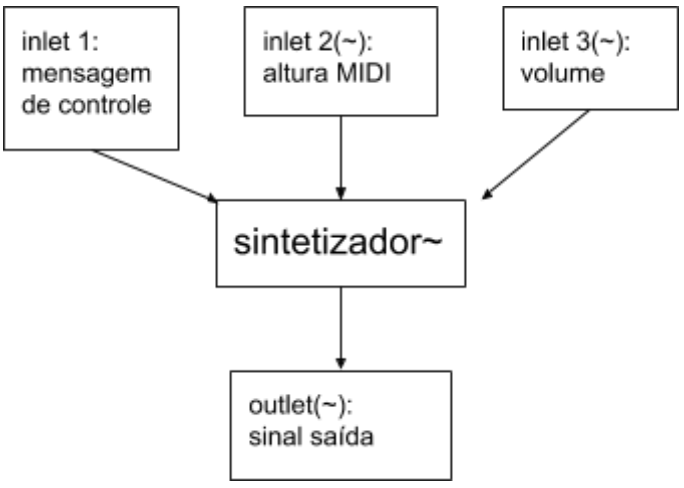
CompMus 2020 — Primeiro Exercício Programa
Sintetizador / Sequenciador
Data de Entrega: 06/11/20 até 23:55

O objetivo deste trabalho é implementar um protótipo de instrumento musical que pode ser usado como sintetizador em tempo real e também processar/sintetizar sequências melódicas. A especificação da interface do instrumento é apresentada na próxima seção, através das mensagens de controle e sinais contínuos aceitos na entrada, bem como da saída esperada. Os detalhes internos de implementação são deixados integralmente ao seu critério, sendo que você pode escolher entre fazer uma implementação usando apenas objetos nativos do Pd, usando objetos de quaisquer bibliotecas disponíveis pelo Deken, usando implementações em Lua através do objeto **[ofelia]**, bem como qualquer combinação dessas possibilidades. Parte importante do trabalho consiste portanto em investigar recursos da linguagem e combiná-los de forma a atender a especificação do instrumento. Considere que essa investigação levará pelo menos tanto tempo quanto a implementação propriamente dita, e por isso é importante começar o trabalho o quanto antes!

Especificação

Seu instrumento deve ser implementado em um único arquivo `sintetizador~.pd`, **autocontido**, ou seja, estruturado de tal maneira que quaisquer módulos em Pd ou códigos Lua utilizados sejam incluídos como subpatches (e **não** como abstrações .pd ou arquivos .lua separados).

Esse instrumento deve possuir três inlets, o primeiro para controle e os demais para sinais, além de um outlet de sinal, de acordo com o diagrama abaixo:



As mensagens aceitas pelo primeiro inlet, indicadas a seguir pela notação `|mensagem<`, são:

- `|liga<`: liga o processamento DSP *dentro do instrumento* (use **[switch~]**).
- `|desliga<`: desliga o processamento DSP dentro do instrumento.
- `|onda senoidal<`, `|onda quadrada<`, `|onda triangular<`, `|onda dente-de-serra<`: seleciona a forma básica de onda do instrumento (veja o exemplo icsm113). Considere que o valor default (inicial) é "onda senoidal".
- `|ataque N<`: configura o tempo de ataque (início ou fade-in) de N milissegundos (valor default = 50) que afeta o “volume interno” do instrumento.
- `|decaimento N<`: configura o tempo de decaimento final (finalização ou fade-out) de N milissegundos (valor default = 50) que afeta o “volume interno” do instrumento.
- `|inicia<`: inicia a produção de som em modo de “síntese livre”, aumentando gradualmente o “volume interno” do instrumento de 0 até 1, durante o tempo configurado para o ataque. No modo de “síntese livre” a frequência e a amplitude do instrumento são controladas dinamicamente pelo 2° e 3° inlets (veja a descrição destes inlets a seguir).
- `|finaliza<`: finaliza a produção de som em “síntese livre”, diminuindo gradualmente o “volume interno” de instrumento até 0, durante o tempo configurado para o decaimento.
- `|notas` N_1 D_1 N_2 D_2 `...` N_k D_k`<`: processa a lista de notas N_i e durações D_i, sintetizando a melodia correspondente de maneira sequencial, iniciando a nota N_{i+1} imediatamente ao terminar a nota N_i. Os valores de notas estão em MIDI (podem ser fracionários) e as durações em milissegundos. O número $k \geq 1$ de notas é arbitrário. Caso o instrumento não esteja produzindo som em modo de “síntese livre” (ou seja, se o “volume interno” for 0) o ataque e o decaimento devem ser aplicados **uma única vez** para a sequência toda (ou seja, apenas a primeira nota é acompanhada de um fade-in, e apenas a última nota é terminada com um fade-out, sendo que os tempos de fade-in e fade-out *contam como parte da duração* dessas notas). Se, por outro lado, a mensagem |notas …< chegar durante a produção de som em “síntese livre” (ou seja, quando o “volume interno” for não-nulo), então a síntese das notas da lista deve ocorrer sem nenhum fade-in ou fade-out, e após o processamento da sequência o instrumento voltará ao estado anterior (“síntese livre”).

