Universidade de São Paulo - USP
Instituto de Matemática e Estatística – IME
Mestrado em Ciência da Computação
Tópicos em Ciência da Computação

Sistemas Musicais Interativos

Mariana Zaparolli Martins
Orientador: Marcelo Gomes de Queiroz

São Paulo, junho de 2005

Índice

1. Introdução

A computação é um campo de pesquisa e desenvolvimento extremamente dinâmico que tem conquistado, com os avanços tecnológicos, sua importância em todas áreas, inclusive as ligadas à música. Muitas pesquisas têm sido realizadas com objetivos diversos através da aplicação do poder computacional em vários contextos musicais. Neste relatório serão apresentados os estudos feitos sobre Sistemas Musicais Interativos.

Sistemas Interativos em Computação Musical são aqueles cujo comportamento muda em resposta a uma entrada musical. Existem muitos Sistemas Interativos de perspectivas de diferentes campos como teoria musical, inteligência artificial e ciência cognitiva, cada campo tem desenvolvido técnicas apropriadas [12]. A interatividade muda qualitativamente a natureza dos experimentos com algoritmos de composição: o efeito dos diferentes valores das variáveis de controle na saída sonora do método podem ser percebidas imediatamente, visto que as variáveis podem ser manipuladas em tempo-real.

Este estudo começa abordando alguns conceitos fundamentais para o entendimento do assunto, como a psicoacústica e algumas discussões sobre como a máquina “entende” a música. Em seguida é apresentada uma classificação de Sistemas Interativos, passando então para os estágios de processamento destes sistemas, onde é dada uma visão geral do padrão MIDI. Na seqüência, são mostradas as idéias de alguns trabalhos desenvolvidos com sistemas interativos que fazem acompanhamento computacional e por fim, as conclusões são apresentadas.

2. Fundamentos

2.1 Psicoacústica

A percepção musical humana tende a combinar informações sonoras de acordo com o contexto sonoro de formas muito variadas. Por exemplo, se duas notas musicais são similares e tocadas muito próximas, o cérebro apenas percebe uma delas. Também se dois sons são diferentes, mas um muito mais forte que o outro, o cérebro humano não notará o som mais fraco. Essa percepção do que é captado ou não pelo ouvido humano ainda pode variar de pessoa para pessoa. O estudo desse fenômeno é chamado psicoacústica, que utiliza modelos matemáticos para representar padrões da audição humana, descritos em tabelas e quadros.

Esses modelos são aplicados, por exemplo, na maioria dos codificadores de MP3 (*Motion Picture Expert Groups Layer 3*), que analisam o som original, modelam-no em um padrão matemático e comparam esse padrão com os modelos da psicoacústica humana [10].

2.1.1 Formas de onda

Vários tipos de informações consistem em composto de sinais de natureza ondulatória, que apresentam diferentes comprimentos de onda. Entre os extremos do espectro de ondas estão os comprimentos de ondas que podem ser captados pelos seres humanos, como a luz e o som.

Além do domínio da luz e do som, estão vibrações sub e ultra-sônicas, raios ultravioletas e infravermelhos, e uma infinidade de outras freqüências imperceptíveis aos seres humanos, como rádio e microondas. Os próprios instrumentos musicais criados pelo homem geram muitas freqüências que nossos ouvidos não podem reconhecer. Em geral, seres humanos não podem ouvir freqüências abaixo de 20Hz (20 ciclos por segundo) nem acima de 20kHz (20.000 ciclos por segundo), como mostrado na Figura 1. Embora a capacidade de audição varie de indivíduo para indivíduo, é genericamente aceito que os seres humanos percebem freqüências médias mais fortemente do que as mais altas e mais baixas, também, que a sensibilidade para as freqüências altas diminui com a idade e a constante exposição a volumes altos. De fato, pessoas adultas dificilmente escutam algo acima de 16kHz (embora as mulheres tendem a manter a capacidade de ouvir freqüências altas por mais tempo que os homens). A maioria das pessoas ouve sons entre 2kHz a 4kHz, um nível provavelmente relacionado com o alcance da voz humana, que fica entre 500Hz e 2kHz [10].

