HOOFDSTUK 3
Een digitale implementatie:"A Book of moves"

HOOFDSTUK 3:

Een digitale implementatie:

"A Book of Moves"

3.0: Doel

In dit hoofdstuk willen we aantonen dat:

1. De mislukkingen waarvan we gewag maakten in ons besluit van hoofdstuk 2, principieel kunnen vermeden worden.

2. Dat het daarvoor noodzakelijk is vanuit de hiervoor beschreven en ontwikkelde ultrasoontechnologie -meer bepaald de analoge komputer- het non-impakt instrument te ontwikkelen als een op zichzelf staand 'user-interface' waarbij het klankopwekkend gedeelte volkomen los komt te staan van de apparatuur die instaat voor de omzetting van de bewegingsparameters in relevante informatie.

3. Dat dergelijk 'user-interface' kan worden gebouwd met deels hardware, deels software.

4. Dat door gebruik te maken van de mogelijkheden van digitale komputers in het user-interface, aan de program-meerbaarheidsvereiste kan worden voldaan.

5. Dat zulk interface met eenvoudige middelen waarover in principe elke muzikus kan beschikken, geprogrammeerd kan worden. Een elementaire kennis van en kundigheid in informatika is daarvoor voldoende.

3.1: Algemeen opzet

3.1.1: Hardware versus Software

Na de ervaringen met de laatste analoge versies van 'Holosound' en de mislukking van de eerste poging tot digitalisering van het non-impakt instrument, beslisten we de boeg helemaal om te gooien en de richting van de informatizering van het projekt in te gaan.

Vanuit elektrotechnisch standpunt gezien komt het hele probleem voor de bouw van ons instrument dan neer op een klassiek probleem van signaalverwerking, waarbij we dan volgende stappen kunnen onderscheiden:

1.- signaal verwerving

Dit is het nivo van de gebruikte transducers. De eerste omzetting dus van het fenomeen waarover we gegevens in elektrische vorm willen (beweging en motoriek) naar die elektrische vorm.

2.- signaal verwerking

Dit is de techniek nodig om het signaal in een bruikbare vorm te krijgen. Deze stap omvat dus in eerste plaats de versterking, linearizering van de deficiënties van de gebruikte transducer en alle stappen nodig om een optimale signaal-ruisverhouding te bekomen.

3.- data-extraktie en data-verwerving

Dit is het geheel van technieken die we dienen toe te passen om uit het signaal informatie (data) af te leiden. In het Engelse vakjargon heet dit 'data-acquisition'.

4.- data verwerking

Dit is het geheel van de technieken van algoritmische aard die we dienen toe te passen om op grond van de verworven informatie, nieuwe informatie te genereren in de meestal meer gebalde vorm waarin we haar uiteindelijk nodig hebben.

5.- data output

Dit is het interface dat ervoor zorgt dat de informatie in een technisch geschikt formaat wordt omgezet dat door de aan te sluiten apparatuur kan worden begrepen.

De eerste stap was en is gezet: we beschikken over de nodige transducers en bruikbare relevante signalen.

De tweede stap kon ook als in zekere mate gezet beschouwd worden. Daartoe behoort immers het gehele opzet van de analoge komputer zoals we die in het vorige hoofdstuk uitvoerig hebben beschreven. De vraag of het signaal reeds in voldoende mate verwerkt was om te kunnen worden ingepast in het nieuwe opzet diende uiteraard te worden gesteld. Immers aan een signaal dat uiteindelijk voor rechtstreekse beluistering was bedoeld moeten andere eisen worden gesteld dan aan een signaal dat voor verdere verwerking wordt gebruikt en dus eigenlijk binnen de machine opgesloten blijft.

De laatste stap was in feite ook reeds gezet. We wisten immers dat de output het MIDI-protokol zou dienen te volgen, en een universeel bruikbaar komputer MIDI-interface ontwierpen we reeds in 1984, kort na de publikatie van de officiële technische MIDI-specifikaties (1983). Het was een 'dom' interface dat op eender welke 8-bit Centronics printerpoort van eender welke komputer kon aangesloten worden. De besturing geschiedde op precies dezelfde wijze als die voor een standaard printer. We hebben er geen ogenblik aan getwijfeld dat deze stap in digitale hardware gerealizeerd diende te worden, ondanks het feit dat het ook via software (vandaag) wel haalbaar en mogelijk is.

Volledigheidshalve heb ik het opbouwschema van dit interface -waarvan terloops gezegd zo'n honderd exemplaren in omloop zijn onder diverse musici/komponisten over de gehele wereld evenals onder studenten en oud-studenten van mijn kursus experimentele muziek aan het Konservatorium, verder in dit hoofdstuk toegevoegd. Het interface kent nogal wat sukses omdat het

snel en uitermate eenvoudig te programmeren is, wat niet gezegd kan worden van het door Roland voor de IBM-PC onder de naam MPU401 op de markt gebrachte interface. Ons ontwerp is niet veel meer dan een toepassing van de 6402- UART chip van Intersil. Originaliteit willen we er beslist niet mee claimen.

Voor alle andere stappen (2 , 3 en 4) was een van de grootste dilemmas waarvoor we ons geplaatst zagen, de keuze tussen hardware en software implementaties enerzijds, en binnen de hardware-implementaties, tussen analoog en digitaal. Het is immers zo dat we voor heel wat problemen die te maken hebben met signaalverwerking en data-extraktie, de techniek de vrije keuze toelaat tussen deze drie technische aanpakmogelijkheden.
Met de ons vandaag ter beschikking staande technologie kunnen immers zowat alle signaalverwerkingsproblemen die geen al te strenge eisen stellen inzake snelheid, zowel via hardware als via software aangepakt en opgelost worden. Wanneer echter uiterste eisen aan snelheid en precizie gesteld worden, zal men steeds van hardware oplossingen gebruik maken. Om dit duidelijk te maken, wilden we eerst per stap alle mogelijkheden onderling vergelijken en een vergelijkende kosten en baten analyze ervan maken.

3.1.2: <u>De signaalverwerking</u>

Hier stonden we voor de keuze onze analoge komputer te behouden, ofwel te vervangen door een geheel digitale implementatie in hardware of software.

De signalen zoals die, na voorversterking, beschikbaar zijn als input zijn echter betrekkelijk hoogfrekwent. (35kHz tot 220kHz). Willen we hierop bewerkingen uitvoeren via digitale technieken dan is eerst en vooral een omzetting naar een digitaal formaat onontbeerlijk. Precies hier zit ook de moeilijkheid: immers, om signalen in het hier aangegeven frekwentiegebied om te zetten, dienen we hen te bemonsteren met een frekwentie die minstens het dubbele bedraagt van de hoogste frekwentiekomponenten in het om te zetten signaal. Dit is immers de konsekwentie van het Nyquist theorema. Veronderstellend dat we zouden afzien van alle harmonischen van ons signaal, dan zouden we dus een sampling rate van ca. 440.000 samples per sekonde moeten kunnen realizeren. Dit is ongeveer het tienvoudige van wat voor koerante muzikale toepassingen in digitale audio gangbaar is.

Hiertegen zou men kunnen opwerpen dat we -gezien de eerdere meetgegevens inzake rezolutie- de omzetting zouden kunnen beperken tot 12, 10 of zelfs 8 bits in plaats van de gebruikelijke 16-bits voor digitale audio, maar zelfs in dit geval vallen de noodzakelijke specifikaties van de nodige ADC's buiten datgene wat vandaag koerant en voor ons bereikbaar is.

Lange tijd hebben we gewerkt aan een ontwerp waarbij de TMS320C25 digitale signaal processor (DSP) van Texas zou gebruikt worden.

De Amerikaanse komponist Larry Wendt had ons enige tijd tevoren (1987) enkele proefexemplaren van de mikroprocessor en de volledige dokumentatie van Texas Instruments kado gedaan, zodat we dus reeds een absoluut minimum aan ervaring hadden op dit vlak. Hij werkte op dat ogenblik aan een DSP-board voor de Studio voor Elektronische muziek aan de University of California at Berkeley, *campus San José, onder leiding van Allen Strange, met de bedoeling er een real-time algoritmische soundprocessor mee op te bouwen.*

Maar toen we ook dokumentatie verkregen met betrekking tot de reeds beschikbare toepassingen met deze DSP-chip werkend op een insteekkaart in een IBM-AT kompatibele komputer (de Model 25 Digital Signal Processor card, geproduceerd door Dalanco Spry, Rochester, NY) en de juiste specifikaties ervan, sloeg de angst ons om het hart. We voelden dat we er wel niet in zouden slagen datgene wat de industrie als top-produkten op technologisch vlak aanbood, zo maar eventjes te overtreffen... Dit DSP-board beschikt over 1 ADC kanaal, 12 bit en met een maximale sampling rate van 110kHz. Multiplexing was mogelijk, maar dan werd uiteraard de sampling rate gedeeld door het aantal kanalen. Wij hadden minstens drie kanalen nodig, aangezien er in de Holosound opstelling 3 ultrasoontransducers als ontvangers geschakeld zijn. Dit zou de maximale sampling rate dus hebben teruggebracht tot 36kHz, of, veel te laag om te voldoen aan de eisen van het Nyquist teorema. Bovendien, werd dit board aangeboden voor programmering uitsluitend in TMS320C25 assembler. Routines voor de noodzakelijke Fourier transformaties (256 punten) hadden we weliswaar kunnen overnemen uit de standaard applikatiebibliotheek, maar ook dan bleef het perspektief weinig bemoedigend...

Cfr.: Texas Instruments, "Digital Signal Processong Applications - TMS320 Family" , 1986, "TMS32010 & TMS32020 User's Guide" , "TMS32020 Assembly Language Programmer's Guide".

Heel devoot en nederig besloten we de reeds gerealizeerde print en de bijhorende DSP- en periferie chips terzijde te schuiven en zochten een andere uitweg. We besloten het opzet van de analoge komputer in principe te behouden.

Later pas zijn we -kwestie van toch niet het gevoel te hebben een nederlaag te hebben geleden- gaan narekenen welke 'winst' er te behalen zou zijn door gebruik te maken van zuiver digitale technieken in deze faze van het ontwerp, en toen bleek er eigenlijk helemaal geen winst aan vast te zitten: het oplossend vermogen van de DSP-processor -dan wel uitgerust met een 'anno 1992' snelle ADC-converter- rekening houdend met dit van het signaal waarvan we uitgaan, bleek helemaal niet groter uit te vallen dan de hooguit 7 á 8 bits waartoe de analoge komputer sowieso ook al in staat was.

3.1.3: Data-extraktie en data-acquisition

3.1.3.1.:-De digitale weg

Het probleem om uit de -ingevolge onze vorige beslissing- analoge en laagfrekwente signalen voortgebracht door de analoge komputer een maximum aan relevante informatie af te zonderen, stelde ons eens te meer voor de keuze analoog of digitaal.
Hier was de vraag echter een heel stuk realistischer dan bij de vorige stap. De bandbreedte van de signalen zoals die de analoge komputer verlaten valt immers in het gebied 0 - 1000Hz, wat de eisen inzake sampling rate aanzienlijk gemakkelijker maakt. Een sampling rate van 2kHz is immers ruim voldoende, en iets waaraan gemakkelijk wordt voldaan door eenvoudige en betrekkelijk goedkope 8-10 bits ADC converter chips in de handel.

Vol goede moed, en met de vastberadenheid het ditmaal wel waar te maken, bouwden we een 3 kanaals ADC konverter, grotendeels volgens de lijnen van wat we in het vorige hoofdstuk beschreven sub 2.5., die we echter dit keer rechtstreeks aansloten op de bus van een van onze IBM-AT klonen.
We gebruikten 3 afzonderlijke en goedkope 8-bit ADC-converter chips (type ZN427 van Ferranti) volgens een schakeling grotendeels overgenomen uit de application notes van Tekelec Airtronic. Hiermee is een bemonsteringsfrekwentie van minstens 10kHz haalbaar. Ruimschoot voldoende voor ons doel dus.

Vandaag (01/1993) zou het beter zijn de UVC3130 8-bit video flash omzetterchip te gebruiken, maar die was begin 1992 nog niet vlot beschikbaar op de onderdelenmarkt. Deze chip kan signalen bemonsteren tot 15Mhz en zou dus ook kunnen worden toegepast voor de vorige stap in kombinatie dan met een klassieke 80486 50MHz processor PC... Bovendien is van deze ADC zelfs een type met een rezolutie van 10bits verkrijgbaar!
Voor een typische toepassing, zie G.PELTZ, "AD-DA-omzetters", 1992, p. 666-, en p.90 e.v.

Het bijzondere aan ons ontwerp is dat de ADC-chips vrijlopend gebruikt worden en hun meest recente data-byte dumpen in een latch (de 74HCT573 chips). De komputer heeft dus steeds de meest recente gegevens onmiddellijk voor verwerking ter beschikking, zonder dat hij specifieke konversie-instrukties dient te geven, en dan wachten op het konversierezultaat. We ontwierpen de schakeling voor aansluiting op de bus van de IBM-PC. Een technische verbetering zou erin kunnen bestaan hier in plaats van latches, zgn. dual-ported RAM-chips te gebruiken.

De uitlezing van de X,Y,Z signalen kan heel eenvoudig gebeuren alsvolgt:

```
X% = INP(&H300)
Y% = INP(&H301)
Z% = INP(&H302)
```

Na opbouw van deze eenvoudige ADC schakeling beschikten we over de mogelijkheid onze transducersignalen rechtstreeks als een reeks samples in de komputer in te lezen.

De amplitude van het afgeleverde signaal is een maat voor de hoeveelheid bewegend oppervlak van de speler. (cfr. de simulatorkurves uit hoofdstuk 2). Dus wilden we deze amplitude uit het signaal kunnen afleiden. Deze kan van elk kanaal eenvoudig worden bepaald wanneer we N monsters van dat kanaal nemen en meten, en daarvan dan het gemiddelde berekenen. Immers in momentane waarden van de golfvorm van het signaal zijn we wat deze parameter aangaat niet geinteresseerd. De amplitude van een signaal is wiskundig gezien equivalent met de oppervlakte van haar kurve. In hardware termen, is het equivalent met een afgevlakte gelijkrichting, waarbij de integratietijd (RC-tijd) evenredig is met de waarde van de afvlakkondensator.

Ter wille van de leesbaarheid van de kode doen we hier alsof het 8bit ingangssignaal uitgelezen wordt als een numerieke waarde met de 0-as als symmetrielijn, anders gezegd als decimale waarden tussen -127 en +127. In werkelijkheid echter bewegen de waarden zich uitsluitend tussen 0 en 255.

In Basic (QBX of QB4.5)) ziet de pseudo-code voor een procedure voor het verkrijgen van bewegingshoeveelheid in elk van de drie vektoren er dan (sterk vereenvoudigd) uit alsvolgt:

Over te dragen variabelen naar deze procedure zijn:
Integratietijd: 0.01s - 0.2s
Overgedragen variabelen zijn:
X , Y , Z : Amplitudes voor de 3 vektoren.
Over te dragen (of vast te leggen) konstanten zijn:
Adr1 , Adr2 , Adr3 : de hardwarebepaalde I/O adressen van de ADC-omzetter.

```
SUB GetAmplitude:
       DEFINT A-S
       DEFDBL T, I
T = TIMER : N = 0 :X = 0 : Y = 0 : Z = 0
DO
              N = N + 1
              X = INP(Adr1) : Y = INP(Adr2) : Z = INP(Adr(3)
              ' bereken de permanente som der ingangssignalen
              X = X + ABS(INP(Adr1))
              Y = Y + ABS(INP(Adr2))
              Z = Z + ABS(INP(Adr3))
LOOP UNTIL TIMER - T >= Integratietijd
X = X / N : Y = Y / N : Z = Z / N
END SUB
```

Puur teoretisch gezien zou hier gebruik gemaakt moeten worden van de gebruikelijke differentiaal-vergelijking voor een RC-integrator :

$$V(t) = 1/RC \cdot \int_{-\infty}^{t} Vi(t) \cdot e^{-(t-t)/RC} \cdot dt$$

Ook wanneer deze vergelijking finitistisch wordt uitgewerkt (wat we deden in het programma 'Holosimi' gebruikt voor de kurves besproken aan het eind van hoofdstuk 2, en waarvan de kode aan de appendix werd toegevoegd, is de winst aan precizie niet van dien aard dat de vertraging van het programma die erdoor wordt veroorzaakt gejustifieerd kan worden. Wanneer middelen beschikbaar komen om het wiskundig korrekte algoritme gebruikt in Holosimi ook in real-time toe te passen, mag dat uiteraard niet nagelaten worden.

De volgende parameter waarvoor we gezien ons opzet uiteraard belangstelling hebben en die we ook uit het ingangssignaal wilden kunnen afleiden, was de bewegingssnelheid. Zoals we vroeger hebben aangetoond, is deze een vektoriele funktie van de hoogste dopplerfrekwentie aanwezig in het signaal. Hier nu kwam de aap uit de mouw en stootten we op -zoals uiteindelijk bleek- onoverkomelijke wiskundige, teoretische én praktische problemen...

Intuitief gezien lijkt het op het eerste gezicht erg eenvoudig te zijn uit een signaal waarin we auditief wel degelijk een toonhoogteverloop kunnen waarnemen, dit verloop langs technische

weg te rekonstitueren. Deze intuitie blijkt evenwel ongegrond, niet in de laatste plaats omwille van de hoge komplexiteit van het toonhoogtewaarnemingsverschijnsel van het menselijk oor.

Cfr.:LEMAN, Marc "Een Model van Toonsemantiek", 1991, p.99-117.

Leman geeft een goed overzicht van de stand van zaken en de problemen inzake toonhoogte-waarneming en extraktie. De kontekst waarin hij het probleem behandelt is daarbij signaaltechnisch gesproken heel wat 'eenvoudiger', aangezien de muzikale signalen waarover hij het heeft in hoge mate als periodiek beschouwd kunnen worden, terwijl wij eigenlijk slechts met a-periodieke ruisbanden te maken krijgen.

Informatie over de frekwentie van een signaal op grond van een reeks monsters van dat signaal kan teoretisch worden bekomen door het toepassen van wiskundig (voor ons althans) relatief komplexe diskrete F.F.T. (*Fast Fourier Transforms*), een techniek die onder meer toelaat van een signaal de spektrale samenstelling te bepalen. Voor een reeks samples geldt dan:

$$X(w)= \sum_{n=-8}^{8} x(n).e^{-j.w.n}$$

Voor de implementatie van een volledige FFT in real-time is het hier toegepaste komputersysteem en de gebruikte ADC evenwel lang niet snel genoeg. Daarvoor zou immers een speciaal DSP-systeem gebruikt moeten worden. Rekening houdend met het feit dat de hoogste grond-frekwentie komponenten in het holosoundsignaal zelden groter zijn dan ca. 400Hz, en met het feit dat de sampling rate van de ADC ongeveer rond 800 a 900 samples per sekonde ligt, bleek het -overeenkomstig het teorema van Nyquist- mogelijk iets over frekwentie en spektrale samenstelling van het signaal te berekenen uit X samples door het aantal keer dat de signaalkurve van richting verandert te tellen. Voor zulk algoritme zijn slechts vergelijkingen van gehele getallen nodig waardoor het erg snel door het komputersysteem afgehandeld kan worden. De interpretatie van de bekomen informatie bleek echter zeer problematisch.

Een tweede, en wellicht fundamenteler probleem, is gelegen in het feit dat de FFT-analyze staat of valt met de periodiciteit van het te analyzeren signaal. Ons signaal voldoet beslist niet aan deze voorwaarde.

Ontgoocheld door de weinig bemoedigende rezultaten van deze pogingen tot software implementatie, (zowel naar verwerkingssnelheid als naar relevantie van de bekomen data voor de bewegingsinput) hebben we toen besloten ook hier uit te kijken naar een oplossing via een 'kunstgreep' met een analoge komputer.

Verder onderzoek op het gebied van de real-time signaal analyze is hier echter zeker aangewezen.

3.1.3.2: Analoge komputer voor data-extraktie

Bij het ontwerpen van deze eigenlijk tweede analoge komputer (om in de terminologie van ons vorige hoofdstuk te blijven, gaat het dan eigenlijk over de toevoeging van enkele extra rekenblokken aan de bestaande blokkendoos) zijn we uitgegaan van de vereisten en mogelijkheden van de apparatuur aan dewelke de informatie diende afgeleverd te worden. Het probleem dat hier -omwille van de goede orde van de uiteenzetting- pas in de volgende paragraaf sub 3.1.4 wordt te berde gebracht, was dan uiteraard reeds opgelost.

