere agent die de beat volgens een ander tempo volgt. Wanneer een gebeurtenis door de ene agent verworpen wordt, maar door de andere als beat tijdstip wordt aanvaard, dan kan dit ervoor zorgen dat de andere agent hierdoor de hoogste score krijgt. De voorspelling van de andere agent zal naar buiten gebracht worden, net zolang tot de ene agent opnieuw genoeg gebeurtenissen als beat kan aanvaarden om terug de hoogste score te bekomen. Dit resulteert in een opeenvolging van voorspellingen met totaal verschillende intervallen. Verder onderzoek is hier dus aangewezen.

De voorgestelde oplossing gebruikt enkel agents die de beat volgen vanaf het begin van de performance tot het einde. De score die ze krijgen toegekend is globaal, dit wil zeggen dat de score een beeld geeft van hoe goed de agent de volledige performance volgt. Het feit dat er gedurende het beat tracking proces geen nieuwe agents worden aangemaakt die de beat beginnen volgen beginnend vanaf de gebeurtenis waarvoor ze zijn aangemaakt, zorgt er voor dat de beat tracker enkel in

staat is om geleidelijke overgangen in het tempo te volgen. Een oplossing die in de toekomst verder onderzocht kan worden maakt gebruik van een complexer systeem om de agents te beheren. De score van de agents krijgt een meer lokaal karakter. Een mogelijke implementatie maakt gebruik van een *afkoelingsfactor* a (in het interval $[0, 1]$) om de nieuwe score S_{nieuw} uit de oude score S_{oud} te berekenen, met een eventuele verhoging van de score S_{inc} wanneer de voorspelling van de agent met een ritmische gebeurtenis overeenstemt, zoals te zien in vergelijking 4.4.

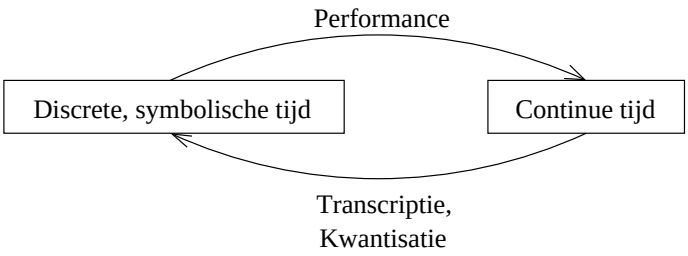
$$S_{nieuw} = a \cdot S_{oud} + S_{inc} \tag{4.4}$$

Om tijdens de performance nieuwe agents te kunnen aanmaken die in staat zijn om abrupte tempo-overgangen te volgen is er steeds nood aan recente tempo-informatie. Hiervoor kan tempo-inductie uitgevoerd worden op gegevens die zich in een glijdend venster bevinden dat steeds de ritmische gebeurtenis die in behandeling is bevat. Voor alle gegevens in dit venster kunnen telkens agents aangemaakt worden voor de tempohypothesen die uit de gegevens in dit venster volgen. Agents die ouder zijn dan een gegeven leeftijd worden verwijderd van zodra hun score onder een bepaalde drempelwaarde terecht komt. Aangezien de levensduur van de agents nu niet noodzakelijk de gehele performance bestrijkt, is er nood aan een globale geschiedenis voor de beat tracker. Telkens een nieuwe agent de hoogste score krijgt wordt zijn volledige geschiedenis aan de geschiedenis van de beat tracker toegevoegd, waarbij beats die met de nieuwe geschiedenis overlappen overschreven worden. Dit systeem zou in staat zijn om abrupte tempowijzigingen te kunnen volgen terwijl het daarnaast waarschijnlijk in real time een beter resultaat geeft dan het eerder beschreven systeem. Er is echter nood aan een meer ingewikkeld systeem om de agents te beheren en de implementatie moet voorzichtig gebeuren om te zorgen dat de performantie niet uit de hand loopt. Verder onderzoek in deze richting is aangewezen.

Hoofdstuk 5

Kwantisatie

Om kwantisatie te verduidelijken is het nodig om een onderscheid te maken tussen twee interpretaties van het begrip muzikale tijd (Clarke geciteerd door Desain & Honing, 1992). Enerzijds is er de discrete onderverdeling die gebruikt wordt bij muzieknotatie en andere symbolische representaties, waarbij de tijdschaal wordt onderverdeeld in maten, beats en gehele onderverdelingen daarvan. Anderzijds wordt de tijd op een continue manier onderverdeeld wanneer een muzikant een muziekstuk voordraagt. Deze continue onderverdeling is het gevolg van de expressiviteit van de muzikant die tot uiting komt wanneer hij een muziekstuk voordraagt, maar ook van de beperkte nauwkeurigheid van het menselijk motorisch systeem en van de vaardigheid van de muzikant. Hieruit wordt het duidelijk dat de overgang van de discrete onderverdeling van muzieknotatie naar de continue onderverdeling gerealiseerd wordt door het uitvoeren van het muziekstuk (de *performance*). De omgekeerde overgang is echter ook mogelijk en wordt gerealiseerd door *transcriptie*. Dit wordt afgebeeld door figuur 5.1. Het proces waarbij de continue tijdstippen van de performance gealigneerd worden met de discrete onderverdeling van een symbolische notatie wordt *kwantisatie* genoemd.



Figuur 5.1: De relaties tussen de twee interpretaties van muzikale tijd

Het kwantiseren van een performance vindt zijn toepassingen onder andere in automatische muziektranscriptie, het automatisch genereren van een begeleidend muziekspoor of het modelleren van hoe een luisteraar ritme ervaart. In het kader van deze masterproef wordt kwantisatie echter gebruikt om te achterhalen wat de muzikant net bedoelde te spelen en om vervolgens te analyseren hoeveel hetgeen de muzikant speelde hiervan afwijkt. Dit op zich is reeds een indicator van de prestatie van de muzikant, maar kan daarnaast ook gebruikt worden om kleine fluctuaties in het tempo van de muzikant te detecteren (zie §6.2.2).

5.1 Verschillende benaderingen

Het kwantisatieprobleem kan op verschillende manieren benaderd worden. De verschillende oplossingsmethoden voor het probleem gaan van heel eenvoudig (maar naïef) tot heel intelligent (en

soms nog steeds zeer elegant).

De meest eenvoudige methoden implementeren de kwantisatie door het afronden van een bepaalde waarde tot een geheel veelvoud van een vooraf bekend kwantum. Er kunnen 2 verschillende methoden onderscheiden worden: *interonsetkwantisatie* en *onset-* of *gridkwantisatie*. Bij interonsetkwantisatie worden de interonsetintervallen[1] afgerond tot een geheel veelvoud van het kwantum. Wanneer een kwantisatiefout optreedt (het nieuwe gekozen interonsetinterval is langer of korter dan wat de muzikant bedoelde te spelen), worden alle volgende aanslagen mee opgeschoven over het foutieve tijdstip. Dit kan voor synchronisatieproblemen zorgen in meerstemmige muziek.

Bij onsetkwantisatie wordt een rooster waarvan de resolutie gelijk is aan het kwantum op de performance geprojecteerd. Vervolgens worden de aanslagtijdstippen afgerond naar het dichtstbijzijnde punt van dat rooster. Het synchronisatieprobleem van interonsetkwantisatie is hierbij opgelost, de methode is echter niet in staat om kleine tempowijzigingen te volgen. Onset- of gridkwantisatie wordt verder besproken in §5.2.

De eenvoudige methoden zijn eerder naïef in die zin dat ze alle noten afzonderlijk bekijken alsof ze onafhankelijk van elkaar zijn. Door ritme wordt echter een hiërarchische structuur opgelegd die een verband creëert tussen de verschillende noten. Er zijn dan ook methoden ontwikkeld die deze informatie proberen te gebruiken bij het kwantiseren. De methoden die voorgesteld worden in Longuet-Higgins (1979) en Rosenthal (1992) komen tot een kwantisatie door deze hiërarchische structuur te parsen uit de performance. Het is niet verwonderlijk dat de resultaten aanzienlijk beter zijn dan die van de eenvoudige methoden. Naast de kwantisatie wordt bovendien de hele hiërarchische structuur van de performance verkregen.

In Desain & Honing (1992) wordt de onafhankelijkheid tussen de verschillende noten gemodelleerd door middel van een geconnecteerd netwerk. Dit netwerk bevat knopen voor de interonsetintervallen van de performance en voor de relaties tussen de verschillende interonsetintervallen. De knopen interageren met elkaar tot een evenwichtstoestand bereikt is, die de uiteindelijke kwantisatie voorstelt.

Modernere methoden, zoals voorgesteld in Cemgil et al. (1999) en Hamanaka et al. (2001) maken gebruik van probabilistische methoden om een correcte kwantisatie te vinden. Een nadeel aan deze methoden is dat de parameters voor het gebruikte model verkregen worden door training met representatieve data. Mits de juiste training produceren ze echter uitstekende resultaten. Verder wordt in Hamanaka et al. (2003) onderzocht of het of de parameters van het model ook gevonden kunnen worden zonder voorafgaande training.

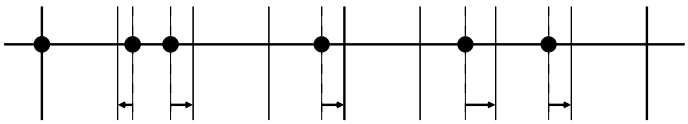
5.2 Gridkwantisatie

Gridkwantisatie is een eenvoudige methode om een performance te kwantiseren. De methode wordt, wellicht dankzij haar eenvoud, vaak teruggevonden in commerciële muzieksoftwarepakketten. De methode bekijkt elke aanslag afzonderlijk, zonder rekening te houden met enige vorm van structuur in de muziek of afhankelijkheden met andere aanslagen.

De methode vereist dat op voorhand een kwantum wordt opgegeven. Dit kwantum is de waarde van de kortste afstand (in de tijd) tussen twee aanslagen, die voorkomt in het muziekstuk. Wanneer de beats van het muziekstuk gekend zijn kan het kwantum opgegeven worden als een gehele onderverdeling van een interbeatinterval. Bijvoorbeeld: een beat is gelijk aan een kwartnoot, de kortste noot in het muziekstuk is een zestiendenoot, het tempo is 120bpm. Dan is het kwantum gelijk aan het interbeatinterval ($\frac{60}{120} \cdot \frac{\text{s}}{beat} = 0.5\text{s}$) gedeeld door 4 ($= \frac{1}{4} \cdot \frac{16}{1}$), resulterend in 0.125s.

Eens het kwantum gekend is kan een rooster waarvan de resolutie gelijk is aan dit kwantum over de performance geprojecteerd worden. Het gekwantiseerde tijdstip van elke aanslag is vervolgens gelijk aan het dichtstbijzijnde roosterpunt.

[1] Hier: de intervallen tussen 2 opeenvolgende aanslagen.



Figuur 5.2: Kwantisatie ten opzichte van een vast tempo

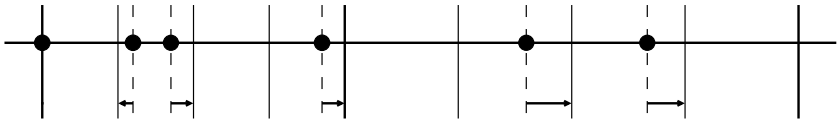
Wanneer het tempo van het muziekstuk vast is (de muzikant heeft bijvoorbeeld meegespeeld met een metronoom), dan is het kwantum over het hele muziekstuk gelijk en kan het gekwantiseerde tijdstip berekend worden zoals in vergelijking 5.1. Hierbij is $T_{gekwantiseerd}$ het gekwantiseerde tijdstip, $T_{aanslag}$ het tijdstip van de aanslag, I het interbeatinterval en D de gehele waarde waardoor het interbeatinterval gedeeld wordt (zodat $\frac{I}{D}$ gelijk is aan het kwantum).

$$T_{gekwantiseerd} = \left\lfloor \frac{T_{aanslag}}{I} \cdot D + 0.5 \right\rfloor \cdot \frac{I}{D} \tag{5.1}$$

Het resultaat wordt geïllustreerd in figuur 5.2. De bolletjes op de figuur zijn de aanslagen, de volle verticale lijnen geven het rooster weer en de pijltjes wijzen naar het gekwantiseerde tijdstip van elke aanslag.

Wanneer het tempo van het muziekstuk niet vast (of niet gekend) is kan alsnog gekwantiseerd worden door gebruik te maken van de beatposities die gevonden worden via het beat trackingalgoritme (zie hoofdstuk 4). Aangezien het interbeatinterval niet vast is, is het kwantum (en bijgevolg de resolutie van het rooster) voor elke beat verschillend. Dit wordt geïllustreerd in figuur 5.3. Voor elke aanslag wordt de beat gezocht die er vlak voor valt (B_{voor}) en de beat die er vlak na valt (B_{na}), zodat het interbeatinterval I op dat ogenblik gelijk is aan $B_{na} - B_{voor}$. Wanneer een aanslag vóór de eerste beat valt wordt het interbeatinterval van de eerste 2 beats genomen, wanneer een aanslag ná de laatste beat valt wordt het interbeatinterval van de laatste beat genomen. Het gekwantiseerde tijdstip kan vervolgens berekend worden zoals in vergelijking 5.2.

$$T_{gekwantiseerd} = B_{voor} + \left\lfloor \frac{T_{aanslag} - B_{voor}}{I} \cdot D + 0.5 \right\rfloor \cdot \frac{I}{D} \tag{5.2}$$



Figuur 5.3: Kwantisatie ten opzichte van berekende beats

Hoofdstuk 6

Metrieken

De analyse van de performance omvat het bepalen van de beatposities, het kwantiseren van de aanslagen en vervolgens het berekenen van waarden volgens verschillende metrieken die op de performance gedefiniëerd worden.

Volgende metrieken komen aan bod:

- Timing
- Tempo
- Strakheid
- Dynamiek

Om de resultaten van de metrieken te testen werd een reeks eenvoudige drumstudies en een reeks geavanceerde drumstudies geselecteerd (zie bijlage A). Vervolgens werden de studies ingespeeld door twee drummers:

- Drummer 1 is een beginnende drummer die nooit enige muzikale opleiding gevolgd heeft.
- Drummer 2 is een meer ervaren drummer die een opleiding slagwerk aan de muziekacademie gevolgd heeft.

De eenvoudige studies werden door beide drummers ingespeeld, met uitzondering van de tweede drumsolo (Henk Mennens - Solo 7), die niet door drummer 1 werd ingespeeld. De geavanceerde studies werden enkel door drummer 2 ingespeeld. De studies werden door elke drummer tweemaal ingespeeld, éénmaal met metronoom en éénmaal zonder.

De resultaten van de analyse zijn terug te vinden in bijlage B.

6.1 Timing

De metriek timing baseert zich op de afstand (in de tijd) tussen elke aanslag en het het gekwantiseerde tijdstip ervan. Om een waarde te verkrijgen die weergeeft hoe goed een muzikant zijn timing is lijkt het voor de hand liggend om het gemiddelde te nemen van al deze afstanden. Deze waarde is echter niet altijd representatief aangezien een aanslag zowel vóór als na zijn gekwantiseerd tijdstip kan liggen. Slechts wanneer de aanslagen allemaal vóór of allemaal na hun gekwantiseerd tijdstip vallen zou deze waarde een mogelijke indicatie zijn van de timing van de muzikant.

Een mogelijke oplossing is om eerst voor elke aanslag de absolute waarde van de afstand tot het gekwantiseerde tijdstip te berekenen, en vervolgens van al deze absolute waarden het gemiddelde

te berekenen. Een minpunt dat geldt voor beide benaderingen die de afstand als maat beschouwen is dat de eventuele vertraging (*latency*[1]) van het opnamesysteem het resultaat beïnvloedt.

Om ook dit te omzeilen kan een spreidingsmaat zoals de standaardafwijking van de afstanden berekend worden. Deze waarde geeft een heel representatief beeld van de timing van de muzikant.

Aangezien deze metriek zich baseert op de kwantisatie van de performance is het van belang dat de kwantisatie correct is. Bij een incorrecte kwantisatie zijn is het resultaat niet meer betrouwbaar. In het geval van de eenvoudige gridkwantisatie (zie §5.2) treedt een kwantisatiefout op wanneer een aanslag verder dan de helft van het kwantum van het bedoelde tijdstip ligt, en de aanslag bijgevolg naar een roosterpunt vóór of na het bedoelde punt gekwantiseerd wordt.

De resultaten voor de metriek timing worden weergegeven in tabellen B.1, B.2, B.3 en B.4. In elke tabel wordt voor elke studie het gemiddelde van de afstanden van de aanslagen tot hun gekwantiseerd tijdstip, het gemiddelde van de absolute waarde van deze afstanden en de standaardafwijking van deze afstanden weergegeven. Alle waarden zijn afstanden in de tijd, in seconden.

Voor elke studie ligt zowel het gemiddelde van de absolute afstanden als de standaardafwijking van de afstanden lager bij de meer ervaren drummer dan bij de beginnende drummer, ondanks enige kwantisatiefouten. Dit voldoet aan de verwachtingen, en toont aan dat de metriek zinvol is.

6.2 Tempo

De metriek tempo berekent een waarde die weergeeft hoe constant het tempo van de muzikant is. Er worden twee benaderingen toegelicht die een tempocurve van de performance proberen op te stellen. Vervolgens wordt de standaardafwijking berekend van de tempo's op elk punt van deze curve, die aantoont hoe constant het tempo is.

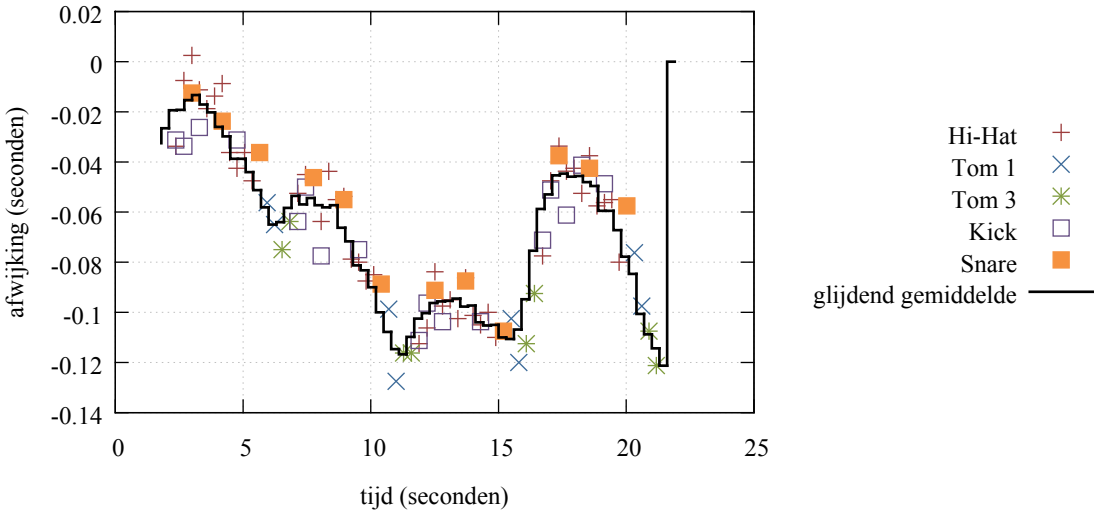
6.2.1 Tempo gebaseerd op de berekende beatposities

De tempocurve gebaseerd op de berekende beatposities kan eenvoudig worden opgesteld. Voor elke beatpositie (met uitzondering van de laatste) wordt de afstand tot de volgende beat berekend. Het tempo (in beats per minuut) wordt vervolgens gegeven door $\frac{60}{interbeatinterval}$. Dit tempo is geldig tussen de twee beats. Voorbeelden van tempocurves die op deze manier werden opgesteld zijn te zien in figuren 6.4 en 6.5 (rode lijn). De schijnbaar grote sprongen die de curve maakt kunnen worden toegeschreven aan het gedrag van de beattracker. Wanneer een aanslag in de nabijheid van een beat valt zal deze het tijdstip van de aanslag (of het gemiddelde tijdstip van de aanslagen indien er meerdere zijn) aanvaarden als beatpositie, daar waar in werkelijkheid er nog een klein timingverschil is tussen de aanslag en de werkelijke beatpositie. Zo zal bijvoorbeeld een aanslag die net na een beat valt, gevolgd door een aanslag die net voor een beat valt resulteren in een korter interbeatinterval en bijgevolg een hoger tempo.

