UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LUCAS DA ROCHA MINÁ

DESENVOLVIMENTO INTERATIVO DE PLUGINS PARA PRODUÇÃO MUSICAL COM TESTES EM TEMPO REAL

CAMPINA GRANDE - PB

2024

LUCAS DA ROCHA MINÁ

DESENVOLVIMENTO INTERATIVO DE PLUGINS PARA PRODUÇÃO MUSICAL COM TESTES EM TEMPO REAL

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Dalton Dario Serey Guerrero

CAMPINA GRANDE - PB

2024

LUCAS DA ROCHA MINÁ

DESENVOLVIMENTO INTERATIVO DE PLUGINS PARA PRODUÇÃO MUSICAL COM TESTES EM TEMPO REAL

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Dalton Dario Serey Guerrero

Orientador – UASC/CEEI/UFCG

Leandro Balby Marinho

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em 15 de Maio de 2024.

CAMPINA GRANDE - PB

RESUMO

Plugins, programas auxiliares na produção musical, são muito desejados por artistas nos dias de hoje por expandirem as possibilidades na criação de música. Em seu processo criativo, necessidades específicas aparecem frequentemente, então a ideia de criar seu próprio plugin pode ser muito atrativa. Porém, o processo de programação de plugins pode ser consideravelmente contra intuitivo, pois para sequer poder ouvir os resultados que estes podem trazer à composição, normalmente é necessário escrever e compilar o programa de forma separada, para somente então testar. Este projeto, então, visa demonstrar a viabilidade de uma alternativa existente, muito mais interativa, apropriada para músicos que tenham menos afinidade com tecnologias de programação. A solução proposta envolve utilizar dois programas, plugdata e hvcc, para criar plugins de diversos tipos, sem a necessidade de escrever código e podendo fazer os testes destes em tempo real. O plugdata fornece ao usuário um ambiente para construir diversas funcionalidades úteis para uma composição, sendo possível ouvir os resultados em tempo real. Utilizando então o outro programa, hvcc, é possível converter essa construção do plugdata em código na linguagem C++, compilar esse código e transformá-lo em um plugin nos formatos desejados. Assim, a criação de plugins torna-se muito mais acessível aos artistas musicais. Para demonstrar isso, foram criados três plugins, de funcionalidades diferentes, os quais foram utilizados na composição de uma faixa musical original.

INTERACTIVE DEVELOPMENT OF PLUGINS FOR MUSIC PRODUCTION WITH REAL TIME TESTING

ABSTRACT

Plugins, auxiliary programs used in music production, are sought after by artists in recent times for how they can increase the possibilities in music making. In their creative process, specific necessities arise frequently, so the idea of creating their own plugin can sound very attractive. However, the process of programming plugins can be considerably counter intuitive, since for simply hearing the results that these can bring to the composition, it's necessary to write and compile the program separately, and only then test it. This project proposes a demonstration of the viability of an existing alternative, which is much more interactive, and appropriate for artists without much knowledge of programming. The proposed solution involves the use of two programs, plugdata and hvcc, for creating several kinds of plugins, which don't require writing code and allow for real time testing. Plugdata offers the user an environment for creating many useful functionalities that may be used in a composition, while having it be possible to hear the results in real time. By using the other program, hvcc, it's possible to convert what was created in plugdata into code in the C++ language, compile it and transform it into a plugin in the desired format. This way, the creation of plugins can be much more accessible to musicians. To demonstrate this, three plugins, each with different functionality, were created and used to compose an original musical track.

Lucas da Rocha Miná
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
lucas.mina@ccc.ufcg.edu.br

Dalton Dario Serey Guerrero
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
dalton@computacao.ufcg.edu.br

RESUMO

Plugins, programas auxiliares na produção musical, são muito desejados por artistas nos dias de hoje por expandirem as possibilidades na criação de música. Em seu processo criativo, necessidades específicas aparecem frequentemente, então a ideia de criar seu próprio plugin pode ser muito atrativa. Porém, o processo de programação de plugins pode ser consideravelmente contra intuitivo, pois para sequer poder ouvir os resultados que estes podem trazer à composição, normalmente é necessário escrever e compilar o programa de forma separada, para somente então testar. Este projeto, então, visa demonstrar a viabilidade de uma alternativa existente, muito mais interativa, apropriada para músicos que tenham menos afinidade com tecnologias de programação. A solução proposta envolve utilizar dois programas, plugdata e hvcc, para criar plugins de diversos tipos, sem a necessidade de escrever código e podendo fazer os testes destes em tempo real. O plugdata fornece ao usuário um ambiente para construir diversas funcionalidades úteis para uma composição, sendo possível ouvir os resultados em tempo real. Utilizando então o outro programa, hvcc, é possível converter essa construção do plugdata em código na linguagem C++, compilar esse código e transformá-lo em um plugin nos formatos desejados. Assim, a criação de plugins torna-se muito mais acessível aos artistas musicais. Para demonstrar isso, foram criados três plugins, de funcionalidades diferentes, os quais foram utilizados na composição de uma faixa musical original.