Omdat de analoog naar digitaal konverter in het komputersysteem waarmee we verder wilden gaan werken (een NEC ADC7004C met een rezolutie van 10bit) alleen ingangssignalen verwerkt tussen 0 en +4V, was voor de aansluiting van de analoge Holosound-komputerblokkendoos alvast een minimale signaal-konditionering noodzakelijk: de uitgangsspanningen mogen alleen positief zijn en mogen de 4V niet overschrijden. De uitgangssignalen van de eerste analoge komputer zijn echter van nature uit en gezien hun oorspronkelijk doel, audio-signalen met een nominaal nivo van +6dB (1.55Vrms).

Voorts hebben we gebruik gemaakt van de eigenschap van de toegepaste ADC-konverter, dat hij over 8 onafhankelijke ingangskanalen beschikt. Dit bracht ons op het idee elke bewegingsparameter via een afzonderlijk kanaal aan de omzetter aan te bieden. Dus, 3 kanalen voor bewegings-intensiteitsinformatie, 3 voor de bewegingssnelheidsinformatie en dan hadden we nog twee kanalen op overschot, waarvoor we eveneens een bestemming vonden: deze twee kanalen zouden gebruikt worden voor de vektoriele som van de bewegings- en snelheidsinformatie der drie kanalen.

3.1.3.2.1: Bewegingshoeveelheid-data

Wat we aanvankelijk via software hadden 'opgelost', bleek via hardware een flink stuk sneller te realizeren te zijn.
Om de amplitude informatie te extraheren uit het signaal van de eerste analoge komputer, maakten we gebruik van wat in het technisch jargon een *true-RMS convertor* heet, onmiddellijk gevolgd door een integrator. Teoretisch gezien dus eigenlijk een gewone amplitudedemodulator zoals we die al bespraken sub 2.3.2.6.1.
Het schema van deze schakeling, zowat de eenvoudigste van alle technische mogelijkheden, kwam er dan uit te zien alsvolgt:

De gebruikte diodes dienen nauwkeurig gekoppelde exemplaren te zijn, en worden tegenelkaar gemonteerd zodat ze zeker dezelfde temperatuur hebben. Alle weerstanden zijn 1% metaalfilm. De ingang is gebalanceerd uitgevoerd. In ons prototype hebben we audiotransformatoren toegepast, wat het voordeel van een perfekte blokkering van DC-offsetspanningen oplevert, en bovendien -door de 1/f karakteristiek van een transformator die sterk opvalt wanneer de frekwenties beneden de 30Hz zakken- de gevoeligheid voor trage bewegingen van het gehele lichaam aanzienlijk vermindert. Uiteraard brengt dit de wiskundige precizie van de schakeling in het gedrang, maar voor ons opzet was dat geen probleem, aangezien hiermee in de faze van de data-verwerking rekening kon gehouden worden. Naar 'betekenis' van het verkregen signaal is het een zuivere informatiewinst. Trage bewegingen van het gehele lichaam van de speler zijn immers expressief minder van belang dan snelle bewegingen van kleinere lichaamsdelen, onder meer omdat ze veelal ontsnappen aan een bewuste kontrole.

De RC-tijdkonstanten voor de integratietijd van de kanalen x,y,z werd gekozen op 15ms, (47kO * 320nF = 15ms) wat overeenkomt met een laagdoorlaatfilter met -3dB punt rond 10Hz.
Deze waarde werd bepaald op grond van experimenteel onderzoek in kombinatie met de verder beschreven software.
De uitgangssignalen worden na deze bewerkingen inverterend gemengd, en nogmaals geinverteerd en geintegreerd waardoor we op de uitgang SA_{xyz} het geintegreerd rezultaat krijgen van de bewerking

$$A_x + A_y + A_z$$

De RC-tijdkonstanten voor de integratietijd van het amplitude-som-kanaal werd bepaald op 220ms (47kO * 4700nF = 220ms).

Dit is equivalent met een laagdoorlaatfilter met een afsnijfrekwentie $f_c = 1/ (2 * p * R * C) =$ 0.723Hz. Met deze eveneens experimenteel bepaalde waarden, bleek het mogelijk zoiets als de omhullende van de totale bewegings-hoeveelheidskurve te verkrijgen. Het signaal kan vrij goed worden gebruikt om kwazi rechtstreeks toegepast te worden op een muzikale parameter zoals de globale dynamiek van de muzikale gebeurtenissen.

Een extra toegevoegde analoge rekenschakeling laat ons toe, in plaats van de A_x, A_y, A_z signalen, de onderlinge verschilsignalen te berekenen en als informatie voor verdere verwerking te gebruiken. Het is een gelijkaardige konfiguratie als die welke we gebruikten bij de vermenigvuldigers die we bespraken sub 2.3.3 en 2.3.3.3.7.

De uitgangssignalen hier -uitgedrukt in funktie van A_x tot A_z- zijn $\{2 - [(A_x-A_y)/2]\}$, $\{2 -[(A_y-A_z)/2]\}$, $\{2-[(A_z-A_x)/2]\}$, en moeten na de verdere analoog digitaal konversie gelezen worden als in binair 'two's-complement' gekodeerde getallen. Het nulpunt is hier immers verschoven naar 2Volt, het middelpunt van het 0-4V bereik.

Voor beide schakelingen is de keuze van het gebruikte type operationele versterkers vooral in de laatste trap van overlevingsbelang voor de ernavolgende schakeling. De ICL7641 op-amp wordt gevoed uit een enkele spanning van 5 Volt. Het is een type dat ook in die omstandigheden zijn uitgangsspanning volledig naar 0V en +5V kan laten gaan en bovendien vrij is van hinderlijke latch-up verschijnselen wanneer het ingangssignaal tegen het plafond aangaat. Hierdoor kan ons uitgangssignaal nooit groter noch kleiner zijn dan de maksimale waarden die toelaatbaar zijn voor de analoog-digitaal konverter.

Door deze opamp keuze hebben we dus eigenlijk meteen een absolute begrenzer/limiter geïmplementeerd.

3.1.3.2.2: Bewegingssnelheidsdata

Zoals we eerder uitvoerig hebben aangetoond, is de bewegingssnelheid in het signaal aanwezig in de vorm van de frekwentiebandbreedte van het signaal. Puur teoretisch gezien zouden we hier dus een analoge omzetter moeten bouwen die de bandbreedte van een aperiodisch signaal omzet in een analoge spanning. Analytisch wiskundig gezien is dit een behoorlijk zwaar probleem, dat we hebben proberen omzeilen alsvolgt:

Stel dat we het signaal eerst ontdoen van zijn amplitudemodulatie. Dit kan eenvoudig gebeuren via een komparator-schakeling die men een nuldoorgangsdetektor noemt. Hoe meer nul-doorgangen het signaal nu heeft, hoe meer hoge frekwenties erin aanwezig zullen zijn. Stel nu dat we elke puls eenzelfde lading in een kondensator laten opslaan en dat we de spanning over de kondensator zouden integreren, dan moeten we de klus min of meer hebben geklaard. Immers de spanning op de kondensator zal een funktie zijn van de mate waarin hogere frekwenties in het signaal zijn terug te vinden.

Na wat opzoekingswerk in het overweldigende aanbod aan geintegreerde schakelingen van diverse fabrikanten, viel ons oog op een verzameling chips die als tacho-konverters bekend staan.
Het opzet van de interne schakelingen bleek goed overeen te komen met wat we nodig hadden, hoewel de chip-funktie aangeduid wordt als *Frequency to voltage converter* en wij eigenlijk geen periodieke signalen ter verwerking konden aanbieden, bleek het probleemloos te funktioneren.

We kozen voor de LM2907 chip van National Electronics. Volgens de databoeken is de uitgangsspanning van deze LM2907 chip een eenvoudige funktie van de ingangsfrekwentie en wel alsvolgt:

$$V_{out} = F_{in} * V_{cc} * R1 * C1$$

Omdat we toch reeds een perfekte en alleen in het analoog gedeelte aangewende 5V voedingsspanning hadden gemaakt voor de voeding van de opamps in de bewegingshoeveelheidsschakeling (dit is overigens de laagste toelaatbare spanning voor de betrokken chips), en omdat het frekwentiegebied dat in deze toepassing voor ons van belang is beperkt kan worden tot 0 - 350 Hz, (overeenkomstig bewegingssnelheden tot circa 3 m/s wanneer een draaggolffrekwentie van 40 a 50kHz gebruikt wordt) berekenden we de waarden voor R1 en C1 als 22nF en 100kO. De hoogste meetbare ingangsfrekwentie wordt bepaald door:

$$f_{max} = I2/(C1 \times V_{cc})$$

De toegepaste schakeling wordt voorafgegaan door een eenvoudig p-low-pass LC-filter (82mH en 100nF) met een afsnijfrekwentie van 1200Hz. Hierdoor worden alle eventueel resterende 40kHz signalen buiten de tacho-schakeling gehouden. Zonder deze voorzorgsmaatregel bleef de uitgangsspanning van de frekwentie naar spanningsomzetter toch nog hangen op V_{cc}!

Over R1 staat een kondensator die bepalend is voor de stabilizatietijd van de omzetting en tevens voor de grootte van de ingangsrimpelspanning aanwezig in het uitgangssignaal.

Deze rimpelspanningskomponent (die na de analoog-digitaal omzetting uiteraard zal verschijnen als een ruissignaal -d.w.z. een signaal dat geen drager is van relevante informatie) wordt berekend als:

$$V_{rpp} = (V_{cc}/2) * (C1/C2) * (1- ((V_{cc} * f_i * C1)/I2))$$

De rimpelspanning wordt 0Volt vanzodra de ingangsfrekwentie naar haar hoogste waarde gaat. Dit komt ons goed uit, want het betekent dat de rezolutie voor hoge bewegingssnelheid groter zal zijn dan voor kleine. De term $1- ((V_{cc} \times f_i \times C1)/I2)$ wordt dan immers 0. Voor de laagste voor ons relevante waarden van f_i krijgen we een rimpelspanning van:

stel $f_i = 10Hz$ $U_{r\text{-}pp} = 15mV$

Voor een 7 bit rezolutie van de ADC convertors moet de rimpelspanning kleiner zijn dan 4V / 127 = 31mV. Bovenstaande waarde zou dus marginaal voldoende zijn om zelfs een 8-bit rezolutie

niet in het gedrang te brengen. Het oplossend vermogen van dit onderdeel van de analoge komputer samen met de verder gebruikte AD-software beperkt dus de laagste meetbare frekwentie, en dus bewegingssnelheid, tot ongeveer 5 Hz wat overeenkomt met een beweging van ca. 5cm/s.

Ter herinnering: $v_b = f_{diff} * v_{geluid} / f_c$

Overigens moet opgemerkt worden dat alhoewel het lijkt dat we eenvoudig een rezolutie van 8 bits voor de laagste frekwenties en zelfs van 10 en meer bits voor de hoogste zouden kunnen halen, dit toch geen zin zou hebben omdat de precizie van de frekwentie naar spanning omzetters in de data sheets voor deze chips toch slechts bepaald wordt op 1%, wat op zich reeds zelfs het LSB van een 7-bit konversie onbetrouwbaar maakt! Bovendien introduceert ook de aperiodiciteit van de signalen waarmee wij hier werken, tegenover die waarvoor de specifikaties van de chip worden gegeven een onzekerheidsfaktor.

Het schema van de uiteindelijk gebouwde en geteste schakeling kwam er uit te zien alsvolgt:

Ook de uitgangssignalen van deze schakeling worden aangeboden aan 3 afzonderlijke ingangskanalen van de analoog-digitaal converters op de 'dedicated single board computer' waarover we het nu gaan hebben.

3.1.4: Data-verwerking

Bij onze eerste pogingen om de informatieverwerving te doen in software, maakten we gebruik van een van onze IBM-AT klonen. Nu is dit soort machine niet bepaald erg handelbaar wanneer het erop aan komt ze te gebruiken in een implementatie van een realistisch muziekinstrument. Ze zijn betrekkelijk groot, lomp, kwetsbaar en... maken lawaai op de koop toe.

> *De lawaaibronnen zijn enerzijds de ventilator in het voedingsblok en anderzijds, de kontinu draaiende motor van de harddisk. De ventilator kan -althans rekening houdend met de gangbare omgevingstemperaturen in onze kontreien- zonder enig gevaar uitgeschakeld worden, de harddisk evenwel niet.*

Bovendien veroorzaakt de ventilator in het voedingsblok een ontoelaatbare verstoring van het ultrasoon veld, dit dan door de beweging van de schoepen. Aangezien we voor vroegere automatiseringsprojekten reeds vaker voor dergelijke problemen waren geplaatst, dachten we ook hier onmiddellijk in de richting van een 'microcontroller'. Dit is in wezen een klein zelfstandig werkend komputersysteem, met een processor, RAM en ROM geheugen, enkele I/O poorten en een seriële poort via dewelke middels een 'host'-komputer data en programmas in het systeem kunnen worden geladen. Het zijn dus eigenlijk volwaardige komputersystemen maar dan zonder beeldscherm, zonder klavier en zonder enig perifeer toestel. Hoewel men dat zich zelden realizeert is het eigenlijk het meest toegepaste en het in de grootste aantallen industrieel gefabriceerd soort komputer.

De types waar we vroeger reeds veel ervaring hadden mee opgedaan waren resp. de 8052 van Intel en het Slicer-board (met een 80186 16-bit Intel processor) dat werkt onder CP/M of een primitief soort DOS. Het eerste type (de 8052) leek ons ondanks de vele argumenten die ervoor pleiten, in deze toepassing te beperkt. Tegenover het voordeel van de gemakkelijke programmeerbaarheid stond het nadeel van de trage snelheid vooral voortkomend uit het feit dat de Basic programmas niet konden gekompileerd worden. De Slicer -hoewel we er via een genereuze gift van een sponsor aan Stichting Logos zo'n 30 exemplaren van in voorraad hadden- kon in een komputertaal naar keuze worden geprogrammeerd, maar door de kompleksiteit van het operating system (meer bepaald de interrupt handlers) bleek het erg onbetrouwbaar voor dit soort taak waar het toch in eerste plaats op real-time signaalverwerking aankomt. Bovendien was het systeem behept met enkele storende 'bugs'.

Waaraan we eigenlijk behoefte hadden in deze faze, was aan een komputer die ofwel een parallelle in plaats van een seriele architektuur bood (wat ons zou toelaten elke parameter afzonderlijk te verwerken), ofwel aan eentje dat over intrinsieke multitasking mogelijkheden zou beschikken. Na wat zoeken, kwamen we zoiets ook op het spoor...
n ding...

De komputer die we hiervoor dan uiteindelijk aanschaften en inzetten, was een 'Multitrax' multitasking microcontroller: dit is een gehele komputer die uit niets meer dan 1 enkele printplaat bestaat en noch scherm, noch klavier heeft. De kommunikatie met de programmeur gebeurt via een terminal. Hiertoe wordt de RS232 poort waarover het board beschikt, verbonden met een terminal of eender welke 'gewone' komputer, waarop dan een kommunikatieprogramma loopt. Voor PC's van de IBM-familie bevelen we daarvoor het programma 'Procomm' sterk aan. Wanneer het programma is geschreven en getest, kan het -na kompilatie- op het board zelf in een EPROM worden opgeslagen, waarna het systeem het programma zonder nood aan terminal of externe komputer, volkomen autonoom kan draaien.

Het leuke van dit komputersysteempje is dat het ingebouwde besturingssysteem, werkelijke multi-tasking toelaat. Dit wil zeggen, dat het systeem verschillende taken tegelijkertijd kan afhandelen. Bovendien is het systeem rechtstreeks in Basic te programmeren: daartoe werd een ROM-basic ingebouwd, inklusief een compiler voor de omzetting naar machinetaal. De ontwikkelde software loopt alleen na compilatie op het systeem, zodat alle nadelen verbonden aan Basic-interpreters voor dit systeem niet geldig zijn.

De erin toegepaste mikroprocessor is een snelle 8-bits CMOS 64180 gefabriceerd door het Japanse Hitachi, wat eigenlijk een meer geavanceerde uitgave is van de vanouds bekende Z80 processor, een chip waarmee we reeds in de jaren '70 hadden gewerkt. Naar geheugen is het een voor hedendaagse normen erg bescheiden systeem: 32kByte RAM en 96kByte ROM. Meer ROM dan RAM dus, iets wat logisch is gezien de industriele bedoeling van het ontwerp.

Naast deze aspekten -en dit punt was natuurlijk niet aan onze aandacht ontsnapt- kon het board ook eenvoudig worden uitgebreid met een 8-kanaals ADC (Analog Digital Convertor) met een rezolutie van 10-bits, een ingebouwde real-time klok, 48 digitale I/O bits, een frekwentieteller, een tweede vrij programmeerbare RS232-poort, een LCD-scherm poort en een eenvoudige toetsenborddekoder voor 16 toetsen... Deze uitbreidingen dienden er wel zelf bijgebouwd worden, maar de verschafte dokumentatie was erg goed en bovendien werden de uitbreidingen reeds zowel in het operating system als in de technische Basic implementatie voorzien.

Technische informatie m.b.t. het beschreven komputersysteem:
'Multitrax' - producent: Control Technology, Box 4605, Mountain View, CA 94040, USA.
Distributie: SINTEC Company, P.O. Box 410, Frenchtown NJ 08825 USA (Tel.:800 526-5960).

Wij besloten het board van volgende extra 'features' te voorzien met het oog op gebruik in het non-impakt muziekinstrument:

-AD-konverter met de NEC D7004C chip
Deze chip leek ons ideaal geschikt voor ons doel. Hij is uitgevoerd in CMOS-technologie en het gevolgde AD-konversie procede is dat van de geleidelijke stapsgewijze benadering (Successive Approximation Register -SAR). Via multiplexing beschikt hij over 8 analoge ingangskanalen. Multiplexing is hier het om beurten aftasten van de verschillende kanalen. De konversietijd bedraagt slechts 100µs. De ingangsspanningen mogen in geen geval negatief zijn en ook niet de voedingsspanning (+5V) overschrijden. Daarom beperkten wij deze in ons hiervoor beschreven analoog komputerontwerp tot het bereik 0-4V.

-Een 1-regelig LCD-display
Dit om toch enige feedback naar de gebruiker mogelijk te maken. Vergeten we immers niet dat het de bedoeling was een volstrekt zelfstandig opererend instrument op te zetten en dus niet iets wat alleen middels een bureelkomputer of via een ingewikkelde labo-opstelling aan de praat te krijgen was. Dit is immers de enige mogelijkheid om wanneer onverhoopt eens iets niet helemaal naar behoren zou funktioneren, via software meldingen naar de gebruiker te sturen. Wij pasten een display toe van de firma Varitronics Ltd. Op de aansluitproblemen daarvan gaan we hier niet verder ingaan omdat dit niets te maken heeft met het instrument als dusdanig of met enige estetische konsekwentie.

-Een midi-poort
Over zulk interface beschikten we reeds, maar er dienden stappen ondernomen te worden om de op het board geïmplementeerde printerpoort -die slechts 7-bits gebruikte- om te bouwen naar een volwaardige universele 8-bit poort. Meer hierover sub. 3.2.5.

-Een precieze timer
Gezien we het systeem voor muzikale doelen wilden gebruiken was het essentieel over een precieze en van de kloksnelheid van de mikroprocessor zelf onafhankelijke bron van tijd-informatie te kunnen beschikken. Hiervoor gebruikten we de MM58274 van National Semiconductor.

-Een miniatuur toetsenbord
We besloten het board van een minimaal 'input-device' te voorzien, in de vorm van een toetsenbord met 16 toetsen, een beetje zoals we dat kennen van de elektronische rekenmachines. 16 toetsen achtten we voldoende ook voor het inbrengen van hexadecimale kodes.

3.1.5: Data-output

MIDI INTERFACE

Hiervoor gebruikten we het uit 1984 daterende ontwerp voor een MIDI-interface uitgaand van de signalen beschikbaar op een Centronics printerpoort, zoals we die op bijna alle komputers terugvinden.

Hoewel het interface bidirektioneel is, wordt het in deze toepassing slechts als output-device gebruikt. De besturingswijze blijkt duidelijk uit de hiernavolgende software.

3.1.6: Evaluatie & blokschema van de gehele hardware

Hiermee is de gehele hardware van het non-impakt instrument -of althans het eerste redelijk goed werkend prototype ervan- beschreven en verklaard.

Volgend schema vat samen hoe alle beschreven blokjes met elkaar worden verbonden. Het is belangrijk dit voor ogen te houden voor een goed begrip van de hierna beschreven software, waarzonder het instrument niet kan funktioneren.