Tabellen B.5 en B.7 tonen in de meest linkse kolommen het gemiddelde tempo en de standaardafwijking (beiden in beats per minuut) voor alle studies ingespeeld door respectievelijk drummer 1 en drummer 2 zonder gebruik te maken van een metronoom. Ook hier liggen alle standaardafwijkingen bij de meer gevorderde drummer lager als bij de beginnende drummer, hetgeen erop wijst dat zijn tempo meer constant is, zoals verwacht zou worden.

Tabellen B.6 en B.8 tonen dezelfde waarden voor alle studies ingespeeld door respectievelijk drummer 1 en drummer 2, waarbij wel gebruik gemaakt werd van een metronoom. Ook hier blijkt dat het tempo van drummer 2 constanter is dan dat van drummer 1, met uitzondering van 2 studies waar drummer 1 beter scoort.

[1] De tijd tussen het aanslaan van het instrument en het ogenblik dat deze aanslag door de computer geregistreerd wordt.



Figuur 6.1: Timing - Pop Drumming p. 25, oef. 7 door drummer 1 (met metronoom)

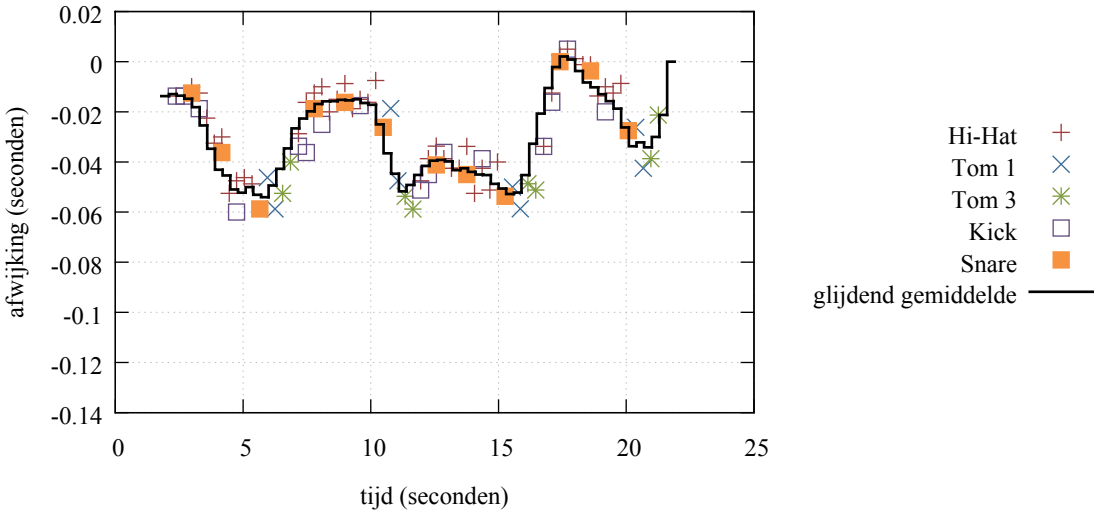
6.2.2 Tempo gebaseerd op de timing van de muzikant

Een andere benadering om een tempocurve te berekenen maakt gebruik van de timingverschillen zoals besproken in §6.1. In Iwami & Miura (2007) wordt gesuggereerd dat er correlaties bestaan tussen de afwijkingen van de aanslagen op verschillende delen van het drumstel (algemener: de verschillende delen van het instrument) die ongeveer op hetzelfde tijdstip vallen. Deze correlaties zouden het gevolg zijn van kleine fluctuaties van het tempo van de muzikant. De auteurs houden het bij deze suggestie, hier zal echter onderzocht worden of deze tempocurve uit de gegevens kan worden afgeleid.

Om duidelijk te maken wat net bedoeld wordt met deze correlaties is het nuttig om de timing weer te geven in een grafiek. Op de horizontale as staat de tijd (in seconden), op de verticale as staat het timingverschil (de afstand tot het gekwantiseerde tijdstip in seconden). Positieve en negatieve waarden betekenen respectievelijk dat een aanslag ná of vóór het gekwantiseerd tijdstip valt. Wanneer nu voor elke aanslag het timingverschil wordt uitgezet met het gekwantiseerde tijdstip als horizontale coördinaat kan worden opgemerkt dat aanslagen op verschillende delen van het drumstel die kort bij elkaar liggen, timingwaarden hebben die ook dicht bij elkaar liggen. Dit wordt aangetoond in figuren 6.1 en 6.2.

De gemiddelde timingwaarde kan op elk ogenblik berekend worden door gebruik te maken van een venster dat glijdt over de performance en vervolgens de gemiddelde timing van alle aanslagen in dit venster te berekenen. Deze werkwijze is analoog aan de werking van een *sweepline*-algoritme (de Berg et al., 2008). De structuur die de *status* van het glijdend venster voorstelt is een lijst met pointers naar de aanslagen die zich in het venster bevinden. Er zijn twee mogelijke gebeurtenissen die de status kunnen beïnvloeden: het toevoegen van een nieuwe aanslag aan het eind van het venster en het verwijderen van een aanslag uit het begin van het venster. Aangezien zowel de lijst van aanslagen als de statusstructuur de aanslagen bevat gesorteerd volgens hun tijdstip is er geen nood aan een aparte wachtrijstructuur (*event queue*). Het algoritme overloopt alle aanslagen. Bij elke nieuwe iteratie wordt het einde van het venster gelijkgesteld aan het gekwantiseerde tijdstip van de laatste aanslag die zich in het venster bevindt. Vervolgens wordt gecontroleerd of het nodig is om eerst aanslagen te verwijderen uit het begin van het venster. Dit komt voor in drie gevallen:

- Wanneer de afstand van het einde van het venster tot het gekwantiseerde tijdstip van de eerstvolgende aanslag voorbij het venster groter is dan de grootte van het venster (en wanneer het venster daarenboven niet leeg is). Zie figuur 6.3a.



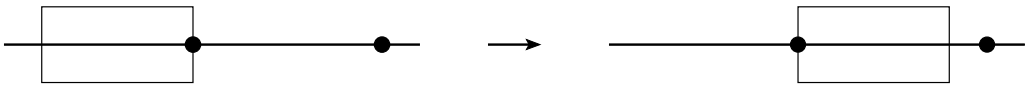
Figuur 6.2: Timing - Pop Drumming p. 25, oef. 7 door drummer 2 (met metronoom)

- Wanneer de afstand van het begin van het venster tot het gekwantiseerde tijdstip van de eerste aanslag in het venster kleiner is dan de afstand van het einde van het venster tot het gekwantiseerde tijdstip van de eerstvolgende aanslag voorbij het venster. Zie figuur 6.3b.
- Wanneer het venster niet leeg is en er geen aanslagen meer voorbij het einde van het venster liggen.

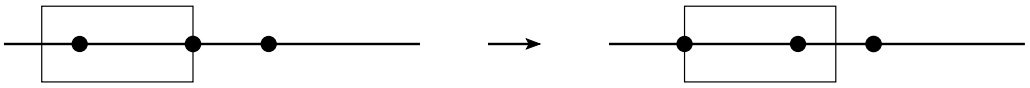
Bij het verwijderen van een aanslag uit het begin van het venster wordt het begin van het venster gelijkgesteld aan het gekwantiseerde tijdstip van de aanslag die net verwijderd werd. Vervolgens wordt een nieuw punt toegevoegd aan de curve met als horizontale coördinaat het midden van het venster en als waarde het gemiddelde van de afstanden van elke aanslag in het venster tot hun respectievelijke gekwantiseerde tijdstip. Indien geen aanslagen verwijderd dienen te worden aan het begin van het venster kan de volgende aanslag worden toegevoegd aan het einde van het venster. In dit geval wordt het einde van het venster gelijkgesteld aan het gekwantiseerde tijdstip van de aanslag die net toegevoegd werd (zie figuur 6.3c). Vervolgens wordt een nieuw punt toegevoegd aan de curve op dezelfde manier als bij het verwijderen van een aanslag uit het venster. Het algoritme stopt wanneer er geen aanslagen meer na het venster vallen en wanneer het venster daarenboven geen aanslagen meer bevat. De resulterende curve wordt weergegeven in figuren 6.1 en 6.2, en werd berekend met een venster dat een afstand van twee beats overspant.

Een verandering in de waarde van de resulterende curve (de timingcurve) wijst erop dat de afstanden van de aanslagen tot de punten van het kwantisatierooster veranderen. Dit wijst op een wijziging van het tempo. Een stijging van de timingcurve is het resultaat van aanslagen die iets later gespeeld werden ten opzichte van het kwantisatierooster dan de vorige aanslagen en wijst bijgevolg op een tempo dat trager is dan dat dat gebruikt werd om het kwantisatierooster op te stellen, omgekeerd wijst een daling van de timingcurve op een sneller tempo. Het tempo is bijgevolg afhankelijk van de verandering van de timingcurve.

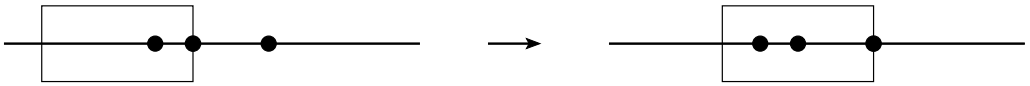
Om op basis van de timingcurve een tempocurve op te stellen is het nodig om eerst de afgeleide van de timingcurve te berekenen, deze geeft namelijk de verandering aan. Het berekenen van de afgeleide gebeurt door voor elke twee opeenvolgende punten van de curve het verschil tussen de timingwaarden dT te delen door het verschil tussen de tijdstippen dt. Het interval dt geeft aan hoe lang dit interval verwacht werd te duren volgens het tempo van de kwantisatiegrid. Het interval dT geeft aan hoeveel langer of korter (wanneer de waarde respectievelijk positief of negatief is) dit interval in werkelijkheid gespeeld werd. Bijgevolg geeft de waarde van de afgeleide $\frac{dT}{dt}$ de fractie



(a) Verwijderen: afstand einde venster tot volgende aanslag is groter dan de grootte van het venster

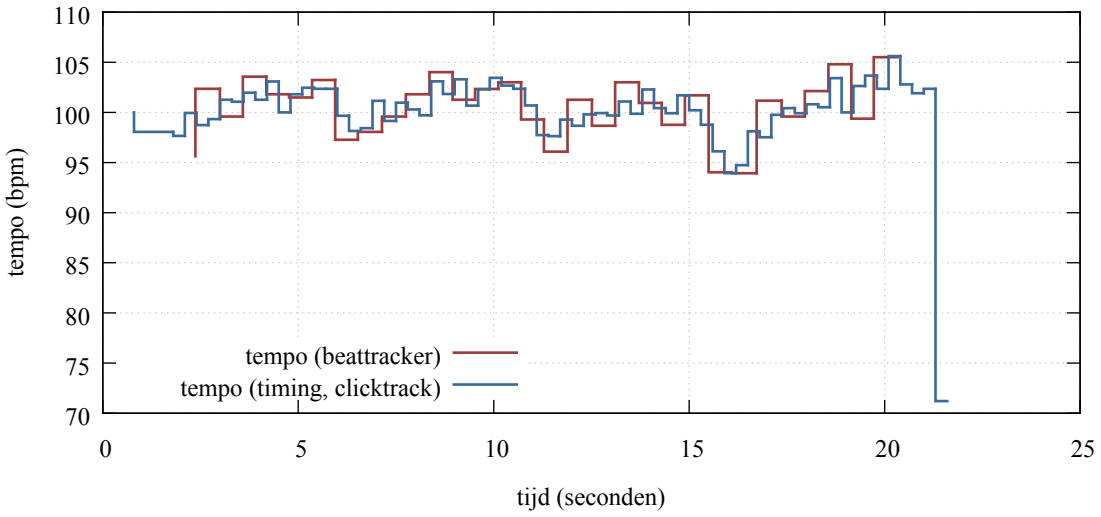


(b) Verwijderen: afstand begin venster tot eerste aanslag is kleiner dan afstand einde venster tot volgende aanslag



(c) Toevoegen: afstand einde venster tot volgende aanslag is kleiner dan afstand begin venster tot eerste aanslag

Figuur 6.3: Mogelijke situaties bij het glijdend venster



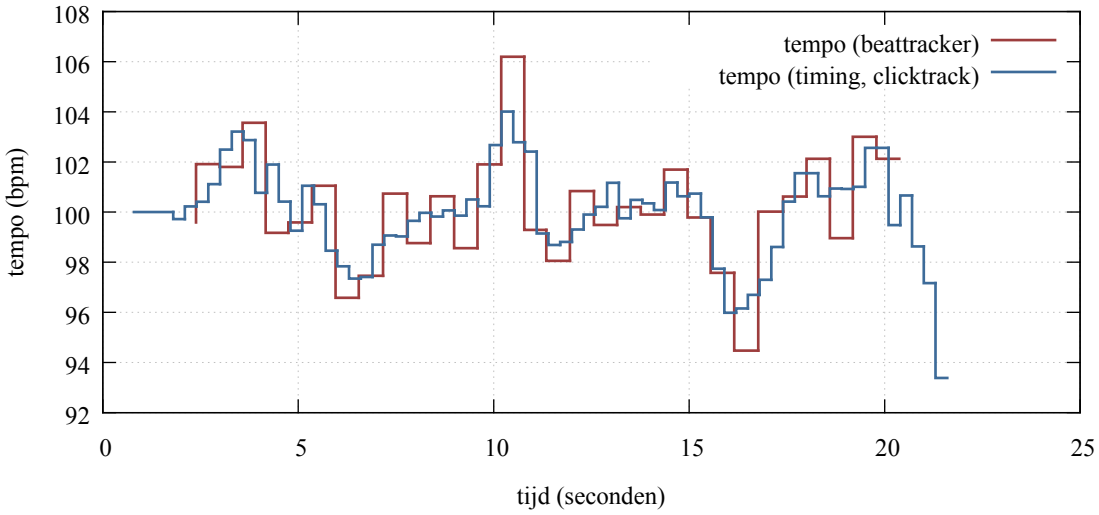
Figuur 6.4: Tempo - Pop Drumming p. 25, oef. 7 door drummer 1 (met metronoom)

aan waarmee een tijdsinterval op dat ogenblik in werkelijkheid verschilt van een verwacht tijdsinterval. De verwachte tijdsintervallen zijn gebaseerd op de interbeatintervallen die gebruikt werden bij de kwantisatie. Hieruit volgt dat het werkelijke interbeatinterval $I_{werkelijk}$ op elk tijdstip van de afgeleide timingcurve verschilt van het verwachte interbeatinterval $I_{verwacht}$ met een fractie gegeven door de waarde van de afgeleide timingcurve op dat tijdstip. Het werkelijke interbeatinterval wordt bijgevolg voor elk tijdstip van de afgeleide timingcurve gegeven door vergelijking 6.1.

$$I_{werkelijk} = I_{verwacht} \cdot \left(1 + \frac{dT}{dt}\right) \tag{6.1}$$

De tempocurve kan vervolgens berekend worden door voor elk berekend werkelijk interbeatinterval het tempo $\frac{60}{I_{werkelijk}}$ te berekenen. Voorbeelden van tempocurves die op deze manier berekend werden zijn te zien in de figuren 6.4 en 6.5 (blauwe lijn). In deze figuren is te zien hoe de tempocurve die gebaseerd is op de timingcurve een analoog verloop heeft aan de tempocurve die berekend werd op basis van de berekende beatposities, maar met meer geleidelijke tempowijzigingen een een meer verfijnde resolutie.

Ook van de waarden van deze tempocurves kan de standaardafwijking berekend worden die een beeld geeft van hoe constant het tempo van de muzikant is. Hierbij dient opgemerkt te worden



Figuur 6.5: Tempo - Pop Drumming p. 25, oef. 7 door drummer 2 (met metronoom)

dat het resultaat beïnvloed wordt door kwantisatiefouten, aangezien deze in sprongen van de timingcurve en bijgevolg in pieken in de tempocurve resulteren. Aangezien de performances van drummer 1 die ingespeeld werden met metronoom vaak heel wat kwantisatiefouten bevatten (met uitzondering van onder andere de performance die gebruikt werd in de getoonde figuren, wegens een groter kwantum) zijn de resultaten niet volledig representatief. Bijgevolg geven de waarden aan dat het tempo van drummer 2 in alle gevallen constanter was dan dat van drummer 1, wanneer met metronoom werd meegespeeld. Zonder metronoom is het tempo van drummer 2 in grote lijnen nog steeds constanter dan dat van drummer 1, met uitzondering van 2 studies waar drummer 1 beter scoort.

6.3 Strakheid

De metriek strakheid analyseert de spreiding van alle aanslagen die naar hetzelfde tijdstip kwantiseren. Met andere woorden: hoe ver liggen de aanslagen die de muzikant op hetzelfde tijdstip bedoelde te spelen uit elkaar. Dit wordt grafisch voorgesteld in figuren 6.1 en 6.2 door de verticale spreiding van symbolen die dezelfde horizontale coördinaat hebben.

Voor elk uniek gekwantiseerd tijdstip worden alle aanslagen verzameld die naar dat tijdstip kwantiseren. Indien meer dan één aanslag gevonden werd, wordt de standaardafwijking van de afstanden van deze aanslagen tot het gekwantiseerd tijdstip berekend. Het gemiddelde van al deze standaardafwijkingen geeft vervolgens een beeld van de strakheid van de performance van de muzikant. In figuur 6.6 worden twee situaties geïllustreerd die resulteren in een verschillende timing, maar eenzelfde strakheid. De horizontale as stelt de tijd voor, de verticale lijn geeft een punt van het kwantisatierooster weer en het venster, dat de grootte van het kwantum heeft, omvat alle aanslagen die naar het roosterpunt kwantiseren.

De resultaten van deze metriek worden weergegeven in tabellen B.9 en B.10. De waarden zijn de

Figuur 6.6: Situaties met verschillende timing maar dezelfde strakheid

gemiddelden van de standaardafwijkingen van de timingverschillen, dit zijn afstanden in de tijd, in seconden. Ook hier zorgen kwantisatiefouten voor een onbetrouwbaar resultaat, tenzij de fouten zo groot zijn dat álle aanslagen die normaal naar dezelfde waarde kwantiseren, naar dezelfde foutieve waarde kwantiseren. De resultaten tonen aan dat de metriek zinvol is: de waarden van drummer 2 liggen voor alle studies onder de waarden van drummer 1, hetgeen wijst op minder spreiding en bijgevolg een strakkere performance.

6.4 Dynamiek

In de muziek is dynamiek een maat voor de verschillen in sterkte waarmee het muziekstuk wordt uitgevoerd. Een performance met veel dynamiek bevat bijgevolg passages met zeer uiteenlopende sterkten, een performance met weinig dynamiek heeft daarentegen een eerder constante sterkte.

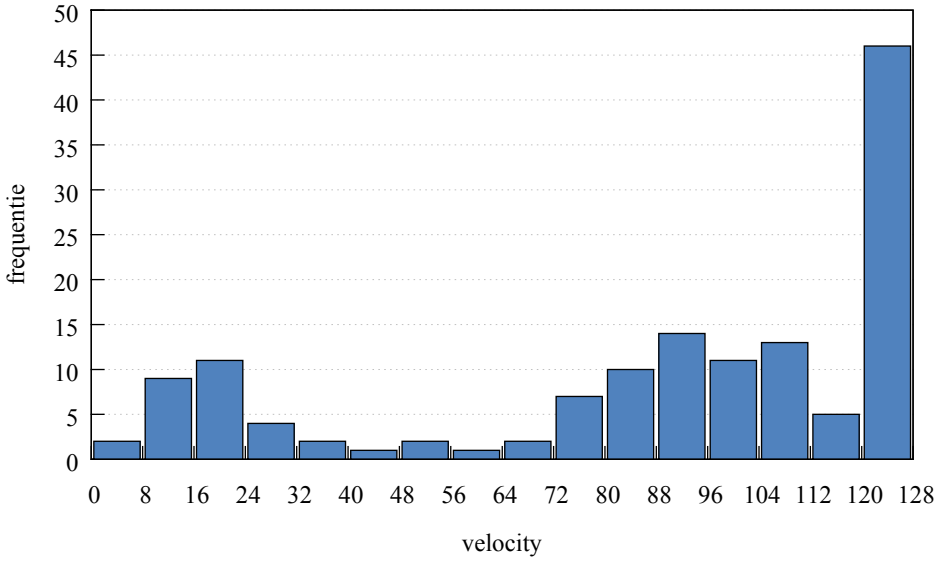
Deze maat kan in een performance in MIDI-formaat worden afgeleid uit de aanslagsterkten die worden aangegeven in de *velocity*-velden van de MIDI-berichten. Bij een performance met veel dynamiek zullen de sterkten van de aanslagen een grote spreiding vertonen. Ook hier is het dus weer zinvol om de standaardafwijking als maat voor de metriek te nemen.

