Improvisação de Ventos

Eduardo Vinícius Piva Furtado
e-mail: evpf@hotmail.com

José Eduardo Fornari Novo Júnior
IA (Professor) e NICS (pesquisador)
e-mail: tutifornari@gmail.com
web: www.nics.unicamp.br

Resumo: este trabalho pretende relatar a criação de um recital, onde estarão interagindo um instrumento acústico (saxofone barítono) e musica gerada por computador. A musica gerada pelo computador será produzida em um ambiente de programação gráfica para áudio e vídeo chamado **Pure Data (PD)**. Explicaremos o processo de criação do *patch* (conjunto de objetos que formam um módulo) que ira interagir com o saxofone até a apresentação do recital.

Método

Para a criação deste recital utilizaremos o PD, um ambiente de programação gráfica em tempo real para áudio e vídeo. O PD é um software livre e foi desenvolvido originalmente por Miller Puckette. Para programar no PD usamos pequenas caixas onde colocamos informações que são processadas pelo computador para gerar um resultado. Estas caixas podem ser objetos, mensagens ou números, e são ligadas umas as outras através de linhas, como cabos que interligam equipamentos físicos.

Implementação

Iniciamos a implementação inserindo um objeto que controla o áudio do computador. Existem varias maneiras de se fazer isto e optamos por um objeto chamado [pddp/dsp]. Este objeto cria um ícone que muda de cor quando esta ativado, facilitando a visualização de seu estado.

Em seguida criamos um objeto [noise~]. Este objeto cria um ruído. Ligamos a ele outro objeto chamado [lop~], um filtro passa-baixa. O filtro passa-baixa permite que as frequências que estão em sua banda de passagem não sofram modificações em sua amplitude, minimizando as frequências que se encontram na faixa de atenuação. Criamos então um objeto [dac~] para podermos ouvir o resultado sonoro. O [dac~] é um conversor de áudio de digital para analógico, o que possibilita que os sons gerados pelo PD sejam reproduzidos em alto-falantes. Entre [lop~] e [dac~] conectamos um objeto [*~] para controlar o volume e o silenciamento do som. Em

seu inlet direito conectamos [vline~] e conectado a este duas caixas de mensagem: [0.8 2000], para que o volume chegasse a 0.8 em 20 segundos e [0 30000], para o volume chegar à zero em 30 segundo. Em [0 30000] conectamos [delay 60000] para que esta caixa fosse acionada após 60 segundos, assim o *patch* seria executado num total de 90 segundos. Para modificar as frequências criamos um [slider] vertical e conectamos ao inlet direito de [lop~]. Variamos os valores da frequência ate encontrarmos os pontos que eram de nosso interesse. Resolvemos utilizar as frequências entre100 e 600 Hz.

Criamos então seis caixas de mensagem onde colocamos as frequências desejadas. Criamos um objeto [line] e conectamos as caixas de mensagem a este. Achamos interessante que a transição entre as frequências não fossem abruptas, então colocamos na caixa de mensagem de cada frequência um valor que corresponde ao tempo em milissegundos que levara para chegar nesta frequência, de forma gradual. O objeto [line] cria uma rampa que faz a transição entre as frequências. Todas as mensagens foram ligadas ao inlet do [line] e de seu outlet ao inlet do [slider]. Como as frequências ficariam entre 100Hz e 600 Hz clicamos com o botão direito do mouse sobre o [slider]. Este procedimento faz abrir uma caixa com algumas opções. Clicamos em ´´properties`` para ter acesso às propriedades do slide. Assim diminuímos seu tamanho e o ajustamos para trabalhar entre 100Hz e 600Hz.