Uiteraard kan software niet meer dan een akademisch bestaan kennen, wanneer er geen hardware gespecifieerd wordt waarop die software dient te lopen. Niettemin, kan men opteren voor een minimale hardwarematige signaalverwerking en zoveel mogelijk via software-technieken afhandelen. De weg dus die wij hier niet hebben bewandeld. Het meest aangehaalde voordeel van een dergelijke softwarematige aanpak, zou zijn dat dit een grotere flexibiliteit mogelijk zou maken. Het zou immers mogelijk zijn via aanpassingen van de software of via geheel nieuwe software, totaal andere machines te koncipieren en te doen funktioneren.

Uit o.m. onze hiervoor geschetste persoonlijke ervaring terzake evenwel, menen we te mogen stellen dat dit een fabeltje is of toch minstens gerelativeerd dient te worden. De ontwikkelingstijd nodig voor het schrijven van een werkend stuk signaalverwerkingssoftware is beslist in dezelfde orde van grootte dan de tijd nodig voor het berekenen, ontwerpen en bouwen van een modern stuk hardware. Het fabeltje blijft evenwel hardnekkig standhouden, vooral m.i. omdat beide aanpakken een grondig verschillende technische kunde en kompetentie veronderstellen. Het zijn onderscheiden specializaties geworden die naarmate de specializatie verder doorgedreven raakte, verder en verder uiteen zijn gegroeid. Het aantal software ontwikkelaars dat vandaag nog op de hoogte is van digitale hardware, is (helaas) uiterst klein. Dit legt onvermijdelijk een hypoteek op de vinding van optimale oplossingen voor gestelde problemen in het domein van de informatieverwerking in het algemeen.

Vooral in het domein dat ons bezighoudt, de muziek, konstateren we de laatste jaren bij de al te weinigen die zich überhaupt nog met de 'konstruktieve' aspekten van het muziekmaken bezighouden, een alsmaar sterkere verschuiving naar software-ontwikkeling ten nadele van de hardware. De hardware wordt al te vaak bekeken als 'loodgieterij'. Er wordt vaak, vooral vanuit akademische hoek, op neergekeken. Hier zien we een vervreemding ontstaan die we ook zien bij musici die van hun eigen instrument vervreemd raken. Hoeveel pianisten slagen er vandaag nog in een piano open te maken en er herstellingen aan uit te voeren? Welke pianist kan nog een piano stemmen of weet zelfs hoe dat teoretisch zou moeten gedaan worden?

Verder nog konstateren we, binnen de software, een evolutie van echt programmeerwerk, naar louter gebruik van zuivere toepassingssoftware (sequencers, sound-editors, notators). De muzikus lijkt in deze ontwikkeling de o.i. noodzakelijke greep op zijn klinkende materie en vooral ook op de wijze waarop hij daarmee zal omgaan te verliezen. Hij wordt -extreem gesteld- veeleer een slachtoffer van de technologische ontwikkeling, dan een bevoordeligde ervan.

3.2. Software

3.2.1.1:'Mapping'

Uiteraard doet al deze hardware helemaal niets, zolang er geen software wordt geschreven om op een relevante manier de aangeboden informatie te verwerken en tot muzikaal aanvaardbare rezultaten te transformeren.

Nu is het precies op het vlak van deze software dat de uiteindelijke bepaling van het instrument tot stand zal kunnen komen. Taak van de software zal er moeten in bestaan uit de beschikbare data op grond van beslissingsregels muzikaal relevante en samenhangende informatie af te leveren. Het centrale begrip hierbij is dat van de 'mapping'.

'Mapping' noemen we de wijze van afbeelden van informatie uit een eerste domein naar informatie uit een ander domein. De Nederlandse vertaling 'inkaartbrenging' is niet echt geschikt, zodat we ons aan de Engelse terminologie zullen houden. Het begrip mapping is ook veel ruimer dan de korrelaties die we bespraken sub 1.1 en 2.3.4 tot 2.6. Hier gaat het immers om informatie en niet om fyzische parameters.

De 16 verschillende programmabrokken die samen het 'Book of Moves' vormen zijn niets anders dan 16 verschillende 'mappings' in deze zin. Elke mapping doet in menig opzicht, een nieuw instrument ontstaan. Ook de instruktie van het via een mapping geïmplementeerd instrument, hangt volledig af van de aard van die mapping. We hopen dit in de hierna volgende kode en de erbij horende kommentaar duidelijk te maken.

Vooraf willen we er wel nog op wijzen, dat de hier gegeven kode gegeven is bij wijze van voorbeeld. Het is de kode zoals wij die voor de definiering van enkele praktische instrumenten voor eigen gebruik ontwikkelden. De muzikus die zich van ons instrument zou wensen te bedienen, dient zich -via het vastleggen van een voor zijn voorkeur geschikte mapping- dit instrument eerst te bepalen. Hij zal daartoe dus ook een eigen stukje software moeten schrijven. Moeilijk is dit echter niet.

3.2.1.2.: Multitasking Basic

Het BASIC dat voor deze processor en dit komputerboard werd geimplementeerd is niet helemaal gelijk te stellen met het 'plain-vanilla' Basic zoals we dat kennen van simpele home-komputers. Het beschikt immers over multi-tasking mogelijkheden die op eenvoudige machines niet voorhanden zijn.

Typische instrukties die betrekking hebben op deze multi-tasking mogelijkheden en waarmee de lezer allicht niet vertrouwd is zijn:

```
START
SUSPEND
EXIT
TASK
```

De START instruktie wordt gebruikt om een taak van start te laten gaan. Elke gedefinieerde taak heeft een nummer evenals een periodiciteit, waarmee tevens de prioriteit tegenover andere taken kan worden vastgelegd. De syntax is dan ook:

```
START x,y.
```

waarbij x het taaknummer is en y de periodiciteit ervan.
Met het kommando EXIT kan een taak verlaten worden.

Voor het overige is dit Basic echter helemaal niet bijzonder geavanceerd. Klassieke strukturen zoals IF-THEN-ELSE (het ELSE deel werd niet voorzien) en de Booleaanse negatie NOT werden helaas niet geimplementeerd. Ook onze favoriete DO ... LOOP UNTIL... struktuur moesten we ontberen evenals het programmeren in modulaire strukturen zoals we dat kennen in Microsoft Professional Basic. Het is evenmin mogelijk zelf de ROM-library uit te breiden, althans niet zonder eerst het hele Basic zelf te gaan herschrijven, iets waartoe we ons niet geroepen voelden.

Ook de mogelijkheden tot 'nesting' zijn erg beperkt. Om die redenen konden we dan ook de (door de 'schoolmeesters' van de informatika) verfoeilijk geachte GOTO's niet steeds vermijden in ons programma.

Daar waar meer geavanceerde Basics (Quick-Basic, Visual Basic of PDF-Basic van Microsoft of GFA-Basic voor IBM-PC's bijvoorbeeld) de programmeur toelaten procedures te definieren, hebben we dit hier via 'user functions' moeten doen.

Gosub's zijn in dit stukje software dan ook nagenoeg niet aan te treffen. Dit komt -zo hopen we althans- de leesbaarheid zeer ten goede.

De subroutine voor het spelen van samenklanken opgebouwd uit 8 noten kon echter onmogelijk als een funktie herschreven worden omdat funkties in dit Basic de overdracht van array-variabelen niet toelaten.

Gezien het technische karakter van dit soort komputer -een industriele mikrokontroller- beschikt het geimplementeerde Basic over enkele Booleaanse funkties geschikt voor gegevensmanipulatie op het nivo der individuele bits in hexadecimale getallen. Funkties dus, die zeer dicht staan bij machinekode en uiteindelijk bij de hardware zelf. We hebben er dankbaar gebruik van gemaakt. Het is niet in het minst door deze mogelijkheid dat dit Basic zo'n goede prestaties kan halen met betrekking tot uitvoeringssnelheid.De funkties zijn:

```
BAND
BOR
BXOR
```

Het zijn de zuiver binaire equivalenten van de logische funkties waarover elk gewoon Basic beschikt: AND, OR, XOR. De gebruikelijke waarheidstabellen zijn hier uiteraard van toepassing.

Zoals bij zowat alle professionele talen, eist ook dit Basic dat variabelen worden gedeklareerd op voorhand als geheel getal (INTEGER), reeel getal (REAL) of tekenreeks (STRING). Andere mogelijkheden zijn er niet.

3.2.2: Kode: "A BOOK OF MOVES"

We hebben de diverse programmasegmenten die volgen in ruime mate voorzien van kommentaren teneinde de precieze werking duidelijk te maken. We hopen dat het programma zodoende ook voor de niet in Basic onderlegde lezer goed te volgen is. Alleen regels die aanvangen met een getal zijn eigenlijke programmaregels. Al het overige is kommentaar.

> *Kode geschreven voor het Multitrax board met een 'extended-MTBasic' (multitasking Basic), EPROM-chip. De kode loopt slechts na kompilatie. De wijze waarop dit dient te gebeuren, is in de appendix toegevoegd. De hier weergegeven listing volgt de versie met nummer MT42 d.d. 19.08.1992). Deze versie werd gebruikt in een twintigtal publieke en internationale voorstellingen en lezingen. Ze mag dus als getest worden beschouwd. Gezien het feit dat aan dit projekt doorlopend wordt voortgewerkt, komt dit op het ogenblik dat dit gelezen wordt, niet noodzakelijk overeen met de meest recente staat van het programma. Deze opmerking is overigens ook van kracht met betrekking tot de gebruikte hardware zelf.*

3.2.2.1.- Deklaraties

In volgende lijst, uitmondend in de eerste twee programmaregels, worden alle verder in het programma gebruikte variabelen gedeklareerd en hun gebruik binnen het programma vastgelegd. Dit terwille van de leesbaarheid en het onderhoud van de software.

A
naam van de variabele gebruikt voor de ingelezen som der bewegingsamplitudes van de 3 kanalen x,y,z. Hierin staan de laatst gelezen inhoud van ADC(3)
C
variabele gebruikt voor de ruimtelijke plaats waar een beweging werd gedetekteerd. (memo: CASE)
D
dummy-variabele die gebruikt wordt in de 'valse' funkties waarmee U(), UPAR(),P() en andere worden aanroepen. De waarde van D heeft zelf geen enkele betekenis.
E
Deze variabele wordt alleen gebruikt om de volgorde waarin de diverse algoritmische mappings worden uitgevoerd in op te slaan. Deze variabele wordt door de gebruiker via het toetsenbordje ingebracht. Er zijn slechts 16 mogelijke waarden: 0-F hexadecimaal.
I
Algemene teller voor gebruik in lussen. Eveneens toegepast als variabele voor het opslaan van een vorige toestand in taak 5 evenals als variabele voor het tonaal-centrum in bepaalde algoritmes.
J
Algemene teller variabele in lussen
L
Teller variabele, meestal gebruikt voor tijdsbepalende routines via de SYSTIC-instruktie.

M
Variabele gebruikt als kanaal-teller in de midi-kommandos.
T
Tijdsvariabele gebruikt in de funktie S()
X,Y,Z
Variabelen gebruikt in de taken waarin de ADC waarden worden uitgelezen voor de drie bewegingsvektoren. X heeft steeds betrekking op een parameter van het x-kanaal (links), Y op het Y-kanaal (rechts) en Z op het Z-kanaal (boven). Deze variabelen worden door de multitasking ververst (updated) gedurende de procedures.
XH,YH,ZH

Deze variabelen worden gebruikt als houd-variabelen wanneer een verversing van de inhoud ongewenst is. Bvb. wanneer na een IF konditie een bewerking wordt uitgevoerd waarin die variabele wordt gebruikt.

> *Dit is wellicht niet duidelijk voor wie niet vertrouwd is met multitasking. Daarom een klein voorbeeld. Onderstel dat we een zin formuleren zoals: IF XH < 64 THEN XH=XH+63. In een 'normaal' programma mag aangenomen worden dat de waarde van XH wanneer aan de initiële konditie XH < 64 voldaan is, niet groter kan zijn dan 127. Indien evenwel XH behoort tot de klasse der variabelen die op dat ogenblik in een taak worden ververst, dan is het mogelijk dat eerst de voorwaarde XH < 64 voldaan is, vervolgens XH ververst wordt en de waarde 120 krijgt, waarna ingevolge XH=XH+63 aan XH uiteindelijk de waarde 183 zou verkrijgen. Dit was uiteraard niet de bedoeling van onze voorwaardelijke bepaling. Vandaar dus de noodzaak om tussenvariabelen te gebruiken in dergelijke gevallen.*

Deze variabelen worden ook gebruikt voor de opslag van de vorige waarde van X, Y, Z in vergelijkingsalgoritmen.

XF,YF,ZF

Deze variabelen worden gebruikt voor de bewegingssnelheden in de drie vektoren. Via deze variabelen worden dus de uitgangen van de analoge tacho-komputer uitgelezen.

U

Dit is de naam van een gedefinieerde funktie. Namen van funkties moeten in MT-Basic gedeklareerd worden alsof het variabelen waren.

UPAR

Naam voor de funktie die gebruikt wordt om parameter-veranderingen in de Proteus synthesizer te bewerkstelligen. Dit is niet in alle versies van dit programma geimplementeerd.

P

Naam van de funktie waarmee voorinstellingen en panning van de synthesizer audiokanalen wordt ingesteld.

S

Naam van de funktie die gebruikt wordt in de tijdsbepalende routine.

> *Aanvankelijk werd dit gedaan door gebruik te maken van het SUSPEND kommando, maar de implementatie daarvan in MT-Basic heeft een merkwaardige Bug die het onbruikbaar maakt...*

B

Variabele gebruikt voor de opslag van een Midi-byte zoals gebruikt in de midi-uitstuurfunktie U()

B0,B1,B2,B3,B4,B5,B6,B7

Deze variabelen worden gebruikt voor de opslag van de registratiegevens voor de Proteus synthesizer. Zij worden gebruikt in de P funktie.

Alle hiervoor opgesomde variabelen worden bepaald als gehele getallen:

```
1  INTEGER  A,B,B0,B1,B2,B3,B4,B5,B6,B7,C,D,E,I,J,L,M,P,S,T,U,X,
            Y,Z,XH,YH,ZH,XF,YF,ZF
```

N(7)

Deze gedimensioneerde variabele wordt gebruikt om de namen van de via MIDI uit te sturen noten op de kanalen 0 tot 7 op te slaan.

V(7)

Deze gedimensioneerde variabele wordt gebruikt om de aanslagsterkte van de via midi te spelen noten op de kanalen 0-7 op te slaan.

NO(7)

Deze gedimensioneerde variabele wordt gebruikt om de waarde van de vorige noot op het overeenkomstige kanaal op te slaan en vast te houden.

Ook alle gedimensioneerde variabelen worden als gehele getallen gedefinieerd:

```
2 INTEGER N(7),V(7),NO(7)
```

3.2.2.2.-Initializatie van hardware en software

Het eerste wat nu dient te gebeuren is het toewijzen van ASCII-waarden aan de 16 schakelaars van het matrix-toetsenbord. We kozen deze waarden zo dat wanneer de hexadecimale konstante $30 ervan afgetrokken wordt, we het decimaal equivalent van de eerste 16 hexadecimale getallen ($0-$F) overhouden. MT-Basic heeft een ingebouwde funktie KEYPAD waarmee de toetsenmatrix uitgelezen kan worden. Wanneer de KEYPAD funktie een 0 waarde oplevert, werd er geen enkele toets ingedrukt. In het omgekeerde geval levert deze funktie ons de ASCII waarde op van de toets zoals we haar tijdens de initializatie middels KEYSTR definieerden. De KEYPAD funktie maakt gebruik van een hardware interrupt. Het is dan ook van groot belang deze interrupt niet buiten werking te stellen, wat via software hier wel mogelijk is.

Een opmerking wat dit betreft: wanneer een toets ingedrukt wordt gehouden, dan valt het hele processorgebeuren voor die tijd stil en herbegint pas wanneer de toets terug wordt losgelaten.

De MIDI-poort dient eveneens via software geinitializeerd te worden, aangezien deze poort hardwarematig gebruik maakt van een programmeerbare PIO chip, de 8255. De werkingsmodus wordt ingesteld via I/O adres 195. De voor ons gewenste konfiguratie is alsvolgt:

Poort A - als uitgang voor de midi-databytes
Poort B - als uitgang voor het Strobe-signaal
Poort C - als ingang voor het lezen van de toestand

cfr. INTEL-databoek m.b.t. de 82C55-chip voor details met betrekking tot de programmatie van deze chip.

De volgende stap in de initializatie is het invullen van de opzoekingstabellen waarvan het programma gebruik maakt. Daarna volgen nog een instruktie om de toevalsgenerator te herbronnen. Deze drie stappen worden uitgevoerd door volgende zin:

```
3 KEYSTR "0123456789:;<=>?"
  OUT $C3,$81 : OUT $C1,1 : OUT $C0,$FF
  GOSUB 900
  RANDOMIZE
```

De toevalsnummergenerator wordt alleen gebruikt voor de panning van het geluid in de module 'Open'.

3.2.2.3.- Definitie van de midi-funktie

Volgende funktie, luisterend naar de naam U, (memo-tip: Uit) omvat al het nodige om ons midi-interface aan de praat te krijgen. De over te dragen parameter is B, het te verzenden midi-byte (7-of 8 bits). (Cfr. sub. 3.2.1.5)

```
4 DEF U(B)
5 IF B<$80 THEN OUT $C1,0
6 IF B>$7F THEN OUT $C1,1: B=B-$80
7 IF BAND(INP($C2),2) THEN GOTO 7
8 OUT $C0,B+$80
  OUT $C0,B
  OUT $C0,B+$80
  OUT $C0,$FF
9 FNEND
```

3.2.2.4.- Initializatie van de Midi-poort en het LCD scherm

De initializatie van de midi-poort maakt gebruik van de zopas gedefinieerde funktie. Voor de initializatie van het LCD-schermpje is een kommando voorzien in MT-basic: INITLCD.

De betekenis van de parameters bij het INITLCD kommando is alsvolgt:Het eerste cijfer dient 0 te zijn bij gebruik van een display dat lettertekens heeft die opgebouwd zijn uit een matrix van 5 op 7 punten. Het tweede cijfer bepaalt hoe de loper zich op het scherm zal gedragen: 0= meegaand met de verschijnende lettertekens, 1= terplaatse blijvend terwijl de tekst naar rechts doorschuift, 2= zoals op een gewoon komputerscherm- de tekst schuift naar links door. Voor het derde cijfer gelden dan volgende betekenissen: 0-3: scherm onleesbaar, 4= loper onzichtbaar, geen geknipper, 5= geen loper, wel geknipper, 6=wel een loper geen geknipper, 7=loper zichtbaar en knipperend. De initializatie instruktie INITLCD plaatst tevens de loper op positie 1, wat ook mogelijk is via het kommando OUTLCD $80. Er zijn slechts 16 plaatsen voor letters op het schermpje dat wij dit instrument meegaven. De posities lopen van $80 tot $8F. Het kommando OUTLCD 1 wordt gebruikt om het scherm te wissen.

```
10 D = U($FF)
11 INITLCD 0,0,5
   OUTLCD 1

12 STIME 0,0,0,2
   SDATE 01,01,93
```

Het instellen van tijd en datum is op zich voor dit programma van geen enkel belang. We namen deze kode alleen op omdat de overgang van tijd 24u00 naar 0u00 een interrupt genereert die ook de verder vaak gebruikte SYSTIC instruktie in de war stuurt. Het getal dat door SYSTIC wordt geretourneerd is een 32 bit waarde in 'two's complement' . Bij de dagovergang springt deze teller van positief naar negatief, waardoor de gebruikte delta-t waarden een fout opleveren. Een dergelijk fout rezulteert dan in een schijnbaar hangende komputer. De instelling van de klok bij het begin van elke uitvoering op het uur 00h00'00" is een goede remedie zolang het instrument tenminste niet langer dan 24u ononderbroken blijft aanstaan.

3.2.2.5.- Definitie van de synthesizerfunkties

De hier gedefinieerde funkties worden gebruikt om de registratie van de Proteus-synthesizer module te programmeren. Elke registratie kan (in deze implementatie) uit 8 verschillende stemmen bestaan. (Het maximum voor dit type synthesizer is echter 16 en mits wat kunstgrepen zelfs 32).

Voor een goed begrip van de uitgestuurde hexadecimale kodes, dient het technisch handboek van de Proteus synthesizer gekonsulteerd te worden, uiteraard na kennisname van de Midi-specifikaties zelf. (cfr. bibliografie).