Tabellen B.11 en B.12 bevatten het gemiddelde en de standaardafwijking van de aanslagsterkte voor de verschillende performances respectievelijk door drummer 1 en drummer 2. De waarden zijn *velocities* en kunnen dus in het interval $[0, 127]$ liggen. De resultaten zijn voor interpretatie vatbaar: indien het de bedoeling was van de muzikant om heel consistent te spelen zal hij een lage standaardafwijking verwachten, indien hij heel dynamisch bedoelde te spelen zal hij een hoge standaardafwijking verwachten. Voor de opnames die hier gebruikt worden kan worden opgemerkt dat drummer 2 minder moeite had om de stukken correct in te spelen, waardoor hij met iets meer gevoel speelde. Het dynamisch bereik en bijgevolg ook de standaardafwijking wordt verwacht hoger te liggen dan bij drummer 1. Dit blijkt echter niet altijd uit de resultaten.

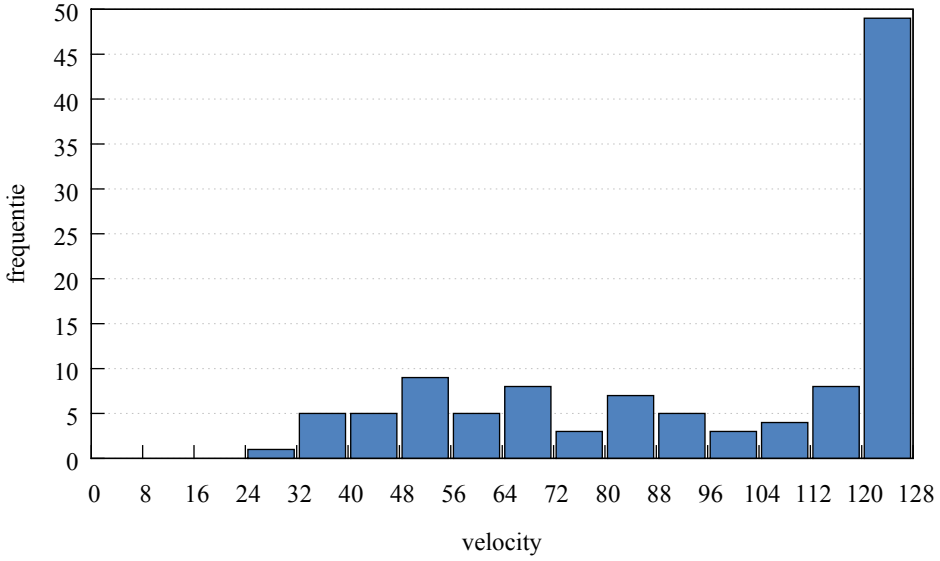
Bij een diepere analyse van de resultaten komen enkele verklaringen naar boven. Het histogram van de aanslagsterkten geeft een goed beeld van het dynamisch bereik van de performance. Bij sommige performances van drummer 1 blijkt dat er een aantal heel stille aanslagen aanwezig is (zie figuur 6.7). Een dichtere kijk op de performance toont dat dit aanslagen zijn die de muzikant niet bedoeld heeft te spelen, maar die het resultaat zijn van het laten rusten van een drumstok op de *snare drum*, waardoor de stok tijdens het spelen zachtjes op de trommel botste. Deze zachte aanslagen vergroten het dynamisch bereik.

Bij drummer 2 daarentegen is de aanslagsterkte op de *snare drum* consistent maximaal, de aanslagen op de *hi-hat* die op de beats vallen hebben ook vaak de maximale aanslagsterkte, de aanslagen op de *hi-hat* die tussen 2 beats vallen hebben vaak een zachtere aanslagsterkte. Dit resulteert in een piek voor de maximale aanslagsterkte, waardoor het dynamisch bereik verkleint. Deze piek kan in sommige gevallen ook worden opgemerkt bij drummer 1, maar is consistenter aanwezig bij drummer 2 (zie figuur 6.8).

Uit deze inzichten blijkt dat verder onderzoek voor deze metriek is aangewezen, waarbij de spreiding van de aanslagsterkten voor de verschillende delen van het drumstel apart in rekening gebracht kan worden.



Figuur 6.7: Histogram van de aanslagsterken van drummer 1 - Pop drumming p. 25, oef. 8 (met metronoom)



Figuur 6.8: Histogram van de aanslagsterken van drummer 2 - Pop drumming p. 25, oef. 8 (met metronoom)

Hoofdstuk 7

De applicatie

Het resultaat van deze masterproef is een applicatie die in staat is een performance in MIDI-formaat in te laden, te analyseren op basis van verschillende metrieken (zie hoofdstuk 6) en vervolgens de performance en gegevens van de metrieken te visualiseren. De configuratie die bij een performance hoort kan worden opgeslagen in de vorm van een project, zodat het achteraf mogelijk is om een project te laden zonder opnieuw de hele configuratie in te geven. Figuur 7.1 toont het hoofdscherm van deze applicatie, waarin een project geladen is.

7.1 Het model

De voornaamste klasse van de applicatie is de klasse *ApplicationModel*. Vooreerst beheert deze modelklasse alle ingeladen gegevens en alle instellingen (kortweg: de *staat*) van de applicatie. Daarnaast is ze verantwoordelijk voor het aansturen van alle operaties. In figuur 7.2 is het klassendiagram geschetst van het model en de voornaamste gerelateerde klassen.

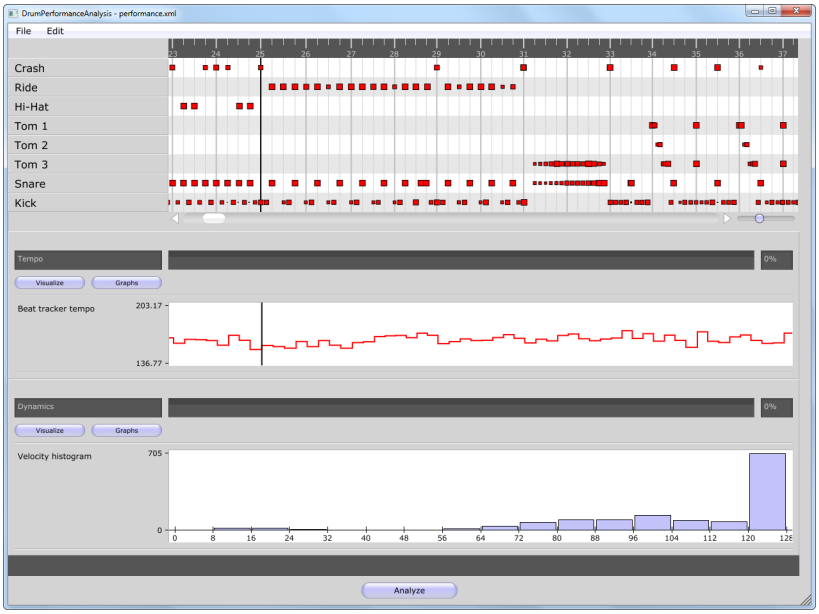
De performance bevindt zich in het model in de vorm van een object van de klasse *MidiMessageSequence* (zie §2.2.2). Het model bevat verschillende eigenschappen van de performance:

- Een object van de klasse *Instrument* dat de nootnummers van de MIDI-berichten afbeeldt op de delen waaruit het instrument bestaat (bijvoorbeeld: de verschillende toms en cimbalen van een drumstel, of de verschillende toetsen van een piano).
- De maatsoort, voorgesteld door een object van de klasse *TimeSignature*.
- Een booleaanse waarde *useClickTrack* die aangeeft of het tempo al dan niet vast is.
- Het tempo van de performance in beats per minuut (indien het tempo vast is).
- Een *vector* die de tijdstippen van de opeenvolgende beats bevat (indien het tempo niet vast is, en nadat deze berekend zijn, zie §7.5).

Om de performance te analyseren beschikt het model over een beattracker (in de vorm van een object van de klasse *MidiBeatTracker*), een quantizer (in de vorm van een object van de klasse *Quantizer*) en een verzameling modules (een vector van *Module* objecten, zie 7.4) die elk een bepaalde metriek implementeren. De analyse gebeurt in de lidfunctie `processOffline()` en wordt verder besproken in §7.5.

Het model beheert de verzameling *Renderers* die instaan voor de visualisatie van de performance en van de resultaten van de analyse (zie §7.6.1).

Voor communicatie met de datalaag (de in- en uitvoer van en naar bestanden, zie §7.2) beschikt het model over twee *data access objects*: een object van de klasse *ProjectDAO* om de staat van



Figuur 7.1: Het hoofdscherm van de applicatie

het model op te slaan naar of te lezen uit projectbestanden, en een object van de klasse *InstrumentDAO*, dat de configuratie van een instrument kan lezen uit een instrumentbestand.

De applicatie beschikt over een GUI die de staat van het model visueel weergeeft. Om dit mogelijk te maken werd het *Observer pattern* (Gamma et al., 1994) geïmplementeerd. De modelklasse is een concrete implementatie van de *Subject* klasse, waarbij componenten die op de hoogte wensen te blijven van de staat van het model (*observers*) zich kunnen registreren. Bij elke wijziging van de staat van de modelklasse wordt de operatie `notifyObservers()` aangeroepen, die alle observers op de hoogte brengt. Een component die geïnteresseerd is in updates van het model dient de *Observer* interface te implementeren. De component dient de `update()` operatie zodanig te implementeren dat hij zichzelf bijwerkt om de nieuwe staat van het model te reflecteren.

7.2 In- en uitvoer van en naar bestanden

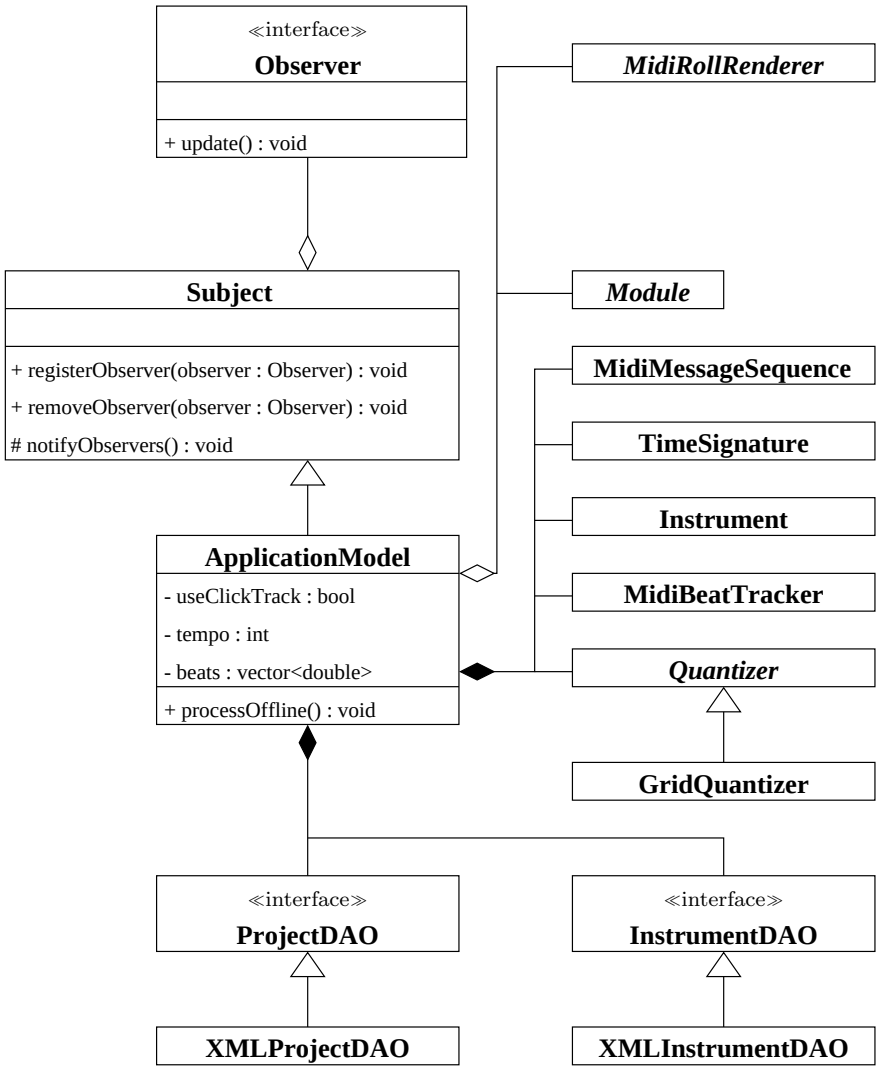
Om in- en uitvoer van en naar bestanden mogelijk te maken zijn twee *data access objects* voorzien:

- *ProjectDAO* voor projectbestanden,
- *ProjectDAO* voor instrumentbestanden.

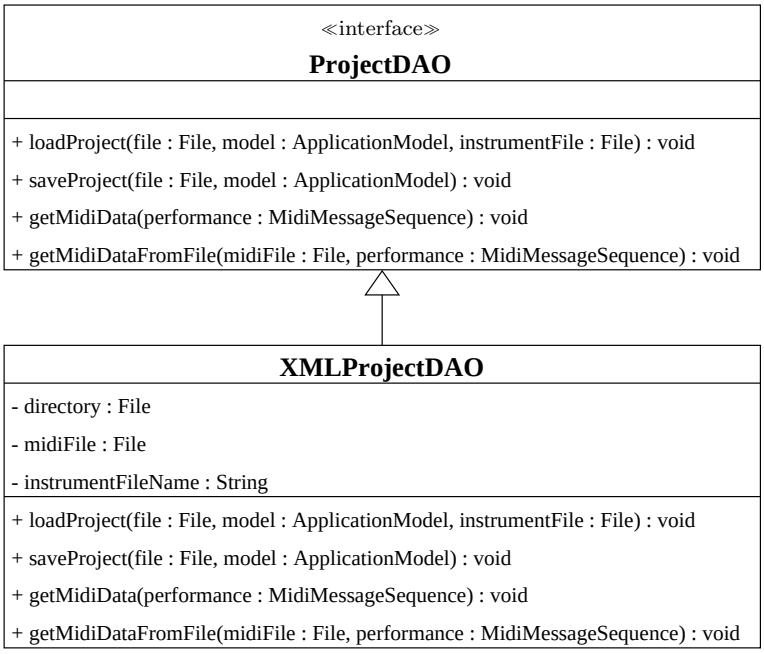
Het klassendiagram van deze data access objects is te zien in figuur 7.3.

7.2.1 Projectbestanden

Een projectbestand is een bestand dat de staat van het model van de applicatie beschrijft. De manipulatie van projectbestanden gebeurt via een object van een klasse die de interface *ProjectDAO* (zie figuur 7.3a) implementeert. Deze interface schrijft de operaties `loadProject()` en `saveProject()` voor respectievelijk om projectbestanden te lezen en te schrijven. Deze operaties hebben beiden de invoerparameter `file` die een verwijzing naar het bestand op de schijf bevat. De parameter `model` komt in de loadfunctie voor als uitvoerparameter, zodat de instellingen die in het bestand beschreven staan in het model kunnen worden geladen, en in de savefunctie als invoerparameter.



Figuur 7.2: Klassendiagram van het model en de voornaamste gerelateerde klassen



(a) Data access object voor projectbestanden

(b) Data access object voor instrumentbestanden

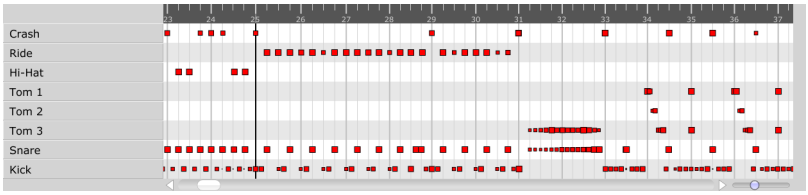
Figuur 7.3: Klassendiagrammen van de *data access objects*

Bij een performance die wordt ingeladen hoort een object van de klasse *Instrument*, dat de gebruikte nootnummers op de delen van het gebruikte instrument afbeeldt. Dit instrument wordt ook ingelezen uit een bestand (zie 7.2.2), waarvan de padnaam eventueel vermeld staat in het projectbestand. Daarom is bij de loadfunctie is nog een derde optionele (invoer)parameter `instrumentFile` voorzien. Als deze niet wordt opgegeven wordt hij gelijkgesteld aan een onbestaand *File* object (*File::nonexistent*) en wordt het instrumentbestand dat in het projectbestand vermeld staat geladen. Indien deze wel wordt opgegeven, dan wordt het instrumentbestand dat in het projectbestand vermeld staat genegeerd, en wordt het opgegeven instrumentbestand in de plaats geladen.

De performance zelf (indien aanwezig) bevindt zich in een apart bestand in het Standard Midi File formaat. In het projectbestand dient bijgevolg de padnaam naar dit bestand te worden opgenomen. De voorgeschreven operatie `getMidiData()` (die pas kan worden aangeroepen na het aanroepen van de loadfunctie) dient het bestand dat de performance bevat te laden in de uitvoerparameter `performance`.

Om een performance te laden uit een ander bestand kan de operatie `getMidiDataFromFile()` aangeroepen worden, waarbij het gewenste bestand als parameter wordt meegegeven.

Als formaat voor projectbestanden werd het project XML formaat ontworpen, dat volledig beschreven staat in bijlage C. Om bestanden in dit formaat te manipuleren is de klasse *XMLProjectDAO* voorzien, die de *ProjectDAO* interface implementeert. Tijdens het inlezen van het projectbestand wordt de directory waarin het bestand zich bevindt, een verwijzing naar het MIDI-bestand en de padnaam van het instrumentbestand bijgehouden in de private velden van het data access object. Wanneer achteraf een ander MIDI-bestand wordt ingeladen, wordt de verwijzing ernaar ook geüpdatet. Bij het opslaan van het projectbestand wordt vervolgens gekeken of het MIDI-bestand dieper in de bestandshiërarchie ligt dan het projectbestand of niet. Indien dit het geval is zal het MIDI-bestand met een relatieve padnaam vermeld worden in het projectbestand. Indien het hogerop in de bestandshiërarchie voorkomt zal een absolute padnaam gebruikt worden. Het instrumentbestand wordt steeds met zijn absoluut pad vermeld.



Figuur 7.4: De *MidiRollEditor* component

7.2.2 Instrumentbestanden

De manipulatie van instrumentbestanden gebeurt via een object van een klasse die de interface *InstrumentDAO* (zie figuur 7.3b) implementeert. De voorgeschreven operatie `setFile()` laat toe het bestand in te stellen waaruit het instrument gelezen wordt. Met `getFile()` kan dit bestand terug worden opgevraagd. Indien geen bestand werd opgegeven, dan geeft deze operatie een *File* object terug dat gelijk is aan *File::nonexistent*.

Nadat een verwijzing naar het instrumentbestand is opgegeven, kan het instrument worden opgevraagd met de operatie `getInstrument()`.

Als formaat voor instrumentbestanden werd het instrument XML formaat ontworpen, dat volledig beschreven staat in bijlage D. Manipuleren van bestanden in dit formaat kan met de klasse *XMLInstrumentDAO*, die de *InstrumentDAO* interface implementeert.

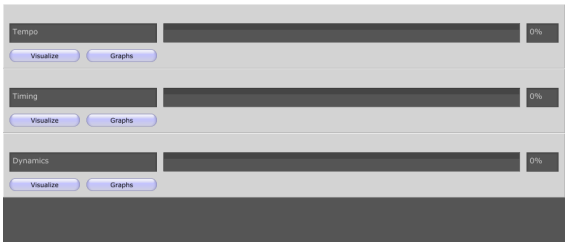
7.3 De MidiRollEditor component

De *MidiRollEditor*-component is een grafische component die instaat voor het visualiseren van de performance en van bepaalde gegevens van de metrieken die gebruikt worden om de performance te analyseren. Op de horizontale as wordt de tijd weergegeven. Verticaal geeft de component voor elk deel van het instrument een horizontale balk weer waarop de aanslagen van dat deel van het instrument getoond kunnen worden. De component ontleent zijn naam aan de papierrol (in het Engels: *pianoroll*) waarop de muziek van een pianola wordt opgeslagen. Aangezien de component echter muziek voor verschillende instrumenten kan weergeven werd dit omgevormd tot *MidiRoll*.