Desejando que a mudança entre as frequências ocorresse de forma aleatória, criamos um objeto chamado [random], que faz com que seu outlet forneça um numero aleatório toda vez que seu inlet recebe um sinal, em nosso caso de um objeto chamado [bang]. Criamos este objeto [bang] e conectamos seu outlet ao inlet do [random]. Como tínhamos seis mensagens com frequências criamos o objeto [random] com o numero 6, fazendo com que ele forneça uma sequencia aleatória de números de 0 a 5. Criamos em seguida um objeto [select 0 1 2 3 4 5], que toda vez que um destes números é recebido em seu inlet ele manda uma mensagem no outlet correspondente. Conectamos o outlet do [random 6] ao inlet do [sel 0 1 2 3 4 5] e cada um dos 6 primeiros inlets as mensagens de frequência e tempo.

Para acionar o [random 6] de forma automática criamos o objeto [metro 1500], que gera um sinal (bang) a cada 1,5 segundos e ligamos seu outlet ao inlet do [random 6]. Para ligar o [metro 1500] criamos um objeto [toggle] que quando clicado alterna entre 1 e 0 (ligado e desligado).

Repetimos este procedimento desde a criação do objeto [noise~], mas ao invés de utilizarmos o objeto [lop~] utilizamos [bp~], um filtro passa-faixa. Um filtro que permite a passagem de algumas faixas de frequência e rejeita outras. Este objeto possui três inlets. O primeiro recebe o sinal, no nosso caso um ruído gerado por [noise~], o segundo inlet recebe a faixa de frequência que deve passar, e o terceiro

recebe a largura desta faixa, conhecida como fator Q. Quanto maior este numero, mais estreita a banda de passagem e quanto menor este fator mais larga a banda de passagem. A partir deste ponto repetimos o procedimento que utilizamos na montagem do filtro anterior criando valores que seriam selecionados aleatoriamente, agora para a faixa de frequência e também para o fator Q. Os objetos utilizados e os métodos de seleção foram os mesmos, porém com valores diferentes. Esta parte do *patch* simularia o vento.

Estudando o objeto [array] criamos um *patch* que fazia tocar algumas notas MIDI. Este *patch* funcionava da seguinte maneira: o objeto [array] armazena números em uma matriz. Estes números são endereçados a ela através de uma caixa de mensagens e podem ser lidos depois com um objeto [tabread]. Chamamos nossa matriz de números de ´´som`` e para armazenar os números nesta criamos mensagens com as notas MIDI desejadas, utilizando caixas de mensagens como a seguinte: [; som 0 72 71 69 67 65 64 62 60]. O numero 0 determina em que ponto da matriz o primeiro valor vai ser introduzido e os demais números são notas MIDI. Para fazer a leitura destas notas criamos o objeto [tabread som]. Este objeto faz a leitura do ponto que é solicitado. Como estamos trabalhando com uma matriz de oito posições os valores estarão endereçados entre 0 e 7. Estes números serão lidos de forma sequencial e para isto criamos um metrônomo [metro 500] para fazer uma leitura a cada meio segundo. Este [metro 500] esta conectado a um contador [cup], que esta conectado a [% 8], que faz com que a contagem se repita entre os números 0 e 7, que são os endereços de nossa matriz. Para transformar estes números em sons criamos os objetos [makenote] e [noteout]. Com o primeiro objeto identificamos a nota MIDI e acrescentamos as informações de ´´velocity`` e duração da nota (em milissegundos) e com o segundo enviamos as informações a uma porta de saída. Precisamos acessar o painel do PD na opção ´´Media`` e neste ´´MIDI settings...``. onde escolheremos o ´´output device`` que iremos utilizar para executar os sons.

Apresentando este *patch* durante a aula nos foi sugerido pelo professor que fizéssemos com que os ritmos também variassem. Assim criamos um segundo grupo de objetos para realizar esta função. Como a base do funcionamento desse grupo de objetos é a mesma dos que fazem a escolha das notas, selecionamos todos os objetos entre [metro 500] e [tabread som] e copiamos. Criamos então um novo [array] e o nomeamos ´´tempo``. Após isto selecionamos o objeto [metro 500] e mudamos para [metro 1000] para que a leitura de nossa matriz de tempo ocorresse a cada segundo. Alteramos também o objeto [tabread som] para [tabread tempo] e as caixas de mensagens que ficaram como o seguinte exemplo: [; tempo 0 6 5 4 5 4 6 7 8]. Estes números indicam a quantidade de notas que serão executadas em um período de tempo, no nosso caso 1 segundo. Criamos um objeto que realiza uma operação