Terloops weze opgemerkt dat dit Basic de prefix $ gebruikt voor de aanduiding van hexadecimale getallen, waar andere Basic-versies eerder &H gebruiken en het $ teken reserveren voor strings.

```
20 DEF P(B0,B1,B2,B3,B4,B5,B6,B7)
```

Selekteer de presets via de overgedragen variabelen:

```
21 D = U($C0)+U(B0)+U($C1)+U(B1)+U($C2)+U(B2)+U($C3)+U(B3)
22 D = U($C4)+U(B4)+U($C5)+U(B5)+U($C6)+U(B6)+U($C7)+U(B7)
```

Reset de pitch-bend controllers en stel de master volume regelaars op maximum: (=&7F)

```
24 FOR M = 0 TO 7
       D = U($E0+M) + U(0) + U(64) + U($B0+M) + U(7) + U($7F)
   NEXT M
```

Nadat de registratie ingesteld is, programmeren we eveneens de ruimtelijke panning van de muzikale output. 0 komt overeen met volledig links in het stereo-beeld, $7F met volledig rechts, en 64 is dan het midden van het stereobeeld.

```
26 D = U($B0) + U($A) +U(0) + U($B1) + U($A) + U(64) + U($B2)  +
       + U($A) + U($7F) + U($B3) + U($A) + U(20)
27    D  = U($B4) + U($A) + U(64) + U($B5) + U($A)  +  U(107)  +
       + U($B6) + U($A) + U(47) + U($B7) + U($A) + U(80)
29 FNEND
```

De nu volgende midi-synthesizer funktie wordt gebruikt om enkele muzikale parameters te kunnen wijzigen. We doopten deze funktie UPAR. Zij verloopt in twee stappen:

1.- stel de synthesizer in op de registerinstelling waarbinnen parameters veranderd dienen te worden. Dit verloopt middels een 'channel-program-change message'.

2. stuur het parameterwijzigingskommando uit via midi-kanaal 0

```
30 DEF UPAR(B1,B4,B2,B3)
```

De variabele B1 bevat het midi-kanaal, terwijl B4 het volgnummer van de gewenste voorinstelling (preset) bevat. (0-127)

```
31 D=U($C0+B1)+U(B4)
```

De variabele B2 bevat het nummer van de gewenste parameter, B3 de data. Voor deze instellingen zijn 14-bit getallen nodig, die we dus eerst dienen te splitsen in MSB en LSB alsvolgt:
B2 te splitsen in MSB=B4 and LSB=B5
B3 te splitsen in MSB=B6 and LSB=B7

```
32 B4 = (B2 / $80)
   B5 = B2 - (B4 * $80)
   B6 = (B3 / $80)
   B7 = B3 - (B6 * $80)
33  D = U($F0) + U($18) + U(4) + U(0) + U(3) + U(B5) +  U(B4)  +
       + U(B7) + U(B6) + U($F7)
34 FNEND
```

3.2.2.6.:- Kommando voor tijd en ritme

Deze funkties werden geschreven ter vervanging van de binnen MT-Basic voorziene kommandos SUSPEND and WAIT. Het oplossend vermogen van de SYSTIC funktie is een honderdste van een sekonde.

```
40 DEF S(B7)
41 T = SYSTIC
42 IF SYSTIC < (T + B7) THEN GOTO 42
43 FNEND
```

3.2.2.7.:- Opstellingstest

Volgende programmaregels worden gebruikt om een optimale opstelling van de hardware, meer bepaald de precieze plaatsing van de zender en de drie ontvangers, mogelijk te maken, vooral in funktie van de ruimteakoestiek van de plaats waar het instrument wordt opgesteld voor gebruik. Via deze regels worden de waarden van de analoog-digitaal konverter uitgestuurd via de 9600 Baud RS232

verbinding naar het scherm van de host-komputer of terminal. Deze laatste is dus alleen in dit geval ook werkelijk nodig. De opstellings- en afregelingsroutine kan worden gestopt door eender welke toets op het matrix-toetsenbord in te drukken.

```
44 FOR I=0 TO 6
      PRINT ADC(I);"    ";
   NEXT I
45 D = S(50)
   PRINT " "
46 IF KEYPAD = 0 THEN GOTO 44
```

3.2.2.8.- Keuze-menu voor de verschillende modules

In de volgende programmaregels wordt een vrije keuze van de te implementeren mappings door de gebruiker mogelijk gemaakt. Voor de teatrale voorstelling van de "A Book of Moves" implementatie van ons instrument ligt deze volgorde weliswaar vast als kompositorisch gegeven, maar dit werd niet vastgelegd op het nivo van de hardware of de software.

De selektie van de geimplementeerde algoritmen en mappings gebeurt via het matrix-toetsenbord en is niet bepaald erg gebruikersvriendelijk. Alleen hexadecimale getallen kunnen immers ingebracht worden. Omdat we geen 'enter' toets voorzagen, is bovendien een foutief ingetoetst cijfer, niet te korrigeren.
Gekoppeld aan de keuze van een module in dit programma, impliceert een bepaalde keuze ook een bepaalde instelling van de synthesizer. Het is in deze implementatie dus niet mogelijk -althans niet zonder kleine wijzigingen in dit programma aan te brengen- de instrumentatie los te koppelen van de kompositorische algoritmiek.

De eerste regel maakt de keyboard-buffer leeg:

```
50 E = KEYPAD
   IF KEYPAD >< 0 THEN GOTO 50
```

Vervolgens schakelen we het schermpje aan en wissen wat er eventueel nog op te zien was.
We vragen de gebruiker zijn keuzecijfer in te brengen:

```
51 DEVICE 6
   OUTLCD 1
   PRINT "Move? ";
```

Wacht nu tot er een toets ingedrukt wordt:

```
52 E = KEYPAD
   IF E = 0 THEN GOTO 52
```

Bereken nu de waarde van de variabele E (cfr. sub.3.2.2.2.).
Druk de waarde van de keuze af op het scherm.
Stuur eventuele output van het board terug naar de terminal uitgang.

```
53 E = E - $30
   OUTLCD $80
   PRINT "Move= ";E
   DEVICE 0
54 IF E = 0 THEN STOP
```

Op grond van de gemaakte keuze wordt nu de synthesizer ingesteld en springt het programma - omwille van de snelheid gebeurt dit via goto's- naar de betreffende programmamodules. Omdat de regelnummers niet direkt de leesbaarheid van de kode bevorderen, hebben we de namen van de modules als labels toegevoegd:

Open:

```
55 IF E = 1 THEN D = P(26,26,26,26,26,26,26,26)
   GOTO 70
```

Topoi:

```
56 IF E = 2 THEN D=P(5,35,19,25,32,45,79,65)
   GOTO 80
```

Minor:

```
57 IF E = 3 THEN D = P(10,10,10,10,10,10,10,10)
   GOTO 120
```

Beat:

```
58 IF E = 4 THEN D = P(8,18,75,89,52,58,63,67)
   GOTO 140
```

Rising:

```
59 IF E = 5 THEN D = P(12,10,6,16,66,36,40,60)
   GOTO 160
```

Sforte:

```
60 IF E = 6 THEN D = P(76,76,76,76,76,76,76,76)
   GOTO 185
```

Lead:

```
61 IF E = 7 THEN D = P(60,61,15,35,50,45,125,56)
   GOTO 200
```

Canvas:

```
62 IF E = 8 THEN D = P(57,74,93,126,84,9,14,106)
   GOTO 220
```

Close:

```
63 IF E = 9 THEN D = P(6,10,12,16,26,36,40,47)
   GOTO 250
```

Prime-time:

```
64 IF E = 10 THEN D = P(85,52,50,44,54,63,32,85)
   GOTO 280
```

Call:

```
65 IF E = 11 THEN D = P(11,11,11,11,11,11,11,11)
   GOTO 300
```

Spooky:

```
66 IF E = 12 THEN D = P(9,29,4,9,9,9,9,9)
   GOTO 350
```

Rec-Play:

```
67 IF E = 13 THEN D = P(55,55,55,88,55,55,55,55)
   GOTO 400
```

Lock-Unlock:

```
68 IF E = 14 THEN D = P(19,13,58,102,112,92,67,67)
   GOTO 450
```

Hammers:

```
69 IF E = 15 THEN D = P(36,36,36,22,115,50,8,8)
   GOTO 475
```

3.2.2.9.: De modules van 'A Book of Moves'

De hierna volgende stukjes programma vormen elk voor zich een verschillende mogelijkheid om het instrument te programmeren. Zij leggen, afhankelijk van het geval tevens in meer of mindere mate kompositorische en estetische samenhangen vast tussen de sonore gebeurtenissen onderling. De kontrole gaat in elk geval steeds uit van de bewegende uitvoerder. De mate waarin het muzikale rezultaat door de bespeler beinvloed of bepaald kan worden is voor elke module verschillend.

In deze modules wordt uitgebreid gebruik gemaakt van de multi-tasking mogelijkheden van dit komputersysteem. De kode voor de taken (tasks) die in de modules gebruikt wordt vormt het onderwerp van een afzonderlijke paragraaf.

3.2.2.9.1.: " Open "

In deze module wordt de bewegingsinput tot een strikt minimum beperkt. Het algoritme speelt een enkel breed en komplex akkoord bestaande uit 8 noten:

De uitvoerings-instruktie is teatraal en retorisch gedacht: bij het begin van de kompositie 'A Book of Moves' zit de uitvoerder samengebald in het centrum van de denkbeeldige tetraeder gevormd door de hoekpunten van het Holosound-systeem. De achtklank wordt alsmaar opnieuw aangeslagen.

Pas na een tijdje gaat de uitvoerder bewegen. Telkens wanneer hij beweegt, stopt het akkoord. Wanneer hij niet meer beweegt, herbegint de dreun. Echter, bij elke herhaling van dit gebeuren, verstilt het akkoord een stap. De beweger dient evenwel alsmaar bewegingsenergie te investeren om het geluid nog tot zwijgen te brengen. De beweging gaat dus in een diskontinu crescendo, terwijl de klank in decrescendo gaat. De module eindigt wanneer de uitvoerder het akkoord definitief het zwijgen heeft opgelegd.

Het instrument wordt hier dus in hoofdzaak aangewend als een 'trigger' systeem via de bewegingsdetektie. Technisch gezien dus eigenlijk bijzonder triviaal, echter volkomen ingegeven door overwegingen van kompositorische en teatraal-retorische aard. Het richt de aandacht van het publiek naar het allereenvoudigste kwazi binaire verband tussen klank en beweging.

J stelt het geluidsvolume bij het begin in op maximum.
I stelt de gevoeligheid van de bewegingsdetektie bij de aanvang van de module in.
De START instruktie vraagt de multitasker om taak 1 met een periodiciteit van 200ms uit te voeren. De bewegingsparameters worden dus vijfmaal per sekonde ververst.

```
70 J=$7F
  I=3
  START 1,20
```

Ingangspunt van de programmalus:
De lus speelt 8 noten na elkaar, wat gezien de uitvoeringssnelheid klinkt als een akkoord. De waarden van de noten worden gelezen uit een opzoekingstabel, vastgelegd aan het eind van het programma, (Een 'look-up table') en overgebracht naar de gedimensioneerde variabele N(M) gebruikt in de eerder gedefinieerde midi-funkties.
Voor elke afzonderlijke noot uit het akkoord wordt een verschillend midi-kanaal gebruikt. De panning geeft aan het klinkend rezultaat een frenetiek karakter: immers zij wisselt na elke herhaling gestuurd door een toevalsfunktie RND.

```
71 FOR M = 0 TO 7
   D = U($B0+M) + U(10) + U(64 + (RND / 512))
   D = U($90+M) + U(PEEK($DF00 + M)) + U(J)
   NEXT M
   U(BXOR(BOR(RND,$FF80),$FF80))
```

Gebruik nu de gedefinieerde tijdfunktie voor de vastlegging van het speeltempo:

```
72 D = S(50)
```

Wanneer de som van de gedetekteerde bewegingsamplitudes in de drie kanalen kleiner is dan de ingestelde gevoeligheid (I), herhaal het akkoord:

```
73 IF X + Y + Z < I THEN GOTO 71
```

Verminder bij elke doorgang van de lus het volume (J) met 5-midi eenheden. Laat de waarde van J echter niet negatief worden:

```
74 J = J - 5
   IF J <= 0 THEN J = 0
   GOTO 78
```

Indien de gedetekteerde bewegingshoeveelheid groter was dan de ingestelde gevoeligheid, verminder ook deze gevoeligheid (I wordt in stapjes van 2 eenheden vergroot). Dit maakt het voor de speler alsmaar lastiger om het geluid te doen stoppen.
Gebruik de tijdsfunktie om een rust van minstens een sekonde in te brengen:

```
75 I = I + 2: D = S(100)
```

Wanneer de bewegingsamplitude de gevoeligheid overtreft, worden geen verdere akkoorden tot klinken gebracht:

```
76 IF X + Y + Z > I THEN GOTO 76
```

Het algoritme kan ook onderbroken worden door een willekeurige toets op het toetsenbordje in te drukken. Dit is voorzien voor het geval dat de speler er niét zou in slagen het akkoord tot stilte te brengen...

```
77 IF KEYPAD = 0 THEN GOTO 71
```

Schakel taak 1 voor de multitasker weer uit (Cancel), en spring weer naar het begin van de lus:

```
78 CANCEL 1
   GOTO 50
```

3.2.2.9.2.: " Topoi "

Dit algoritme vormt een goede demonstratie van de mogelijkheid van ons systeem om via bewegingsdetektie aan plaatsbepaling te doen. De kode probeert om aan de hand van de informatie van de analoge komputer, te achterhalen waar de speler zich binnen het veld bevindt wanneer hij beweegt. Men zou een soort topografische analyze van de data kunnen noemen.

Op grond van de berekende plaats waar de speler zich bevindt, worden dan andere akkoordsamenstellingen tot klinken gebracht. Deze akkoorden zijn ook hier weer op voorhand ondergebracht in een 'look-up' tabel. Hun samenstelling en onderling harmonisch verband behoort tot het strikt kompositorisch aspekt van het stuk. Wel werden de akkoorden zo gekozen, dat de mapping van de bewegingsplaatsen op hun al dan niet klinken, een spelen van melodische en harmonische muzikale zinnen mogelijk maakt.

De eerste analoge komputer dient wel ingesteld te worden voor de ring-vermenigvuldiger patch. (Onderlinge vermenigvuldiging der signalen). Anders klopt de onderliggende rekenkundige analyze niet meer. De software in deze module laat toe 15 verschillende plaatsen te onderscheiden. De achterliggende logika laat zich wellicht het best aan de hand van een tekening illustreren. Wel moet men zich hierbij voor ogen houden dat de in de tekening uiteraard scherp getekende grenslijnen en ingekeurde vlakafbakeningen, in de praktijk niet zo heel precies te bepalen zijn. Een en ander hangt sterk af van de akoestische eigenschappen van de ruimte waarbinnen wordt gespeeld. Korrekties zijn overigens wel aan te brengen via enkele regelaars in het analoog-komputergedeelte. Een opstellingshulproutine werd overigens ook voorzien op het nivo van deze software.

De module maakt gebruik van gegevens verkregen via taak 18, (zie verder) die tienmaal per sekonde wordt aanroepen. De analyze van de bewegingsinformatie maakt gebruik van de vektoriele bewegingssnelheden XF,YF,ZF.

Tekening van de plaatsbepalingslogika in "Topoi":

De taak wordt tienmaal per sekonde in de agenda van de multitasker opgenomen:

```
80 START 18,10
```

Ingangspunt van de programmalus:
Wanneer niets verandert in vergelijking met de voordien waargenomen bewegingssituatie, gebeurt er ook in de muziek niets:

```
81 IF (XF = XH) AND (YF = YH) AND (ZF = ZH) THEN GOTO 81
```

Ook wanneer de amplitudes 0 zijn, gebeurt niets. Dit is niet helemaal redundant, omdat de frekwentiedata iets trager reageren op de werkelijkheid dan die van de amplitude.

```
82 IF BOR(BOR(X,Y),Z) = 0 THEN GOTO 81
```

Hier volgt dan de analyze van de plaats van de beweger:
Wanneer geen beweging gedetekteerd wordt zetten we de case-variabele gelijk aan 1.

```
83 IF A = 0 THEN C = 1
   GOTO 110
```

Een beweging in het middelpunt van de denkbeeldige tetraeder, geeft aanleiding tot geval 2:

```
84 IF BOR(YF,ZF) = 0 THEN C = 2
   GOTO 99
```

Beweging in de linker-hoek:

```
85 IF BOR(XF,YF) = 0 THEN C = 3
   GOTO 99
```

Beweging in de rechterhoek:

```
86 IF BOR(XF,ZF) = 0 THEN C = 4
   GOTO 99
```

Beweging die als identiek gezien wordt vanuit de drie vektoren. Dit is het geval wanneer de speler zich in het centrum van de opstelling 'opblaast' of 'ineenkrimpt'.

```
87 IF BAND(XF,YF) = ZH THEN C = 5
   GOTO 99
```

Beweging op de centrale as:

```
88 IF YF = ZF THEN C = 6
   GOTO 99
```

Beweging op de linker X-Z as:

```
89 IF XF=YF THEN C=7: GOTO 99
```

Beweging op de rechter Z-Y as:

```
90 IF XF=ZF THEN C=8: GOTO 99
```

Bepaling van de tusseninliggende overgangsposities:

```
91 IF (XF < YF AND XF < ZF) AND (YF < ZF) THEN C = 9
```

```
   GOTO 99
92 IF (XF < YF AND XF < ZF) AND (YF > ZF) THEN C = 10
   GOTO 99
93 IF (YF < XF AND YF < ZF) AND (XF < ZF) THEN C = 11
   GOTO 99
94 IF (YF < XF AND YF < ZF) AND (XF > ZF) THEN C = 12
   GOTO 99
95 IF (ZF < XF AND ZF < YF) AND (XF < YF) THEN C = 13
   GOTO 99
96 IF (ZF < XF AND ZF < YF) AND (XF > YF) THEN C = 14
   GOTO 99
```

Indien geen van de vorige gevallen gevonden werd, voert het programma de analyze opnieuw uit:

```
97 GOTO 81
```

De C-variabele is nu in elk geval bepaald. Op grond hiervan zal de mapping van de plaats op de muzikale samenklanken gebeuren. De look-up tabel start op geheugenadres $DF00:

```
99 FOR M = 0 TO 7
```

Lees per kanaal de van het geval afhankelijke noot, en wanneer geen noot wordt opgegeven, zoek dan de opgegeven noot voor de volgende muzikale stem of midi-kanaal.

```
100 N(M) = PEEK($DF00 + M + (C * 8))
```

Indien een noot in de tabel werd gevonden, transponeer haar dan naar een bruikbare oktaafligging en transpositie. De konstante 36 schuift de noten 3 oktaven naar omhoog. Vanaf deze ligging hebben we voorts de uiteindelijke oktaafligging afhankelijk gemaakt van de globale bewegingsamplitude door bij de numerieke waarde voor de noot (12*(A/30)) op te tellen.

```
101 IF N(M) THEN N(M)=N(M)+36+(12*(A/30))
```

Wanneer de beweging gelokalizeerd werd in 'overgangszones', en wanneer er geen noot in de tabel gegeven wordt, laat de te spelen noot afhangen van de vektoriele bewegingssnelheid op het moment nu:

```
102 IF (C > 8 AND M = 0) AND (N(M) = 0) THEN N(M) = 60 + XF
103 IF (C > 8 AND M = 1) AND (N(M) = 0) THEN N(M) = 60 + YF
104 IF (C > 8 AND M = 2) AND (N(M) = 0) THEN N(M) = 60 + ZF
```

Wanneer een noot lager zou klinken dan MIDI-waarde 36, schrap ze:

```
105 IF N(M) < 36 THEN N(M) = 0
    V(M) = 0
```

Wanneer er een noot staat, geef haar dan een geluidssterkte afgeleid van de momentane bewegingsparameters:

```
106 IF N(M) THEN V(M) = A
```

Indien de gevonden noten verschillend zijn van de voordien in dezelfde stem gespeelde, schakel dan de vorige noot uit en de nieuwe aan:

```
107 IF N(M) >< NO(M) THEN
    D = U($80+M) + U(NO(M)) + U(0) + U($90+M) + U(N(M))+U(V(M))
```

Sla de zopas gespeelde noten op in de gedimensioneerde variabele NO():

```
108 NO(M) = N(M)
```

Sluit de kanaalteller-lus:

```
109 NEXT M
```

Merk op dat in deze module nergens een tijd- of ritme instruktie voorkomt. Deze muzikale parameters volgen hier vooral op grond van de opgegeven periodiciteitsparameter bij het opstarten van de gebruikte taak.