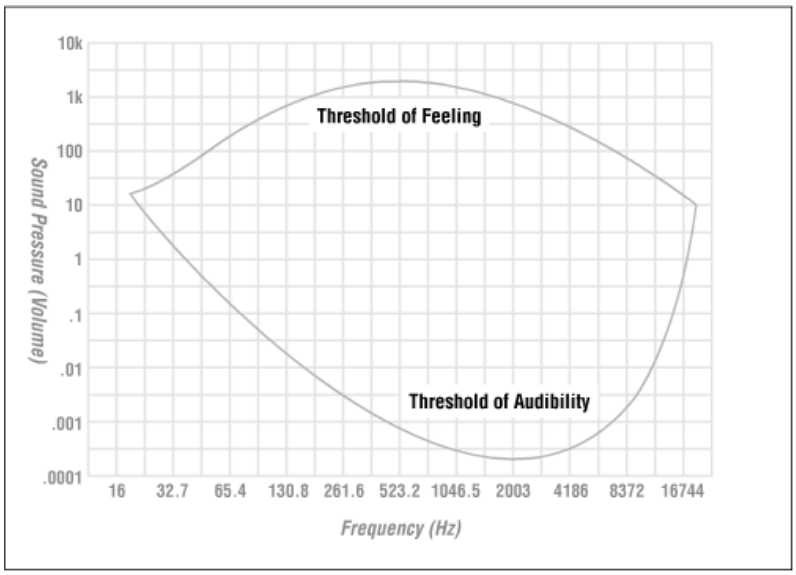


Figura 1

2.2 Máquina com conhecimento musical

Ao usar um computador para composição ou performance, a pergunta fundamental a ser feita sobre qualquer sistema em particular é: para que propósito musical isto serve?

A resposta de sistemas interativos requer que eles façam alguma interpretação da entrada fornecida. Entretanto, uma questão central é: o que a máquina pode ouvir? A maioria dos sistemas fazem uso da abstração oferecida pelo padrão MIDI (*Musical Instrument Digital Interface*) que será melhor abordado no próximo capítulo. Um nível adicional de informação pode ser coletado da análise do sinal de áudio emitido por instrumentos musicais acústicos. Uma entrada MIDI e sinais de áudio são representações que devem ser processadas pelo programa para alcançar qualquer objetivo musical particular.

Muitos programas interpretam a entrada através da emulação do entendimento musical humano. Entretanto, programar um computador para exibir aptidão musical semelhante a um humano é um objetivo com implicações para um grande conjunto de aplicações. Claro que para alguns programas de computação musical, algum grau de conhecimento específico poderia prover maior performance e utilidade nas aplicações. Por exemplo, na edição automática de gravações de áudio digital, a combinação de acessos simultâneos a representações baixo-nível no sinal com construções alto-nível, familiares aos músicos, podem trazer ótimos resultados. Com sistemas interativos, interpretar a entrada é indispensável.

2.2.1 Capturando Conceitos Musicais

Na interpretação de uma entrada musical, sistemas interativos implementam algumas coleções de conceitos, normalmente relatados nas estruturas que os músicos comumente assumem. Cada sistema interativo também inclui métodos para construção de respostas a serem geradas quando construções particulares de entrada são encontradas. Os métodos de interpretação abordam a bem sucedida representação de conceitos musicais humanos e os algoritmos de resposta movem-se na direção de emular as práticas de performance humana.

Novamente uma pergunta surge, porque é importante para o programa de computador abordar práticas da performance humana? De fato, passando pelos recentes estágios da eletrônica e do desenvolvimento da computação musical, segundo alguns pesquisadores, um objetivo foi à eliminação de músicos humanos com todas as suas limitações e variabilidade. Para eles, a computação musical tem sido um veículo perfeito para a eliminação da personalidade do músico.

A eliminação total de músicos humanos não é desejada, pois os músicos entendem o que é música, como ela funciona e podem comunicar o entendimento musical a uma audiência. Fazer uma coordenação entre uma gravação fixa e uma performance expressiva e variável de instrumentistas pode ser problemática. Algumas dificuldades são mais pronunciadas quando a improvisação faz parte do discurso. E, se a gravação e a realização da performance forem justapostas, a disparidade entre os níveis de musicalidade é evidenciada [12].

2.2.2 Composição por Refinamento

Sistemas musicais interativos contribuem para o processo de composição por refinamento, o programa reage imediatamente às mudanças na configuração da entrada. Assim, um usuário pode desenvolver uma aplicação de composição pelo refinamento contínuo das idéias iniciais.

Muitos sistemas interativos podem ser considerados Teorias Musicais Aplicadas. A teoria musical descreve o processo de compor ou ouvir música. A implementação em um programa de computador demanda formalização da teoria de uma forma que uma série de instruções de máquina possam realizá-la. Para teorias aceitáveis, a adição de rigor para realização pelo computador pode clarear suas formulações e torná-las disponíveis de uma forma na qual elas possam ser extendidas ou usadas em outras tarefas computacionais. Quando a teoria é trazida para o ponto da interatividade, pode ser aplicada para a produção e análise de música no seu ambiente nativo, isto é, tocada e experimentada como a música ao vivo [12].