aritmética: [/] . Conectamos ao inlet direito deste objeto uma caixa de mensagens com o numero 1000 e nesta caixa de mensagens um objeto [loadbang] que faz com que a caixa de mensagem seja selecionada assim que o *patch* é iniciado. Ao inlet direito de [/] conectamos o outlet de [tabread tempo]. Conectamos então o outlet de [/] a um objeto [s metro] que vai enviar os valores recebidos ao grupo de objetos do *patch* que controla as notas. Criamos então [r metro] e conectamos ao inlet direito de [metro 500]. Dessa forma as notas não seriam mais executadas uma a cada meio segundo mas no tempo determinado pelo outlet de [/].

Seguindo o mesmo procedimento selecionamos novamente os objetos que já haviam sido duplicados e repetimos a duplicação. Nosso objetivo agora era fazer com que algumas notas fossem acentuadas mais forte que outras. Novamente criamos outro [array] e demos o nome de ´´swing``. Mudamos [metro 500] para [metro 250] e [tabread som] para [tabread swing]. A mesma mudança ocorreu nas caixas de mensagem que ficaram: [; swing 0 9 6 6 8 8 6 6 8]. Como os valores que colocamos nas caixas de mensagem eram muito baixo, criamos um objeto [* 10], que multiplicaria por dez estes valores. Ligamos seu inlet esquerdo ou outlet de [tabread swing] e seu outlet a um objeto [s vel]. Criamos um objeto [r vel] e conectamos ao segundo inlet de [makenote], fazendo com que as notas geradas recebessem uma informação de ´´velocity``.

Faltava fazer com que estas notas geradas de forma aleatória interagissem com um instrumento acústico, e para este fim tivemos a ideia de fazer com que a tonalidade mudasse conforme o comando dado pelo instrumento acústico. Escolhemos três tonalidades que foram: dó maior, que era a tonalidade que já havíamos escrito nas caixas de mensagem, meio tom acima (dó sustenido maior) e um tom abaixo (Si bemol maior). Estas tonalidades mudariam conforme tocássemos notas chave, como as notas dó, dó sustenido e sí bemol em determinada oitava do instrumento. Estas notas teriam de ser fora da oitava que estava sendo gerada pelo *patch* e por isso escolhemos notas mais graves. Assim a mudança de tonalidade ocorreria com as notas MIDI: 48 (dó), 49 (dó sustenido) e 46 (si bemol). Escolhemos por utilizar como instrumento acústico um saxofone barítono. Assim criamos os seguintes objetos: [adc~], que transforma os sinais analógicos em sinais digitais para serem compreendidos pelo PD. Conectamos a este o objeto [fiddle~] para transformar estes sons em números MIDI. Agora precisávamos selecionar os numero MIDI 48,49 e 46 e criamos o objeto [sel 48 49 46]. Em seu primeiro oulet da esquerda conectamos um a caixa de mensagens com o numero 0, no segundo inlet um a caixa de mensagens com o numero 1 e no terceiro com o numero -2. Ligamos os outlets destas três caixas a um objeto [s ton]. Voltamos aos objetos que executam as notas e criamos entre [tabread som] e [makenote] um objeto [+]. No inlet direito deste objeto conectamos [r ton].