O segundo inlet recebe um sinal interpretado como valores MIDI, que determinam a frequência instantânea do oscilador. Ele é usado para controlar livremente a altura musical da síntese após uma mensagem `|inicia<`. Essa entrada é ignorada durante o processamento de uma sequência de notas (quando a frequência instantânea é determinada pelos valores MIDI da mensagem `|notas` N_1 D_1 `...` N_k D_k`<`).

O terceiro inlet recebe um sinal interpretado como amplitude instantânea do instrumento. Esse valor deve ser usado para ajustar a amplitude do sinal devolvido pelo outlet, através de sua multiplicação pelo “volume interno” do instrumento (que depende se ele está ou não em modo de “síntese livre” e é afetado pelos fade-ins e fade-outs).

Algumas dicas

Seu instrumento pode imprimir o que você quiser na janela de interação do Pd. Use esse recurso para depurar o funcionamento do seu instrumento, indicando através dessa saída o tratamento dado às mensagens da entrada (você pode por exemplo imprimir "ajuste do ataque = N milissegundos" quando o instrumento receber a mensagem `|ataque N<`). Você pode inclusive começar a implementação por um tratamento puro e simples das mensagens, certificando-se de que cada formato de mensagem no 1° inlet gera uma frase indicativa, antes mesmo de executar as ações correspondentes.

Tenha atenção à qualidade sonora do resultado: barulhinhos no início e fim de notas são indício de erro no tratamento dos sinais (que ocorrem por exemplo ao aumentar ou diminuir o "volume interno" de forma descontínua). Variações instantâneas de frequência não devem gerar nenhum artefato sonoro, mas variações instantâneas de amplitude no 3° inlet inevitavelmente geram barulho (que corresponde à descontinuidade do sinal), *o que não está errado* (seu instrumento não é responsável pela adequação dos sinais da entrada). Trocas de forma de onda durante a síntese de uma nota ou melodia também gera barulho (o que está ok). Variações contínuas de frequência com uma onda quadrada ou dente-de-serra também podem gerar artefatos devidos ao rebatimento, o que não está errado (ver o exemplo icsm114).

Use o padrão `identificador_$0` para nomear todas as suas tabelas, bem como os nomes associados aos objetos **[value]**, **[send]**, **[receive]**, etc. Esse `$0` é um identificador único para cada instância do patch, e esse padrão de nomeação transformará todos os identificadores em nomes locais, permitindo o uso simultâneo de mais de uma instância do mesmo instrumento, sem risco de confusão das variáveis nem erros do tipo "multiply defined".

Todos estão convidados a postarem no e-disciplinas exemplos de teste/uso do sintetizador: para isso, verifique que seu patch de teste utiliza o objeto **[sintetizador~]** de acordo com a especificação acima, mas certifique-se também de que seu patch de teste não inclua nenhuma parte da implementação propriamente dita do instrumento. Você pode acompanhar seu exemplo de teste com alguma descrição da saída esperada (por exemplo, "escala de Dó maior de três oitavas durante 5 segundos com som senoidal" ou "varredura contínua de onda quadrada com oscilação de amplitude").

Essa é uma lista (não-exaustiva) de alguns objetos que você muito provavelmente irá precisar usar: **[delay]**, **[list]**, **[mtof~]**, **[multiplex~]** (depende de **[zexy]**), **[route]**, **[select]**, **[switch~]**, **[tabosc4~]**, **[trigger]**, **[value]**, **[vline~]**. Abra o help de cada um deles para saber como usá-los.

Finalmente...

- Esse trabalho é individual: conversar com os colegas para tirar dúvidas da linguagem é absolutamente normal, mas compartilhar soluções específicas e códigos não...
- Leia o enunciado mais de uma vez. É comum surgirem dúvidas que já estão respondidas no enunciado, mas que não demos atenção na primeira leitura.
- Use o fórum do e-disciplinas para tirar dúvidas! Se sua dúvida é "posso usar o objeto X da biblioteca Y?", leia o enunciado mais uma vez! :-) Evidentemente, não está permitido compartilhar código do sintetizador, mas podemos postar códigos de teste, inclusive como forma de tirar dúvidas (tipo "qual seria a saída esperada desse teste?").
- Entregue quantas versões parciais no e-disciplinas você quiser, antes do prazo, por precaução. Imprevistos sempre acontecem (difícil saber por que têm esse nome): Internet cair, HD pifar, cachorro mastigar o computador, tudo isso faz parte.
- Divirta-se programando! (esse deve ser o objetivo de todo aprendizado, certo?)