Sla ook de vorige bewegingstoestandsparameters op in variabelen:

```
110 XH = XF
    YH = YF
    ZH = ZF
```

Ga na of er zich een exit-konditie voordoet (wanneer er een toets gedrukt wordt op het toetsenbordje). Zoniet, ga terug naar het begin van de lus.

```
111 IF KEYPAD = 0 THEN GOTO 81
```

Indien er wel een toetst werd ingedrukt, beëindig deze module en ga terug naar het hoofdmenu.

```
112 CANCEL 18
    GOTO 50
```

3.2.2.9.3.: " Minor"

Deze module is een software implementatie van een onzichtbare piano. De via de bewegingen gespeelde noten cirkelen centripetaal rond een a priori vastgelegd tonaal centrum. De gebruikte noten maken gebruik van een kleine tertstoonladder.
De oktaafpositie waarin de noten worden gespeeld is afhankelijk gemaakt van de globale bewegingshoeveelheid.

De intervallische afwijking tegenover het tonaal centrum is afhankelijk gemaakt van de vektoriele bewegingssnelheden. (De XF,YF,ZF parameters). Wanneer de uitvoerder redelijk symmetrisch beweegt ten opzichte van de drie ontvangers op de hoekpunten van de denkbeeldige tetraeder, klinkt de noot die overeenstemt met het tonaal centrum van de muziek via midi-kanaal 7.

De intervallen ontleend aan de mineur toonladder zijn ingeschreven in een look-up tabel in het geheugenadresgebied dat uitgelezen wordt met PEEK($DFF0) tot PEEK($DFFF). Het tessituurbereik is twee oktaven en een kleine terts.

Muzikaal gesproken klinkt deze module eerder 'romantisch'.

Taak 1, de enige die in multitasking gebruikt wordt in deze module, levert de bewegingsparameters in de variabelen:

XF,YF,ZF bereik 0 -127 (7 bits)
A bereik 0 -127
X,Y,Z bereik 0 -127

Reset alle variabelen en start taak 1 tienmaal per sekonde:

```
120 XF = 0
    YF = 0
    ZF = 0
    START 1,10
```

Ingangspunt van de programmalus:

```
121 IF A = 0 THEN GOTO 121
```

De variabele I in volgende formule bepaalt de oktaafligging van de klinken noten. Het bereik van I loopt van 24 tot 72. De oktaafpositie zal afhangen van de bewegingssnelheden. Een konstante bepaalt een minimum waarde voor de geluidssterkte, om de dynamiek in deze rustig klinkende module niet te groot te maken:

```
122 I = 36 + (12 * ((XF + YF + ZF) / 92))
```

Wanneer de bewegingsamplitudes in de X,Y,Z vektoren groter zijn dan nul, voer dan de mapping uit van de bewegingssnelheden naar de muzieknoten:

```
123 IF X THEN D = U($90) + U(I+PEEK($DFF0+(XF/8))) + U(X)
124 IF Z THEN D = U($91) + U(I+PEEK($DFF0+(ZF/8))) + U(Z)
125 IF Y THEN D = U($92) + U(I+PEEK($DFF0+(YF/8))) + U(Y)
```

Als de speler zich in het centrum van de tetraeder bevindt, speel dan ook de noot die het tonaal centrum vormt. De konditie met ZF werd toegevoegd om deze noot toch niet teveel te laten klinken.

```
126 IF ZF > 24 AND BAND(XF,YF) = ZF THEN
                                    D = U($97) + U(I) + U(A / 2)
     GOTO 130
```

Wanneer hij zich ook helemaal in het centrum bevindt, speel dan ook het onderoktaaf van het tonaal centrum:

```
127 IF YF = ZF THEN
    D = U($93) + U(I+12) + U(A/2) + U($96) + U(I-12) + U(A/2)
```

Beweegt hij links van het centrum, voeg dan aan het tonaal centrum de bovenkwint toe:

```
128 IF XF = YF THEN D = U($94) + U(I+7) + U(A)
```

Beweegt hij rechts, speel dan een kwint omlaag:

```
129 IF XF = ZF THEN D = U($95) + U(I - 7) + U(A)
```

De ritmische struktuur is afhankelijk gemaakt van de bewegingsamplitude: hoe groter deze wordt, hoe sneller de noten elkaar zullen opvolgen:

```
130 D = S(46 -(A / 3))
```

Test de exit-konditie:

```
131 IF KEYPAD = 0 THEN GOTO 121
```

Indien die konditie is voldaan, schakel taak 1 uit en spring terug naar het hoofdmenu.

```
132 CANCEL 1
   GOTO 50
```

`3.2.2.9.4.: " Beat "

In deze algoritmische module worden de bewegingsparameters gemapt op een denkbeeldig uitgebreid slagwerkinstrumentarium. Om dit effektief te laten zijn moet het uiterst responsief zijn. De te instruktie schrijft bruuske karate-achtige bewegingen voor: kort en met een duidelijk gemarkeerd begin en einde. Geen enkele 'elegantie' is hier toegestaan. Zoniet onstaat er sonore rommel.

Start taak 2 met een interval van 80ms:

```
140 START 2,8
```

De slaginstrumenten worden konditioneel gespeeld:

```
141 IF C = 0 THEN GOTO 155
```

Links in het stereobeeld komt een eerste slagwerkgroep:

```
142 IF XF AND X THEN D = U($90) + U(XF * 2) + U($7F)
```

Rechts plaatsen we de tabla:

```
143 IF YF AND Y THEN D = U($92) + U(YF * 2) + U($7F)
```

Centraal staat een tweede slagwerkensemble:

```
144 IF ZF AND Z THEN D=U($91)+U(ZF*2)+U($7F)
```

Een bijzondere konditie bepaalt wanneer de Shinto-gong gespeeld wordt:

```
145 IF X + Y + Z > 36 AND XF + YF + ZF > 90 THEN
    D = U($96) + U((XF + YF + ZF + 24) / 2) + U(A)
    GOTO 150
```

Rammelaars en Japanse bellen samen:

```
146 IF X AND Y THEN
    D = U($93) + U(XF+YF) + U(X) + U($94) + U(YF*2) + U(Y)
```

Rammelaars en gestemde glissando-trommels:

```
147 IF X AND Z THEN
    D = U($93) + U(XF + ZF) + U(X) + U($95) + U(ZF * 2) + U(Z)
```

Japanse bellen en glissando-trommels samen:

```
148 IF Y AND Z THEN
    D = U($94) + U(YF + ZF) + U(Y) + U($95) + U(ZF * 2) + U(Z)
```

Laat de grote Miya Daiko trommel klinken:

```
150 IF (X + Y + Z) > 120 THEN
    D = U($97) + U((X + Y + Z) / 3) + U($7F)
```

De volgende instruktie maakt de tijdsduur van de gesampelde slaggeluiden afhankelijk van de piek-amplitude van de geregistreerde beweging:

```
151 D = S(1 + L)
```

Voor het merendeel der gebruikte slaginstrumenten is het niet nodig de aangeslagen noten ook uit te schakelen. Ze sterven vanzelf uit. Sommige instrumenten moeten terwille van de muzikale transparantie, toch gedempt worden:

```
154 D=U($B4)+U($7B)+U(0)+U($B5)+U($7B)+U(0)+U($B6)+U($7B)+U(0)
```

Test de exit-konditie. Indien voldaan, schakel taak 2 uit en ga terug naar het hoofdmenu:

```
155 IF KEYPAD = 0 THEN GOTO 141
156 CANCEL 2
    GOTO 50
```

Deze implementatie staat of valt met de gebruikte synthesizer evenals met de instrumentatie waarmee de module wordt aanroepen. De Proteus-synthesizer waarvoor ze gedacht is, beschikt over een ruim arsenaal aan ingebouwde 16-bit samples van etnische slaginstrumenten. Deze samples werden in ruime mate op voorhand 'ge-edit' in funktie van hun gebruik in deze module. Deze editing is evenwel niet mogelijk binnen het opzet van het hier voorgestelde programma. Daarvoor schreven we een afzonderlijk programma in Microsoft PDF-Basic dat evenwel buiten het opzet van deze studie valt. De naamgeving der instrumenten is overgenomen uit de handleiding bij de betreffende synthesizermodule.

3.2.2.9.5.: " Rising "

Dit algoritme bestaat uit een vanuit de bewegingsinformatie gestuurde langzame en extreme opwaartse akkoordprogressie.
De bewegingsparameters worden hier o.m. gebruikt als informatie voor de toonwielkontrole (PMD, pitch-modulation device) van de gebruikte synthesizer. Hierdoor kan de toonhoogte na de aanslag van elk afzonderlijk akkoord vanuit kleine bewegingen gemoduleerd en gebogen worden, wat een dramatisch en spannend effekt geeft.

De interaktiviteit is in dit algoritme gesteund op een eenvoudige topologische analyze van de bewegingsvektoren.
Wanneer de ruimtelijke plaats zich wat de beweging betreft herhaalt of konstant blijft, worden de toonbuigingen waarvan we spraken uitgevoerd. Vanzodra de uitvoerder op een iets andere plek bewegingen uitvoert, zal de akkoordprogressie een stapje omhoog klimmen.

Taak 1 levert ons hier volgende parameters:

A bereik 0 -127
X,Y,Z bereik 0 -127
XF,YF,ZF worden niet gebruikt.

Reset de teller voor de progressie. Start taak 3 om de 30ms. Reset de vorig-geval variabele op 1:

```
160 J = 0
    I = 1
    START 1,3
```

Ingangspunt van de programmalus:

```
161 IF X + Y + Z = 0 THEN C = 1
    GOTO 170
```

Wanneer de beweging in het centrum plaatsvindt, verhoog de J teller met een waarde evenredig met de amplitude van de gedetekteerde beweging. Hierdoor wordt de stijgende progressie mogelijk gemaakt:

```
162 IF A AND A / 3 = (X + Y + Z) / 9 THEN C = 15
    J = J + 1 + (A / 3)
    GOTO 170
```

Voer de plaatsanalyzes uit en bepaal een waarde voor C in funktie daarvan:

```
163 IF X = Y THEN C = 16
    GOTO 170
164 IF Y = Z THEN C = 17
    GOTO 170
165 IF X = Z THEN C = 18
    GOTO 170
166 IF X > Y AND X > Z THEN C = 19
    GOTO 170
167 IF Y > X AND Y > Z THEN C = 20
    GOTO 170
168 IF Z > X AND Z > Y THEN C = 21
```

Ingangspunt van de programmaspong na de plaatsbepaling:

```
170 IF C=I THEN GOTO 177
```

Wanneer het geval (C) gelijk is aan het vorige, realiseer dan de toonhoogtebuigingsmogelijkheid voor de bespeler via een sprong naar 530, is dat niet het geval, speel dan de noten van het nieuwe akkoord in de progressie:

De M-lus loopt alle muzikale stemmen of midi-kanalen af die hier gebruikt worden (8):

```
171 FOR M=0 TO 7
```

De noten worden opgezocht in een speciaal segment van de look-up tabel in het RAM geheugen:

```
172   N(M) = PEEK($DF00 + (C * 8) + M)
```

Wanneer er geen noot staat, zet dan ook het volume op nul. Dit heeft te maken met de 'stommiteit' van midi waarbij noot 0 overeenstemt met een werkelijk opgewekte toonhoogte, een hele diepe Do.

```
   IF N(M) = 0 THEN V(M) = 0
      GOTO 175
```

De konstante 36 bepaalt in welke oktaafligging de akkoordprogressie een aanvang zal nemen. Het volume wordt bewegingsafhankelijk opgeslagen in de gedimensioneerde variabele V(). De IF konditie verhindert overflow. Het midi-bereik is immers slechts 7-bits, of decimaal 0-127:

```
173   N(M) = N(M) + 36 + J
      V(M) = 63 + ((A) / 2)
      IF N(M) > $7F THEN N(M) = $7F
```

Aankomstpunt van de programmasprong. De vorige noot wordt uitgeschakeld als er een nieuwe te spelen is en de nieuwe wordt aangeslagen:

```
174 IF N(M) AND N(M) >< NO(M) THEN
    D = U($80+M) + U(NO(M)) + U(0) + U(144+M) + U(N(M)) +U(V(M))
```

De pas gespeelde noot wordt in de geheugenvariabele NO() opgeslagen voor gebruik in de volgende lusdoorgang.

```
175 NO(M) = N(M)
```

De kanaal-lus wordt terug afgesloten:

```
176 NEXT M
```

Ingangspunt van de instrukties die betrekking hebben op de tijdsstruktuur van de progressie. Kijk op de interne klok en sla de waarde op in T:

```
177 T = SYSTIC
```

Uitvoering van de pich-bend (toonhoogtemodulatie) via de PMD midi-sekwens. Ook hier wordt een kanaaltellerlus gebruikt met M:

```
178 FOR M = 0 TO 7
179 IF N(M) THEN D = U($E0 + M) + U(0) + U(A)
180 NEXT M
```

Hoe groter de bewegingsamplitude hoe sneller de progressie vooruitgaat en hoe responsiever het instrument gaat reageren:

```
181 IF SYSTIC < T + 64 - (A / 2) THEN GOTO 178
```

Sla het nummer van het vorig geval op in de variabele I en test de interne exit-konditie voor variabele J. De module eindigt dus op grond van haar intern muzikaal verloop. Niettemin werd toch een mogelijkheid ingebouwd haar te kunnen onderbreken via het kleine toetsenbord door het indrukken van een willekeurige toets:

```
182 I = C
    IF J < 75 AND KEYPAD = 0 THEN GOTO 161
```

Exit-punt van de module, waarbij taak 1 wordt opgeheven en terug wordt gekeerd naar de hoofdmenu-routine:

```
183 CANCEL 1
    GOTO 50
```

De initiele mapping van de akkoorden ziet eruit alsvolgt:

3.2.2.9.6.: " Sforte "

In dit algoritme wordt een imaginaire klavierspeler geimplementeerd. Er wordt gebruik gemaakt van de verschillen (delta-waarden) tussen zes opeenvolgend genomen samples van de bewegingsinformatie.
Er wordt gebruik gemaakt van taak 4, een taak die in de variabelen X, Y, Z waarden tussen -30 en +30 retourneert. Deze variabelen bevatten dus de versnellingsinformatie met betrekking tot de beweging van de speler.
Indien de bekomen intervallen te klein worden om muzikaal bruikbaar te zijn, moet de tijd tussen het nemen van de individuele samples vergroot worden. In het omgekeerde geval kan ofwel de tijd verkleind worden, ofwel kan de waarde van de deler vergroot worden. Voorts wordt nog gebruik gemaakt van de globale amplitude-variabele A die hier echter diskontinu werkt: hij is 0 ofwel iets tussen 64 en 127.

Start taak 4 elke 40ms:

```
185 START 4,4
```

De waarden van de amplitude en van de te spelen noten worden gedurende deze module doorlopend ververst iets wat alleen bij gebruik van multitasking tot de programmeermogelijkheden behoort. Hier begint de programmalus:

```
186 IF A = 0 THEN GOTO 186
```

Volgende noten worden op het linkerkanaal uitgestuurd:

```
188 IF X THEN D = U($90) + U(64+XF)+U(A)+U($95)+U(64-XF)+U(A)
```

Volgende klinken rechts:

```
189 IF Y THEN D = U($92) + U(64+YF)+U(A)+U($97)+U(64-YF)+U(A)
```

De volgende worden in het midden van het geluidsbeeld geplaatst:

```
190 IF Z THEN D = U($91) + U(64+ZF)+U(A)+U($96)+U(64-ZF)+U(A)
```

Bijzondere kondities met betrekking tot te spelen noten waardoor een bepaald tonaal centrum wordt gerealizeerd:

```
191 IF A > 48 AND X + Y + Z = 0 THEN D = U($93) + U(64)+U(120)
192 IF X + Y + Z > 64 THEN D = U($94) + U(40) + U(A)
```

Tempobepalende routine:

```
193 D = S(16)
```

Test de exit konditie om de module te verlaten:

```
194 IF KEYPAD = 0 THEN GOTO 186
```

Wanneer aan de exit konditie is voldaan, keer terug naar het hoofdkeuzemenu:

```
195 CANCEL 4
    GOTO 50
```

3.2.2.9.7.: " Lead "

Deze module beoogt een eenvoudige implementatie te zijn van een enigszins bizar samengesteld groot symfonisch orkest. Het wordt gespeeld door het denkbeeldig aanwezige orkest te dirigeren.
De samenklanken zijn 'modernistisch' , dissonant, dreigend en met abrupte inzetten. Samenklanken worden aangehouden door de beweging te stoppen.

Er wordt gebruik gemaakt van taak 3, een taak die volgende zeven variabelen retourneert:

A bereik: 0 - 127
X,Y,Z bereik: 0 - 127
XF,YF,ZF bereik: 0 - 127

Start taak 18 om de 80ms:

```
200 START 18,8
```

Startpunt van de programmalus:

```
201 IF X + Y + Z = 0 THEN GOTO 201
```

De mapping gebeurt op een uiterst direkte wijze. Het is door dit 'gebrek aan muzikale samenhang' op het harmonische vlak dat het kompleks en dreigend karakter tot stand komt. De noot N(0) klinkt links, N(2) rechts, en N(1) in het centrum:

```
 202 IF X THEN N(0) = X
203 IF Y THEN N(2) = Y
204 IF Z THEN N(1) = Z
```

Wanneer de noten te laag zouden uitvallen, transponeer ze dan drie oktaven omhoog:

```
205 IF N(0) AND N(0) < 48 THEN N(0) = N(0)+36
206 IF N(1) AND N(1) < 48 THEN N(1) = N(1) + 36
207 IF N(2) AND N(2) < 48 THEN N(2) = N(2) + 36
```

Bepaal een minimum geluidssterkte voor de A variabele. Bescherm evenwel A tevens voor overflow-kondities:

```
208 IF A THEN A = A + 48
    A = BXOR(BOR(A,$FF80),$FF80)
```

Speel de noten uit de mapping, waarbij telkens de vroegere noot samen met de nieuwe noot wordt gespeeld. Dit verhoogt het idee van muzikale samenhang aanzienlijk. Merk op dat de A variabele gedurende de loop van deze instruktie kan veranderen, wat een rijk geschakeerde dynamiek oplevert:

```
209 FOR M = 0 TO 5
    D = U($90 +M) + U(N(M)) + U(A)
    NEXT M
```

Onthoud de zopas gespeelde nieuwe noten en roep de tijdsroutine aan:

```
210 N(3) = N(0)
    N(4) = N(1)
    N(5) = N(2)
    D = S(15)
```

Test het toetsenbordje op een exit konditie:

```
211 L = KEYPAD
    IF L = 0 THEN GOTO 201
```

Indien echter toets 7 werd ingedrukt, verander dan de bron van bewegingsinformatie van taak 18 naar taak 1 en ga door met de muziek:

```
212 IF L = $37 THEN CANCEL 18
    START 1,8
    GOTO 201
```

Invalspunt voor de beeindiging van de module, waarbij tevens het slotakkoord wordt gespeeld:
C - G# - Eb - F - D

```
214 D=U($90) + U(72) + U($7F) + U($91) + U(80) + U($7F)
215 D=U($92)+U(87)+U($7F)+U($93)+U(89)+U($7F)+U($94)+U(98)+U($7F)
```

Haal de timerwaarde op ter bepaling van de duur van het slotakkoord:

```
216 T = SYSTIC
```

Tijdens de duur van dit akkoord staat de dirigent hopeloos te zwaaien en verliest de kontrole over zijn orkest. Alleen de globale dynamiek kan hij nog beinvloeden. Dit gebeurt via de midi-parameter voor 'master'-volume:

```
217 FOR M = 0 TO 5
    D = U($B0 + M) + U(7) + U(64 + (A / 2))
    NEXT M
```

Dit akkoord wordt 10 sekonden aangehouden:

```
218 IF SYSTIC < (T + 1000) THEN GOTO 217
```

Indien nu de bediener van het komputerboard een vergissing zou maken en een verkeerde toets zou indrukken, zou een taak niet worden opgeheven. Daarom schakelden we hier beide taken uit hoewel een van beide steeds overbodig is. De kompiler scheen hiertegen echter geen bezwaar te maken.

```
219 CANCEL 1
    CANCEL 18
```

Keer terug naar het hoofdmenu:

```
  GOTO 50
```

3.2.2.9.8.: " Canvas "

Deze mapping herschept ons onzichtbaar instrument in het schilderdoek van een sonore 'action-painter'.

Het algoritme is bijzonder responsief en maakt gebruik van toonladders, toonhoogtebuigingen en glissandi evenals hoog-plastische volumekontrole.