Após alguns testes percebemos que não funcionou como o esperado e para entender melhor o problema criamos uma caixa de números conectada ao primeiro outlet de [fiddle~]. Percebemos que as notas recebidas do saxofone não estavam sempre afinadas e uma nota mesmo que pouquíssimo desafinada para baixo não era reconhecida pelo [sel 48 49 46]. Precisávamos que o seletor reconhecesse uma nota mesmo que esta estivesse desafinada para cima ou para baixo. Selecionamos o objeto [sel 48 49 46] e o deletamos. Em seu lugar colocamos [moses 47.5]. Conectamos ao seu inlet direito o outlet do [fiddle~]. Todo numero inferior a 47.5 sairia pelo outlet esquerdo de [moses 47.5] e todo numero superior a 47.5 pelo seu outlet direito. Neste caso só nos interessavam os números maiores que 47.5. Criamos então outro objeto [moses] com o valor 48.5. Conectamos o outlet direito do [moses 47.5] ao inlet esquerdo de [moses 48.5]. Assim todo numero entre 47.5 e 48.5 sairia pelo outlet esquerdo de [moses 48.5] e todo numero superior a 48.5 pelo outlet direito. Conectamos então o outlet esquerdo a caixa de mensagens [0]. Repetimos o procedimento para os valores 48.5 e 49.5 e ligamos a caixa de mensagem [1], e com os valores 45.5e 46.5 a caixa de mensagem [-2]. Dessa maneira conseguimos com que o PD se comunicasse de forma amigável com o instrumento acústico.

Figuras:

Após a implementação da parte sonora passamos a montar o *patch* que controlará a parte visual.

Escolhemos uma figura geométrica para começar: um quadrado, que foi substituído logo em seguida por um cubo. Para visualizarmos as formas geométricas precisamos criar uma janela onde estas formas serão geradas. Criamos um objeto chamado [gemwin]. Para que as figuras sejam transmitidas do *patch* para a janela precisamos de uma mensagem chamada [render]. Esta mensagem pode ser anexa à mensagem que cria a janela: [create, 1].

Criamos então um objeto [gemhead]. Toda figura precisa deste objeto para funcionar. Logo em seguida criamos o objeto [cube], e ligamos o outlet do [gemhead] ao inlet do [cube]. Desfizemos esta ligação e criamos um objeto [translateXYZ], notando que o XYZ devem ser em maiúsculas. Este objeto permite controlar a posição da figura nos eixo X Y e Z. Criamos três caixas de números e conectamos aos inlets 2,3 e 4 do objeto. Pudemos então movimentar o objeto para qualquer parte da janela que fosse de nosso interesse.

Criamos em seguida um objeto [rotateXYZ] e conectamos entre os objetos [translateXYZ] e [cube]. Este objeto controla os movimentos que a figura faz em torno de seus próprios eixos. Criamos também três caixas de números e conectamos aos inlets do objeto. Variamos os números observando os movimentos realizados pelo objeto.

Conectamos ao inlet esquerdo do objeto [cube] duas mensagens: [draw line] e [draw fill] quem fazem com que o cubo tenha somente suas linhas desenhadas, ou seja, completamente preenchido, respectivamente. Após algumas experimentações chegamos a conclusão que estas mensagens não seriam necessárias em nossa implementação e deletamos ambas. Conectamos no inlet direito do cube uma caixa de números que conectada a este inlet permite a mudança de tamanho da figura. Após algumas experimentações deletamos também esta caixa de números.

Decidimos criar uma iluminação para a figura geométrica e para isso criamos um objeto [gemhead] e um objeto [world_light] que simula uma iluminação a uma grande distancia como a iluminação pelo sol. Conectamos uma caixa de números ao terceiro inlet do objeto [world_light] fazendo com que a iluminação gire ao redor do eixo Y. Para melhor visualizar esta movimentação criamos uma mensagem [debug $1] e conectamos ao objeto [world_light]. Em [debug $1] conectamos um objeto [toggle] para permitir ativar e desativar a mensagem [debug]. O inlet direito de [world_light] permite que mudemos a cor da iluminação. Este inlet recebe informação das três cores básicas numa só mensagem. Por isso criamos um objeto [pack f f f] que permite agrupar três informações numéricas. Nos três inlets superiores deste objeto conectamos caixas de números programadas para variar de 0 a 1. O objeto [pack] só envia a informação ao seu outlet quando tem uma mudança numérica no seu primeiro inlet ou quando recebe um impulso. Para isso criamos um objeto [bang]. Conectamos os outlets das caixas de números dois e três ao inlet do [bang] e conectamos seu outlet ao primeiro inlet do objeto [pack f f f], assim toda vez que mudarmos os valores das caixa de numero teremos este resultado no outlet do objeto [pack f f f]. Para receber esta informação criamos uma caixa de mensagem [$1 $2 $3]. Conectamos ao seu inlet o outlet de [pack f f f], e seu outlet ao inlet direito de [world_light].