3. Classificação de Sistemas Interativos

A classificação apresentada a seguir foi proposta por Robert Rowe em [12], motivada não simplesmente para dar rótulos aos programas, mas reconhecer similaridades entre eles e ser capaz de identificar as relações entre novos sistemas e seus predecessores.

Esta classificação de sistemas foi construída sob uma combinação de três dimensões, cujos atributos ajudam a identificar as motivações musicais como tipos de interpretação da entrada e métodos de resposta.

3.1 Dirigido por Partitura x Dirigido por Performance

A primeira dimensão distingue sistemas dirigidos por partitura de daqueles que são dirigidos por performance.

Programas dirigidos por partitura usam coleções de eventos pré-determinados ou fragmentos de música armazenados para correlacionar (*match*) com a música que chega da entrada. Eles são amigáveis para organizar eventos usando as categorias tradicionais de pulso, compasso e andamento.

Programas dirigidos por performance não pressupõem a realização de qualquer partitura particular. Eles não possuem uma representação da música armazenada que eles esperam encontrar na entrada. Estes programas tendem a não empregar as categorias métricas tradicionais, usam normalmente parâmetros mais gerais envolvendo medidas de percepção como densidade e regularidade para descrever o comportamento temporal da música que chega.

3.2 Métodos de Resposta

A segunda dimensão de distinção são os métodos de resposta, podendo ser transformativo, generativo ou seqüenciado.

Métodos Transformativos coletam material musical existente e aplicam transformações para produzirem variantes. De acordo com a técnica, essas variantes podem ou não ser reconhecidamente relacionadas com o original. Para algoritmos transformativos, o material fonte é a entrada musical completa, mas não necessariamente armazenada, as transformações são aplicadas à entrada que chega.

Nos algoritmos Generativos, por outro lado, o material fonte será elementar ou fragmentado, por exemplo, escalas armazenadas ou conjuntos de durações. Os métodos generativos usam conjuntos de regras para produzir uma saída musical completa vinda do material fundamental armazenado.

Técnicas Seqüenciadas usam fragmentos de música pré-gravada em resposta à entrada em tempo-real. Alguns aspectos desses fragmentos podem ser variados na reprodução, como o tempo de *playback*, a forma da dinâmica, variações rítmicas, etc.

3.3 Modelos

E por último, pode-se diferenciar modelos Instrumento e Instrumentista.

Sistemas com modelo Instrumento são envolvidos com construção de um instrumento musical extendido: gestos executados de um instrumentista são analisados pelo computador e guiam a saída elaborada, excedendo a resposta instrumental normal. Imaginando como se o sistema fosse tocado por um único músico, o resultado musical seria como um solo.

Sistemas que seguem o modelo Instrumentista tentam construir um tocador artificial, uma presença musical com personalidade e comportamento próprios e pode variar em um grau no qual é seguido pela liderança de um parceiro humano. Um sistema modelo Instrumentista tocado por um único humano produziria uma saída como um dueto.

3.4 Exemplo

Score Followers são um grupo de programas capazes de acompanhar um solo instrumental humano pela correlação da realização de uma partitura particular com a representação armazenada da partitura, simultaneamente tocando uma parte do acompanhamento armazenada. Suas aplicações são exemplos perfeitos de sistemas dirigidos por partitura. A técnica de resposta é seqüenciada, tudo que a máquina toca vai sendo armazenada antes de um tempo particular. Finalmente, *score followers* podem ser baseados nos sistemas de modelo instrumentista, porque eles percebem uma linha musical reconhecidamente separada, assumindo o papel tradicional de acompanhante na performance de sonatas instrumentais.

4. Estágios de Processamento

Os dois pilares básicos de sistemas interativos são: manipulação de MIDI e escalonamento (*scheduling*). O padrão MIDI foi desenvolvido por produtores de instrumentos e sintetizadores e permite que controladores, computadores e sintetizadores passem dados entre eles mesmos. O segundo pilar de programas musicais é capaz de executar tarefas em pontos de tempo específicos. A música é uma arte temporal, qualquer programa de computador que lidar com ela deverá possuir facilidades sofisticadas para representar o tempo e para agendar processos a serem executados em um instante particular.

A cadeia de processamento de sistemas interativos de computação musical pode ser conceitualizada em três estágios [12]:

1) Estágio de Captação: é o estágio inicial, onde os dados são coletados dos controladores pela leitura da informação gestual vinda dos músicos humanos.
2) Estágio de Processamento: é o estágio de processamento, o computador lê e interpreta a informação vinda dos sensores e prepara os dados para a próxima etapa.
3) Estágio de Resposta: é o estágio de construção da resposta, quando o computador e alguma coleção de recursos de produção de som se dividem na produção da saída musical.