Taak 1 wordt gebruikt en retourneert:

X,Y,Z 0-127
A 0-127
XF,YF,ZF 0-127

Taak 1 wordt om de 50ms uitgevoerd:

```
220 START 1,5
```

Ingangspunt van de programmalus:
Klank links:

```
221 IF X > 36 AND X < 96 THEN D = U($90) + U(X+30) + U($7F)
```

Klank rechts:

```
222 IF Y > 36 AND Y < 96 THEN D = U($92) + U(Y + 30) + U($7F)
```

Klank in het midden:

```
223 IF Z > 36 AND Z < 96 THEN D = U($91) +U(Z + 30) + U($7F)
```

Extra klanken en geluiden (er worden immers 'niet-muzikale' geluiden gebruikt zoals kolkend water, wind en storm...), waarbij diverse parameters uit de beweging op diverse klankparameters gemapt worden:

```
224 IF X AND X < 36 THEN D = U($93) + U(X + 36) + U(BOR(A,64))
225 IF X > 64 THEN D = U($94) + U(X) + U($7F)
226 IF Y > 64 THEN D = U($95) + U(Y) + U($7F)
227 IF Y AND Y < 36 THEN D = U($96) + U(Y+36) + U(BOR(A,64))
228 IF Z > 64 THEN D = U($97) + U(Z) + U($7F)
```

Tijdsbepalende routine waarbij eerst op de systeemklok wordt gekeken en de tijd opgeslagen in de variabele L.

```
229 L = SYSTIC
```

Via de volgende instrukties worden de glissandi en toonhoogtemodulaties uitgevoerd:

```
230 D = U($E0) + U(0) + U(X) + U($E2) + U(0) + U(Y)
    D = U($E1) + U(0) + U(Z) + U($E4)
    D = U(0) + U(X) + U($E5) + U(0) + U(Y)
```

Test de tijd-verstreken konditie gekoppeld aan de bewegingsamplitude:

```
232 IF SYSTIC < L + 25 AND A THEN GOTO 230
```

Test de exit voorwaarde:

```
233 IF KEYPAD = 0 THEN GOTO 221
```

Verlaat het algoritme en keer terug naar het hoofdmenu:

```
234 CANCEL 1
    GOTO 50
```

3.2.2.9.9.: " Close "

Deze module vormt de antipode of het komplement van de eerste. Ze wordt gebruikt ter afsluiting van het Book of Moves en is dus muzikaal gezien een finale.

Hier echter worden akkoorden gespeeld dan en dan alleen wanneer er beweging gedetekteerd wordt. De akkoorden doorlopen een gradueel decrescendo dat alleen helemaal aan het einde weer even in een crescendo overgaat. Parallel met dit -klassiek retorisch- dynamisch verloop, volgen de toonhoogtes een dalend patroon tot de finale unisono in de acht stemmen (lage C) is bereikt.

Dit algoritme maakt slechts van de totale bewegingsinformatie gebruik, via taak 1.

Vul de gedimensioneerde variabelen (arrays) met nieuwe noten uit de look-up tabel die hier begint op het RAM-geheugenadres $DF00. Het beginakkoord is identiek aan het openingsakkoord van het stuk.

```
250 FOR M = 0 TO 7
    N(M) = PEEK($DF00 + M) + 24
    NEXT M
```

Set (J=beginvolume) en reset (I=teller) de variabelen en start taak 1 om de 30ms:

```
251 J = $7F
    I = 0
    START 1,3
```

Ingangspunt van de programmalus:

```
252 IF A < 32 THEN GOTO 252
```

Indien dit niet het geval is, speel de noten. Dit maakt een minimum bewegingsdrempel noodzakelijk waaronder er niets gebeurt.

```
253 FOR M = 0 TO 7
    D = U($90 +M) + U(N(M)) + U(J +(I * 8))
```

```
    NEXT M
```

Bepaling van de aanhoudtijdsduur voor het gespeelde akkoord:

```
254 D = S(16 + I + I + I)
```

Ga door met het spelen van hetzelfde akkoord zolang beweging gedetekteerd wordt:

```
255 IF A > 16 THEN GOTO 253
```

Wanneer echter geen beweging werd waargenomen, verminder dan het geluidsvolume via de J variabele (decrecendo). Begrens echter de laagste waarde van J tot 12 (muzikaal ppp).

```
256 J = J - 1
    IF J < 12 THEN J = 12
```

Zoek een willekeurig random getal tussen 0 en 7 dat kan gebruikt worden voor een willekeurige kanaaltoewijzing van de noten behorende tot het akkoord. (Dit implementeert een soort 'Klangfarbenmelodie'). Indien de laagste noot op sommige van de kanalen reeds werd bereikt en indien niet alle kanalen daar reeds zijn aangekomen, dan, als de willekeurig gekozen stem reeds op de laagste noot was aangekomen, zoek dan voort tot een stem is gevonden die nog niet op die laagste noot is aangekomen:

```
257 L = BXOR(BOR(RND,$FFF8),$FFF8)
     IF I AND (I < 8) AND (N(L) <= 36) THEN GOTO 257
```

Reset de I-teller waarin het aantal stemmen is opgeslagen die reeds hun laagste noot hebben bereikt. In de stemmen waarvoor dat niet het geval is, wordt de nootwaarde met een interval tussen een halve toon en een grote terts verminderd. Indien dit de noot te laag zou maken, begrens haar tot de laagste noot.

```
258 I = 0
    N(L) = N(L) - (BXOR(BOR(RND,$FFFC),$FFFC) + 1)
    IF N(L) < 36 THEN N(L) = 36
```

Tel het aantal stemmen waarin reeds de lage Do bereikt is:

```
259 FOR M = 0 TO 7
260     IF N(M) <= 36 THEN I = I + 1
261 NEXT M
```

Ga terug naar het begin van de lus wanneer aan de exit voorwaarde nog niet is voldaan:

```
262 IF I < 8 THEN GOTO 252
```

Exit instruktieregel van de hoofdprogrammalus, speel het slotakkoord (8 x unisono DO).

```
263 FOR M = 0 TO 7
        D = U($90+M) + U(36) + U($7F)
    NEXT M
```

Deze module kan niet door ingrepen via het toetsenbord worden onderbroken. Wanneer het slotakkoord is bereikt stopt ze vanzelf en heft haar informatieverwervende taak op. Op het akkoord wordt een orgelpunt geplaatst:

```
264 CANCEL 1
```

```
    D = S(1000)
```

Keer terug naar het hoofdkeuzemenu:

```
 265 GOTO 50
```

3.2.2.9.10.: " Prime Time "

Deze algoritmische module verloopt volledig polyritmisch. De zeven muzikale stemmen klinken alleen wanneer bewegingen gedetekteerd worden.

Voor het spelen van de noten wordt gebruik gemaakt van bijzondere taken in de multitasker. Dit maakt de polyritmiek erg eenvoudig realizeerbaar.

Start alle taken (nrs. 5-10) waardoor noten worden uitgestuurd in de diverse stemmen. De metrische verhoudingen voor de periodiciteit in de individuele stemmen zijn afgeleid van priemgetallen (x10). Elke taak vraagt een variabele voor de te spelen noot evenals een voor de aanslagsterkte van die noot.

Voor de informatieverwerving wordt gebruik gemaakt van taak 4. Deze taak wordt in het 'agenda' van de multitasker geplaatst om de 50ms. De taak levert de variabelen XF,YF,ZF met het bereik -30 tot +30, evenals de amplitude variabele 0-127, diskontinu.

```
280 START 5,20
    START  6,33
    START  7,50
    START 8,70
    START 9,110
    START 10,130
    START 11,170
    START 4,5
```

Ingangspunt van de programmalus:

De noten (array-variablele N()) worden toegewezen in funktie van de versnellingswaarden van de beweging en wel in termen van symmetrische afwijkingswaarden naar boven en naar onder ten opzichte van de centrale do van het klavier.

```
281  N(0) = 60 + XF
     N(1) = 60 + YF
     N(2) = 60 + ZF
     N(3) = 60 - XF
     N(4) = 60 - YF
     N(5) = 60 - ZF
     N(6) = 60 + (A / 2)
```

Voor de bepaling van de individuele volumes der stemmen wordt de vektoriele amplitudeinformatie rechtstreeks gebruikt.

```
282 V(0) = X :     V(3) = X
    V(1) = Y :     V(4) = Y
    V(2) = Z :     V(5) = Z
    V(6) = A
```

Test de voorwaarde -het indrukken van een toets- voor het verlaten van deze module:

```
283 IF KEYPAD=0 THEN GOTO 281
```

Verlaat de module , hef alle taken op en keer terug naar het hoofdmenu:

```
284 CANCEL 5: CANCEL 6: CANCEL 7: CANCEL 8: CANCEL 9
    CANCEL 10: CANCEL 11: CANCEL 4: GOTO 50
```

3.2.2.9.11.: " Call "

Deze module is muzikaal gezien driestemmig ontworpen. De stemmen cirkelen rond een tonaal centrum dat echter verschuift telkens de uitvoerder een beweging maakt die als zijnde in het centrum van de denkbeeldige tetraeder wordt waargenomen. De toonhoogte is een direkte mapping vanuit de vektoriele bewegingssnelheden.

Bij het begin wordt het tonaal centrum gelijk gesteld aan de centrale Do van het pianoklavier (60). De module reageert aanvankelijk aarzelend en traag, door de tijdsintervalvariabele L op 1.6s te bepalen. De geheugen-variabelen XH,YH,ZH worden gereset:

```
300 I = 36: L = 160
    XH = 0: YH = 0: ZH = 0
```

Ingangspunt van de programmalus:
Haal data voor het X-kanaal en beperk het bereik van XF:

```
301 XF = I + ADC(4)/24
```

Haal ook een amplitude parameter voor X en map deze in het bereik 0 of 37 tot 87:

```
302 V(0) = ADC(0) / 20
    IF V(0) THEN V(0) = 36 + V(0)
```

Wanneer de gevonden noot verschillend is van de vorige gespeelde noot binnen deze stem, speel haar:

```
303 IF XH >< XF THEN D = U($90) + U(XF) + U(V(0))
```

Herhaal nu deze procedure van het Y kanaal en laat deze noten klinken door het rechteraudiokanaal:

```
305 YF = I + ADC(5) / 24
306 V(1) = ADC(1) / 20
    IF V(1) THEN V(1) = 36 + V(1)
307 IF YH >< YF THEN D = U($92) + U(YF) + U(V(1))
```

Herhaal de procedure voor het Z-kanaal en map deze noten in het centrum van het stereobeeld:

```
309 ZF = I + ADC(6) / 24
310 V(2) = ADC(2) / 20
    IF V(2) THEN V(2) = 36 + V(2)
311 IF ZH >< ZF THEN D = U($91) + U(ZF) + U(V(2))
```

Schakel alle eerder gespeelde noten uit:

```
312 D = U($81)+U(ZH)+U(0)+U($82)+U(YH)+U(0)+U($80)+U(XH)+U(0)
```

Sla de gespeelde noten op in het geheugen.
Haal een nieuwe aktuele waarde voor de bewegingsamplitude op, en afhankelijk van het rezultaat, wordt het tonaal centrum verschoven. De konditie wordt bepaald als XH=YH=ZH. Het bereik voor de variabele I is beperkt tot het trajekt 47 tot 72.

```
313 XH = XF
    YH = YF
    ZH = ZF
    IF XH = YH OR YH = ZH THEN I = 36 + (512 - ADC(3)) / 20
```

In funktie van het tijdsverloop wordt de instrumentatie van de verschillende stemmen gewijzigd. De waarden van de hier gebruikte parameters staan uiteraard geheel in funktie van de specifikaties van de gebruikte Proteus module en de daarin opgeslagen en door ons op voorhand gewijzigde en aangepaste samples.

```
314 IF L = 1 THEN GOTO 320
```

Harmonium, Ney Fluit, Shofar

```
315 IF L = 50 THEN D = P(27,5,11,6,6,6,6,6)
```

Harmonium, Musette, Accordion

```
316 IF L = 10 THEN D = P(27,41,1,31,45,45,45,45)
```

Driestemmig harmonium:

```
317 IF L = 2 THEN D = P(27,27,27,27,27,27,27,27)
```

Tijdsbepalende routine, waarbij een gradueel accellerando wordt doorgevoerd, aan het einde waarvan de maximaal haalbare responsiviteit van dit onzichtbaar instrument bereikt wordt:

```
318 L = L - 1
    IF L < 1 THEN L = 1
320 D = S(L)
```

Test de exit konditie:

```
321 IF KEYPAD = 0 THEN GOTO 301
```

Indien deze exit konditie voldaan is, keer terug naar het hoofdmenu:

```
322 GOTO 50
```

3.2.2.9.12.: " Spooky "

In deze module wordt de vektoriele bewegingsinformatie gemapt op driestemmige nootinformatie. De amplitude informatie bepaalt de muzikale geluidssterkte. De noten worden gespeeld door gebruik te maken van de taken 6,7,8. Er wordt geen gebruik gemaakt van multitasking voor de informatieverwerving, wel voor het spelen van de verschillende stemmen.

Start de noot-speel taken voor de drie stemmen, met voor elke stem een andere periodiciteit:

```
350 START 5,59
    START 6,67
    START 7,71
```

Ingangspunt van de programmalus:

```
351 FOR M = 0 TO 2
```

Haal de nootinformatie uit de analoog-digitaal konverterkanalen 4,5,6 (Xf,Yf,Zf) voor de bewegingssnelheid.
Haal tevens de amplitudes (als 5-bit waarden) vanuit de AD kanalen 0,1,2 (X,Y,Z):

```
352 N(M) = ADC(M + 4) / 8: V(M) = ADC(M) / 8
```

Bepaal een minimale geluidssterkte:

```
353 IF V(M) THEN V(M) = BOR(V(M),48)
```

Filter al te lage noten weg:

```
354 IF N(M) < 24 THEN N(M) = 0 : V(M) = 0
```

Sluit de stemmenteller:

```
355 NEXT M
```

Test de exit konditie:

```
356 IF KEYPAD=0 THEN GOTO 351
```

Indien voldaan, schrap al de taken, en keer terug naar het hoofdselektiemenu:

```
357 CANCEL 5
    CANCEL 6
    CANCEL 7
    GOTO 50
```

3.2.2.9.13.: " Record-Playback "

Deze module is een poging om een soort muzikale lijntekening mogelijk te maken. De speler 'tekent' een melodische lijn in de lucht. De bewegingslijn wordt als een reeks samples opgeslagen en, nadien weergegeven als een melodische lijn. Het is een muzikale toepassing van wat we in het simulatieprogramma 'Holosimi', uitvoerig besproken in hoofdstuk 2, deden om de kurves om te zetten in muzikale notatie.

Van elk informatiekanaal worden 32 samples genomen. De waarden worden via POKE instrukties in een stukje 'kladblok-geheugen' (scratchpad memory) weggeschreven. Aan 128 bytes hebben we genoeg. De geheugenadressen legden we tussen $DE00 en $DEFF, het bovenste topje van het voor de gebruiker beschikbare geheugen. Aan een procedure met rechtstreekse benadering van de geheugencellen werd de voorkeur gegeven boven het gebruik van variabelen, vanwege de veel grotere uitvoeringssnelheid van de verkregen kode.

Ingang van de hoofdlus voor opname en weergave:

```
400 L = -1
    I = 0
    D = U($93) + U(36) + U(120)
```

Ingangspunt van de opname-lus:
J is de teller waarmee de analoog-digitaal konverter kanalen worden afgelopen. De teller loopt alsvolgt: ZF, YF, XF, Amplitude.

```
401 FOR J = 6 TO 3 STEP -1
```

Haal een 10-bit waarde op van de transducers en sla 7-bits van die informatie op in variabele M.

```
402 M = ADC(J) / 8
    IF J = 3 AND M THEN M = BOR(M,24)
403 IF M < 24 THEN M=0
```

Ga naar een volgende geheugencel (L-teller), en schrijf de informatie weg naar een daarvoor via USERMEM $200 gereserveerd stuk van het werkgeheugen:

```
404 L = L + 1
    POKE $DE00 + L,M
```

Sluit de kanaallus:

```
405 NEXT J
```

Voeg een 100ms wachttijd in tussen het nemen van de verschillende samples:

```
406 D=S(10)
```

Indien voldoende waarden (32) voor de bewegingssnelheid werden verzameld, verlaat dan de opnamelus. Zoniet verhoog te teller met een eenheid en keer terug naar het begin van de lus:

```
407 IF I<31 THEN I=I+1
    GOTO 401
```

Hierna volgt dan de weergave-lus voor de muzikale verklanking van de opgenomen beweging. De weergave verloopt in slow-motion op grond van een kompositorische beslissing.

```
410 D = U($93) + U(41) + U(110)
```

De I teller wordt gebruikt om alle geheugencellen af te lopen en uit te lezen:

```
411 FOR I=0 TO L-1 STEP 4
```

De inhoud van de geheugencellen wordt gedeeld door 4 om de afbeelding van beweging op melodie zo te vereenvoudigen dat het verband voor de kijker-luisteraar eenvoudig valt in te zien.

```
412 N(0) = (PEEK($DE00 + I)) / 4
    N(1) = (PEEK($DE01 + I)) / 4
    N(2) = (PEEK($DE02 + I)) / 4
    A = ((PEEK($DE03 + I)) / 3) * 3
```

De noten worden gespeeld, indien hun waarde niet 0 zijn:

```
413 IF N(0) THEN N(0) = 36 + N(0): D = U($90) + U(N(0)) + U(A)
414 IF N(1) THEN N(1) = 36 + N(1): D = U($92) + U(N(1)) + U(A)
415 IF N(2) THEN N(2) = 48 + N(2): D = U($91) + U(N(2)) + U(A)
```

Het afspeeltempo wordt door volgende instruktie vastgelegd en bepaald op 100ms.

```
419 D = S(10)
```

De tellerlus voor de geheugenplaatsen wordt gesloten:

```
420 NEXT I
```

Alvorens een nieuwe reeks samples te nemen van een nieuwe bewegingsboog, worden de gespeelde noten uitgeschakeld:

```
421 FOR M = 0 TO 3
    D = U($B0 + M) + U($7B) + U(0) + S(10)
    NEXT M
```

Test de exit konditie. Indien geen toets werd ingedrukt, wordt terug gesprongen naar het begin van de programmalus (opname).

```
422 IF KEYPAD = 0 THEN GOTO 400
```

Indien aan de exit-voorwaarde is voldaan, keer dan terug naar het openingsmenu:

```
423 GOTO 50
```

3.2.2.9.14.: " Lock & Unlock "

In deze tweeledige module wordt een zevenstemmig eenvoudig slagwerkensemble geimplementeerd. Elke stem speelt in een net ietsje verschillend tempo, waardoor we geleidelijke fazeverschuivingen te horen krijgen. De gespeelde toonhoogtes evenals de dynamiek zijn gestuurd door de bewegingen.

In het tweede gedeelte van de module (Unlock) wordt dit patroon doorbroken en komen de stemmen metrisch gezien van elkaar los te staan. De metriek wordt hier een funktie van de bewegingsparameters.

3.2.2.9.14.1: "lock...

De noten worden per stem in een eigen tempo gespeeld door taken toegewezen aan de multitasker. N(6)-V(6) is het MIDI-informatiekoppel (F#) voor de enkele trommelslag die het samenvallen van de fazecyclus markeert (taak 11).

Vervolgens worden de taken opgestart.De taken 5,6,7 besturen de muzikale stemmen 0,1,2 als afzonderlijke midikanalen.

```
450 N(6)=30
    V(6)=$7F
    START 11,10300
    START 5,100
    START 6,101
    START 7,102
```

De taken 8,9,10 sturen de muzikale stemmen 3,4,5 via de overeenkomstige midikanalen.

Taak 1 wordt gebruikt als bron van bewegingsinformatie en retourneert XF,YF,ZF, X,Y,Z.

```
451 START 8,103
    START 9,104
    START 10,105
    START 1,4
```

De bewegingssnelheidsinformatie vervat in XF,YF,ZF bepaalt de toonhoogtes. Het bereik voor de kanalen 0-2 loopt van 60 tot 92. Voor de kanalen 3 tot 5 loopt het van 28 tot 60.

De bewegingshoeveelheidsinformatie wordt rechtstreeks gebruikt ter kontrole van de aanslagsterkte van de instrumenten.

```
452 N(0) = 60 + ((XF - 64) / 2)
    V(0) = X
    N(1) = 60 + ((YF - 64) / 2)
    V(1) = Y
    N(2) = 60 + ((ZF - 64) / 2)
    V(2) = Z
453 N(3) = 60 - ((XF - 64) / 2)
    V(3) = X
    N(4) = 60 - ((YF - 64) / 2)
    V(4) = Y
    N(5) = 60 - ((ZF - 64) / 2)
    V(5) = Z
```

Test de exit konditie om wanneer dat gewenst is naar het tweede luik van deze module over te gaan:

```
454 IF KEYPAD = 0 THEN GOTO 452
```

Schakel de eerder gebruikte taken waarmee de noten gespeeld worden uit:

```
455 CANCEL 5
```

```
    CANCEL 6
    CANCEL 7
    CANCEL 8
    CANCEL 9
    CANCEL 10
    CANCEL 1
    CANCEL 11
```

3.2.2.9.14.2: "...Unlock "

In dit tweede luik wordt het tempo waarin de noten worden gespeeld in de diverse stemmen een funktie van de bewegingshoeveelheid. Hierbij wordt taak 3 gebruikt als informatiebron voor de bewegingsdata.
Deze taak levert V(0),V(1),V(2), als aanslagsterktevariabelen afgeleid van de X,Y,Z, amplitudekanalen. De noten, N(0),N(1),N(2) worden kwa oktaafligging bepaald op grond van de snelheidinformatie van de parameters XF,YF,ZF. De precieze noot echter binnen dit oktaaf laten we afhangen van de bewegingsversnellingsinformatie. De globale amplitude bepaalt het tijdsinterval tussen de noten.