Era nosso desejo que as figuras interagissem com o som que estava sendo criado. Voltamos ao *patch* recital vento e criamos um objeto [s chuva]. A letra s é uma abreviação para send. Este objeto manda uma informação recebida para um objeto [receive], que pode ser abreviado pela letra r que tenha o mesmo nome que ele, no caso chuva. Conectamos seu inlet a caixa de números que estava conectada ao [slider] vertical.

Voltamos ao *patch* figuras e criamos o objeto [r chuva]. Conectamos seu outlet ao inlet da caixa de números que já estava conectada ao terceiro inlet de [rotateXYZ].

Experimentamos fazer o mesmo procedimento com os cubos e no *patch* recital vento criamos [s vento] e [s vento2] e conectamos as caixas de números dos dois [sliders] horizontais.

Em figuras criamos o objeto [r vento] e conectamos a segunda caixa de números de [rotateXYZ]. Observamos que a rotação era muito rápida, pois os números eram muito altos. Para diminuir os valores criamos um objeto [/ 5]. Este objeto divide o numero que esta chegando a seu inlet da esquerda pelo numero 5. Achamos que ainda estava muito rápido e fomos variando o numero do objeto ate chegar ao numero que foi de nosso agrado, neste caso o numero 10.

Criamos em seguida o objeto [r vento2]. Como já observamos que os números que este objeto receberia eram baixos, criamos um objeto [* 10], que multiplica por 10 o numero recebido e ligamos este objeto a terceira caixa de números de [rotateXYZ].

Após observar por algum tempo o funcionamento do *patch*, resolvemos aumentar a quantidade de quadrado para nove. Assim selecionamos todos os objetos entre [gemhead] e [cube] e clicamos Ctrl d. Isto faz com que todo o esquema selecionado seja duplicado. Após a primeira duplicação percebemos que só havia um cubo na tela. Isto ocorreu porque os dois cubos tinham os mesmos valores em [translateXYZ]. Variamos então os números das caixas de números que estavam conectadas em [translateXYZ] e colocamos os cubos nos locais que eram de nosso interesse.

Repetimos este procedimento mais sete vezes até completar os nove cubos que queríamos.

Resolvemos experimentar outros tamanhos para os cubos e para isso criamos uma caixa de números que ligamos ao inlet direito de todos os objetos [cube], assim podíamos modificar o tamanho de todos ao mesmo tempo. Após alguns experimentos chegamos ao numero 0.7 que foi de nosso agrado.

Quando o computador foi desligado e ligado novamente, tivemos a desagradável surpresa de ter novamente só um cubo na tela. Percebemos que todas as caixas de numero tinham o numero zero e só então lembramos que elas não guardam os valores quando o programa é desligado.

Tivemos que fazer novamente todos os ajustes de posicionamento e tamanho e depois disso colocamos os valores dentro do objeto [translateXYZ] que passou a ser, por exemplo, [translateXYZ 0 0 0] e do objeto [cube], que ficou [cube 0.7]. desconectamos e deletamos todas as caixas de números que estavam ligadas aos objetos [translate] e [cube].

Passamos agora a relatar a montagem do *patch* que funcionara conjuntamente com o *patch* do piano:

Resolvemos fazer um um *patch* que usasse os cubos já existentes, mas que interagisse com a nova parte sonora que iniciaria, girando conforme as notas fossem tocadas. Como estamos trabalhando com 8 notas MIDI e temos 9 cubos decidimos não usar o cubo central junto ao piano MIDI. Os outros cubos responderiam às notas

MIDI começando de baixo, da esquerda para a direita. Assim, se a nota MIDI 60 fosse executada, o primeiro cubo inferior da esquerda giraria ao mesmo tempo, e assim por diante.