4.1 Captação

Os produtores comerciais dominam os estágios de captação e resposta através de controladores e sintetizadores MIDI. A sofisticação desses recursos, que implementam vários modos de operação e podem ser programados, requer uma discussão desses estágios, ao invés de simplesmente um programa de computador em tempo-real com entrada e saída.

4.1.1 O padrão MIDI

O padrão MIDI é uma especificação de *hardware* e protocolo de comunicação que permitem que os computadores, controladores e sintetizadores sejam projetados de forma a passarem informação entre eles. MIDI faz a abstração do nível do sinal acústico da música para uma representação mais alto-nível baseada no conceito de eventos musicais que compreendem o início e o fim e cada nota e a intensidade (*velocity*).

A abstração MIDI é eminentemente bem sucedida para instrumentos com teclado, como o piano, que podem ser representados como uma série de *switches* (chaves que ligam/desligam). De fato, a captação MIDI, em alguns instrumentos, opera tratando cada tecla separada como um *switch*. Quando a tecla é pressionada, uma mensagem *Note On* é enviada, indicando qual tecla foi apertada e com qual intensidade. Quando a tecla é solta,

uma mensagem *Note Off* (ou equivalentemente *Note On* com intensidade e zero) é transmitida com o número da tecla.

O conceito da nota é o paradigma fundamental MIDI, todos os instrumentos MIDI implementam isto. Tratando de controladores de teclado comercial, MIDI representa variações de controle contínuas. Uma delas é conhecida com *pitchbend*, normalmente implementada com uma pequena roda colocada no instrumento. A cada tempo que a roda é movida, um novo valor de *pitchbend* é transmitido ao estágio de processamento. Assim, contínuas mensagens de controle são transmitidas com um número e um valor controladores. Muitos instrumentos permitem ao usuário assinalar os números de controle a recursos físicos, remapeando a roda de *pitchbend*, por exemplo, para outro canal. A geração de controles contínuos é, na maioria dos casos, facilmente reconfigurável.

Apesar da introdução do padrão MIDI ter tido um grande efeito de expansão na pesquisa e na performance de sistemas musicais interativos, o uso deste padrão impôs limitações de várias formas. Primeiro porque MIDI é um protocolo de controle – performance de eventos gestuais ao invés de qualquer representação de sinal de áudio – o padrão não pode ser usado para descrever ou controlar mais que o aspecto tímbrico de uma performance musical. O controle sobre a síntese e a evolução no tempo de um som particular está codificado em cada sintetizador e pode ser afetado através do padrão MIDI apenas pelo uso de uma específica coleção de *triggers* (gatilhos), controles contínuos e comandos de sistema exclusivos, limitando cada máquina sobre a idéia do padrão.

Em uma típica transação, uma mensagem *Note On* emitida de um computador ou controlador, irá desencadear uma complexa reação do sintetizador, percebendo a porção correspondente ao ataque do som. Controles contínuos podem afetar variáveis do algoritmo de síntese durante todos os estágios da síntese sonora, modificando alguma característica, como as freqüências de corte do filtro ou a amplitude de modulação do oscilador.

Uma das limitações principais para a performance está no fato de que o padrão MIDI força a banda de transmissão em 31.250 bits por segundo. Cada byte (8 bits) MIDI é circundado por um bit de início e um bit de parada, fazendo com que seu tamanho efetivo seja de 10 bits. Entretanto, uma mensagem padrão MIDI *Note On* requer 3 bytes (30 bits) de informação, levando aproximadamente 1 milisegundo para transmitir, assim, a performance com MIDI de um acorde com 10 notas irá introduzir um atraso de 10 milisegundos entre a primeira e a última nota do acorde. Entretanto, 10 milisegundos não são suficientes para afetar a percepção do evento como um simples acorde, isto pode ter um efeito na qualidade tímbrica do som. Porém, quando nós consideramos o impacto de 1 milisegundo no tempo de transmissão na performance de um acorde de 10 notas, mandado simultaneamente para 10 canais, os 100 milisegundos de atraso entre a primeira e a última notas, certamente serão ouvidos [12].

Este padrão foi originalmente projetado para prover uma baixa latência de transmissão de mensagens musicais entre os recursos, porém alguns questionamentos têm sido feitos a respeito de sistemas altamente interativos em tempo-real. Quantificar a latência do padrão MIDI é crucial porque mesmo uma pequena variação de tempo pode ser musicalmente perceptível, especialmente quando pequenos ornamentos estão presentes, como trilos e glissados.