De component is opgebouwd uit een horizontale en een (optionele) verticale scrollbar, een *slider* die gebruikt wordt om in of uit te zoomen, en 3 belangrijke subcomponenten:

- *TimelineComponent*: De component bovenaan waarop de muzikale tijdschaal (maten en beats) wordt weergegeven. Door de muis verticaal te slepen op deze component wordt in- of uitgezoomd. Rechtsklikken op deze component toont een menu waarin de gebruiker kan kiezen om de tijdschaal volgens het vaste opgegeven tempo te gebruiken, of om de beats die door de beattracker berekend worden weer te geven.
- *InstrumentComponent*: De uiterst linkse component. Deze component toont de namen van de delen van het gebruikte instrument.
- *NoteComponent*: De centrale component van de *MidiRollEditor*. Hierop kunnen de performance en de gegevens van de metrieken gevisualiseerd worden door middel van *renderers* (zie §7.6.1).

De component implementeert de *Observer* interface en is geregisteerd bij het model om meldingen te ontvangen van wijzigingen (zie §7.1). Hij beschikt over een verwijzing naar een object van een klasse die de interface *MidiRollEditorModel* implementeert, een implementatie van deze interface dient alle nodige gegevens voor de component te voorzien. Het model bevat de staat van de applicatie, en implementeert bijgevolg deze interface, zo kan de component aan het model alle courante waarden opvragen.



Figuur 7.5: De *ModulesComponent*

7.4 Modules

Om de verschillende aspecten van de performance te analyseren beschikt de applicatie over *modules*. Een module analyseert de performance door het berekenen van verschillende gegevens op basis van een bepaalde metriek.

Achter de schermen bestaat een module uit twee delen: een object van de klasse *Module* die deel uitmaakt van het model en een object van de klasse *ModuleComponent* dat zich in de presentatielaag bevindt en de module grafisch weergeeft. Een *ModuleComponent* beschikt over een pointer naar de *Module* die hij grafisch voorstelt.

De klasse *Module* is een abstracte klasse. Gegevens waarover elke module sowieso beschikt worden in deze klasse bijgehouden:

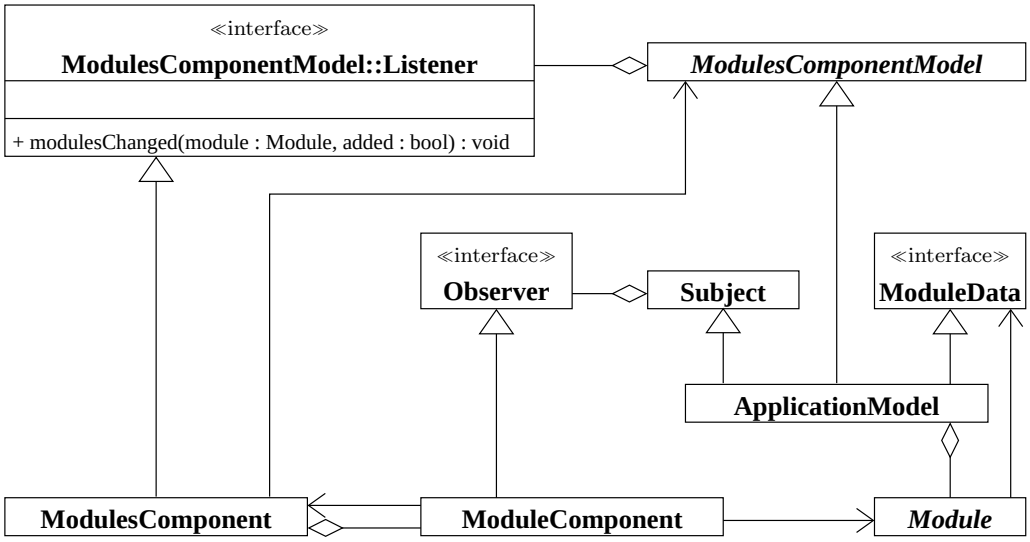
- De naam van de module.
- Een pointer naar een *ModuleData*-object, dat alle gegevens voorziet die de module kan gebruiken bij het analyseren van de performance (bijvoorbeeld: de performance zelf, het gebruikte instrument, het tempo, de maatsoort, etc. . .).
- Een vector die pointers naar *VisualisationData*-objecten bijhoudt die door *renderers* gevisualiseerd kunnen worden (zie §7.6.1).
- Een vector die pointers naar *NoteRendererData*-objecten bevat, indien de module over gegevens beschikt die door *NoteRenderers* kunnen worden weergegeven (zie §7.6.1).
- Een vector die pointers naar *GraphData*-objecten bijhoudt, die de gegevens bevatten die als grafieken getoond kunnen worden (zie §7.6.2).

Deze gegevens worden (met uitzondering van de naam) bijgehouden in *protected* velden van de *Module*. Verder schrijft de klasse de abstracte operatie `process()` voor, die aangeroepen wordt bij het analyseren van de performance.

Een module die een bepaalde metriek implementeert dient deze abstracte klasse uit te breiden. Ze kan beslissen welke gegevens beschikbaar zijn voor visualisatie door deze in de juiste vector van de abstracte bovenklasse te plaatsen. Tenslotte dient ze de `process()`-operatie op een zinvolle manier te implementeren, zodat hierin alle gegevens correct berekend worden.

De *Module*-objecten worden beheerd door het *ApplicationModel*, daar waar de *ModuleComponent*-objecten worden bijgehouden door de *ModulesComponent*. Op deze component worden de *ModuleComponent*-objecten geplaatst om de gegevens van de *Module* grafisch weer te geven (zie figuur 7.5).

Er bestaat een losse koppeling tussen de *Modules* die in het model geladen zijn en de *ModuleComponents* die op de *ModulesComponent* getoond worden, die ervoor zorgt dat wijzigingen in de configuratie van de modules in het model (toevoegen, verwijderen of wijzigen van volgorde van de modules) ook worden doorgevoerd in de *ModulesComponent*. Om dit te realiseren



Figuur 7.6: Klassendiagram van de modules en hun grafische componenten

beschikt de *ModulesComponent* over een *ModulesComponentModel* (dat geïmplementeerd wordt door het *ApplicationModel*). Dit *ModulesComponentModel* voorziet verschillende operaties om de modules in het model te beheren (toevoegen, verwijderen, wijzigen van volgorde). Bij het model kunnen zich verschillende luisteraars (objecten van een klasse die de interface *ModulesComponentModel::Listener* implementeert) registeren, die telkens op de hoogte gebracht worden wanneer de configuratie van de modules in het model werd gewijzigd. De *ModulesComponent* implementeert zelf deze interface, en kan zo automatisch de juiste *ModuleComponents* laden en weergeven, in overeenstemming met de *Modules* die in het model geladen zijn. Figuur 7.6 geeft een overzicht van de relaties tussen alle klassen die hierbij betrokken zijn.

Er werden 5 modules geïmplementeerd:

- *TempoModule:* Analyseert het tempo van de performance volgens de getrackte beats en volgens de variaties in timing, zoals besproken in §6.2..
- *TimingModule:* Analyseert de timing van de performance, zie §6.1.
- *TightnessModule:* Analyseert de strakheid van de performance, zie §6.3.
- *DynamicsModule:* Analyseert de dynamiek van de performance, zie §6.4.
- *NoteLengthModule:* Berekent de lengte van de gespeelde noten. Deze module is niet zo zinvol voor een drum performance, maar kan nuttig zijn voor gebruik bij andere instrumenten. Ze kan de lengte van de noten ook visualiseren op een *NoteRenderer* (zie §7.6.1), om zo het uitzicht van een pianoroll te verkrijgen.

7.5 Analyse van de performance

Nog vóór de analyse begint zijn reeds enkele waarden berekend, om de analyse te vereenvoudigen:

- De tijdstippen van de aanslagen (de *note-on*-berichten) van de performance worden in de vector *onsetTimes* gekopieerd.
- Er wordt een vector *onsetsToSequence* samengesteld die even lang is als de *onsetTimes* vector, die voor elke aanslag de index bevat van het overeenkomstige MIDI-bericht in de *MidiMessageSequence* van de performance.

Het analyseren van de performance gebeurt door het aanroepen van de lidfunctie `processOffline()` van het model. De analyse gebeurt in volgende stappen:

1. **Beat tracking:** De *MidiBeatTracker* wordt gereset, en zoekt de beats in de performance. De beats worden opgeslagen in de vector *beats*.

2. De *onsetTimes* vector wordt samen met de *beats* vector overlopen om voor elke aanslag de index van de laatste beat te vinden die ervoor valt. Deze indices worden opgeslagen in de *onsetsToBeats* vector.

3. **Kwantisatie:** De *beats* vector wordt doorgegeven aan de *Quantizer*, die vervolgens, afhankelijk van de instellingen, de aanslagen kwantiseert ten opzichte van een vast tempo of ten opzichte van de getrackte beats. De gekwantiseerde aanslagtijdstippen worden opgeslagen in de *quantizedOnsets* vector.

4. **Verwerking door de *Modules*:** Voor elke module die in het model geladen is wordt de `process()` lidfunctie aangeroepen. Elke geladen module analyseert de performance volgens een bepaalde metriek en berekent alle waarden die gebruikt kunnen worden voor visualisatie (zie §7.6). Elke module heeft hiervoor toegang tot alle hiervoor vermelde vectoren via het *ModuleData* object (zie §7.4).

Een direct gevolg van deze volgorde is dat wanneer een nieuwe module wordt toegevoegd, de analyse volledig opnieuw moet gebeuren opdat deze module een zinvolle uitvoer kan produceren.

7.6 Visualisatie

Om de muzikant meer inzicht te geven in zijn performance werden 2 systemen voor visualisatie geïmplementeerd. Het eerste systeem bestaat uit *renderers* die in staat zijn om de performance en resultaten van de analyse weer te geven op de *NoteComponent* van de *MidiRollEditor*. Het tweede systeem voorziet elke module van een manier om bepaalde gegevens te visualiseren door middel van grafieken.

7.6.1 Renderers

Visualisatie op de *NoteComponent* van de *MidiRollEditor* gebeurt door middel van *renderers* (objecten van de klasse *MidiRollRenderer*). Er zijn twee soorten renderers: de *NoteRenderer* en de *ZoneRenderer*.

De *NoteRenderer* is in staat om noten weer te geven door op de juiste horizontale positie en in de balk van het juiste deel van het instrument een symbool (hier: een vierkant/rechthoek) te tekenen. Om dit te kunnen realiseren heeft de *NoteRenderer* nood aan bijkomende informatie over de noten die gevisualiseerd moeten worden, deze informatie wordt aangeboden door een *NoteRendererData*-object. Dit object bevat een vector die alle aanslagtijdstippen bevat, de MIDI-sequentie die gevisualiseerd moet worden, een vector die de indices van elke aanslag in de sequentie bevat, en een instrument dat elk nootnummer dat gebruikt wordt in de sequentie afbeeldt op het juiste deel van het instrument (dit instrument moet uit dezelfde delen zijn opgebouwd als het instrument dat gebruikt wordt door de *MidiRollEditor*, de delen dienen daarenboven in dezelfde volgorde voor te komen).

De *ZoneRenderer* kan gegevens visualiseren door het inkleuren van horizontale zones over de volledige hoogte van de *NoteComponent*.

Om de eigenschappen van de visualisatie aan te passen beschikt elke renderer over parameters (objecten van de klasse *RendererParameter*). Een parameter heeft een standaardwaarde, een factor waarmee elke waarde vermenigvuldigd wordt, en een afstand (*offset*) waarover elke waarde verschoven wordt (na vermenigvuldiging met de factor. Tabel 7.1 geeft een overzicht van de beschikbare parameters van *Note*- en *ZoneRenderers*.

Tabel 7.1: Beschikbare parameters van *Note-* en *ZoneRenderers*

NoteRenderer			***ZoneRenderer***		
Parameter	Interval	Relatief?	Parameter	Interval	Relatief?
Kleurtint	[0,1]	ja	Kleurtint	[0,1]	ja
Verzadiging	[0,1]	ja	Verzadiging	[0,1]	ja
Helderheid	[0,1]	ja	Helderheid	[0,1]	ja
Ondoorzichtigheid	[0,1]	ja	Ondoorzichtigheid	[0,1]	ja
Absolute breedte	$[-\infty,+\infty]$	neen	Breedte	$[-\infty,+\infty]$	neen
Relatieve breedte	[0,1]	ja			
Horizontale verschuiving	$[-\infty,+\infty]$	neen			
Verticale verschuiving	[-1,1]	ja			

Met behulp van parameters is het mogelijk om bepaalde resultaten van de analyse grafisch weer te geven. Aan elke parameter kan nu bijkomstige data gekoppeld worden, die voor elk datapunt dat de renderer visualiseert (hetzij een aanslag, hetzij een gekleurde zone) een waarde voorziet die door de parameter gevisualiseerd wordt. Deze bijkomstige data bevindt zich in een object van de klasse *VisualisationData*, en wordt aangeboden door de modules.

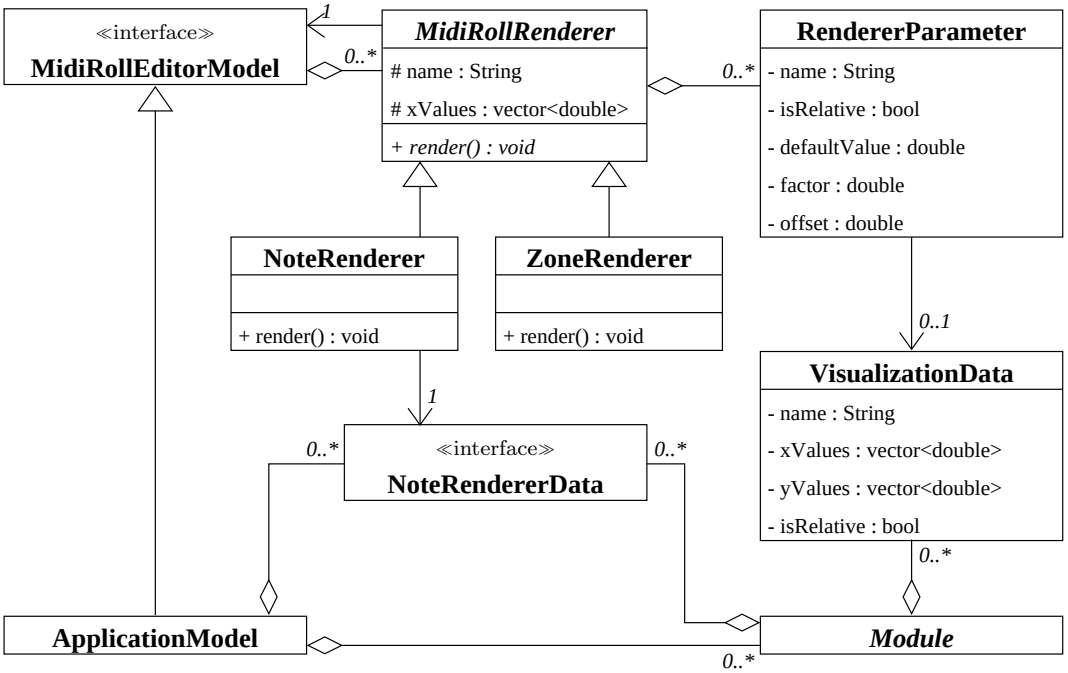
Een *VisualisationData*-object kan niet zomaar aan eender welke parameter gekoppeld worden. Een *VisualisationData*-object bevat een pointer naar een vector die de x-coördinaten bevat van de datapunten waarvoor de waarden gelden. Een renderer heeft op dezelfde manier een pointer naar een vector die de x-coördinaten bevat van de gegevens die hij visualiseert. Een *VisualisationData*-object komt pas in aanmerking om aan de parameters van een bepaalde renderer gekoppeld te worden indien beide pointers naar dezelfde vector wijzen (en zowel het *VisualisationData*-object als de renderer dus dezelfde x-coördinaten gebruiken). Verder dient een onderscheid gemaakt te worden tussen parameters die relatieve waarden in het interval [-1,1], [0,1] of [-1,0] (afhankelijk van het interval kan ervoor gecorrigeerd worden door het instellen van de *factor* en de *offset* van de parameter) kunnen weergeven (zoals bijvoorbeeld de kleurtint of de verticale verschuiving) en parameters die absolute tijdswaarden weergeven (zoals de absolute breedte van een symbool of een zone). Ditzelfde onderscheid wordt gemaakt voor de gegevens die een *VisualisationData*-object voorstelt. Een *VisualisationData*-object kan slechts aan een *RendererParameter* gekoppeld worden indien het hetzelfde type data aanbiedt als dat de renderer aanvaardt. Figuur 7.7 biedt een verduidelijkend overzicht van alle betrokken klassen en hun onderlinge relaties.

Het visualiseren van de data gebeurt door de lidfunctie `render()` aan te roepen, waaraan de volgende parameters worden meegegeven:

- De grafische context (een object van de klasse *Graphics*) waarop getekend wordt.
- De x-coördinaten van de meest linkse en de meest rechtse pixel van de component. Deze zijn nodig om horizontaal te kunnen scrollen en zoomen.
- De y-coördinaten van de bovenste en onderste pixel van de component. Deze zijn nodig om verticaal te kunnen scrollen.

7.6.2 Grafieken

Een module kan beschikken over gegevens die kunnen voorgesteld worden door middel van een grafiek. Deze gegevens worden gegroepeerd in objecten van de klasse *GraphData*. Een *GraphData*-object beschikt minstens over een beschrijvende naam, een vector die de coördinaten op de x-as bevat, een vector die de waarden horend bij de x-coördinaten bevat en een *Range*-object dat het bereik van de waarden beschrijft.



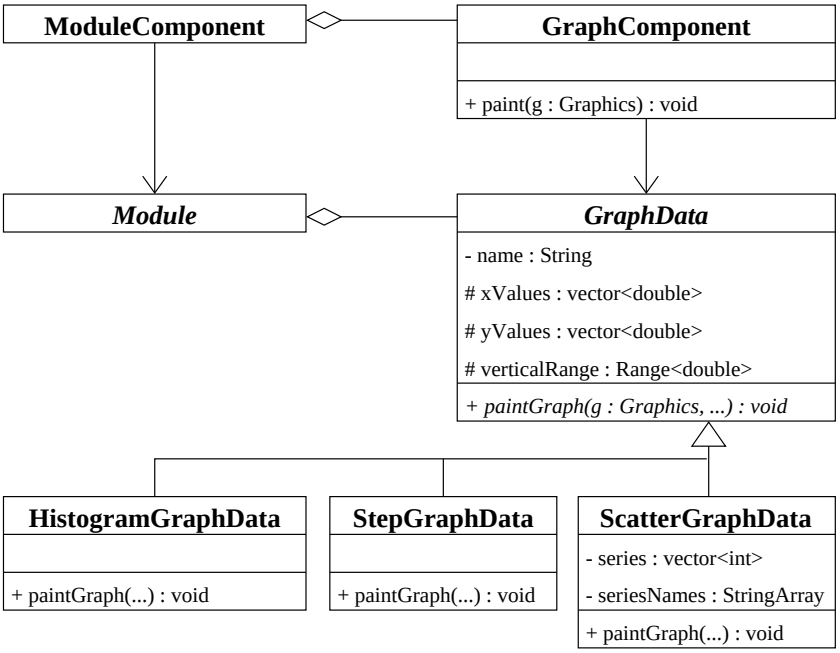
Figuur 7.7: Klassendiagram van de renderers

Grafieken worden getekend op objecten van de klasse *GraphComponent*. Deze objecten worden op de bijhorende *ModuleComponent* geplaatst. Hiertoe heeft de *ModuleComponent* een vector van pointers naar *GraphComponent*-objecten, deze vector bevat evenveel elementen als de bijhorende module *GraphData*-objecten heeft. Wanneer een grafiek getoond wordt, bevat deze vector een pointer naar het juiste object, wanneer een grafiek niet getoond wordt wijst de pointer naar 0.

Vanzelfsprekend heeft een *GraphComponent* een pointer naar het *GraphData*-object dat hij voorstelt. Om verschillende soorten grafieken met verschillend gedrag te kunnen weergeven werd het *Strategy pattern* geïmplementeerd. Een *GraphData*-object beschikt over de mogelijkheid om zichzelf weer te geven op een grafische context door middel van de operatie `paintGraph()`. Elke implementatie beslist zelf hoe dit gebeurt. Wanneer een *GraphComponent* op het scherm getekend wordt roept deze de `paintGraph()`-operatie aan van zijn bijhorend *GraphData*-object, waaraan de grafische context van de *GraphComponent* wordt meegegeven. Figuur 7.8 toont het bijhorende klassendiagram.