Começamos criando um objeto [send], que chamamos de [s note] e conectamos ao outlet de [tabread som].

Criamos depois um objeto [receive] e chamamos de [r note]. Conectamos este objeto a um novo objeto que criamos para selecionar as notas MIDI que estávamos usando: [sel 60 62 64 65 67 69 71 72]. Logo após este objeto criamos um [bang]. Colocamos o *patch* em funcionamento e pudemos confirmar que funcionava conforme o esperado.

Criamos um objeto [vline~] para movimentar o cubo. Como este objeto é específico para áudio, não podemos conecta-lo direto ao inlet do [cube]. Tivemos de criar um objeto [snapshot~], que transforma um sinal de áudio em um sinal de dados sempre que recebe um *bang*. Como precisamos de uma resposta continua criamos um [metro 50], que emite um *bang* a cada 50 milissegundos, suficiente para o que precisamos. Criamos então uma caixa de mensagem e conectamos ao inlet do [vline~]. Fomos testando valores para movimentar o cubo. Quando achamos valores satisfatórios conectamos o outlet do *bang* ao inlet da caixa de mensagem. Assim a caixa de mensagem ficou da seguinte maneira: [350 500, -60 2000 900, 50 500 3000, -20 500 3500, 0 500 4000], fazendo com que o cubo gire no sentido horário quando recebe um bang, depois gire mais lentamente no sentido anti-horário e repete este procedimento até estabilizar parado.

O *patch* inicial estava ficando muito grande e difícil de visualizar. Criamos então um *sub-patch* chamado [pd cubos piano], onde copiamos e colamos os itens anteriores referentes ao movimento dos cubos durante a execução do acompanhamento de piano.

Criamos outro *sub-patch* onde colocamos todos os objetos referentes à criação e posicionamento dos cubos chamado [pd figuras].

Continuamos a trabalhar no sub-*patch* [pd cubos piano]. Colocamos os objetos [r note], [sel 60 62 64 65 67 69 71 72] e os bangs no centro da tela do computador e copiamos os objetos responsáveis por fazer o cubo girar até atingirmos o numero de oito. Posicionamos estes objetos ao redor da tela e conectamos cada um dos oito bangs a uma caixa de mensagens. Criamos 8 objetos send que chamamos de [s ca] (send cubo a), [s cb] (send cubo b), e assim por diante até o [s ch], não criamos um objeto [s ce] pois não iremos movimentar este cubo durante a execução com piano.

Voltamos ao *patch* principal e começamos a trabalhar em um efeito de luz para a primeira parte do recital, fazendo com que a cor da iluminação mudasse quando tivéssemos algum som mais forte sendo captado pelo microfone. Criamos outro objeto [adc~] para captar os sons externos e o conectamos a um objeto [env~] que

recebe um sinal em seu inlet e demonstra esse sinal RMS em seu outlet em dB. Conectamos a outro objeto [moses] onde escolhemos o valor 70, portanto [moses 70] e o outlet direito deste objeto a uma caixa de mensagem. Assim, toda vez que tivéssemos um valor maior que 70, a caixa de mensagem receberia um bang. Esta caixa de mensagem possui a informação [0 10, 1 2000 500]. Conectamos esta caixa de mensagem a um objeto [vline] que como dito anteriormente precisa de um objeto [snapshot~] para funcionar. Conectamos o [snapshot~] às duas caixas de números da esquerda, já criadas anteriormente.

Criamos um sub-*patch* [pd iluminação] onde realocamos todos os objetos relativos a esta execução.