Palavras-chave

Música, Desenvolvimento, Acessibilidade, Teste, Pure Data, hvcc.

1. INTRODUÇÃO

Atualmente, o processo de criação de música envolve, muito frequentemente, o uso de um computador. Programas criados especialmente com o objetivo de fornecer um ambiente de produção musical para artistas são geralmente chamados de DAWs (Digital Audio Workstation, ou Área de Trabalho Digital de Áudio). Dentro de sua DAW de preferência, o artista pode organizar suas composições e gravações, manipular o áudio, utilizar instrumentos digitais, entre outras atividades recorrentes.

É comum, então, que as DAWs não tenham tudo que o artista possa desejar. Afinal, o processo criativo é algo pessoal, e varia muito entre os músicos. Dessa forma, a maioria destes busca plugins, ou seja, software externo e adicional com capacidades específicas para atender a certas necessidades, que pode ser inserido na DAW e expandir as possibilidades na produção musical.

Esses plugins, por sua vez, normalmente possuem diversos parâmetros a serem manipulados pelo artista, a fim de personalizar sua utilização e adaptar o plugin para o uso desejado.

No entanto, as opções ainda podem ser insuficientes. Uma peça específica pode necessitar de efeitos ou sons específicos que não são exatamente o que se encontra em plugins com propósitos mais abrangentes. O artista pode, ainda, buscar uma maneira própria de compor e de se apresentar que se beneficiaria de um maior controle das ferramentas utilizadas, quando as interfaces propostas pelos plugins que encontrou não são exatamente o que é procurado. Por tratar-se de um processo criativo, é esperado que novas soluções sejam criadas ao longo do tempo. Porém, o processo de programação de um novo plugin pode ser algo com o que o artista não tem nenhuma experiência, além de apresentar obstáculos particularmente contra intuitivos.

Plugins precisam ser programas de alto desempenho, com mínima latência. Caso contrário, o artista não irá ouvir sua música no tempo correto, o que pode, com certeza, atrapalhar muito o processo de composição. Dessa forma, é comum que os plugins sejam escritos em linguagens de programação relativamente baixo nível como C++, reconhecidas por seu alto desempenho, o que também torna necessário um processo de compilação. O problema encontrado nesse método é que o processo de criação do plugin fica completamente removido do uso deste, pois é necessário escrever, compilar, e somente então importar o plugin numa DAW e fazer o teste prático. Assim, mesmo com experiência de programação, um artista teria que lidar com um processo frustrante e pouco interativo.

Dessa forma, a utilização de certos programas em particular, plugdata[1] e hvcc[2], pode ser uma alternativa desejável. São programas gratuitos e de código livre, sendo assim muito acessíveis. O plugdata é uma adaptação e nova interface para o programa Pure Data[3], ou simplesmente Pd, desenvolvido pelo professor Miller Puckette[4] do Departamento de Música na Universidade de Califórnia San Diego (UCSD) para análise, síntese e processamento de som. As inovações propostas pelo plugdata incluem, principalmente, a possibilidade de utilizar as capacidades de Pd no formato de um plugin, sendo assim possível inseri-lo numa DAW, além de apresentar uma interface mais amigável. Já o hvcc (Heavy Compiler Collection), ou simplesmente Heavy, é um compilador capaz de transformar algo construído em Pd (nesse caso, no plugdata) e gerar código em C++ para este. Hoje, o plugdata já inclui uma interface para o Heavy, que também permite a automatização do processo de compilação deste código para gerar o plugin nos formatos desejados.

Assim, um artista pode simplesmente inserir o plugdata em sua DAW de preferência, construir interativamente o que for necessário para sua composição até estar satisfeito, podendo testar livremente e em tempo real, e então utilizar o Heavy para gerar o plugin, agora otimizado por ser um programa escrito em C++ e compilado, que estará pronto para ser usado por qualquer um que se interesse.

Com base no exposto, a alternativa proposta pode ser de grande interesse para artistas musicais, com experiência em programação ou não, podendo expandir as possibilidades e incentivar a inovação. Este trabalho tem como objetivo demonstrar a viabilidade dessa alternativa, através de seu uso na construção de três plugins de tipos diferentes: Um sintetizador, um efeito de áudio para gerar ecos, e um sequenciador, bem como uma faixa musical original que os utilize.