Er werden drie soorten grafieken geïmplementeerd:

- **Histogram:** Toont de frequentieverdeling van informatie die in klassen gegroepeerd is door middel van verticale balken. Het histogram is steeds even breed als de omvattende *GraphComponent* en reageert niet op scrol- of zoomacties op de *MidiRollEditor*. Als x-coördinaten worden de middelste waarden van elke klasse opgegeven. De y-coördinaten bevatten de frequentie voor elke klasse. Een histogram wordt weergegeven door een object van de klasse *HistogramGraphData*.
- **Spreidingsdiagram of *scatter plot*:** Toont een verzameling waarden (y-as) op verschillende tijdstippen (x-as) door middel van symbolen. De punten kunnen in verschillende reeksen gegroepeerd worden die elk met een uniek symbool worden weergegeven. Hiertoe dient een extra vector te worden opgegeven die voor elk datapunt een index bevat die de reeks aangeeft waartoe het punt behoort. Daarnaast is een tabel van *Strings* aanwezig die



Figuur 7.8: Klassendiagram van de grafieken

de naam van elke reeks bevat. Aangezien het spreidingsdiagram tijdwaarden weergeeft op de x-as, scrolt en zoomt het mee met dezelfde horizontale acties op de *MidiRollEditor*.

- **Traplijndiagram:** Toont waarden op verschillende tijdstippen door middel van een trapvormige lijn. Bij elk datapunt verspringt de hoogte van de lijn naar de nieuwe waarde. Aangezien de x-coördinaten weer tijdwaarden zijn scrolt en zoomt deze grafiek weer samen met de *MidiRollEditor*.

Om de GUI niet te overladen met informatie is er geen legende aanwezig. De gebruiker kan echter wel meer informatie verkrijgen via de *tooltips* die getoond worden wanneer met de muis over een grafiek bewogen wordt.

Hoofdstuk 8

Besluit

In deze masterproef werden de mogelijkheden onderzocht om verschillende aspecten een muzikale performance op een drumstel (in MIDI-formaat) te analyseren. Daarnaast werd een applicatie ontwikkeld die zowel de performance zelf als de resultaten van de analyse kan visualiseren op verschillende manieren.

Alvorens de verschillende aspecten van de performance te analyseren met behulp van verschillende metrieken, werd de muzikale structuur gezocht in de gegevensstroom van de performance door middel van *beat tracking* en *kwantisatie*. Vermits verschillende metrieken gebruik maken van de gevonden structuur is het van belang dat de algoritmes die gebruikt worden om deze structuur te vinden goede resultaten produceren. *Beat tracking* werd verwezenlijkt door het algoritme uit Dixon (2001), dat mits enkele kleine aanpassingen er in slaagt om op correcte wijze de beats te vinden in bijna alle testperformances. De eenvoudige methode die geïmplementeerd werd om de performance te kwantiseren vertoont goede resultaten indien ze samen gebruikt wordt met de resultaten van de *beat tracker*. Indien ze echter gebruikt wordt bij performances die zijn ingespeeld met een metronoom zijn de resultaten, afhankelijk van de bekwaamheid van de muzikant, vaak ondermaats.

De onderzochte metrieken proberen een waarde te berekenen die weergeeft hoe goed de muzikant presteert voor een bepaald aspect van de performance. De resultaten van de metrieken timing, tempo en strakheid weerspiegelen duidelijk het verschil in ervaring tussen een beginnende en een meer ervaren drummer, hetgeen het nut en de correctheid van deze metrieken bewijst. Daarenboven werd bij de metriek tempo een methode onderzocht om, zonder gebruik te maken van de *beat tracker*, tempofluctuaties te detecteren in een performance die met een metronoom werd ingespeeld. Het resultaat van deze methode loopt gelijk met, maar is verfijnder dan het resultaat van de *beat tracker*. De waarden voor de metriek dynamiek zijn voor interpretatie vatbaar, en wijzen op de nood aan verder onderzoek voor deze metriek.

De ontwikkelde applicatie stelt de muzikant in staat om zijn performance op een gebruiksvriendelijke manier in te laden en te analyseren. Om verschillende aspecten van de performance te analyseren kan de gebruiker modules laden die zich elk op een verschillende metriek baseren. Via een systeem van renderers is het mogelijk om zowel de performance als de analyseresultaten van de verschillende modules te visualiseren. Daarnaast kan een module berekende gegevens weergeven in de vorm van grafieken. Deze twee visualisatiemethoden stellen de applicatie in staat om de muzikant op een grafische manier feedback te geven over zijn performance. Daarnaast is de applicatie, dankzij de visualisatiemogelijkheden, een ideale omgeving om nieuwe metrieken in te ontwerpen in de vorm van modules. De applicatie is gemakkelijk uitbreidbaar in die zin dat het weinig moeite kost om een nieuwe module aan de applicatie toe te voegen. Een ontwikkelaar kan op een eenvoudige wijze de gegevens die door de module berekend worden grafisch voorstellen, hetgeen hem in staat stelt om sneller fouten (maar ook sterke punten) op te sporen in de ontworpen metriek.

Naar de toekomst toe zou het interessant zijn om de applicatie uit te breiden met enkele bijkomstige features. Voor de verschillende metrieken kunnen functies ontworpen worden die de resultaatwaarde omzetten naar een zinvolle quotatie. Dit geeft een duidelijker referentiekader voor de muzikant en stelt hem in staat om zich doelen voorop te stellen (bijvoorbeeld: "Ik blijf deze oefening hernemen tot ik een score van 90% haal"). Voorts zou het handig zijn om opname- en afspeelfunctionaliteit te voorzien. Met opnamefunctionaliteit hoeft de muzikant niet meer eerst zijn performance op te nemen met behulp van een ander programma alvorens ze in deze applicatie te laden. Afspeelfunctionaliteit stelt de muzikant in staat om achteraf te beluisteren wat hij net speelde, zo kan hij voor zichzelf verifiëren waarom een bepaalde metriek een goed of slecht resultaat aangaf. Om eerder aangehaalde redenen zou het interessant zijn om de mogelijkheden voor kwantisatie verder uit te diepen, een betere *quantizer* zorgt namelijk voor een meer betrouwbaar resultaat van de analyse. Tot slot kunnen de mogelijkheden onderzocht worden om de analyse in real-time uit te voeren, zodat de muzikant al feedback kan krijgen tijdens het spelen.

Lijst van figuren

Lijst van tabellen

Literatuurlijst

Cemgil, A. T., Desain, P., & Kappen, B. (1999). Rhythm Quantization for Transcription.

de Berg, M., Cheong, O., van Kreveld, M., & Overmars, M. (2008). *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3^{rd} edition.

Desain, P. & Honing, H. (1992). The quantization problem: traditional and connectionist approaches. In M. Balaban, K. Ebcioglu, & O. Laske (Eds.), *Understanding Music with AI: Perspectives on Music Cognition* (pp. 448–463). Cambridge: MIT Press.

Dixon, S. (2001). Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1), 39–58.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Glatt, J. (2009). MIDI Technical Fanatic's Brainwashing Center. `http://home.roadrunner.com/~jgglatt/`.

Hamanaka, M., Goto, M., Asoh, H., & Otsu, N. (2001). A Learning-Based Quantization: Estimation of Onset Times in a Musical Score. *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics 2001*, 10(1), 374–379.

Hamanaka, M., Goto, M., Asoh, H., & Otsu, N. (2003). A Learning-Based Quantization: Unsupervised Estimation of the Model Parameters. In *International Computer Music Conference Proceedings* (pp. 369–372).

haydxn (2005). Juce programming tutorial. `http://code.google.com/p/juced/wiki/JuceTutorial`.

Iwami, N. & Miura, M. (2007). A support system for basic practice of playing the drums. In *Proceedings of the International Computer Music Conference*.

Longuet-Higgins, H. C. (1979). The perception of music. In *Proceedings of the Royal Society*, volume 205 (pp. 307–322).

Rosenthal, D. (1992). Emulation of Human Rhythm Perception. *Computer Music Journal*, 16(1), 64–76.

Storer, J. (2013). JUCE: Jules' Utility Class Extensions (Version 2.0) [klassenbibliotheek]. `http://www.juce.com/`.

Stroustrup, B. (2008). *De programmeertaal C++*. Pearson Education, 2^{nd} edition.

Bijlage A

Testgegevens

Als materiaal om de analyse te testen werden enkele studies gebruikt die typisch aan bod komen tijdens een slagwerkopleiding. De gebruikte studies worden opgedeeld in eenvoudige studies en geavanceerde studies. Elke categorie bevat twee volledige drumsolo's, en twee pagina's met korte oefeningen die viermaal herhaald worden. De eenvoudige studies staan uitgeschreven in tabel A.1, de geavanceerde studies in tabel A.2.

Alle studies werden tweemaal ingespeeld: éénmaal terwijl de muzikant meespeelt met een metronoom en éénmaal zonder metronoom. Volgende formaten werden opgenomen:

- MIDI, opgeslagen in het *Standard Midi File* formaat (.mid).
- Audio, 44kHz, 16-bit, 1 kanaal (.wav).

Tabel A.1: Eenvoudige drumstudies

Afkorting	**Titel**	**Auteur**	**Maatsoort**
es1	Solo 2	Henk Mennens	$\frac{4}{4}$
es2	Solo 7	Henk Mennens	$\frac{6}{8}$
epd1 t.e.m. epd5	Independent Pop Drumming pagina 24 (oef. 1 t.e.m. 5)	Frans De Witte	$\frac{4}{4}$
epd6 t.e.m. epd11	Independent Pop Drumming pagina 25 (oef. 6 t.e.m. 11)	Frans De Witte	$\frac{4}{4}$

Tabel A.2: Geavanceerde drumstudies

Afkorting	**Titel**	**Auteur**	**Maatsoort**
as1	Solo 8	Gert Bomhof	$\frac{4}{4}$
as2	Solo 10	Gert Bomhof	$\frac{2}{2}$
apd1 t.e.m. apd6	Independent Pop Drumming pagina 42 (oef. 1 t.e.m. 6)	Frans De Witte	$\frac{4}{4}$
apd7 t.e.m. apd12	Independent Pop Drumming pagina 47 (oef. 1 t.e.m. 6)	Frans De Witte	$\frac{4}{4}$

Bijlage B

Resultaten van de analyse

B.1 Beat tracking

In de volgende performances werden fouten gemaakt door de *beat tracker*:
drummer 1

- *Pop Drumming - p. 25, oef. 9 (met metronoom)*: De laatste beat komt iets te laat.
- *Pop Drumming - p. 25, oef. 9 (zonder metronoom)*: Het gevonden interbeatinterval is correct, de beatfase is met een halve tel opgeschoven.

drummer 2

- *Gert Bomhof - Solo 8 (zowel met als zonder metronoom)*: De laatste beat van de voorlaatste maat komt iets te laat, waardoor ook de beats van de laatste maat verkeerd zijn.
- *Gert Bomhof - Solo 10 (zowel met als zonder metronoom)*: In grote lijnen wordt het correcte interbeatinterval gevonden, maar is de beatfase met een halve tel opgeschoven. Bij lange fill-in passages worden kleine foutjes gemaakt.

Bij alle andere performances werden de correcte beatposities gevonden.

B.2 Kwantisatie

In de volgende performances werden ernstige kwantisatiefouten vastgesteld:
drummer 1

- *Henk Mennens - Solo 2 (met metronoom)*
- *Pop Drumming - p. 24, oef. 1 (met metronoom)*
- *Pop Drumming - p. 24, oef. 2 (met metronoom)*
- *Pop Drumming - p. 24, oef. 3 (met metronoom)*
- *Pop Drumming - p. 24, oef. 4 (met metronoom)*
- *Pop Drumming - p. 24, oef. 5 (met metronoom)*
- *Pop Drumming - p. 25, oef. 6 (met metronoom)*
- *Pop Drumming - p. 25, oef. 8 (met metronoom)*

In de volgende performances werden kleine kwantisatiefouten vastgesteld:
drummer 1

- *Pop Drumming - p. 25, oef. 9 (met metronoom)*
- *Pop Drumming - p. 25, oef. 10 (met metronoom)*: Klein foutje bij versiernoten.
- *Pop Drumming - p. 25, oef. 11 (met metronoom)*: Enkele foutjes in de eerste helft van de eerste maat.

drummer 2

- *Gert Bomhof - Solo 8 (zowel met als zonder metronoom)*: Kleine foutjes wegens een te groot kwantum. Een kleiner kwantum levert een incorrecte kwantisatie op.
- *Gert Bomhof - Solo 10 (met metronoom)*
- *Pop Drumming - p. 25, oef. 6 (met metronoom)*
- *Pop Drumming - p. 42, oef. 1 (met metronoom)*
- *Pop Drumming - p. 42, oef. 4 (met metronoom)*
- *Pop Drumming - p. 42, oef. 6 (met metronoom)*
- *Pop Drumming - p. 47, oef. 1 (met metronoom)*
- *Pop Drumming - p. 47, oef. 2 (met metronoom)*

Ook bij de performance van *Gert Bomhof - Solo 10 (zonder metronoom)* kunnen kwantisatiefouten worden vastgesteld. De kwantisatie is echter niet betrouwbaar wegens het foutieve resultaat van de *beat tracker*. Alle andere performances werden correct gekwantiseerd.

B.3 Metrieken

Tabel B.1: Waarden voor metriek timing, ingespeeld door drummer 1 zonder metronoom

Studie	Gemiddelde	Standaardafwijking	Gemiddelde (absoluut)
es1	-0,00221	0,00886	0,00561
epd1	0,00361	0,01034	0,00669
epd2	-0,00263	0,00746	0,00573
epd3	-0,00305	0,01443	0,00847
epd4	-0,00061	0,00887	0,00573
epd5	-0,00089	0,01076	0,00729
epd6	-0,00424	0,00834	0,00646
epd7	-0,00350	0,01495	0,00802
epd8	-0,00424	0,01492	0,00944
epd9	-0,00061	0,01119	0,00733
epd10	-0,00190	0,01792	0,01117
epd11	-0,00104	0,01015	0,00642

Tabel B.2: Waarden voor metriek timing, ingespeeld door drummer 1 met metronoom, gekwantiseerd volgens de metronoom

Studie	**Gemiddelde**	**Standaardafwijking**	**Gemiddelde (absoluut)**
es1	-0,00777	0,03197	0,02753
epd1	0,00020	0,05717	0,05427
epd2	0,00233	0,05635	0,05331
epd3	0,00298	0,04973	0,04752
epd4	-0,00612	0,04663	0,04055
epd5	0,02997	0,04715	0,05299
epd6	-0,00902	0,05167	0,04948
epd7	-0,06698	0,03207	0,06704
epd8	0,00444	0,04863	0,04390
epd9	-0,03268	0,03925	0,04791
epd10	-0,07286	0,05135	0,08464
epd11	-0,02289	0,02870	0,03251

Tabel B.3: Waarden voor metriek timing, ingespeeld door drummer 2 zonder metronoom

Studie	**Gemiddelde**	**Standaardafwijking**	**Gemiddelde (absoluut)**
es1	-0,00115	0,00629	0,00455
es2	-0,00177	0,00629	0,00400
epd1	-0,00213	0,00583	0,00455
epd2	-0,00220	0,00505	0,00406
epd3	-0,00223	0,00654	0,00475
epd4	-0,00327	0,00600	0,00466
epd5	-0,00047	0,00445	0,00345
epd6	-0,00090	0,00553	0,00389
epd7	-0,00055	0,00460	0,00321
epd8	-0,00172	0,00688	0,00490
epd9	-0,00011	0,00700	0,00488
epd10	-0,00050	0,00754	0,00519
epd11	-0,00134	0,00662	0,00444
as1	-0,00292	0,01283	0,00832
as2	0,00321	0,01796	0,01148
apd1	-0,00200	0,00862	0,00601
apd2	0,00030	0,00650	0,00417
apd3	0,00178	0,01078	0,00778
apd4	-0,00226	0,01233	0,00882
apd5	0,00004	0,01186	0,00810
apd6	-0,00010	0,00946	0,00558
apd7	0,00327	0,01029	0,00593
apd8	-0,00295	0,01126	0,00799
apd9	-0,00295	0,01130	0,00709
apd10	-0,00245	0,00712	0,00552
apd11	-0,00283	0,01062	0,00638
apd12	-0,00111	0,00802	0,00604

Tabel B.4: Waarden voor metriek timing, ingespeeld door drummer 2 met metronoom, gekwantiseerd volgens de metronoom

Studie	Gemiddelde	Standaardafwijking	Gemiddelde (absoluut)
es1	-0,02643	0,01596	0,02681
es2	-0,02974	0,01502	0,02997
epd1	-0,02137	0,01349	0,02212
epd2	-0,02350	0,01400	0,02395
epd3	-0,03565	0,01752	0,03688
epd4	-0,02900	0,01272	0,02900
epd5	-0,02657	0,00904	0,02657
epd6	-0,03820	0,01992	0,04012
epd7	-0,02904	0,01794	0,02935
epd8	-0,03083	0,01663	0,03089
epd9	-0,03147	0,01173	0,03147
epd10	-0,03445	0,01555	0,03445
epd11	-0,02543	0,01164	0,02543
as1	0,02698	0,02007	0,02984
as2	-0,00583	0,02242	0,01830
apd1	-0,03098	0,02816	0,03620
apd2	-0,03177	0,02256	0,03495
apd3	-0,00961	0,01396	0,01298
apd4	-0,00616	0,02261	0,01706
apd5	-0,02968	0,02455	0,03258
apd6	-0,03676	0,01688	0,03813
apd7	-0,03390	0,02774	0,03796
apd8	-0,02951	0,02287	0,03117
apd9	-0,03978	0,01774	0,04014
apd10	-0,02573	0,01207	0,02595
apd11	-0,03814	0,01798	0,03851
apd12	-0,04116	0,01717	0,04191

Tabel B.5: Waarden voor metriek tempo, ingespeeld door drummer 1 zonder metronoom

	Beattracker		Timing	
Studie	Gemiddelde	Standaardafwijking	Gemiddelde	Standaardafwijking
es1	113,889	4,015	114,416	4,509
epd1	102,163	2,393	102,729	2,884
epd2	105,025	2,714	105,214	2,789
epd3	105,595	3,353	105,629	3,492
epd4	105,380	4,046	106,286	5,762
epd5	110,485	3,434	110,074	3,952
epd6	105,920	2,166	105,946	2,399
epd7	108,134	6,529	107,848	6,309
epd8	115,163	2,223	115,190	2,817
epd9	99,940	2,322	99,934	2,238
epd10	106,835	2,689	106,425	2,995
epd11	100,858	2,326	101,183	5,418

Tabel B.6: Waarden voor metriek tempo, ingespeeld door drummer 1 met metronoom, gekwantiseerd volgens de metronoom

	Beattracker		Timing	
Studie	**Gemiddelde**	**Standaardafwijking**	**Gemiddelde**	**Standaardafwijking**
es1	110,142	3,193	110,238	5,494
epd1	100,232	1,854	100,645	10,794
epd2	100,347	1,965	100,657	9,118
epd3	100,311	2,673	100,327	5,487
epd4	100,771	3,481	100,287	6,530
epd5	100,293	2,496	100,772	9,190
epd6	100,392	2,913	100,913	9,426
epd7	100,489	2,936	100,058	4,128
epd8	100,307	2,597	101,048	8,712
epd9	99,809	4,899	100,141	5,858
epd10	100,799	3,075	100,116	4,678
epd11	99,684	2,631	100,833	5,834

Tabel B.7: Waarden voor metriek tempo, ingespeeld door drummer 2 zonder metronoom

	Beattracker		Timing	
Studie	**Gemiddelde**	**Standaardafwijking**	**Gemiddelde**	**Standaardafwijking**
es1	99,070	2,104	99,970	2,415
es2	203,222	6,062	203,997	6,783
epd1	100,096	1,509	100,056	1,625
epd2	100,364	1,887	100,416	1,913
epd3	100,304	2,316	100,171	2,890
epd4	99,792	1,749	99,767	1,889
epd5	100,887	1,313	100,862	1,276
epd6	100,002	2,111	99,752	2,419
epd7	100,094	1,786	100,250	1,908
epd8	100,439	1,700	100,720	2,457
epd9	103,297	2,290	103,384	2,644
epd10	103,243	1,606	103,197	1,633
epd11	102,287	1,679	102,728	3,324
as1	110,169	3,581	110,116	4,148
as2	78,327	4,797	78,584	5,549
apd1	99,476	1,667	99,470	1,983
apd2	99,882	1,270	99,894	1,263
apd3	98,721	2,192	98,668	2,342
apd4	77,910	1,503	77,801	1,722
apd5	90,846	2,031	90,845	2,240
apd6	91,714	1,115	91,816	1,725
apd7	92,574	1,693	92,731	1,940
apd8	91,801	2,325	91,598	2,400
apd9	92,858	2,175	92,791	2,135
apd10	90,895	1,191	91,018	1,282
apd11	94,337	1,537	94,364	1,537
apd12	91,543	2,217	91,573	2,523