Para mudar as cores durante a execução do piano criamos os objetos: [r ton] e [sel 0 1 -2], criamos um bang para cada um dos três primeiros inlets e conectamos caixas de mensagem a estes, cada caixa contendo valores referentes as cores vermelho, verde e azul. Assim ficaram as caixas de mensagem: [1 1 0.77], [1 0.88 0.77] e [0.91 0.69 0.41]. Estas caixas foram conectadas ao inlet direito de [world_light].

Observamos que o valor 70 do objeto [moses 70] não funcionava corretamente em varias situações, então criamos uma caixa de números que conectamos ao inlet direito deste objeto e um objeto [inlet], para que pudéssemos mudar os valores sem precisar entrar nos sub-*patch*. Criamos também um objeto [outlet], que conectamos ao outlet de [env~] para monitorar os valores recebidos.

No *patch* principal conectamos uma caixa de números ao outlet de [pd iluminação] e outra a seu inlet, à qual conectamos um objeto [Knob].

Após um ensaio, preparando para o dia do recital, notamos a necessidade de deixar mais claro o final da apresentação, pois o piano parava de tocar repentinamente sem dar ideia de finalização. Criamos então um sub-*patch* dentro de [pd pianista] chamado [pd final]. Este sub-*patch* faria tocar um acorde de dó maior quando uma nota especifica fosse tocada no saxofone. Para isso usamos os objetos [moses] da mesma maneira que anteriormente e selecionamos a nota MIDI 36. Quando esta nota fosse executada acionaria um [triger] com três [floats], um para cada nota do acorde. Em caixas de mensagem colocamos as notas desejadas. Na primeira mensagem colocamos também as informações de ´´velocity`` e duração das notas. Como não havíamos usado nenhum acorde até o momento, optamos por não utilizar a terça do acorde para não mudarmos a textura abruptamente.

Conectamos as três caixas de mensagem a um objeto [send chord].

No sub-*patch* [pd pianista] criamos um objeto [receive chord], que conectamos ao inlet de [makenote].

Executamos o *patch* novamente e percebemos que este acorde era acionado algumas vezes, mesmo que a nota MIDI 36 não fosse tocada no saxofone. Para

solucionar este problema criamos um objeto [* 0] e conectamos ao inlet do primeiro [moses] de [pd final]. No inlet direito de [* 0] conectamos duas caixas de mensagem. Uma com o numero zero e outra com o numero 1. Na caixa com o numero zero conectamos [r in]. Esta informação não seria necessária se usássemos o *patch* uma única vez depois de aberto, mas no caso de executarmos o *patch* outras vezes ela se faz necessária. Na mensagem com o número um conectamos [receive final]. Este objeto recebe a mensagem que o tempo de execução do *patch* terminou e libera o reconhecimento da nota MIDI 36. Para que o acorde não fosse tocado varias vezes conectamos o outlet esquerdo do segundo [moses] a mensagem zero, assim quando o acorde fosse executado a primeira vez o reconhecimento da nota 36 seria cancelado. Por motivos de organização refizemos esta ultima conexão utilizando os objetos [send end] e [receive end].

Para escurecer a tela ao final da execução, abrimos o sub-*patch* [pd iluminação] e criamos uma mensagem [0 0 0] com seu inlet conectado ao outlet de outro objeto [receive end]. Conectamos o outlet de [0 0 0] ao inlet direito de [world_light]. Apesar de funcionar corretamente, o escurecimento acontecia de forma imediata e optamos por fazer isto de forma gradual, aproveitando para que o mesmo ocorresse durante as mudanças de tonalidade.

Desconectamos de [world_light] as quatro caixas de mensagem que possuíam números e deixamos conectado apenas [$1 $2 $3]. Conectamos estas quatro caixas a um objeto que separa as informações de cada mensagem: [unpack f f f]. em cada um dos outlets deste objeto conectamos uma caixa de mensagem [$1 500] conectada a [line]. Os outlets dos três [lines] foram ligados ao objeto [pack f f f], para reagrupar as mensagens que foram direcionadas ao inlet direito de [world_light].