Eerst wordt de instrumentatie naar de Proteus doorgegeven:

```
460 D = P(38,67,92,22,34,50,8,8)
```

Vervolgens worden de taken waarmee de noten worden gespeeld opgestart evenals de taak die de informatie ophaalt:

```
461  START 3,10
     START 12,20
     START 13,20
     START 14,20
     START 15,30
     START 16,30
     START 17,30
```

Test voor de exit-konditie van de module, indien voldaan, schakel alle lopende taken terug uit.

```
462 IF KEYPAD = 0 THEN GOTO 462
463 CANCEL 3
    CANCEL 12
    CANCEL  13
    CANCEL 14
    CANCEL 15
    CANCEL 16
    CANCEL 17
    GOTO 50
```

3.2.2.9.15.: " Hammers "

Deze mapping is een poging om een maximum aan relevante bewegingsinformatie zo rechtstreeks mogelijk op overeenkomstige muzikale parameters af te beelden.
Daarom vertaalt het de vektoriele bewegingshoeveelheid in geluidssterkte, de vektoriele snelheid in de oktaafligging en de vektoriele versnelling van de beweging in de noten binnen elk oktaaf. De versnellingsinformatie wordt verworven via taak 3.

Start de informatieverwervingstaak (3).
Start de taken waarmee de noten worden uitgestuurd over de gebruikte midikanalen (3-stemmig met een extra toegevoegd klankeffekt).

```
475 START 3,5
    START 5,11
    START 6,12
    START 7,13
    START 9,19
```

Wanneer de drie noten van elkaar verschillend zijn, wordt het effekt gespeeld:

```
476 IF N(1) >< N(0) AND N(1) >< N(2) AND N(0) >< N(2) THEN
            N(4) = N(1)
            V(4) = V(1)/2
```

Test voor de exit-konditie:

```
477 IF KEYPAD = 0 THEN GOTO 476
```

Schakel alle taken uit en spring terug naar het hoofdselektiemenu.

```
478 CANCEL 3
    CANCEL 5
    CANCEL 6
    CANCEL 7
    CANCEL 9
    GOTO 50
```

3.2.2.10: De Taken

De hier volgende kodesegmenten illustreren overduidelijk de mogelijkheden van multitasking. De hier beschreven kode wordt gestart via het START [n,m] kommando. [n] is daarbij het volgnummer van de taak, en [m] de periodiciteit waarmee ze dient te worden uitgevoerd. Deze periodiciteit wordt uitgedrukt in hondersten van een sekonde.

De eerste taken worden gebruikt om informatie te verwerven vanuit de analoog-digitaal konverterkanalen. De konversietijd voor het hier toegepaste type konverter wordt door de fabrikant gespecifieerd als 100 µs. In teorie zou dus een sampling-rate van 10ks/s haalbaar moeten zijn. Deze snelheid moet echter worden gedeeld door het aantal kanalen dat we willen uitlezen (10ks/s / 7 = 1428s/s) en verder is er nog de tijd die de softwarekode zelf nodig heeft tussen de verschillende omzettingen. In de praktijk bleek voor dit projekt de hoogste haalbare sampling rate rond 900s/s te liggen. Het teorema van Nyquist volgend, betekent dit dat de hoogste signaalfrekwentie die nog zinvol te meten valt hooguit 450Hz bedraagt. Gezien het feit dat we het grootste deel van de rekenkundige bewerkingen eigenlijk reeds op het nivo van de analoge komputerschakelingen konden uitvoeren, is dit hier ruimschoots voldoende.
De specifikaties van de filters in de analoge sektie zijn zo berekend, dat in het aan de AD-konverters aangeboden signaal geen frekwentiekomponenten meer aanwezig zijn die de Nyquist grens te boven gaan.

3.2.2.10.1.: Taak 1

```
500 TASK 1
```

Deze taak leest de beschikbare informatie van de analoog-digitaal konverterkanalen 0-6. (7 kanalen). De geretourneerde variabelen zijn X,Y,Z en de waarden ervan vallen binnen het bereik 0-127. (7-bit) Deze waarden komen overeen met de geintegreerde amplitude van de overeenkomstige kanalen. Alleen de hoogste 7-bits worden dus gebruikt. (De omzettingen zelf grijpen plaats met een rezolutie van 10-bit). Dit werd gedaan omdat de laagste bits op grond van de precizie in de analoge komputerschakelingen sowieso beperkt is tot niet veel beter dan 1% terwijl ook de signaal-ruisverhouding deze lage bits volstrekt onbetrouwbaar en irrelevant maakt. Bovendien is een oplossend vermogen ruim voldoende voor de mogelijkheden die het MIDI-systeem kan bieden en, last but not least, betwijfelen we of de mens tot een grotere lichaamskontrole in staat is. Maar, dus zelfs indien dat toch het geval ware, zou een grotere rezolutie gezien de beperkingen van onze hardware, geen aarde aan de dijk brengen.
Verder levert deze taak ook de variabelen XF.YF en ZF op binnen hetzelfde bereik: 0 - 127. Deze variabelen zijn evenredig met de vektoriele bewegingssnelheid. De overdrachtfunktie en vooral het verloop ervan (lineair, exponentieel, logaritmisch) hangt uiteraard af van de ingestelde rekenkundige funktie op de analoge komputer. Als laatste variabele levert deze taak ons nog een waarde voor de globale bewegingshoeveelheid, geintegreerd over een langer tijdsinterval dat voor de vektoriele bewegingshoeveelheden X,Y,Z. Ook A heeft een rezolutie van 7 bits en valt in het bereik 0-127.

```
501 XF = ADC(4) / 8
     X = ADC(0) / 8
502 YF = ADC(5) / 8
     Y = ADC(1) / 8
503 ZF = ADC(6) / 8
     Z = ADC(2) / 8
504 A = ADC(3) / 8
506 EXIT
```

3.2.2.10.2.: Taak 2

Deze taakkode wordt gebruikt voor de implementatie van het onzichtbaar slagwerk in module 4 ("Beat").
De variabelen die erdoor geretourneerd worden zijn:

L = waarin de piek-amplitude (grootste waarde van twee lezingen) wordt opgeslagen.

X,Y,Z zoals in taak 1 (0 -127)

C is een logische variabele en is 0 (vals) wanneer de taak niet beeindigd werd, -1 (waar) wanneer ze werd afgewerkt.

Deze konstruktie is ook weer typisch voor multitasking, waarbij het immers mogelijk is dat de taak-kode onderbroken wordt voor het uitvoeren van een andere opdracht. Wanneer die andere kode gegevens uit de taak gebruikt, kan het van belang zijn te weten in welke staat van afwerking deze gegevens verkeerden.

XF,YF,ZF, zoals in taak één maar met een beperkter bereik van 0-63, en dus met een rezolutie van slechts 6-bits.

A zoals in taak 1.

```
510 TASK 2
```

Reset alle variabelen:

```
511 C = 0
    A = ADC(3) / 8
    X = ADC(0) / 8
    Y = ADC(1) / 8
    Z = ADC(2) / 8
```

Indien geen beweging gedetekteerd werd, verlaat dan de taak zonder enige variabele te wijzigen:

```
513 IF (X + Y + Z = 0) OR (A = 0)  THEN GOTO 518
```

Lees de bewegingssnelheidvariabelen. Konverteer via deling door 16 tot een 6 bits getal (0-64).

```
514 XF = ADC(4) / 16
    YF = ADC(5) / 16
    ZF = ADC(6) / 16
516 L = ADC(3) / 8
    IF L > A THEN A = L
517 C=1
518 EXIT
```

3.2.2.10.3: Taak 3

Deze taak retourneert rechtstreeks MIDI noten en aanslagsterkten via gedimensioneerde variabelen. In V(0), V(1), V(2) worden de bewegingshoeveelheden opgeslagen. De bewegingssnelheden worden in de oktaafligging ondergebracht en de noten worden afgeleid van de hier berekende vektoriele versnellingen van de bewegingen. De uiteindelijke notenwaarden worden geretourneerd in N(0),N(1),N(2).

Op grond van onze ervaring met dit instrument levert een dergelijke mapping een erg muzikaal aanvoelend reageren van het instrument op de beweging van de uitvoerder. Talloze musici aan wie we hebben gevraagd het uit te proberen deelden deze mening. Een inherente logika voor deze mapping lijkt ons echter moeilijk te geven. Daarom menen we dat het aan de gebruiker overgelaten moet worden de mapping naar keuze ook anders te organizeren en bijvoorbeeld de versnelling op de oktaafligging (of een ander interval indien gewenst) en de snelheid op de noot af te beelden.

Taak 3 wordt gebruikt in de modules "Hammers" en "...unlock".

```
520 TASK 3
```

X,Y,Z worden hier gebruikt voor de eerste inlezing van de snelheidsinformatie. Merk op dat we hier de volle 10 bit waarde opslaan.

```
521 X = ADC(4)
```

```
    Y = ADC(5)
    Z = ADC(6)
```

De volumes worden rechtstreek ingelezen in de betreffende Midi-variabelen:

```
522 V(0) = ADC(0) / 8
    V(1) = ADC(1) / 8
    V(2) = ADC(2) / 8
```

De globale amplitude wordt afgeknot gebruikt:

```
527 A=ADC(3) / 3
    IF A > 127 THEN A = 127
```

De booleaanse somfunktie wordt gebruikt ter bepaling van de uit te sturen aanslagsterkte:

```
530 V(0) = BOR(V(0),A)
    V(1) = BOR(V(1),A)
    V(2) = BOR(V(2),A)
```

Bepaal de tessituurligging van de te spelen noten in grove stappen:

```
532 N(0) = (X / 210) * 24
    N(1) = (Y / 210) * 24
    N(2) = (Z / 210) * 24
```

In de volgende berekeningen wordt de versnellingsinformatie berekend op grond van een nieuwe opmeting van de snelheid. Het verschil tussen X en ADC(4) is de versnelling. Omwille van de responsiviteit van het algoritme worden de metingen voorwaardelijk uitgevoerd, alleen wanneer er reeds eerder een beweging was vastgesteld.
Het bereik van de versnellinginformatie werd zo gekozen dat het binnen -32 tot +32 valt.

```
533 IF N(0) THEN N(0) = N(0) + (ADC(4)-X)/32
534 IF N(1) THEN N(1) = N(1) + (ADC(5)-Y)/32
535 IF N(2) THEN N(2) = N(2) + (ADC(6)-Z)/32
539 EXIT
```

3.2.2.10.4: Taak 4

Wanneer de uitlezingen van de kanalen 4,5,6 (XF,YF,ZF) waarden opleveren evenredig met de Dopplerfrekwenties en dus met de vektoriele bewegingssnelheden, dan levert het verschil tussen twee opeenvolgende waarden van deze parameter ons een waarde proportioneel met de versnelling (positief of negatief).

De door deze taak geretourneerde variabelen zijn:

- X,Y,Z bereik: 0 - 127 kanaal amplitudes
- XF,YF,ZF bereik: - 30 to 30 vektoriele versnellingen
- XH,YH,ZH bereik: 0 - 63 vektoriele snelheden
- A bereik: in diskontinue stapjes.

Taak 4 wordt gebruikt in de modules 'Sforte' en 'Prime Time'.

```
540 TASK 4
```

Lees de waarden van kanalen 0.1.2, en draag de 7-bit waarden over in de amplitudevariabelen X,Y,Z.
Lees de waarden van de kanalen 4,5,6 en draag de 6bit waarden over in de snelheidsvariabelen XH,YH,ZH.

```
541 X = ADC(0) / 8
    Y = ADC(1) / 8
```

```
    Z = ADC(2) / 8
    XH = ADC(4) / 16
    YH = ADC(5) / 16
    ZH = ADC(6) / 16
```

Lees de globale amplitude, en voeg er door afknotting, een duidelijk sforzando effekt aan toe. Dit aspekt wordt dus een eigenschap van de aanslag van het geimplementeerde instrument.

```
543 A = ADC(3) / 4
    IF A > 64 THEN A = 127
```

Lees nieuwe snelheidinformatie in als 6-bit waarden, bereken de versnelling en sla het rezultaat op in XF,YF,ZF

```
544 IF X THEN XF = XH - (ADC(4) / 16)
545 IF Y THEN YF = YH - (ADC(5) / 16)
546 IF Z THEN ZF = ZH - (ADC(6) / 16)

547 EXIT
```

3.2.2.10.5.: Taken 5 tot 11

De reeks taken van 5 tot en met 11 worden uitsluitend gebruikt voor het spelen van MIDI-noten door per taak een andere muzikale stem. Aangezien de taken in het 'agenda' van de multitasker kunnen opgenomen worden met een verschillende periodiciteit van uitvoering, wordt het implementeren van ingewikkelde muzikaal ritmische patronen een fluitje van een cent. Aangezien we zeven taken definieerden, kunnen we zoiets zevenstemmig uitvoeren.

Vanzelfsprekend dienen de parameters elders een waarde toegekend te krijgen: zowel noten en volumes moeten toegewezen worden voor elke afzonderlijke taak.

Eerste stem:

```
550 TASK 5
551 D = U($90) + U(N(0)) + U(V(0))
552 EXIT
```

Tweede stem:

```
560 TASK 6
561 D = U($91) + U(N(1)) + U(V(1))
562 EXIT
```

Derde stem:

```
570 TASK 7
571 D = U($92) + U(N(2)) + U(V(2))
572 EXIT
```

Vierde stem:

```
580 TASK 8
581 D = U($93) + U(N(3)) + U(V(3))
582 EXIT
```

Vijfde stem:

```
590 TASK 9
```

```
591 D = U($94) + U(N(4)) + U(V(4))
592 EXIT
```

Zesde stem:

```
600 TASK 10
601 D = U($95) + U(N(5)) + U(V(5))
602 EXIT
```

Zevende stem:

```
610 TASK 11
611 D = U($96) + U(N(6)) + U(V(6))
612 EXIT
```

3.2.2.10.6.: Taken 12 tot 17

De taken in de reeks 12 tot en met 17 spelen net zoals de vorige reeks noten via de midi-kanalen, maar hier schrapt elke taak zichzelf uit de agenda van de multitasker wanneer ze beindigd wordt. De SUSPEND-tijd is een lineaire funktie van de geintegreerde som der bewegingsamplitudes van de X,Y,Z kanalen, dus de A-waarde.

De parameters voor deze takenreeks worden normaal gezien ingevuld door taak 4.

Eerste stem: (X-vektor)

```
620 TASK 12
621 IF N(0) THEN D = U($90) + U(N(0)) + U(V(0))
    SUSPEND 32 - (A / 4)
622 EXIT
```

Tweede stem: (Y-vektor)

```
630 TASK 13
631 IF N(1) THEN D = U($92) + U(N(1)) + U(V(1))
    SUSPEND 32 - (A / 4)
632 EXIT
```

Derde stem: (Z-vektor)

```
640 TASK 14
641 IF N(2) THEN D = U($91) + U(N(2)) + U(V(2))
    SUSPEND 32 - (A / 4)
642 EXIT
```

Vierde stem:
De te spelen noot is het gemiddelde tussen N(0) en N(1).

```
650 TASK 15
651 IF N(0) THEN D = U($93) + U((N(0) + N(1)) / 2) + U(V(0))
     SUSPEND 32 - (A / 4)
652 EXIT
```

Vijfde stem:
De te spelen noot is het gemiddelde tussen N(1) en N(2).

```
660 TASK 16
```

```
661 IF N(1) THEN D = U($95) +U((N(1) + N(2)) / 2) + U(V(1))
     SUSPEND 32 - (A / 4)
662 EXIT
```

Zesde stem:
De te spelen noot is het gemiddelde tussen N(0) en N(2).

```
670 TASK 17
671 IF N(2) THEN D = U($94) + U((N(0) + N(2)) / 2) +U(V(2))
     SUSPEND 32 - (A / 4)
672 EXIT
```

3.2.2.10.7.: Taak 18

Deze taak is eigenlijk een vroegere versie van wat eerst taak 1 was. In principe vervult ze precies dezelfde opdracht. Het eigenaardige -zoals bleek uit de praktijk- is echter dat ze toch een heel verschillend estetisch rezultaat blijkt op te leveren. Aangezien ik dit rezultaat eigenlijk heel goed vond behield ik dit stuk kode in het programma. De taak wordt gebruikt in de modules "Lead" en "Topoi". In de module "Lead" worden beide achtereenvolgens gebruikt en leveren daar een muzikale variatie op.

```
800 TASK 18
```

De ADC-kommandos zijn hier kwazi in assembler gekodeerd. Er wordt gebruik gemaakt van OUT en IN kommandos, die alleen op byte nivo funktioneren.
De syntax is bepaald door de hardware uiteraard:
OUT $91, <kommando>
kommando: moet 3 zijn, omdat daarmee de klokfrekwentie van de AD-konverter wordt aangepast aan de hardware.
OUT $90, <kanaal kommando>
kanaal-kommando: het volgnummer van het te bemonsteren kanaal.
<Var> = INP($91)
de 8-bit waarde van de AD-converter kan worden ingelezen in Var.
Om een 7-bit waarde te verkrijgen delen we het rezultaat uiteraard door 2.
De geretourneerde waarden zijn identisch met die van taak 1: dus X,Y,Z, (0-127), XF,YF,ZF, (0-127), A(0-127).

```
801 OUT $91,3: OUT $90,0: X = INP($91)/2
     OUT $91,3: OUT $90,1: Y = INP($91)/2
802 OUT $91,3: OUT $90,2: Z = INP($91)/2
     OUT $91,3: OUT $90,4: XF = INP($91)/2
803 OUT $91,3: OUT $90,5: YF = INP($91)/2
     OUT $91,3: OUT $90,6: ZF = INP($91)/2
```

Voor module 2 ("Topoi") moet de snelheidsinformatie omgezet worden naar een andere schaal:

```
804 IF E = 2 THEN XF = XF / 9 : YF = YF / 9 : ZF = ZF / 9
```

De globale amplitude wordt ingelezen en omgezet in een diskontinue variabele. Delen door 16 en vermenigvuldigen met 8 levert dit rezultaat wanneer men er zich tenminste rekenschap van geeft dat alle bewerkingen ingevolge onze deklaraties uitgevoerd worden met als rezultaat gehele getallen.

```
805 OUT $91,3 : OUT $90,3 : A = (INP($91) / 16) * 8
806 EXIT
```

3.2.2.11.: Opzoekingstabel voor muzikale gegevens

De hier volgende data-statements bevatten eigenlijk niets anders dan de 'a priori kennis' van het instrument. Het zijn de tabellen met definities van het openingsakkoord, de harmonische progressies gebruikt voor "Topoi" , de struktuur van de mineur toonladder gebruikt in 'Minor' enz...

De data zien eruit als 16-bit getallen. Ze moeten dan ook bij gebruik gesplitst worden in MSB en LSB. Dit werd gedaan om geheugen te sparen.

```
900 DATA 4047,5055,6266,7381,0,0,0,0
901 DATA 6,0,0,1100,1100,7,2,0
902 DATA 0,900,500,12,3224,2820,1609,1204
903 DATA 12,0,0,400,1200,9,6,0
904 DATA 0,1100,700,4,0,9,5,20
905 DATA 0,0,1100,307,700,4,2,0
906 DATA 12,9,5,300,0,1100,700,5
907 DATA 0,200,900,705,12,12,4,800
908 DATA 211,5,8,0,9,1206,0,300
909 DATA 4,100,700,10,7,5,0,211
910 DATA 12,200,500,9,104,6,9,0
915 DATA 2,305,709,1112,1415,1719,2324,2627
```

3.2.2.12: Subroutines:

Deze subroutine wordt gebruikt om de in de data-statements opgesloten informatie terug te ontbinden naar enkelvoudige bytes die rechtstreeks in het geheugen geplaatst worden via POKE statements. Dit geheugen ($DF00-$DFFF) dient wel voorafgaand aan de kompilatie van het programma vrijgemaakt te worden via het systeemkommando USERMEM $200.

```
916 FOR M = 0 TO $FE STEP 2
917 IF M < $B0 OR M > $EF THEN READ I:
        POKE ($DF00 + M),I / 100
        POKE ($DF01 + M),I - ((I / 100) * 100)
918 NEXT M
919 I = 0 : RETURN

USERMEM $200
NOERR
END

CODE ENDS
[EOF]
```

3.2.3: Geheugenstruktuur van het Multitraxboard na kompilatie van het "A Book of Moves" programma:

Opgemerkt moet worden dat we het komputersysteempje werkelijk tot op zijn uiterste grenzen hebben gebruikt.