2. FUNDAMENTAÇÃO TEÓRICA

Três tipos de plugins foram selecionados para serem desenvolvidos. Seguem as características principais de cada um:

2.1 Sintetizador

Sintetizadores são instrumentos eletrônicos que geram som a partir de ondas de diversos formatos, para gerar diferentes timbres. O som gerado utilizando a onda pode então ser modificado, por exemplo, utilizando um filtro para cortar certas frequências do som. Para manipular a frequência da onda, sintetizadores comumente contam com uma entrada MIDI[5] para gerar tons referentes a notas musicais. As notas MIDI recebidas também podem afetar a dinâmica do som, por meio de sua velocidade, da mesma forma que em um instrumento acústico como o piano, é possível gerar sons diferentes pela velocidade em que se tocam as teclas.

2.2 Efeito de áudio (delay)

Efeitos de áudio podem ter diversos propósitos. De forma geral, som é recebido pelo efeito, modificado, e então devolvido com a modificação aplicada. O efeito selecionado neste trabalho é um que gera ecos, ou seja, resulta em repetições do som recebido, e é comumente chamado de "delay". Normalmente, num efeito de delay, é possível controlar a quantidade (volume) das repetições geradas, e o tempo entre cada repetição.

2.3 Sequenciador

Sequenciadores permitem a manipulação de sequências de eventos em um determinado ritmo. Esses eventos normalmente se referem a notas sendo enviadas a um instrumento, como um sintetizador. Comumente, é possível manipular que notas são tocadas, e em qual ritmo. As notas são normalmente enviadas através do protocolo MIDI.

3. METODOLOGIA

Para atingir os objetivos apresentados, foi seguida a seguinte metodologia:

3.1 Preparação do ambiente

A DAW escolhida para servir de ambiente base de desenvolvimento, onde será carregado o plugdata, foi BespokeSynth[6] que, por funcionar de forma modular, oferece diversos módulos para diferentes propósitos, e plugins carregados também são tratados como módulos.

Foi preparado um computador com sistema operacional Linux, e neste foram instaladas as versões estáveis mais recentes de plugdata (versão 0.8.3) e BespokeSynth (versão 1.2.1). Também foi utilizado um teclado controlador da marca Arturia conectado ao computador, para que fosse possível realizar testes mais interativos pressionando teclas. No BespokeSynth, foi carregado o plugdata como um plugin no formato VST3, e dentro deste foi ativado o "Compiled Mode" ("Modo Compilado") para limitar a seleção de objetos aos compatíveis com o Heavy.

3.2 Familiarização com o software

Com o objetivo de aprofundar o conhecimento sobre o plugdata e o Heavy, foram utilizados os seguintes recursos:

- A série de vídeos explicativos e práticos "Real-time Music and Sound with Pure Data"[7], criada pelo professor Andrew R. Brown[8] e disponibilizada no YouTube.
- O manual[9] do Pure Data, encontrado no website do professor Miller Puckette.
- A documentação e os exemplos oferecidos na página do Heavy, já referenciada anteriormente.

3.3 Análise de modelos

A principal referência para o desenvolvimento dos plugins foram módulos já existentes no BespokeSynth, dentre os quais se destacam o sintetizador "oscillator", o efeito de áudio "delay" e o sequenciador "drumsequencer". Suas funcionalidades foram analisadas a fim de determinar as especificações de design dos plugins a serem desenvolvidos.

3.4 Desenvolvimento e teste dos plugins

Com plugdata dentro de BespokeSynth, suas entradas e saídas MIDI e de áudio puderam ser utilizadas para testar as funcionalidades dos plugins enquanto eram desenvolvidas, conectadas a diversos módulos já inclusos no BespokeSynth. Os parâmetros mais importantes foram selecionados para serem tornados acessíveis interativamente no plugin a ser gerado, e objetos interativos do próprio plugdata foram utilizados para testar também essa interatividade.

Após concluído seu desenvolvimento, no projeto de cada plugin foi utilizada a função "Compilar" no menu do plugdata para utilizar o compilador Heavy e gerar os plugins nos formatos desejados. É necessário aceitar a instalação automática do Heavy na primeira vez que a função é utilizada. Para todos os plugins, foi selecionada a opção "DPF Audio Plugin" e formato VST3 para a exportação. A ativação de entradas e saídas MIDI foi feita de acordo com o tipo do plugin.

Então, com os plugins exportados no formato VST3, estes foram então carregados separadamente no BespokeSynth e testados para confirmar que funcionavam como esperado.