Pesquisadores propuseram valores tão baixos como de 1 a 1.5 milisegundos como um intervalo aceitável de variação de latência e cerca de 10 milisegundos como um aceitável limite superior na latência absoluta. Porém quantificar a latência de um sistema depende fortemente das partes do sistema e da aplicação em particular [7].

Uma mensagem MIDI é composta por um byte de status, que identifica o tipo da mensagem, seguido na maioria dos casos por um ou mais bytes de dados. O bit mais significante dos bytes de dados é sempre 0, para distingui-los do byte de status, cujo bit mais significativo é 1. Exceto para mensagens exclusivas do sistema, o byte de status especifica exatamente quantos bytes de dados o sucedem.

Quando nós consideramos o MIDI como um canal de comunicação não apenas para mensagens de nota (*Note messages*), mas para o estado de controladores contínuos, os problemas de largura de banda se tornam mais sérios. O padrão não provê uma boa forma de controlar a evolução interna do som, como facilidades existentes que não podem ser passadas rapidamente através de 31.250 bits por segundo, para permitir ao músico um controle rígido sobre o som resultante. O usuário deve programar o sintetizador com instruções de forma a dar evolução ao som em resposta aos gatilhos mandados pelo computador [12].

O baixo preço dos componentes de *hardware* MIDI e a rápida disponibilidade tornam este padrão uma freqüente escolha dos pesquisadores para construção de sistemas musicais interativos.

4.1.2 Controladores Tradicionais

A informação pode chegar dos sensores em uma forma diferente do padrão MIDI, outro tipo de entrada importante são as amostras (*samples*), a representação digital de um sinal de áudio variante no tempo. Normalmente, o som de algum instrumento musical é captado por um microfone, passado por um conversor analógico-para-digital (ADC) e então para o computador. A taxa de amostragem para produção de áudio digital em qualidade de *Compact Disc* (CD), produz pelo menos 44.100 amostras de 16 bits por segundo, então lidar com amostra de fluxo contínuo de dados não tratados demanda alto poder de processamento. Por esta razão, sistemas interativos projetados para manipular áudio possuem recursos de *hardware* dedicados com a habilidade de processar sons de intensidades requisitadas.

Controladores MIDI podem ser pensados como transdutores de gestos provendo uma representação de uma performance musical humana. Performances com teclado são representadas muito bem, mas informações gestuais importantes de performances de outros instrumentos (cordas, sopros, voz), entretanto, não são totalmente capturadas por sensores MIDI. Por esta razão, uma pesquisa significante tem crescido em volta do objetivo de construir controladores melhores, capazes de capturar um conjunto de expressões de instrumentos musicais tradicionais [12].

4.2 Processamento

A informação coletada durante o estágio inicial (captação) é passada para o computador, que começa o estágio de processamento. A comunicação entre os dois estágios assume um protocolo entendido pelos dois lados e o protocolo mais amplamente usado é o padrão MIDI. Outros sinais passam entre eles e podem incluir áudio digital ou informação de controle.

Sistemas interativos sempre necessitam implementar um *driver* MIDI, que é capaz de ir armazenando um fluxo contínuo de informação serial vinda da porta física e empacotar em séries de comandos MIDI. A maioria das plataformas de hardware resolve os problemas da transmissão MIDI através de um *driver* padronizado e de um *software* associado.

Na tarefa de receber e empacotar comandos MIDI válidos, *drivers* MIDI introduzem *time stamps*, uma extensão crítica do padrão. *Time stamps* são indicações do tempo no qual algum pacote MIDI chega no computador. Com a adição do *time stamp*, a crítica informação temporal da produção da música torna-se disponível. Processos de interpretação podem usar essa informação para analisar a parte rítmica da seqüência MIDI que chega. Informação temporal é também necessária para que o computador prepare os dados para a saída através do estágio de resposta e aqui a função de escalonamento (*scheduling*) em tempo-real começa a atuar.

4.2.1 Escalonamento em Tempo-Real

Scheduling é o processo no qual uma ação específica do computador é determinada para ser executada em algum ponto no tempo futuro. Tipicamente, um programador pode usar as facilidades do escalonador para atrasar a execução de um procedimento por alguns milisegundos. Os argumentos da rotina a ser executada são salvos junto com o nome da própria rotina. Quando o escalonador noticia que o ponto específico no tempo chegou, o processo escalonado é chamado com os argumentos salvos.

Uma implementação é apresentada por Robert Rowe em [12]. Nela, um relógio em centisegundos é mantido pelo MIDI *driver* e é referenciado pelo *driver* através dos *time stamps* dos dados MIDI que chegam. O relógio grava o número de centisegundos que vão passando desde que o *driver* MIDI foi aberto. Dessa forma, eventos MIDI que chegam recebem um *time-stamp* através de uma interrupção de *hardware*, quando eles chegam na porta serial. O mesmo relógio é usado pelo escalonador para temporizar a execução das tarefas escalonadas.