Tabel B.8: Waarden voor metriek tempo, ingespeeld door drummer 2 met metronoom, gekwantiseerd volgens de metronoom

	Beattracker		**Timing**	
Studie	**Gemiddelde**	**Standaardafwijking**	**Gemiddelde**	**Standaardafwijking**
es1	100,031	1,832	100,003	1,633
es2	200,187	5,852	200,339	5,666
epd1	100,242	1,856	100,071	2,196
epd2	100,027	1,974	99,928	2,391
epd3	100,223	2,588	99,910	2,546
epd4	100,028	1,708	99,788	1,747
epd5	100,169	1,179	99,980	1,219
epd6	100,199	1,806	100,078	2,782
epd7	100,182	2,233	99,967	1,870
epd8	100,142	1,991	100,072	2,087
epd9	100,147	1,834	99,823	2,393
epd10	100,113	1,818	99,876	1,958
epd11	100,098	1,965	99,807	2,677
as1	109,924	3,850	110,090	3,089
as2	80,997	5,032	80,008	1,729
apd1	100,143	2,107	100,091	3,818
apd2	100,343	1,284	100,267	3,594
apd3	100,010	2,197	99,917	1,545
apd4	99,850	1,872	99,809	2,462
apd5	89,959	2,976	89,973	2,248
apd6	90,026	1,263	89,989	1,902
apd7	90,015	1,683	90,073	2,440
apd8	90,325	1,942	90,189	1,855
apd9	90,155	1,936	89,952	1,679
apd10	90,018	1,171	90,033	1,217
apd11	90,045	1,664	90,043	1,490
apd12	90,086	2,475	90,035	1,666

Tabel B.9: Waarden voor metriek strakheid, ingespeeld door drummer 1 met en zonder metronoom

	Met metronoom	**Zonder metronoom**
Studie	**Gemiddelde**	**Gemiddelde**
es1	0,00667	0,00634
epd1	0,01486	0,00612
epd2	0,00881	0,00653
epd3	0,00704	0,01106
epd4	0,00960	0,00810
epd5	0,00770	0,00505
epd6	0,00559	0,00567
epd7	0,00600	0,00921
epd8	0,01589	0,00756
epd9	0,01085	0,00719
epd10	0,01476	0,01662
epd11	0,00728	0,00493

Tabel B.10: Waarden voor metriek strakheid, ingespeeld door drummer 2 met en zonder metronoom

	Met metronoom	Zonder metronoom
Studie	**Gemiddelde**	**Gemiddelde**
es1	0,00550	0,00510
es2	0,00742	0,00604
epd1	0,00612	0,00524
epd2	0,00486	0,00414
epd3	0,00603	0,00470
epd4	0,00405	0,00379
epd5	0,00467	0,00483
epd6	0,00410	0,00281
epd7	0,00405	0,00420
epd8	0,00547	0,00511
epd9	0,00588	0,00470
epd10	0,00737	0,00673
epd11	0,00424	0,00335
as1	0,00717	0,00680
as2	0,01081	0,01018
apd1	0,00460	0,00694
apd2	0,00474	0,00420
apd3	0,01211	0,00862
apd4	0,01394	0,00767
apd5	0,01073	0,00913
apd6	0,00695	0,00770
apd7	0,00864	0,00772
apd8	0,01103	0,00880
apd9	0,00801	0,00893
apd10	0,00597	0,00671
apd11	0,00619	0,00865
apd12	0,01179	0,00715

Tabel B.11: Waarden voor metriek dynamiek, ingespeeld door drummer 1 met en zonder metronoom

	Met metronoom		Zonder metronoom	
Studie	**Gemiddelde**	**Standaardafwijking**	**Gemiddelde**	**Standaardafwijking**
es1	105,443	23,976	115,080	18,944
epd1	85,274	28,313	90,840	16,782
epd2	86,733	23,060	86,414	14,885
epd3	93,950	16,768	93,720	28,429
epd4	83,291	23,254	92,040	21,507
epd5	106,667	19,973	108,474	19,165
epd6	114,404	16,851	109,600	21,126
epd7	105,534	19,591	112,670	15,577
epd8	88,171	40,356	106,477	30,210
epd9	113,599	26,428	116,152	19,980
epd10	98,386	21,520	93,300	20,908
epd11	119,283	15,384	113,822	27,989

Tabel B.12: Waarden voor metriek dynamiek, ingespeeld door drummer 2 met en zonder metronoom

	Met metronoom		Zonder metronoom	
Studie	**Gemiddelde**	**Standaardafwijking**	**Gemiddelde**	**Standaardafwijking**
es1	99,369	22,607	100,725	22,934
es2	97,089	25,772	95,654	24,892
epd1	101,960	23,180	102,780	23,214
epd2	105,260	22,137	106,140	21,962
epd3	99,970	22,943	103,31	21,752
epd4	93,820	23,313	98,770	21,878
epd5	108,365	21,241	108,417	19,907
epd6	105,019	21,700	101,856	23,752
epd7	106,375	22,581	104,807	20,233
epd8	96,938	32,583	98,348	31,142
epd9	105,779	18,216	102,221	19,727
epd10	104,300	18,888	96,270	22,945
epd11	110,219	20,409	105,146	21,158
as1	89,333	29,387	89,179	30,212
as2	78,758	37,126	81,737	38,577
apd1	104,500	20,331	108,655	19,821
apd2	103,430	26,071	104,969	26,051
apd3	98,951	20,101	100,210	19,603
apd4	96,478	23,384	104,523	17,057
apd5	88,106	21,274	92,579	23,456
apd6	98,472	19,405	108,576	18,224
apd7	97,956	28,490	95,664	28,474
apd8	113,853	13,363	111,265	13,370
apd9	97,948	23,654	99,999	24,685
apd10	94,435	15,468	95,218	17,376
apd11	111,558	20,466	105,909	23,854
apd12	101,611	24,997	101,927	23,122

Bijlage C

Het project XML formaat

Om alle instellingen van het model van de applicatie op te slagen is het project XML formaat ontwikkeld.

Het wortelelement van dit formaat is het element *project*. De attributen van dit element houden het tempo, de signatuur en het gebruik van de clicktrack bij.

Wanneer een audio- en/of MIDI-bestand werd ingeladen, worden hun (absolute of relatieve) padnamen weggeschreven in de optionele kindelementen *audiofile* en/of *midifile* van het optionele *files* element.

Het geladen instrumentbestand komt terecht in het optionele element *instrument*.

De instellingen van de *beat tracker* en de quantizer worden opgeslagen in de attributen van respectievelijk het *beattracker* en *quantizer* element. Van de beat tracker worden volgende instellingen opgeslagen:

- **sensitivity:** De gevoeligheid van de beat tracker. Waarde in het interval [0, 1].
- **windowsize:** De grootte van het initiële venster, in seconden.
- **agenttimeout:** Het aantal beats dat een agent mag missen alvorens hij als *timed-out* gemarkeerd wordt.
- **maxagents:** Het maximaal aantal agents dat gebruikt wordt tijdens het beat trackingalgoritme.
- **hypotheses:** Het aantal tempohypotheses dat gebruikt wordt om in de initialisatiefase van het algoritme de agents aan te maken.

Van de quantizer worden volgende instellingen opgeslagen:

- **type:** Het type quantizer dat gebruikt wordt. Dit is steeds gelijk aan *GridQuantizer* aangezien dit de enige aanwezige implementatie is.
- **clicktrack:** Indien *true* wordt het vaste tempo van de clicktrack gebruikt om de grid te definiëren, anders de beats die door de beat tracker gevonden werden.

De grootte van het quantum kan worden opgegeven in de vorm van een verdeling van het interbeatinterval op meerdere niveaus. Dit gebeurt in het optionele *beatsubdivision* element dat minstens één kindelement *divisioncomponent* heeft. In het geval van de gridquantizer worden alle *divisioncomponents* met elkaar vermenigvuldigd om de onderverdeling van het interbeatinterval te bekomen. Een intelligentere quantizer kan deze niveaus op een andere manier interpreteren.

De modules die in de applicatie geladen zijn worden opgeslagen in het optionele *modules* element. Voor elke geladen module wordt een *module* kindelement toegevoegd, de naam van de geladen module komt in het attribuut *name* van dit kindelement terecht.

Tenslotte worden de geladen renderers opgeslagen in het optionele *renderers* element. Dit element kan kindelementen van het type *noterenderer* of *zonerenderer* bevatten afhankelijk van het type van de geladen renderers. Voor elk renderertype wordt de naam van de renderer in het *name* attribuut opgeslagen. Bij een NoteRenderer wordt het gebruikte NoteRendererData object opgeslagen in de vorm van de index waarop dit object in het model voorkomt. Bij een ZoneRenderer wordt de lijst van horizontale coordinaten opgeslagen door de index van een VisualizationData-object met dezelfde horizontale coördinaten op te geven, samen met de index van de module waarin dit VisualizationData object voorkomt.

De instellingen van de parameters van de renderers komen terecht in de kindelementen *parameter* van de rendererelementen. Deze parameters worden in volgorde dat ze aanwezig zijn in de renderers opgeslagen en hebben als attributen de defaultwaarde van de parameter, de factor en de offset. Optioneel kan een *parameter* element een *visualizationdata* kindelement hebben, dat aangeeft welk VisualizationData object gekoppeld is aan deze parameter door het in de attributen vermelden van de index van de module en de index van het data-object in die module.

De getoonde grafieken kunnen niet worden opgeslagen, aangezien zij niet door het model maar door de presentatielaag beheerd worden.

C.1 Voorbeeldbestand

Het bestand dat gegeven wordt door listing C.1 is een voorbeeld van een project XML bestand. In het project zijn 4 modules geladen en zijn er 2 renderers aanwezig.

De NoteRenderer tekent alle aanslagen van de performance en heeft het *velocity* VisualizationData-object van de *Dynamics* module gekoppeld aan de hoogte en de relatieve breedte van de symbolen. Hierdoor zal de grootte van de symbolen proportioneel zijn met de velocity van de aanslagen.

De ZoneRenderer gebruikt de horizontale coördinaten van het *Total spread* VisualizationData-object van de *Tightness* module. Ditzelfde data-object is gekoppeld aan de parameter die de absolute breedte van de zones instelt. Hierdoor zullen de zones tussen de eerste en laatste aanslag van een groepje aanslagen die naar hetzelfde tijdstip kwantiseren gekleurd worden.

Listing C.1: Een voorbeeld project XML bestand

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE project SYSTEM "project.dtd">
<project tempo="90" meter="4/4" clicktrack="true">
  <files>
    <audiofile>testaudio.wav</audiofile>
    <midifile>testmidi.mid</midifile>
  </files>
  <instrument>D:\testinstrument.xml</instrument>
  <beattracker sensitivity="0.5" windowsize="5.0"
               agenttimeout="8" maxagents="10" hypotheses="10" />
  <quantizer type="GridQuantizer" clicktrack="true">
    <beatsubdivision>
      <divisioncomponent>2</divisioncomponent>
      <divisioncomponent>2</divisioncomponent>
    </beatsubdivision>
  </quantizer>
  <modules>
    <module name="Tempo" />
    <module name="Timing" />
    <module name="Dynamics" />
    <module name="Tightness" />
```

```
  </modules>
  <renderers>
    <noterenderer name="Onsets" data-index="0">
      <parameter default="0.0" factor="1.0" offset="0.0" />
      <parameter default="1.0" factor="1.0" offset="0.0" />
      <parameter default="1.0" factor="1.0" offset="0.0" />
      <parameter default="1.0" factor="1.0" offset="0.0" />
      <parameter default="1.0" factor="1.0" offset="0.0">
        <visualizationdata module-index="2" data-index="0" />
      </parameter>
      <parameter default="0.0" factor="1.0" offset="0.0" />
      <parameter default="1.0" factor="1.0" offset="0.0">
        <visualizationdata module-index="2" data-index="0" />
      </parameter>
      <parameter default="0.0" factor="1.0" offset="0.0" />
      <parameter default="0.0" factor="1.0" offset="0.0" />
    </noterenderer>
    <zonerenderer name="Tightness - Total spread"
                  module-index="3" data-index="0">
      <parameter default="0.0" factor="1.0" offset="0.0">
        <visualizationdata module-index="3" data-index="0" />
      </parameter>
      <parameter default="0.0" factor="1.0" offset="0.0" />
      <parameter default="1.0" factor="1.0" offset="0.0" />
      <parameter default="1.0" factor="1.0" offset="0.0" />
      <parameter default="0.5" factor="1.0" offset="0.0" />
    </zonerenderer>
  </renderers>
</project>
```

C.2 Document Type Definition

Een geldig project XML bestand moet voldoen aan de DTD die gegeven word in listing C.2.

Listing C.2: Document Type Definition voor een project XML bestand

```
<!ELEMENT project (files?, instrument?, beattracker,
                   quantizer, modules?, renderers?)>
<!ELEMENT files (audiofile?, midifile?)>
<!ELEMENT audiofile (#PCDATA)>
<!ELEMENT midifile (#PCDATA)>
<!ELEMENT instrument (#PCDATA)>
<!ELEMENT beattracker EMPTY>
<!ELEMENT quantizer (beatsubdivision?)>
<!ELEMENT beatsubdivision (divisioncomponent+)>
<!ELEMENT divisioncomponent (#PCDATA)>
<!ELEMENT modules (module+)>
<!ELEMENT module EMPTY>
<!ELEMENT renderers (noterenderer, zonerenderer)+>
<!ELEMENT noterenderer (parameter*)>
<!ELEMENT zonerenderer (parameter*)>
<!ELEMENT parameter (visualizationdata?)>
<!ELEMENT visualizationdata EMPTY>
```

```
<!ATTLIST project tempo NMTOKEN #REQUIRED
                 meter CDATA #REQUIRED
                 clicktrack NMTOKEN #REQUIRED>
<!ATTLIST beattracker sensitivity NMTOKEN #REQUIRED
                      windowsize NMTOKEN #REQUIRED
                      agenttimeout NMTOKEN #REQUIRED
                      maxagents NMTOKEN #REQUIRED
                      hypotheses NMTOKEN #REQUIRED>
<!ATTLIST quantizer type NMTOKEN #REQUIRED
                    clicktrack NMTOKEN #REQUIRED>
<!ATTLIST module name NMTOKEN #REQUIRED>
<!ATTLIST noterenderer name CDATA #REQUIRED
                       data-index NMTOKEN #REQUIRED>
<!ATTLIST zonerenderer name CDATA #REQUIRED
                       module-index NMTOKEN #REQUIRED
                       data-index NMTOKEN #REQUIRED>
<!ATTLIST parameter default NMTOKEN #REQUIRED
                    factor NMTOKEN #REQUIRED
                    offset NMTOKEN #REQUIRED>
<!ATTLIST visualizationdata module-index NMTOKEN #REQUIRED
                            data-index NMTOKEN #REQUIRED>
```

Bijlage D

Het instrument XML formaat

Het instrument XML formaat is ontworpen om nootnummers af te beelden op verschillende delen van een instrument. Bij een piano zijn de delen van het instrument de verschillende toetsen, bijgevolg bestaat er een één-op-één relatie tussen de nootnummers en de delen van het instrument. Bij een elektronisch drumstel hebben de verschillende delen van het drumstel vaak verschillende *trigger zones* die elk een verschillend nootnummer gebruiken, bijgevolg zijn er meerdere nootnummers die naar eenzelfde deel van het instrument kunnen verwijzen.

Het wortelelement van het instrument XML formaat is het *instrument* element. Dit element heeft één attribuut *name* dat de naam van het instrument bevat.

Voor elk deel van het instrument dient een *part* element aanwezig te zijn, waarbij in het *name* attribuut de naam van het deel van het instrument wordt ingevuld.

Een *part* element heeft *notenumber* elementen als kinderen. Elk *notenumber* element geeft een nootnummer aan dat op dit deel van het instrument wordt afgebeeld.

D.1 Voorbeeldbestand

Het bestand dat gegeven wordt in listing D.1 bevat de afbeelding van de verschillende nootnummers op de verschillende delen van een elektronisch drumstel van Roland.

Listing D.1: Een voorbeeld instrument XML bestand

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE instrument SYSTEM "instrument.dtd">
<instrument name="Roland TD9-KX2">
  <part name="Crash">
    <notenumber>49</notenumber>
    <notenumber>55</notenumber>
    <notenumber>57</notenumber>
    <notenumber>52</notenumber>
  </part>
  <part name="Ride">
    <notenumber>53</notenumber>
    <notenumber>51</notenumber>
    <notenumber>59</notenumber>
  </part>
  <part name="Hi-Hat">
    <notenumber>46</notenumber>
    <notenumber>26</notenumber>
```

```
    <notenumber>44</notenumber>
  </part>
  <part name="Tom 1">
    <notenumber>48</notenumber>
    <notenumber>50</notenumber>
  </part>
  <part name="Tom 2">
    <notenumber>45</notenumber>
    <notenumber>47</notenumber>
  </part>
  <part name="Tom 3">
    <notenumber>43</notenumber>
    <notenumber>58</notenumber>
  </part>
  <part name="Snare">
    <notenumber>38</notenumber>
    <notenumber>40</notenumber>
  </part>
  <part name="Kick">
    <notenumber>36</notenumber>
  </part>
</instrument>
```

D.2 Document Type Definition

Een geldig project XML bestand moet voldoen aan de DTD die gegeven word in listing D.2.

Listing D.2: Document Type Definition voor een instrument XML bestand

```
<!ELEMENT instrument (part*)>
<!ELEMENT part (notenumber*)>
<!ELEMENT notenumber (#PCDATA)>
<!ATTLIST instrument name CDATA #REQUIRED>
<!ATTLIST part name CDATA #REQUIRED>
```

Bijlage E

Gebruikershandleiding

E.1 Introductie

DrumPerformanceAnalysis is een programma dat bestemd is voor muzikanten. Het programma is in staat om een muzikale performance in MIDI-formaat in te laden en om vervolgens verschillende aspecten van de performance te analyseren en te visualiseren. Deze handleiding behandelt de belangrijkste functies van het programma en schetst enkele scenario's die nuttig kunnen zijn voor de gebruiker. Tot slot worden enkele mogelijke problemen behandeld. Deze handleiding is niet volledig en moedigt de gebruiker aan om zelf te experimenteren met het programma.

E.2 Systeemvereisten

Om dit programma uit te voeren is een computer nodig met een besturingssysteem van de *Microsoft Windows* familie.

E.3 Het programma opstarten

Het programma opstarten doe je als volgt:

1. Open *Windows Verkenner*.
2. Blader in de bestandsstructuur naar de map waar het programma zich bevindt.
3. Dubbelklik op *DrumPerformanceAnalysis.exe*

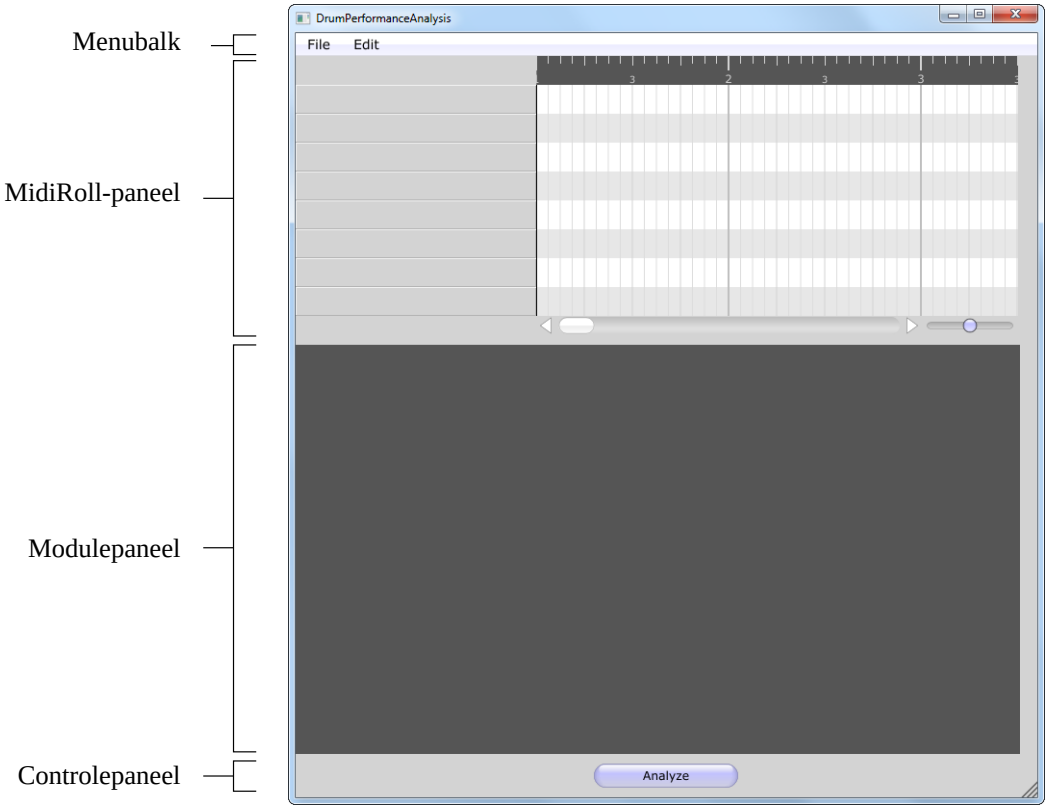
E.4 Rondleiding

Wanneer je het programma opstart krijg je het hoofdscherm te zien (figuur E.1). Dit scherm bestaat uit 4 onderdelen:

- De menubalk (§E.4.1).
- Het MidiRoll-paneel (§E.4.2).
- Het modulepaneel (§E.4.3).
- Het controlepaneel (§E.4.4).