3.5 Composição de uma faixa musical

Para demonstrar a usabilidade dos plugins criados em um cenário real, foi composta uma faixa musical que utiliza cada um deles dentro do BespokeSynth, utilizando também módulos já inclusos neste.

4. RESULTADOS E DISCUSSÕES

O sintetizador gerado conta com quatro formatos de onda disponíveis: Senoidal, triangular, dente de serra e quadrada. Possui um filtro de corte de altas frequências, cuja frequência de corte e ressonância podem ser ajustadas. Também conta com dois geradores de envelope ADSR[10] que podem controlar a frequência de corte do filtro e o volume do sintetizador. Estes parâmetros também podem ser afetados pela velocidade das notas MIDI recebidas.

Já o efeito de delay conta com suporte para som estéreo e controles para o tempo entre repetições, volume do delay e do sinal original, e volume de entrada e saída. Também é possível desligar a função “feedback” para gerar um único eco em vez de múltiplas repetições seguidas.

Finalmente, o sequenciador criado oferece uma sequência de 16 passos que pode tocar notas MIDI entre 0 e 15, ou nenhuma nota. O controle do ritmo pode ser feito internamente, controlando as batidas por minuto (BPM) e subdivisões da medida musical, ou externamente, por meio de sua entrada MIDI: Cada nota recebida, qualquer que seja, avança um passo na sequência. Também é possível modificar a velocidade e duração das notas a serem tocadas.

A faixa original que foi produzida, o arquivo de projeto do BespokeSynth, os arquivos dos projetos criados no plugdata e uma breve documentação de seu desenvolvimento podem ser acessadas num repositório GitHub que foi criado para este fim, no seguinte link: https://github.com/lucminah/plugins-com-plugdata

O desenvolvimento dos plugins decorreu-se majoritariamente como o esperado: graças à possibilidade de realizar testes em tempo real utilizando o plugdata, as funcionalidades de cada

plugin puderam ser testadas livremente antes da fase de compilação, possibilitando assim um desenvolvimento muito intuitivo. Foi possível observar o comportamento dos plugins em interação com outros componentes da DAW escolhida de forma simples e eficiente.

Em certas situações, como descrito na documentação de desenvolvimento, houveram erros inesperados que só puderam ser detectados após compilar e testar o plugin gerado, o que interferiu com a interatividade do processo. Porém, estas foram muito poucas, e se comparado ao ciclo de desenvolvimento mais comum, em que não há a possibilidade de testes em tempo real, utilizar o plugdata com o Heavy ainda se apresenta como a alternativa mais interativa e com menor potencial de retrabalho.

Durante a composição da faixa original produzida, os plugins funcionaram como esperado e exerceram suas funções sem problemas.

5. CONCLUSÃO

Diante dos resultados apresentados, observa-se que o uso de plugdata e Heavy para o desenvolvimento de plugins demonstrou-se viável. Em relação ao método convencional de programação de plugins, a alternativa apresenta grandes vantagens em questão de interatividade, o que torna o processo de desenvolvimento não somente mais acessível como também mais agradável.

Os arquivos e a documentação publicados no GitHub podem servir de inspiração a outros artistas, e espera-se que, com mais reconhecimento, essa alternativa seja mais utilizada por quem tiver interesse em criar seus próprios plugins de forma intuitiva.

6. REFERÊNCIAS

[1] plugdata - A visual programming environment for audio experimentation, prototyping and education. Disponível em: https://plugdata.org/.

[2] Heavy Compiler Collection (hvcc). Disponível em: https://wasted-audio.github.io/hvcc/.

[3] Pure Data. Disponível em: https://puredata.info/

[4] Miller Puckette, UCSD. Disponível em: https://msp.ucsd.edu/

[5] CENTER FOR ELECTRONIC AND COMPUTER MUSIC, Indiana University Bloomington. “Introduction to MIDI and Computer Music”. Disponível em: https://cecm.indiana.edu/361/midi.html

[6] BespokeSynth - A modular DAW for Mac, Windows, and Linux. Disponível em: https://www.bespokesynth.com/

[7] QCGInteractiveMusic. “Real-time Music and Sound with Pure Data”. Disponível em: https://www.youtube.com/playlist?list=PLuxj2jXSuTvvqYcDLJ-poN-JxvqX0wq-m

[8] Andrew R. Brown, Griffith University - Bristane, Australia. Disponível em: https://www.explodingart.com/arb/

[9] Pd Manual. Disponível em: https://msp.ucsd.edu/Pd_documentation/index.htm

[10] JENKINS, Jake. “A Simple Guide to Modulation: Envelope Generators”, 2022. Disponível em: https://www.sweetwater.com/insync/a-simple-guide-to-modulation-envelope-generators/