Tarefas são associadas com um número de atributos, com controle inicial e possíveis execuções repetidas da função escalonada. Pode-se identificar esses atributos na lista de argumentos da rotina *Scheduler_CreateTask* que insere uma função na fila de tarefas do escalonador.

```
TaskPtr
Scheduler_CreateTask (time, tol, imp, per, fun, args)
        short imp, tol, per;
        long time;
        void (*fun)();
        arglist args;
```

O primeiro argumento é o *time:* o tempo absoluto no qual a função deve ser executada. Tempo absoluto significa que o ponto no tempo é expressado em centisegundos desde a abertura do *driver* MIDI.

O segundo argumento é o *tol*: a tolerância é a quantidade de tempo permitida para iniciar a função. Se existir o argumento *tol*, a função será chamada no ponto do tempo calculado subtraindo a tolerância (*tol*) do tempo (*time*). Isto acomoda rotinas cujos efeitos serão notados alguma quantidade fixa de tempo após sua execução. Neste caso, a função pode ser escalonada para o instante de tempo onde o efeito é desejado e o ponto de início é passado como o argumento de tolerância.

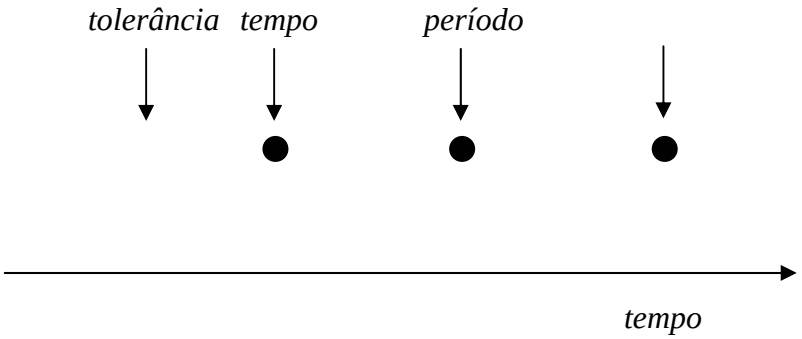


Figura 2

Na figura acima as bolas pretas representam eventos sonoros produzidos por algum processo *player*. O processo necessita de uma quantidade fixa de tempo para produzir o som – uma tolerância – representada pela seta mais à esquerda. A chamada ao *Scheduler_CreateTask*, dará então o tempo (*time*) para a chegada desejada do evento sonoro, com o argumento de tolerância (*tol*) para prover o processamento adiantado necessário.

O escalonador mantém três filas priorizadas em separado e garante que todas as tarefas na espera das filas de alta-prioridade, serão executadas antes que qualquer tarefa das filas de baixa-prioridade seja invocada. O argumento *imp*, importância, determina qual prioridade a tarefa indicada irá receber. Se um argumento *per*, período, não-zero foi incluído, a tarefa irá se reescalonar *per* centisegundos depois de cada execução. Periodicidade é um forte atributo de produção musical e a facilidade do argumento *per* manipula facilmente esta necessidade. Tarefas que são periodicamente reescalonadas por elas mesmas podem ser encerradas por um explícito comando *kill* em qualquer ponto no tempo. Na figura 2, o argumento do período irá continuar os eventos sonoros em intervalos regulares depois do primeiro ser mostrado. O argumento da tolerância continua atuante para cada chamada, provendo o tempo de processamento adiantado requerido. Finalmente, um

ponteiro para a função a ser chamada e os argumentos a serem mandados para função em execução são listados. Os argumentos são avaliados no tempo escalonado, ao invés de quando a função for realmente invocada.

É importante notar que o algoritmo supõe que nenhuma das rotinas escalonadas irá requerer um longo tempo de processamento. Uma vez que um procedimento é chamado pelo escalonador, ele executa até o fim, e então o controle volta para o escalonador novamente. Essas rotinas não são interrompidas exceto por um número estritamente limitado de manipuladores de entrada. Se algum procedimento requerer um processamento maior, ele deverá se reescalonar em intervalos regulares para permitir ao escalonador tempo de CPU suficiente para manter a execução das outras tarefas na fila.

4.3 Resposta

O estágio de resposta se parece com o estágio de captação, mais fortemente no protocolo de comunicação usado. Novamente, o computador e os recursos utilizados para dar as respostas se comunicam comumente através do padrão MIDI. A exata natureza das direções mandadas na saída pelo computador irá depender dos recursos de síntese em uso e dos tipos de efeitos que estes recursos utilizarão. Atualmente, recursos MIDI comerciais tendem a cair em dois grandes grupos: síntese e amostragem (*sampling*).