Mesmo após o escurecimento da tela, as cores poderiam voltar se o [fiddle~] reconhecesse algum dos números MIDI referentes às cores. Consideramos prudente desativar o [fiddle~] e para isso decidimos desativar o pd dsp. Deletamos o objeto que havíamos criado no inicio da implementação: [pddp/dsp]. Criamos em seu lugar uma caixa de mensagem: [; pd dsp $1] com um objeto [toggle] conectado a seu inlet. Ao inlet de [toggle] conectamos o objeto [receive end].

Resultados

Durante o recital pudemos observar o funcionamento do *patch*: ouvimos um som que simulava chuva, variando sua intensidade para mais forte ou mais fraco, após alguns segundos foi introduzido um som de vento, que também variava de intensidade. Durante este acontecimento acústico a parte gráfica do programa era visualizada na tela do projetor onde nove cubos eram iluminados por ângulos diferentes conforme mudava a intensidade da chuva. Quando começamos a ouvir o

vento estes cubos começaram a girar sincronizados com os eventos sonoros. Todos os cubos apresentavam a cor branca e mudavam para azul quando o saxofone emitia algum som, que podia ser uma nota ou um efeito como *slap* ou harmônico. Após um período o som de vento cessou e os cubos moveram-se lentamente para seu estado inicial.

Após os cubos atingirem seus pontos iniciais o som de chuva cessou e começou automaticamente a segunda parte do *patch.*

Pudemos ouvir notas musicais sendo geradas aleatoriamente pelo PD que formavam melodias através de mudanças de altura, duração e intensidade. Estas notas se encontravam dentro da distancia de uma oitava e mudavam de tonalidade conforme o saxofone também mudasse. Assim tínhamos três tonalidades durante a execução da peça. Sempre que o saxofone tocasse uma nota pré-determinada o PD entendia a mudança de tonalidade e mudava também.

As mudanças de tonalidade eram acompanhadas por mudanças de cor na iluminação dos cubos, que giravam sempre que uma nota fosse tocada pelo PD. Cada cubo representava uma nota da escala, começando pela linha de cubos inferiores, da esquerda para a direita e subindo, sempre da esquerda para a direita. Somente o cubo central não representava nenhuma das oito notas da escala de cada tom. Quando a nota representada por este cubo era tocada, ele girava rapidamente sobre seu próprio eixo para a direita e depois voltava lentamente girando para a esquerda. Fazia uma espécie de balanço até se estabilizar e parar. O foco da iluminação mudava conforme as notas executadas pelo saxofone. O foco no centro iluminava todos os cubos quando a nota tocada era uma das tônicas e ia para as laterais com as outras notas.

Após um tempo determinado as notas geradas pelo PD pararam de ser produzidas e o saxofone tocou sozinho, como numa cadência. Ao termino desta cadencia o PD produziu um acorde e a peça foi encerrada.

Conclusões

O PD oferece uma quantidade infinita de possibilidades de criação artística. Como toda ferramenta necessita de treino para ser compreendido e utilizado todo seu potencial. Este *patch* foi somente um pequeno exemplo do que pode ser desenvolvido.

A interação do analógico com o digital foi uma das partes mais sensíveis, pois algumas vezes ruídos externos eram reconhecidos como notas, gerando interpretações errôneas pelo PD. Acreditamos que isso pode ser minimizado com um conhecimento mais profundo das possibilidades do PD.

Durante a fase final do desenvolvimento do *patch*, quando já possuíamos um pouco mais de familiaridade com o PD, notamos que algumas das partes do processo que havíamos desenvolvido poderiam ter sido executadas de maneira bem mais simples. Optamos por não refazer estas partes, visto que estavam funcionando corretamente.

Concluímos que o PD é uma ferramenta muito útil para o músico, podendo ser utilizado tanto numa performance no palco como em projetos de musicalização para crianças, que será nosso próximo objetivo.

Referências Bibliográficas

Kreidler, Johannes. Programming Electronic Music in Pd
Disponível em: http://www.pd-tutorial.com/english/index.html