De geheugenindeling van het board ziet eruit alsvolgt:

	Start Addr	End Addr	Hex Size	Dec Size		
MTOS-MTBasic	0000	7FFF	7FFF	32767	ROM	0 - 7FFF
Source code:	8000	C2BE	42BF	17087	RAM	8000 - C2BE
Compiled code:	4500	BFE2	7AE3	31459	EPROM	
Comp code free:	BFE3	BFFF	001D	29	EPROM	
Src/Vars free:	C2BF	C870	05B2	1458	RAM	C2BF - C870
Variables:	C871	DDFF	158F	5519	RAM	C871 - DDFF
User space:	DE00	DFFF	0200	512	RAM	DE00 - DFFF

De geheugenallokatie in absolute hardware adres-blokken is:

MTOS	00000	07FFF	0 - 32767	
RAM1	08000	0FFFF	32768 - 65535	Source code
RAM2	10000	17FFF	65536 - 98303	Compiled Code
EPROM	18000	1FFFF	98304 -131071	Eprom Source code

Hoewel het Multitrax board zelf een EPROM-programmer aan boord heeft, bleek het onmogelijk om een zelfstartend systeem in een EPROM geprogrammeerd te krijgen. De instrukties USERMEM $200 en NOERR voor de kompiler, kunnen alleen via een externe EPROM-programmer en debugger in de ROM geschoven worden. Dit feit is nergens gedokumenteerd in de manuals van het Multitrax board.

3.3.: Artistieke bespreking en prezentatie:

"A Book of Moves" is de artistieke demonstratie geworden van een stadium in een lang proces van zoeken naar een instrument dat rechtstreeks vanuit de bewegingsmotoriek zou worden gestuurd. Men zou zich hierbij uiteraard de vraag kunnen stellen wat het instrument zou opleveren los van deze of enige artistieke prezentatie. Welnu, wij geloven dat dit onmogelijk is. Meer nog, principieel gezien is ons instrument in deze niet eens fundamenteel verschillend van enig traditioneel instrument. Een traditioneel instrument sluit in zich immers eveneens een heel arsenaal aan artistieke en estetische keuzes van de maker, van de koper en van de bespeler. Zoals we in ons eerste hoofdstuk aantoonden, is het muziekinstrument principieel als expressief werktuig artistiek en estetisch waardegeladen. Het 'bevat', in samenspel met zijn instruktie, een beeld van de muziek waarvoor het werd gekoncipieerd.

Ook een traditioneel instrument kan niet zinnig worden bestudeerd, laat staan beoordeeld, wanneer het niet wordt bespeeld. In de bespeling en de bespeelbaarheid ligt uiteindelijk de enige justifikatie van een muziekinstrument.

Zo ook dus, geldt dit alles voor het hier voorgesteld instrument. Daarom ook zou het geen zin hebben, alleen te volstaan met een technische beschrijving van de hardware. In dit geval ware zoiets nog veel onzinniger dan in het geval van traditionele instrumenten. Immers voor dit instrument bestaat geen standaard muzikaal repertoire. Het betracht immers nieuwe speelwijzen mogelijk te maken.

Nu willen we anderzijds ook geenszins de justifikatie van het door ons hier voorlopig voorgestelde instrument uitsluitend afhankelijk maken van het ene stuk wat we er zelf voor koncipieerden. Niets zegt immers dat dit ook maar ergens het best mogelijke maakbare stuk zou zijn gebruikmakend van dit instrument. Adophe Sax schreef beslist ook niet de beste saxofoonmuziek. "A Book of Moves" moet dan ook gezien worden als een artistiek en demonstratief voorbeeld. Het beoogt het instrument in werking te tonen en te laten horen.

We hopen en geloven echter dat de hier aangeduide weg een mogelijke richting voor de toekomstige instrumentenbouw zou kunnen aanwijzen. Dat er in zo'n proces nog ontelbare verbeteringen aan onze voorstellen en ontwerpen mogelijk zijn beseffen we wellicht zelf beter dan wie ook. Misschien ook hebben we ons op enige belangrijke punten fundamenteel vergist...

Mogelijke perspektieven voor verdere verbeteringen en research hebben we, voorzover we die zelf kunnen zien, aangegeven in het volgende hoofdstuk.

Omdat na al deze kode-beschrijvingen en beschrijvingen van de meest diverse technisch-elektronische details -waarzonder evenwel niets zou funktioneren en waarzonder ook helemaal niets aangetoond zou zijn- de lezer wellicht het muzikale spoor bijster is geraakt, hernemen we hier heel beknopt en in meer muzikaal-artistieke termen de 16 verschillende modules waaruit "A Book of Moves" is opgebouwd. We geven onze beschrijving hier tevens in de volgorde waarin we ze bij de vele koncertuitvoeringen die we er reeds van mochten verzorgen ook prezenteerden.

Open

Een ouverture gebazeerd op een als trage mokerslagen steeds maar herhaald 8-tonig breed en kompleks akkoord.

Bij de aanvang van het stuk zit de speler in het midden van het podium ineengehurkt op de grond. Zij draagt een pakje in hoogglanzend metaalachtig tekstiel.

Na honderdachtentwintig *ff*-aanslagen van het akkoord -waarbij de tonen tussen de luidsprekers over en weer springen- zwaait zij een arm in de lucht. De akkoorden worden onderbroken, maar gaan vanzodra de beweging stopt weer door. Ze zijn nu echter een nauwelijks hoorbare gradatie (2dB) zachter geworden.

De speler maakt langzaam meer en meer bewegingen daarbij verduidelijkend dat zij de klank wég wil. De klank wordt alsmaar stiller. Zij moet echter alsmaar grotere inspanningen leveren om dit verstillingsproces te laten doorgaan. Wanneer de bewegingen een hoogtepunt bereiken, wordt het helemaal stil. De ouverture is afgelopen.

In klassieke zin komt deze ouverture volledig overeen met de opening van een klassieke redevoering: het onderwerp, in dit geval de beweging-klank transformatie- wordt zo duidelijk mogelijk

afgebakend. Het probleem wordt gesteld in een tot zijn meest eenvoudige gedaante gereduceerde vorm. Daarom noemen we deze ouverture een retorische formule.

Call

Nadat het probleem in alle klaarheid en vereenvoudiging is gesteld, wordt nu de algemene geldigheid ervan betoogd.
In de kompositie 'Call' worden gradueel alsmaar meer muzikale eigenschappen aan de bewegingen van de speler gekoppeld. De volledige werking van het onzichtbare instrument wordt gedemonstreerd.
De module opent met de klanken van Shofars vanuit de verte. De koppeling aan de zwaaibewegingen van de speler zijn aanvankelijk erg vaag. De instrumenten komen echter dichterbij en veranderen ook geheel van klank. Ze worden uiteindelijk gemetamorfozeerd tot een enkel akkordeon, waarbij de speler ook werkelijk een fiktieve akkordeon gaat bespelen. Hier kan de speler heel getrouw gaan improvizeren. De klanken sluiten met grote direktheid aan bij de uitgevoerde bewegingen. Gedurende het gehele stuk blijft de speler hier in het midden van het speelvlak.

Topoi

In 'Topoi' wordt de muzikale ruimte -het formaat- van het muziekinstrument afgetast. Daartoe wordt via verplaatsingen van de speler over de scene, aan de diverse plaatsen verschillende harmonische konstellaties toegekend. Er kunnen plaats-afhankelijke akkoordprogressies worden gespeeld. Het tonaal centrum (hoewel het niet om tonale muziek gaat in traditionele zin- reden waarom we dan ook deze term eerder dan 'tonika' gebruiken-) bevindt zich in het centrum van de opstelling. Hoe verder de speler zich van dit centrum verwijdert hoe verder ook de tot klinken gebrachte harmonieën van dit tonaal centrum gaan afwijken.

Rising

In deze formeel gezien uiterst eenvoudige kompositie worden we muzikaal gekonfronteerd met een graduele akkoordenprogressie van laag naar hoog. Naar heel erg hoog, dat wel... De wijze waarop de progressie vooruitloopt is echter geheel afhankelijk van de bewegingen van de bespeler. Hij speelt dus eigenlijk een vastgelegde partituur waar uitsluitend een soort becijferde bas op voorgeschreven staat. De grootte van de intervallen evenals de metrische geleding kan hij volledig zelf interpreteren.

Retorisch gezien krijgen we te maken met een figuur waarin een alsmaar grotere spanning wordt opgebouwd.

Spooky

De schrille ijle en hoge klanken waarmee 'Rising' wordt afgesloten, monden uit in een totaal 'andere' muzikale klankenwereld. Hier krijgen we moeilijk thuis te brengen klanken en geluiden afkomstig van niet-toonhoogte gerichte muziekinstrumenten. Alle harmonie lijkt opgeheven. Ook van enige melodische samenhang is hier geen sprake meer. De klankenwereld is zuiver vloeibaar en timbraal. De bewegingen zijn spookachtig en zwevend. De klanken worden bijzonder plastisch en op kontinue wijze door de bewegingen gestuurd. De speler verdwijnt in de klankennevel van de scene.

Minor

Een nieuwe speler verschijnt ten tonele. Hij speelt een onzichtbaar klavierinstrument. De toonsoort is duidelijk mineur. Hij heeft de melodische patronen letterlijk in de hand. De muziek wordt door haar centrum aangezogen. De melodische patronen cirkelen rond het tonaal centrum. De metriek is rustig maar niet regelmatig. Het dynamisch bereik is beperkt tot het bereik *mp* tot *mf*. De sfeer is vriendelijk. De bewegingen van de speler zijn soepel en beperken zich grotendeels tot de bovenste ledematen.

Beat

In deze kompositie staat op de scene een bijzonder groot aantal slaginstrumenten van het meest diverse pluimage opgesteld: westerse drumkits, symfonische slaginstrumenten, latijns-amerikaans slagwerk, afrikaanse trommels, gamelan-instrumenten en nog meer dergelijk exotika.

Als een parodie op nogal wat solistische slagwerk-literatuur uit de seriele en post-seriele periode, springt de speler van het ene instrument naar het andere waarbij hij zich vaak in de meest onmogelijke bochten dient te wringen om bepaalde klanken te kunnen spelen. De klanken zijn volledig op de bewegingen afgebeeld. Van de bespeler van het onzichtbaar instrumentarium wordt een soort muzikaal karate verwacht. De bewegingen moeten abrupt zijn, zonder de minste elegantie. Immers ook waar 'normaal' slagwerk wordt gespeeld, eindigen de armbewegingen van de speler toch in een abrupte botsing met het klankopwekkend gedeelte van het slaginstrument...

Sforte

Zoals de naam van het stukje al laat uitschijnen, levert het hier bespeelde instrument een uitgesproken sforzando element. De dynamische nuances van de bewegingen van de bespeler worden daarbij dus sterk overdreven. Het melodisch verloop is uitgesproken centripetaal en gericht naar een tonaal centrum.

Anders dan in 'minor' echter, is de sfeer hier geenszins vriendelijk. Er worden talloze smalle klusters gespeeld, waarbij de breedte afhangt van de energie, de inspanning, van de bespeler. Hoe groter de geinvesteerde energie, hoe groter ook de afwijking van de akkoorden en klusters van het tonaal centrum.

Lead

In deze kompositie wordt de bespeler tot dirigent van een betrekkelijk groot 'symfonisch' orkest. De toonspraak is uitgesproken modernistisch en vormt een 'karikatuur' van de orkestratie zoals we die zouden kunnen aantreffen bij komponisten zoals Xenakis, of, bij ons dan, Luc Brewaeys. De dirigent kan met armen -en desgewenst ook met de benen- orkestrale groepen doen inzetten, samenklanken aanhouden en de gehele metriek bepalen. De muzikale samenhang berust op een bijna triviaal maar uitermate effektief middel: de overlapping van de vorige met de volgende samenklank. Dit is het enige 'kompositorische' gegeven dat hier a priori vastligt, afgezien dan van de orkestratie zelf. De orkestbehandeling is volledig acht-stemmig.

Canvas

In dit stuk wordt gespeeld met in hoofdzaak 'niet-muzikale' geluiden. Geluiden die zich in elk geval moeilijk laten inpassen in de klassieke traditie van de westeuropese muziek waarbij in eerste plaats de diskontinue organizatie van de muzikale parameters toonhoogte en tijd centraal heet te staan. Hier krijgen we het omgekeerde: de muzikale parameters toonhoogte -voorzover die al duidelijk bepaalbaar zou zijn in de hier gebruikte klanken-, geluidsterkte en tijdstrukturering worden hier als een plastische massa, een sonore brei, behandeld. Een klankmoeras zo men wil. De uitvoerder speelt en manipuleert de klankmassa een beetje zoals een 'action painter' zijn schilderdoek te lijf zou kunnen gaan.

Rec-Play

Dit stukje heeft een uitgesproken demonstratief karakter. Het laat de speler toe een melodisch verloop min of meer 'in de stille ruimte' te tekenen. Die ruimtelijke tekening wordt als bewegingspatroon opgenomen, en vervolgens als een muzikaal melodisch verloop weergegeven. De opname en weergave verloopt cyclisch waarbij de afzonderlijke cycli door een gongslag en een bel-cimbaal worden aangegeven. De melodieën zijn driestemmig. Voor de instrumentatie worden Nej-fluiten gebruikt.

Prime Time

Deze slagwerkkompositie heeft als genererende kompositorische cel de overlapping van zuiver periodieke metra die in een irrationale verhouding tot elkaar staan. Priemgetallen bepalen de onderlinge verhoudingen der periodetijden. Is de rezulterende ritmische struktuur hier het vaststaande kompositorisch gegeven, dan ligt de interpretatieve vrijheid van de uitvoerder hier geheel op het vlak van de bepaling van de juiste noten (er worden slaginstrumenten voorgeschreven waarvan de toonhoogte bepaalbaar is) evenals van de dynamiek, die bepaald wordt door de inspanning van de speler.

Hammers

In dit stukje wordt een imaginair en hypotetisch cimbalon op de scene geplaatst. Toonhoogte is hier een funktie van de versnelling van de beweging.

Lock ... Unlock

Dit stukje vormt eigenlijk een diptiek met twee kontrasterende panelen. Het eerste luik -Lock- berust op cyclische slagwerkpatronen waarin heel trage fazeverschuivingen optreden. De herhalingstijden zijn in de kompositie vastgelegd, en verschillen slechts een fraktie van elkaar. Een 'klassieke' kompositorische truuk dus die ook aan de bazis ligt van heel wat zogenaamde minimal muziek.

In het tweede luik -unlock- wordt dit repetitief karakter volledig doorbroken en kontroleert de speler de periodetijden van de metrische cycli zelf.

Close

Dit is het sluitstuk van 'A Book of Moves'. Het vormt een antipode van de ouverture 'Open'. Het muzikale verloop is nochtans geen parametrisch omgekeerde imitatie van de ouverture, hoewel ze net zoals die ouverture, een zuiver retorische opbouw kent. Zij is immers opgebouwd als een grote dalende akkoorden progressie eindigend in een finaal achtstemmig unisono. Het verloop van deze progressie -in het bijzonder de ritmische struktuur en de intervalgrootte van de melodische sprongen in de 8 stemmen- wordt volledig door de speler zelf bepaald.

Hoewel 'A Book of Moves' voor vele toeschouwers/luisteraars wellicht zou kunnen overkomen als een toch minstens met dans verwante produktie, is dit toch een fundamentele misvatting. Zoals allicht duidelijk zal zijn uit deze studie gaat het wel degelijk in elk onderdeel van het stuk om een bespeling van instrumenten. Dat deze misvatting kan ontstaan is evident, aangezien het publiek nu eenmaal niet gewoon is koncerten bij te wonen waar de uitvoerders eigenlijk niet veel meer lijken te doen dan wat min of meer expressief over de scene te bewegen.

We kunnen er echter niet genoeg de nadruk op leggen, dat een dergelijke implementatie van een non-impakt instrument zelfs nauwelijks voor toepassingen binnen de wereld van de dans bruikbaar is, en wel omdat dansers vanuit hun vorming en opleiding, steeds de muziek achternahollen. Hun bewegingen staan in funktie van de muziek. Zij genereren deze niet. Dansers produceren gekonfronteerd met dit instrument -en uiteraard hebben we dit bij verschillende gelegenheden uitgeprobeerd- slechts de meest triviale sonore rezultaten. Zij weten niet om te gaan met het bewust hanteren van motoriek in funktie van de klankproduktie. De geproduceerde klanken worden bij hen grotendeels accidenteel, louter sekondair gevolg. Ze slagen er niet in hun beweging in funktie van een klankrezultaat te formuleren. Ook de aangeleerde 'elegantie' die de danstechniek toch kenmerkt, staat een behoorlijk muzikaal rezultaat ernstig in de weg. Hierdoor immers worden bewegingen steeds soepel en afgerond uitgevoerd, terwijl het precies de motorische diskontinuïteiten zijn die de grootste muzikale en expressieve profilering toelaten.

> *Alleen bij ons bezoek aan de Volksrepubliek China in augustus 1992, (Bejing en Shangai) toen we zoals vaak na een opvoering, de scene en daarmee het instrument openstelden voor het publiek, bleek een aantal dansers er toch een en ander van terecht te brengen. We zijn geneigd dit in verband te brengen met de vertrouwdheid van deze dansers met taijiquan enerzijds en gevechtssporten zoals karate anderzijds, waarbij diskontinuïteiten in de bewegingen een grote rol spelen.*

In de enscenering van 'A Book of Moves' wordt normaal gezien een volledig uitgeschreven toneelbelichting voorzien. Deze strekt er uitsluitend toe, de muzikale sfeer te versterken. Zij vormt hier een ondersteunend retorisch element. Het stuk kan overigens ook uitgevoerd worden zonder deze belichtingen.

Een laatste verwarring die bij sommige luisteraars soms kan ontstaan, is dat gedacht wordt dat het hier gaat om een soort bewegingsgestuurde sequencer. Dat dit principieel onjuist is, zal wel duidelijk zijn uit onze studie.

3.4.: Besluit

Tot besluit van dit hoofdstuk menen we, mede aan de hand van de volledige beschrijving van een operationeel prototype van een alternatief non-impakt instrument, aangetoond te hebben dat:

1. Een non-impakt instrument gebouwd kan worden als een user-interface, losgekoppeld van datgene wat de klanken genereert.

2. De programmeerbaarheidseis zoals we die stelden in ons eerste hoofdstuk, en die niet kon worden ingelost in de technische implementatie voorgesteld in het tweede hoofdstuk, kan worden voldaan.
Nochtans dient de aandacht erop gevestigd te worden, dat deze programmeerbaarheid voor de gebruiker niet tijdens het bespelen van het instrument beschikbaar is! Hij dient het instrument op voorhand aan de gewenste wijze van reageren op motorische input aan te passen. Dit kan als een zwak punt van onze realizatie zoals hier voorgesteld gezien worden. Anderzijds maakt deze eigenschap het voorgestelde instrument, wat dit betreft, niet slechter dan enig ander bestaand instrument.

3. Dat voor een praktische realizatie uitgaand van informatieverwerving waarbij beroep wordt gedaan op hard- zowel als software, het punt waar de hardware stopt en de software begint, afhankelijk is van de huidige stand en bereikbaarheid van de technologie en dus gekozen kan worden. De keuzes zoals wij die effektueerden in de hier voorgestelde implementatie zijn dan ook niet beslissend of imperatief. Zij werden in eerste plaats ingegeven door een bekommernis om 'eenvoud'. Andere implementaties kunnen zowel volstrekt evenwaardig als vele malen beter zijn.

4. We erin slaagden uit de globale motorische input relevante informatie te extraheren in digitale vorm en met betrekking tot volgende parameters van de globale beweging:
- bewegingssnelheid
- bewegend oppervlak

en dit in drie ruimtelijke dimensies.

5. Dat globale positionele informatie met betrekking tot de motorische input in beperkte mate verworven kan worden, maar dat de globale non-positionele informatie daarvan onafhankelijk kan worden gemaakt. De keuze tussen een positionele of non-positionele ('energetische') definiering van het instrument, is afhankelijk van het programma geschreven door de gebruiker.