Os módulos de síntese usam um algoritmo - por exemplo, modulação de freqüência – para produzir sons. Módulos de amostragem têm formas de onda armazenadas normalmente gravadas de instrumentos acústicos tradicionais, que são tocadas sob notas específicas em resposta às mensagens MIDI. Comandos de resposta são mandados aos recursos, incluindo mensagens *Note On/Off* e outros valores de controle que são desejados para manipulação da produção de som.

5. Acompanhamento Computacional

Um dos mais interessantes desenvolvimentos em computação musical é uma classe de programas chamada Acompanhante Computacional. Esses programas fazem justamente que seus nomes sugerem: eles tocam o acompanhamento para um solo executado pelo músico.

Os acompanhantes de tempo-real requerem que o computador tenha a habilidade de seguir o solista. Três sub-problemas surgem: a detecção do que o solista está fazendo, a correlação da entrada detectada com a partitura armazenada que contém a entrada esperada e a produção do acompanhamento, cuja partitura também está armazenada [1].

É esperado que o solo tocado contenha erros ou que seja imperfeitamente detectado pelo computador, então é necessário haver uma tolerância a erros na performance quando se correlacionar o solo atual com a partitura.

Dannenberg em [1], foca o segundo subproblema e propõe um eficiente algoritmo de programação dinâmica para encontrar a melhor correlação entre a performance solo e a partitura. A estratégia da programação dinâmica aborda um dado problema, dividindo-o em subproblemas mínimos, solucionando os subproblemas, guardando os resultados parciais, combinando subproblemas menores e sub-resultados para obter e resolver problemas maiores, até recompor e resolver o problema original. O problema é decomposto uma única vez e os subproblemas menores são gerados antes dos subproblemas maiores, até recompor o problema original.

Nesta solução apresentada, utiliza-se uma matriz para encontrar a melhor correlação, onde cada linha corresponde a um evento da partitura e cada coluna corresponde a um evento detectado na performance. Assume-se que o sistema detecta uma classe limitada de eventos pré-determinados, que podem ser, por exemplo, teclas apertadas em um piano ou as posições dos pistões de um trompete. Ainda é implementado um relógio virtual para dar condições para que controles temporais sejam feitos.

Porém, limitações desta implementação foram notadas, então Dannenberg e Mukaino em [2] mostram formas de resolver algumas dessas limitações. As melhorias do sistema são desenvolvidas sob três idéias principais: o uso de uma quantidade limitada de paralelismo, para permitir que o sistema siga duas hipóteses competidoras até que uma lhe pareça superior; o atraso da ação de correlação, criando-se uma pequena janela de tempo enquanto uma decisão pode ser revertida e por fim, métodos para manipulação de ornamentos da partitura (g*race note*, trilos e glissandos [Apêndice A]).

Existem também outras formas de se tratar o problema, alguns pesquisadores utilizam as chamadas HMMs (*Hidden Markov Models*) nas suas implementações pois as HMMs podem lidar com vários níveis de imprevisibilidade. Schwarz, Orio, Lemouton e Schnell mostram em [6] e em [9] uma implementação feita, descrevendo a modelagem utilizada, os conceitos envolvidos e os resultados obtidos.

6. Conclusão

Neste trabalho foram apresentados os conceitos básicos referentes ao projeto de um sistema musical interativo, bem como foram discutidos alguns detalhes de implementação referentes ao estágio de processamento de tal sistema. Estes estudos fornecem um ponto de partida para a elaboração do texto do exame de qualificação a ser realizado brevemente.

Partes da implementação do sistema envolvem a ferramenta MAX/MSP, que é uma linguagem gráfica orientada a objetos para manipulação de informação de áudio e MIDI, bem como possivelmente a ferramenta Jitter para maniputação de imagens. As aplicações específicas de interação entre áudio e vídeo estão sendo estudadas em colaboração com o Professor Fernando Iazzetta, do departamento de Música da ECA/USP.