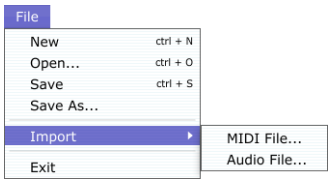
E.4.1 De menubalk

De menubalk bevat twee menu's met acties die het programma kan uitvoeren: het *File*-menu en het *Edit-menu*.



Figuur E.1: Het hoofdscherm van de applicatie

Het File-menu



Figuur E.2: Het *File*-menu

Het *File*-menu (figuur E.2) bevat acties om bestanden in te laden, op te slagen of om het programma af te sluiten:

- `New`: Maak een nieuw bestand aan. Hierbij worden alle instellingen van het programma teruggezet naar de standaardwaarden.
- `Open...`: Open een bestand en laad alle instellingen uit dit bestand (zie §E.5.10). Hiervoor krijg je een dialoogvenster te zien waarmee je kan bladeren naar het bestand dat je wenst in te laden.
- `Save`: Sla alle instellingen van het programma op in het huidig geladen bestand (zie §E.5.10). Indien je nog geen bestand geladen had krijg je de mogelijkheid om het bestand op te geven waarin je de gegevens wil opslaan.
- `Save As...`: Sla alle instellingen van het programma op in een nieuw bestand (zie §E.5.10). Hiervoor krijg je de mogelijkheid om het bestand op te geven waarin je de gegevens wil opslaan.

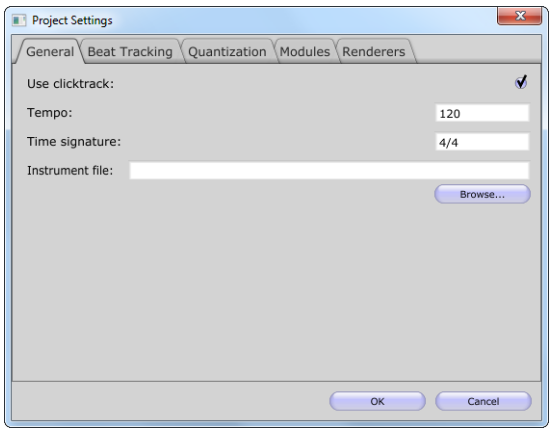
- Het `Import` menu:
  - `MIDI File...`: Laad een nieuwe performance in MIDI-formaat (zie §E.5.1). Indien er reeds een performance geladen is zal je gewaarschuwd worden dat de nieuwe performance de oude zal vervangen.
  - `Audio File...`: Laad een nieuwe performance in een digitaal audioformaat. Indien er reeds een performance geladen is zal je gewaarschuwd worden dat de nieuwe performance de oude zal vervangen.
- `Exit`: Sluit het programma af (zie §E.5.11). Indien er instellingen gewijzigd zijn sinds de laatste keer dat je hebt opgeslagen zal je nog de kans krijgen om deze wijzigingen alsnog op te slagen alvorens het programma definitief afsluit.

Het Edit-menu

Figuur E.3: Het *Edit*-menu

Het *Edit*-menu (figuur E.3) bevat één item `Project settings...` dat een venster opent waar allerhande instellingen van het project geconfigureerd kunnen worden (figuur E.4). De instellingen zijn verdeeld in vijf categorieën die elk een apart tabblad toegewezen gekregen hebben:

- *General*: Algemene instellingen zoals tempo, maatsoort, instrument. Zie §E.5.2 en §E.5.3.
- *Beat Tracking*: Instellingen voor de *beat tracker*. Zie §E.5.4.
- *Quantization*: Instellingen voor kwantisatie. Zie §E.5.5.
- *Modules*: Instellingen met betrekking tot de modules. Zie §E.4.3 en §E.5.6.
- *Renderers*: Instellingen voor de visualisatie. Zie §E.5.7.



Figuur E.4: Het *Project Settings* venster

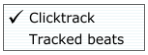
E.4.2 Het MidiRoll-paneel

De hoofdfunctie van het *MidiRoll*-paneel is het weergeven van de performance. Bij verstek worden de aanslagen van de performance weergegeven als rode vierkante symbolen. Het paneel bestaat hiervoor uit drie componenten: de tijdlijn, het instrumentpaneel en het nootpaneel.

De tijdlijn

Figuur E.5: De tijdlijn

De tijdlijn (figuur E.5) is de donkergrijze balk bovenaan het *MidiRoll*-paneel. Hierop wordt de tijd weergegeven als schaal van de horizontale as. De tijd wordt voorgesteld in muzikale eenheden: maten, beats,... Wanneer de performance is ingespeeld met een metronoom (zie §E.5.3) wordt bij verstek de tijdschaal van de metronoom weergegeven. Wanneer de performance zonder metronoom gespeeld is, wordt de tijdschaal aangepast aan de beats die door de *beat tracker* gevonden werden. Om de tijdschaal aan te passen aan de beats van de *beat tracker* wanneer met een metronoom gespeeld is, kan je met de rechtermuisknop klikken op de tijdlijn en vervolgens in het menu `Tracked beats` selecteren (figuur E.6).



Figuur E.6: De schaal van de tijdlijn aanpassen

Het instrumentpaneel

(a) Drumstel **(b)** Piano

Figuur E.7: Het instrumentpaneel

Volledig links in het *MidiRoll*-paneel bevindt zich het instrumentpaneel. Dit paneel dient als legende voor het nootpaneel (zie verder). In het instrumentpaneel worden de verschillende delen van het instrument getoond. Het geeft voor elke horizontale balk in het nootpaneel aan op welk deel van het instrument de aanslagen gespeeld werden die in die balk worden weergegeven. Figuur E.7 toont hoe deze component er kan uitzien voor een drumstel of voor een piano.

Het nootpaneel

De centrale component in het *MidiRoll*-paneel is het nootpaneel (figuur E.8). Op dit paneel worden de aanslagen van de performance bij verstek gevisualiseerd door symbolen. Daarnaast kunnen hierop ook andere resultaten van de analyse gevisualiseerd worden (zie §E.5.7). Dit paneel bestaat uit horizontale balken die gelijk lopen met de delen van het instrument in het instrumentpaneel. Op elke balk worden de aanslagen op het corresponderende deel van het instrument gevisualiseerd door middel van symbolen op het juiste tijdstip.

Zoomen

Om de timing van de aanslagen in detail te kunnen inspecteren of om meer overzicht over de performance te krijgen is het mogelijk om in- en uit te zoomen op de horizontale as van het *MidiRoll*-paneel. Dit kan op twee manieren:

Figuur E.8: Het nootpaneel

- Via de schuif rechtsonderaan het *MidiRoll*-paneel: door de schuif naar rechts of naar links te bewegen kan je respectievelijk in- of uitzoomen rondom het midden van het paneel.
- Via de tijdlijn: Wanneer je met de muis op een punt in de tijdlijn klikt plaats je de cursor op dat punt. Wanneer je vervolgens de muis naar onder of naar boven sleept kan je respectievelijk in- of uitzoomen rondom de cursor.

E.4.3 Het modulepaneel

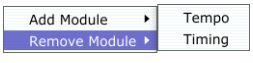
Om verschillende aspecten van de performance te analyseren werkt *DrumPerformanceAnalysis* met modules. Elke module verzorgt een specifiek aspect van de performance. Deze modules kunnen ingeladen worden (zie §E.5.6) en worden weergegeven in het modulepaneel (figuur E.9).

Figuur E.9: Het modulepaneel

Elke module beschikt over een knop `Visualize` en een knop `Graphs`. Wanneer je op de knop `Visualize` klikt komt een menu tevoorschijn waarmee je gegevens van de module kan koppelen aan parameters van *renderers* (zie §E.5.7). Wanneer je op de knop `Graphs` klikt komt een menu tevoorschijn waarin je kan selecteren welke grafieken worden weergegeven (zie §E.5.8).

Wanneer je met de rechtermuisknop in het donkergrijze gebied van het modulepaneel klikt verschijnt een menu waarmee modules kunnen worden ingeladen of verwijderd (figuur E.10).

(a) Inladen



(b) Verwijderen

Figuur E.10: Menu's voor inladen en verwijderen van modules

E.4.4 Het controlepaneel

Op het controlepaneel (figuur E.11) bevindt zich de knop `Analyze`. Wanneer je op deze knop klikt wordt de performance geanalyseerd.

Figuur E.11: Het controlepaneel

E.5 Aan de slag

In deze sectie vind je terug hoe je de basishandelingen van het programma kan uitvoeren. Door de uitgebreide configuratiemogelijkheden zijn echter veel scenario's mogelijk, die niet allemaal aan bod kunnen komen. Sectie E.6 behandelt enkele praktische voorbeelden en geeft een aanzet om zelf verder te experimenteren.

E.5.1 Een MIDI-bestand inladen

Een performance in MIDI-formaat inladen doe je als volgt:

1. Klik in de menubalk op `File` → `Import` → `MIDI File...`
2. Blader in het dialoogvenster naar de map waarin het MIDI-bestand zich bevindt.
3. Selecteer het MIDI-bestand.
4. Klik op `Open`.

Wanneer er reeds een MIDI-bestand geladen is krijg je hiervan een melding (figuur E.12). Klik op `OK` om door te gaan of op `Cancel` om de operatie af te breken.

Figuur E.12: Melding dat er reeds een MIDI-bestand geladen is

E.5.2 Een instrument inladen

Om ervoor te zorgen dat de performance correct wordt weergegeven in het *MidiRoll*-paneel (zie §E.4.2) en dat ze correct geanalyseerd kan worden dien je het juiste instrumentbestand te laden.

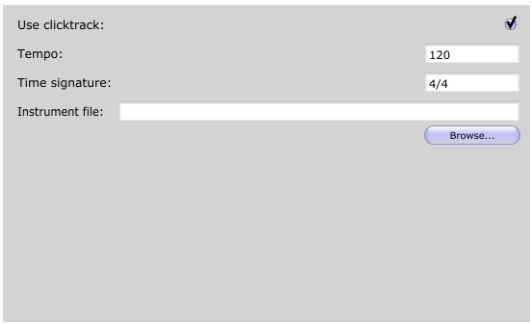
1. Klik in de menubalk op `Edit` → `Project settings...`
2. Selecteer in het *Project Settings* venster het tabblad *General* (figuur E.13).
3. Vul in het tekstveld `Instrument file` de absolute padnaam van het instrumentbestand in.

   OF

   (a) Klik op de knop `Browse...`
   (b) Blader in het dialoogvenster naar de map waar het instrumentbestand zich bevindt.
   (c) Selecteer het instrumentbestand.
   (d) Klik op `Open`.
4. Klik op `OK` om de instellingen op te slagen en het nieuwe instrumentbestand in te laden.

E.5.3 Het tempo en de maatsoort instellen

Om te zorgen dat de maten en de beats correct worden weergegeven op de tijdlijn van het *MidiRoll*-paneel (zie E.4.2) dien je het tempo, de maatsoort en het gebruik van de metronoom in te stellen in het *General* tabblad *Project Settings* venster (figuur E.13).



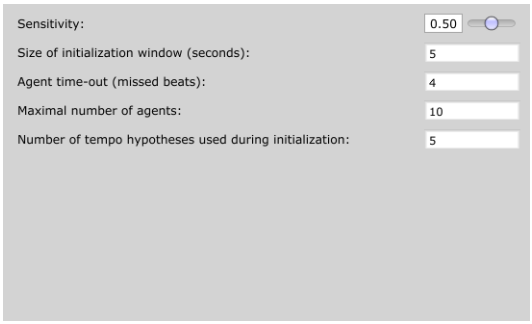
Figuur E.13: Het *General* tabblad van het *Project Settings* venster

1. Klik in de menubalk op `Edit` → `Project settings...`
2. Selecteer in het *Project Settings* venster het tabblad *General* (figuur E.13).
3. `Use clicktrack`: Aanvinken indien de performance werd ingespeeld met een metronoom. Indien niet verwijder je het vinkje.
4. `Tempo`: Het tempo van de metronoom in beats per minuut indien de performance werd ingespeeld met een metronoom.
   Indien de performance zonder metronoom werd ingespeeld wordt het opgegeven tempo gebruikt als vertrekpunt voor de *beat tracker* (zie §E.5.4).
   Indien je de waarde 0 opgeeft zal de *beat tracker* zelf het initiële tempo bepalen.
5. `Time signature`: Hier kan je de maatsoort van de performance opgeven. Je kan deze opgeven als een eenvoudige breuk zoals $\frac{4}{4}$ of $\frac{6}{8}$.
   Om aan te geven waar de *zware* beats vallen kan je de maatsoort ook in het volgende formaat opgeven: $\frac{x+y+z}{u}$, waarbij de som van x, y en z het aantal beats in een maat voorstelt, en u de nootwaarde van één beat aangeeft. Zo kan je de maatsoort $\frac{6}{8}$ opgeven als $\frac{3+3}{8}$, dit geef je in als `(3+3)/8`.
6. Klik op `OK` om de instellingen op te slagen.

E.5.4 De beat tracker configureren

In het tabblad *Beat Tracking* van het *Project Settings* venster (figuur E.14) kan je verschillende parameters van de *beat tracker* instellen.

1. Klik in de menubalk op `Edit` → `Project settings...`
2. Selecteer in het *Project Settings* venster het tabblad *Beat Tracking* (figuur E.14).
3. `Sensitivity`: De gevoeligheid waarmee de *beat tracker* zich aanpast aan tempowijzigingen. Dit is een getal in het interval $[0, 1]$. Wanneer de waarde 1 is zal de *beat tracker* heel gevoelig zijn voor tempowijzigingen, maar wellicht ook voor fouten. Wanneer de waarde 0 is zal de *beat tracker* zich bijna als een metronoom gedragen.
4. `Size of initialization window (in seconds)`: De grootte van het venster dat gebruikt wordt tijdens de initialisatiefase.



Figuur E.14: Het *Beat Tracking* tabblad van het *Project Settings* venster

5. `Agent time-out (missed beats)`: Het aantal beats dat een agent mag missen alvorens als *timed out* gemarkeerd te worden.
6. `Maximal number of agents`: Het maximaal aantal agents dat actief mag zijn telkens een nieuwe ritmische gebeurtenis behandeld wordt.
7. `Number of tempo hypotheses used during initialization`: Het aantal tempohypotheses dat gebruikt wordt tijdens de initialisatiefase, enkel de hypoteses met de hoogste scores worden gebruikt.
8. Klik op `OK` om de instellingen op te slagen.

E.5.5 De quantizer configureren

De configuratie van de *quantizer* gebeurt in het tabblad *Quantization* van het *Project Settings* venster (figuur E.15).

Figuur E.15: Het *Quantization* tabblad van het *Project Settings* venster

De grootte van het kwantum kan je instellen door op te geven in hoeveel delen de beats moeten worden onderverdeeld. Stel dat een beat gelijk is aan een kwartnoot en de kortste noot in de performance is een zestiendenoot, dan geef je 4 op als onderverdeling.

Met het oog op complexere *quantizers* kan je deze onderverdeling opgeven op meerdere niveaus, in het gegeven voorbeeld zou je dus 2 keer de beat verdelen in 2 om zowel achtste als zestiende noten te kwantiseren. Aangezien echter een eenvoudige *gridquantizer* gebruikt wordt is deze functionaliteit overbodig.

1. Klik in de menubalk op `Edit` → `Project settings...`
2. Selecteer in het *Project Settings* venster het tabblad *Quantization* (figuur E.15).

3. `Quantize to clicktrack`: Aanvinken om te kwantiseren ten opzichte van een rooster dat gebaseerd is op het tempo van de metronoom. Verwijderen om te kwantiseren naar een rooster gebaseerd op de beats die gevonden werden door de *beat tracker*.
4. Voor elk niveau waarop de beats worden onderverdeeld:
   (a) Vul in het teksveld de waarde in waardoor de beats gedeeld worden.
   (b) Klik op `Add`
5. Om een niveau van onderverdeling te verwijderen:
   (a) Selecteer de te verwijderen waarde in de lijst.
   (b) Klik op `Remove`
6. Klik op `OK` om de instellingen op te slagen.

E.5.6 Modules inladen en verwijderen

Het inladen en verwijderen van modules kan zowel in het modulepaneel van het hoofdvenster van de applicatie (zie §E.4.3) als in het *Project Settings* venster gebeuren.

Via het modulepaneel

Om een module toe te voegen ga je als volgt te werk:

1. Klik met de rechtermuisknop in het donkergrijze gebied van het modulepaneel.
2. Selecteer `Add Module`.
3. Selecteer de module die je wenst toe te voegen.

Om een module te verwijderen ga je als volgt te werk:

1. Klik met de rechtermuisknop in het donkergrijze gebied van het modulepaneel.
2. Selecteer `Remove Module`.
3. Selecteer de module die je wenst te verwijderen.

Via het Project Settings venster

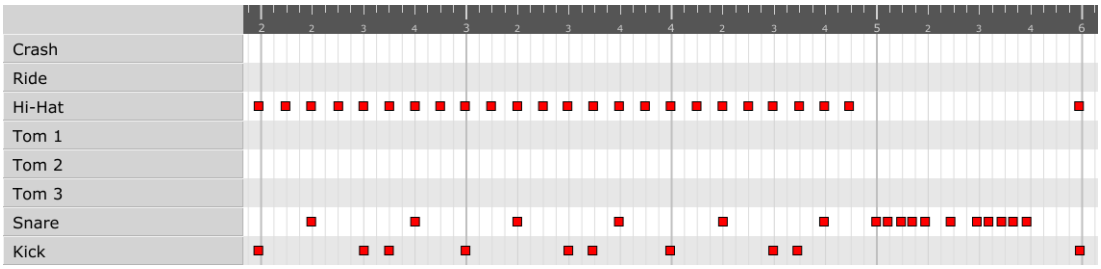
1. Klik in de menubalk op `Edit` → `Project settings...`
2. Selecteer in het *Project Settings* venster het tabblad *Modules* (figuur E.16).
3. Een module toevoegen:
   (a) Selecteer de module die je wenst toe te voegen in de lijst `Available Modules`.
   (b) Klik op `Add ->`.
4. Een module verwijderen:
   (a) Selecteer de module die je wenst te verwijderen in de lijst `Loaded Modules`.
   (b) Klik op `<- Remove`.
5. De volgorde van de modules wijzigen:
   (a) Selecteer de module die je wil verplaatsen in de lijst `Loaded Modules`.
   (b) Klik op `Move Up` om de module één plaats naar boven op te schuiven.
   (c) Klik op `Move Down` om de module één plaats naar onder op te schuiven.
6. Klik op `OK` om de geselecteerde modules in de juiste volgorde in te laden.