7. Referências Bibliográficas

[1] Dannenberg, R. B. 1984. "*An On-Line Algorithm for Real-Time Accompaniment*". Proceedings da *Internacional Computer Music Conference* (ICMC) 1984. San Francisco: Computer Music Association, pp 193-198. Disponível em: http://www-2.cs.cmu.edu/~rbd/papers/icmc84accomp.pdf

[2] Dannenberg, R. B e Mukaino, H. 1988. "*New Techniques for Enhanced Quality of Computer Accompaniment*". Proceedings da *Internacional Computer Music Conference* (ICMC), 1988. Disponível em: http:// www-2.cs.cmu.edu /afs/cs.cmu.edu /user/rbd/www/papers/acp.pdf

[3] Buxton, W., Dannenberg R. e Vercoe B. 1986. "*The Computer as Accompanist*". Proceedings CHI'86, pp 41-43. Disponível em: http://portal.acm.org

[4] Puckette M. e Lippe C.1992. "*Score Following in Practice*". Proceedings da *Internacional Computer Music Conference* (ICMC), 1992. Disponível em: http://www.music.buffalo.edu/lippe/pdfs/sanjose.pdf

[5] Puckette M.. 1990. "*EXPLODE: A User Interface for Sequencing and Score Following*". Proceedings da *Internacional Computer Music Conference* (ICMC) 1990, pp 259-261. Disponível em: http://mediatheque.ircam.fr/articles/textes/Puckette90a/

[6] Orio N., Lemouton S., Schwarz D. 2003. "*Score Following: State of the Art and New Developments*". Proceedings da *Internacional Computer Music Conference* (ICMC) 2003. Disponível em: http://suac.net/NIME/NIME03/NIME03_Orio.pdf

[7] Thom B. e Nelson M. 2004. "*An In-Depth Analysis of Real-Time MIDI Performance*". Proceedings da *Internacional Computer Music Conference* (ICMC) 2004, pp 430-437. Disponível em: http://www.cc.gatech.edu/~mnelson/papers/Thom_and_Nelson_-_MIDI_Performance_-_ICMC04.pdf

[8] Taki Y., Suzuki K., Hashimoto S. 2000. "*Real-time Initiative Exchange Algorithm for Interactive Music System*". Proceedings da *Internacional Computer Music Conference* (ICMC) 2000. Disponível em: http://www.shalab.phys.waseda.ac.jp/~kenji/pdf/ICMC2000-II.pdf

[9] Schwarz D., Orio N. e Schnell N. 2004. "*Robust Polyphonic Midi Score Following with Hidden Markov Models*". Proceedings da *Internacional Computer Music Conference* (ICMC) 2004, pp 442-445. Disponível em: http://recherche.ircam.fr/equipes/analyse-synthese/schwarz/publications/icmc2004/scofo-midi.pdf

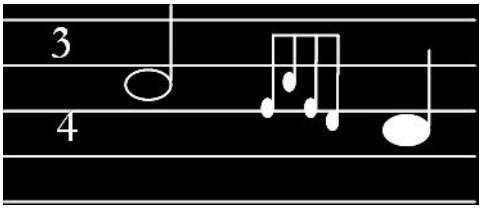
[10] Martins M., Pedroso J. e Vetromille M. "*Especificação de uma Arquitetura para Decoders de arquivos MP3*". Projeto de Graduação em Engenharia de Computação da Fundação Universidade Federal do Rio Grande – FURG RS.

[11] Cycling 74. 2004. "*MAX/MSP 4.5 Documentation*". Disponível em http://www.cycling74.com/products/dldoc.html

[12] Rowe R. 1994. "*Interactive Music Systems: Machine Listening and Composing*". The MIT Press, Cambridge, England.

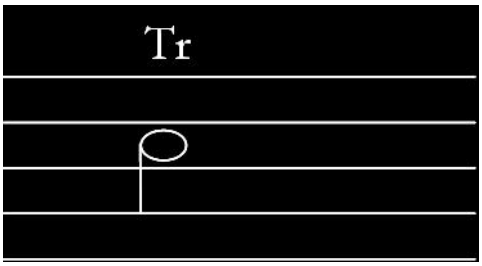
Apêndice A

- ***Grace Notes (ornamentos)***

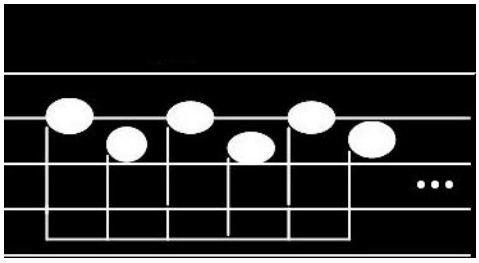


- ***Trilo***

Escreve

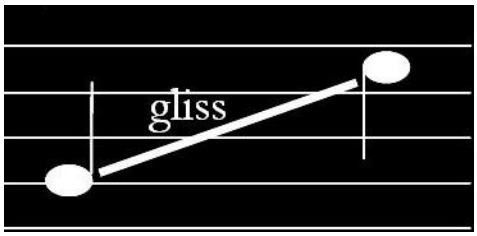


Soa



- ***Glissando***

Escreve



Soa