Figuur E.16: Het *Modules* tabblad van het *Project Settings* venster

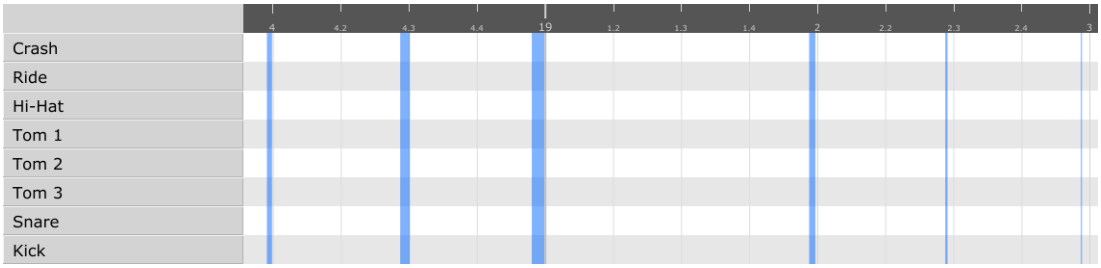
E.5.7 Gegevens visualiseren op het MidiRoll-paneel

Visualisatie op het *MidiRoll*-paneel gebeurt door middel van *renderers*. Je kan zowel de performance zelf als verschillende analyseresultaten visualiseren. Hiervoor zijn twee soorten *renderers* beschikbaar:

- *NoteRenderer*: Deze *renderer* gebruik je om aanslagen te visualiseren op het *MidiRoll*-paneel door middel van symbolen. Zie figuur E.17a.
- *ZoneRenderer*: Deze *renderer* kan zones die de volledige hoogte van het nootpaneel overspannen inkleuren. Zie figuur E.17b.



(a) *NoteRenderer*



(b) *ZoneRenderer*

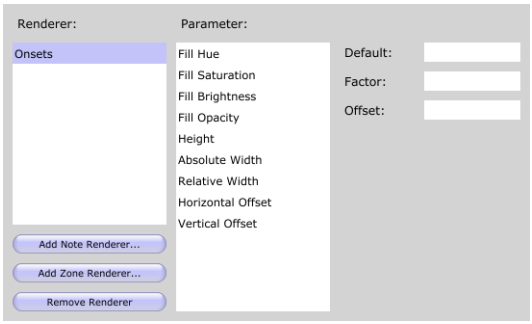
Figuur E.17: Voorbeelden van *renderers*

Elke renderer beschikt over een verzameling parameters waarmee je de eigenschappen van de gevisualiseerde symbolen of zones kan aanpassen. Voor elke parameter kan je de volgende instellingen opgeven:

- *Default*: De standaardwaarde voor elk gevisualiseerd symbool of elke gevisualiseerde zone.
- *Factor*: Een factor waarmee elke waarde vermenigvuldigd wordt.

- *Offset*: Een waarde die bij elke waarde wordt opgeteld, nadat deze eerst vermenigvuldigd werd met de opgegeven factor.

Het configureren van de *renderers* doe je in het *Renderers* tabblad van het *Project Settings* venster (figuur E.18).



Figuur E.18: Het *Renderers* tabblad van het *Project Settings* venster

Een renderer toevoegen of verwijderen

1. Klik in de menubalk op `Edit` → `Project settings...`
2. Selecteer in het *Project Settings* venster het tabblad *Renderers* (figuur E.18).
3. **Een *NoteRenderer* toevoegen:**
   (a) Klik op `Add Note Renderer...`
   (b) Het dialoogvenster *Add Note Renderer* verschijnt (figuur E.19a).
   (c) Selecteer de gegevens die je wenst te visualiseren uit de lijst `Data for new renderer`.
   (d) Vul in het tekstveld `Name for new renderer` een beschrijvende naam voor de nieuwe renderer in.
   (e) Klik op `OK` om de nieuwe *renderer* toe te voegen
4. **Een *ZoneRenderer* toevoegen:**
   (a) Klik op `Add Zone Renderer...`
   (b) Het dialoogvenster *Add Zone Renderer* verschijnt (figuur E.19b).
   (c) Selecteer de gegevens die je wenst te visualiseren uit de lijst `Data for new renderer`.
   (d) Vul in het tekstveld `Name for new renderer` een beschrijvende naam voor de nieuwe renderer in.
   (e) Klik op `OK` om de nieuwe *renderer* toe te voegen
5. **Een *renderer* verwijderen:**
   (a) Selecteer de *renderer* die je wenst te verwijderen in de lijst `Renderer`.
   (b) Klik op `Remove Renderer`.
6. **De parameters van een renderer configureren:**
   (a) Selecteer de *renderer* waarvan je de parameters wil configureren in de lijst `Renderer`.
   (b) Selecteer de parameter die je wenst te configureren in de lijst `Parameter`.
   (c) Vul de gewenste waarden in in de tekstvelden `Default`, `Factor` en `Offset`, waarbij je steeds afsluit met `<ENTER>`.
7. Klik op `OK` om de gekozen renderers in te laden en de configuratie toe te passen.

(a) *Note renderer* **(b)** *Zone renderer*

Figuur E.19: Dialoogvensters om *renderers* toe te voegen

Gegevens van een module visualiseren via een renderer

De gegevens die berekend worden door de modules kunnen gevisualiseerd worden door deze te koppelen aan een parameter van een *renderer*. Dit zorgt ervoor dat elk symbool of elke ingekleurde zone voor die parameter een waarde weergeeft die overeenkomt met een waarde van die aanslag in de performance of dat specifieke analyseresultaat. Om dit te bereiken ga je als volgt te werk:

1. (Indien nog geen renderer aanwezig is om de gegevens op te visualiseren) Voeg de renderer toe om de gegevens op te visualiseren zoals hierboven beschreven staat. Als gegevensbron selecteer je de gegevens die je wenst te visualiseren.
2. In het hoofdscherm: klik op de knop `Visualize` van de module die de gegevens voorziet die je wenst te visualiseren.
3. Selecteer de gegevens die je wenst te visualiseren.
4. Selecteer de renderer waarmee je de gegevens wil visualiseren.
5. Selecteer de parameter waaraan je de gegevens wil koppelen.

Opmerking: De visualisatie zal pas correct worden weergegeven wanneer de performance geanalyseerd is, zie §E.5.9.

E.5.8 Grafieken weergeven

Sommige modules zijn in staat om gegevens te visualiseren door middel van grafieken.

1. Klik in het hoofdscherm op de knop `Graphs` van de module waarvan je een grafiek wil weergeven.
2. Klik op de naam van de grafiek die je wil weergeven.

Om een grafiek te verwijderen ga je hetzelfde te werk.

Door de scheidingslijn onder elke grafiek te verslepen kan je de grootte van de verschillende grafieken aanpassen. Indien je gedetailleerde waarden van verschillende punten van een grafiek te weten wil komen kan je met de muis over de grafiek bewegen, zodat er een tooltip tevoorschijn komt met meer informatie.

E.5.9 De performance analyseren

Wanneer je op de knop `Analyze` in het controlepaneel (§E.4.4) klikt wordt de performance geanalyseerd. Telkens er wijzigingen gebeuren aan de algemene instellingen (§E.5.3), de instellingen van de *beat tracker* (§E.5.4), de *quantizer* (§E.5.5) of de configuratie van de modules (§E.5.6), dient de performance opnieuw geanalyseerd te worden.

E.5.10 Alle instellingen opslaan of inladen

Alle instellingen van het programma kunnen worden opgeslagen in de vorm van een XML-bestand, zodat je dit later opnieuw kan inladen in plaats van alles opnieuw te moeten configureren.

Opslaan

1. Klik in de menubalk op `File` → `Save`.
2. Indien dit de eerste keer is dat je de instellingen opslaat krijg je een dialoogvenster te zien, indien niet wordt de performance nu opgeslagen en kan je de volgende stappen overslaan.
3. Blader naar de map waar je het bestand wil opslaan.
4. Vul de naam van het bestand in. Vergeet de extensie `.xml` niet toe te voegen.
5. Klik op `Save`.

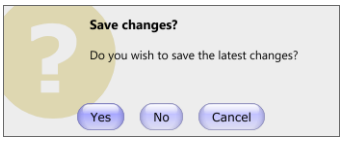
Wanneer je later aangebrachte wijzigingen wil toevoegen aan het opgeslagen bestand klik je in de menubalk op `File` → `Save`. Wanneer je het bestand onder een andere naam wil opslagen klik je in de menubalk op `File` → `Save As...`.

Openen

1. Klik in de menubalk op `File` → `Open...`.
2. Blader naar de map waar het bestand zich bevindt.
3. Selecteer het bestand.
4. Klik op `Open`.

E.5.11 Het programma afsluiten

1. Klik in de menubalk op `File` → `Exit`.
2. Indien de instellingen gewijzigd zijn sinds de laatste keer dat je hebt opgeslagen krijg je hiervan een melding (figuur E.20), en krijg je de kans om de laatste wijzigingen op te slagen.

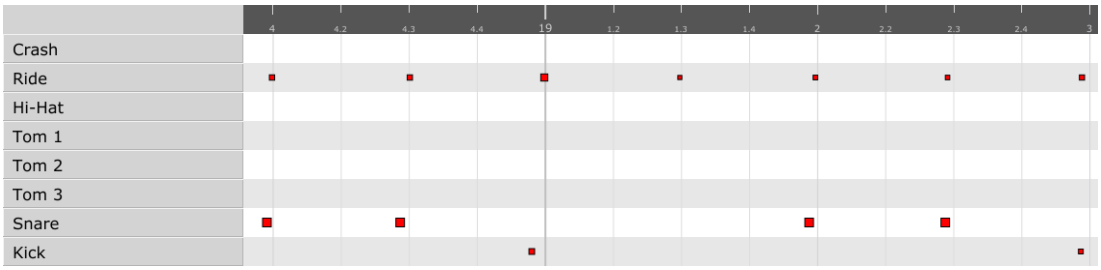


Figuur E.20: Melding dat de instellingen gewijzigd zijn sinds de laatste keer dat opgeslagen werd

E.6 Voorbeelden

Wegens de talrijke configuratiemogelijkheden van de *renderers* behandelt deze sectie enkele mogelijke scenario's die nuttig kunnen zijn voor muzikanten.

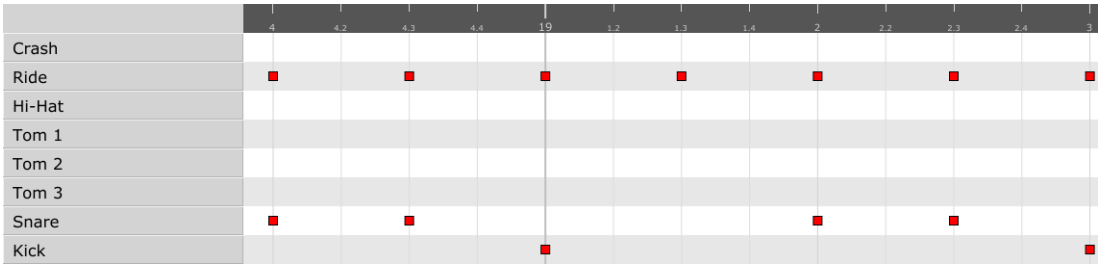
E.6.1 De grootte van de symbolen aanpassen aan de sterkte van de aanslagen



Figuur E.21: Grootte van de symbolen gekoppeld aan de aanslagsterkte

1. Zorg ervoor dat er een *NoteRenderer* genaamd *Onsets* aanwezig is die de aanslagen van de performance (data: *Onsets*) visualiseert (zie §E.5.7).
2. Laad de module *Dynamics* (zie §E.5.6).
3. Klik op de knop `Visualize` van de *Dynamics* module.
4. Selecteer `Velocity` als gegevens die je wenst te visualiseren.
5. Selecteer `Onsets` als de *renderer* waarop je de gegevens wenst te visualiseren.
6. Selecteer `Height` om de aanslagsterkte aan de hoogte van de symbolen te koppelen.
7. Herhaal stappen 3 tot 5.
8. Selecteer `Relative Width` om de aanslagsterkte aan de breedte van de symbolen te koppelen.
9. Zorg ervoor dat de performance geanalyseerd is (zie E.5.9).

E.6.2 De gekwantiseerde performance weergeven

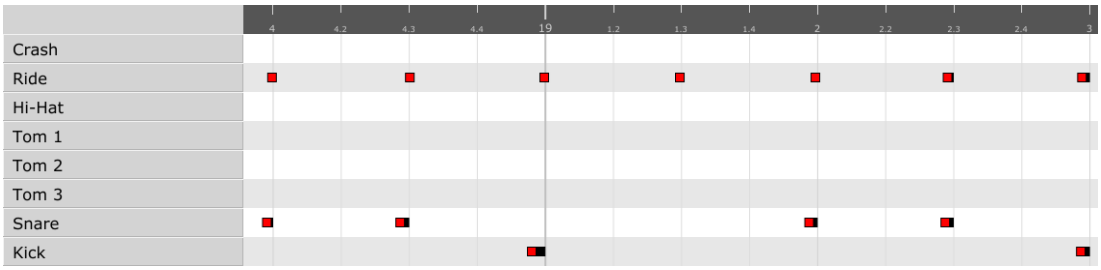


Figuur E.22: De gekwantiseerde performance

- Zorg ervoor dat er een *NoteRenderer* genaamd *Onsets* aanwezig is die de aanslagen van de performance (data: *Onsets*) visualiseert (zie §E.5.7).
- Stel de `Factor` waarde van de parameter `Horizontal Offset` in op -1 (voor de *Onsets-renderer*).
- Laad de module *Timing* (zie §E.5.6).
- Klik op de knop `Visualize` van de *Timing* module.

- Selecteer `Deviation from quantization` als gegevens die je wenst te visualiseren.
- Selecteer `Onsets` als de *renderer* waarop je de gegevens wenst te visualiseren.
- Selecteer `Horizontal Offset` om de timingverschillen aan de horizontale verschuiving van de symbolen te koppelen.
- Zorg ervoor dat de performance geanalyseerd is (zie E.5.9).

E.6.3 De afstand van elke aanslag tot het gekwantiseerd tijdstip weergeven

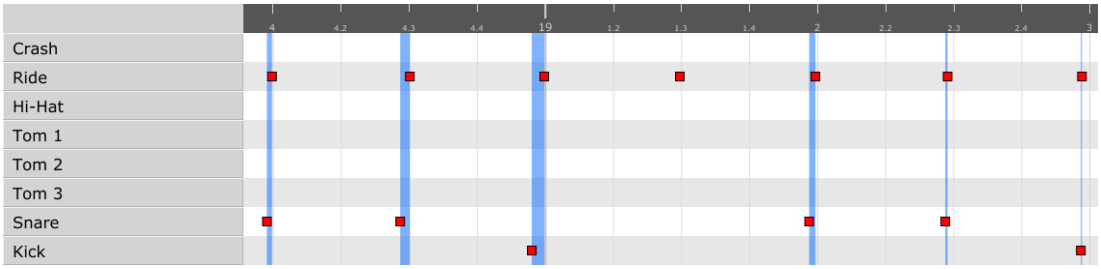


Figuur E.23: De afstand van elke aanslag tot zijn gekwantiseerd tijdstip

- Zorg ervoor dat er een *NoteRenderer* genaamd *Onsets* aanwezig is die de aanslagen van de performance (data: *Onsets*) visualiseert (zie §E.5.7).
- Voeg een nieuwe *NoteRenderer* toe met als gegevens *Onsets* (de aanslagen van de performance) en als naam *Afwijking*.
- Stel de `Factor` waarde van de parameter `Absolute Width` in op -1 (voor de nieuwe renderer).
- Laad de module *Timing* (zie §E.5.6).
- Klik op de knop `Visualize` van de *Timing* module.
- Selecteer `Deviation from quantization` als gegevens die je wenst te visualiseren.
- Selecteer `Afwijking` als de *renderer* waarop je de gegevens wenst te visualiseren.
- Selecteer `Absolute Width` om de timingverschillen aan de breedte van de symbolen te koppelen.
- Zorg ervoor dat de performance geanalyseerd is (zie E.5.9).

E.6.4 De spreiding van alle aanslagen die naar hetzelfde tijdstip kwantiseren weergeven

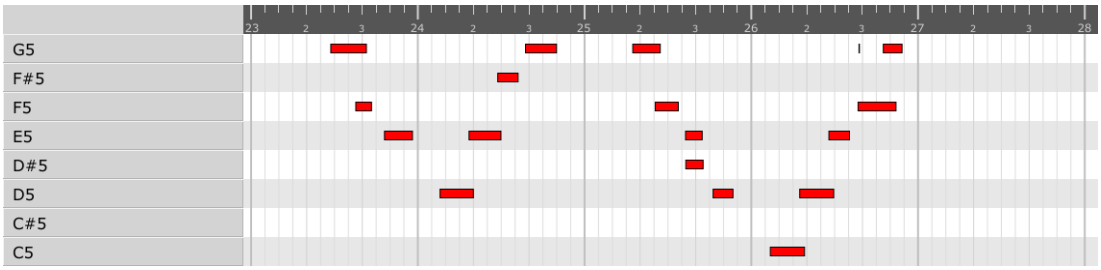
- Zorg ervoor dat er een *NoteRenderer* genaamd *Onsets* aanwezig is die de aanslagen van de performance (data: *Onsets*) visualiseert (zie §E.5.7).
- Laad de module *Tightness* (zie §E.5.6).
- Voeg een nieuwe *ZoneRenderer* toe met als gegevens *Tightness - Total spread* en als naam *Spreiding*.



Figuur E.24: De spreiding van aanslagen die naar hetzelfde tijdstip kwantiseren

- Stel de `Default` waarde van de parameter `Colour Hue` in op 0.6 (voor de nieuwe renderer), zo krijgen de zones een blauwe kleur.
- Klik op de knop `Visualize` van de *Tightness* module.
- Selecteer `Total spread` als gegevens die je wenst te visualiseren.
- Selecteer `Spreiding` als de *renderer* waarop je de gegevens wenst te visualiseren.
- Selecteer `Absolute Width` om de breedte van de zones te koppelen aan de spreiding van de aanslagen.
- Zorg ervoor dat de performance geanalyseerd is (zie E.5.9).

E.6.5 Een performance op een piano weergeven



Figuur E.25: Een performance die werd ingespeeld op een piano

- Zorg ervoor dat er een *NoteRenderer* genaamd *Onsets* aanwezig is die de aanslagen van de performance (data: *Onsets*) visualiseert (zie §E.5.7).
- Laad het piano instrumentbestand (zie §E.5.2).
- Laad de module *Note Length* (zie §E.5.6).
- Klik op de knop `Visualize` van de *Note Length* module.
- Selecteer `Note Length` als gegevens die je wenst te visualiseren.
- Selecteer `Onsets` als de *renderer* waarop je de gegevens wenst te visualiseren.
- Selecteer `Absolute Width` om de lengte van de aanslagen te koppelen aan de breedte van de symbolen.
- Zorg ervoor dat de performance geanalyseerd is (zie E.5.9).

E.7 Probleemoplossing

De noten worden niet gevisualiseerd op het *MidiRoll*-paneel na het inladen van een MIDI-bestand.

- Zorg ervoor dat er een *NoteRenderer* aanwezig is die de aanslagen van de performance (data: *Onsets*) visualiseert (zie §E.5.7).
- Zorg ervoor dat het juiste instrumentbestand geladen is (zie §E.5.1).

De visualisatie wordt niet correct weergegeven (*ZoneRenderers* worden niet weergegeven, koppelen van gegevens aan parameters van een *NoteRenderer* heeft geen resultaat).

- Zorg ervoor dat de performance geanalyseerd is (zie §E.5.9).

De grafieken geven geen gegevens weer.

- Zorg ervoor dat de performance geanalyseerd is (zie §E.5.9).

Na het verwijderen en achteraf opnieuw laden van een module lukt het niet om gegevens aan een *renderer* te koppelen die was aangemaakt voor die module.

- Door het verwijderen van de module is de *renderer* ongeldig geworden. Je dient de oude *renderer* te verwijderen en opnieuw toe te voegen (zie §E.5.7).

Wanneer net een module is toegevoegd via het *Project Settings* venster lukt het niet om een *renderer* voor deze module toe te voegen.

- Klik eerst het *Project Settings* venster weg met `OK` om de nieuwe modules in te laden. Open vervolgens opnieuw het *Project Settings* venster om de *renderer* toe te voegen.

Na het instellen dat een metronoom gebruikt wordt (in het *Project Settings* venster) lukt het niet om aan te duiden dat de performance volgens het tempo van de metronoom gekwantiseerd moet worden.

- Klik eerst het *Project Settings* venster weg met `OK` om de wijzigingen toe te passen. Open vervolgens opnieuw het *Project Settings* venster om aan te duiden dat de performance volgens het tempo van de metronoom gekwantiseerd moet worden.