UNIVERSIDADE FEDERAL DE MINAS GERAIS

CURSO SUPERIOR DE SISTEMAS DE INFORMAÇÃO

Design e Implementação de Sintetizadores VST: Explorando Algoritmos de Síntese Sonora e Processamento de Áudio

Belo Horizonte, Minas Gerais

2024

Giovanni Gontijo Braga

Design e Implementação de Sintetizadores VST: Explorando Algoritmos de Síntese Sonora e Processamento de Áudio

Proposta de pesquisa tecnológica apresentada como requisito para a conclusão da Graduação em Sistemas de Informação da Universidade Federal de Minas Gerais

Orientador(a): Flavio Vinicius Diniz de Figueiredo (UFMG)

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Belo Horizonte, Minas Gerais
2024

Sumário

1. Introdução

O processo de criação de som por meio de um sintetizador é denominado síntese sonora, uma prática que envolve a criação e manipulação de formas de onda com o intuito de atingir resultados sonoros específicos. Inicialmente, as formas de onda são geradas a partir de dispositivos conhecidos como osciladores, sendo posteriormente submetidas a uma variedade de processamentos destinados a conceber timbres distintos.

Este trabalho de conclusão de curso tem como propósito apresentar uma proposta para o desenvolvimento de um sintetizador em formato VST (Virtual Studio Technology), orientado para a compatibilidade com softwares de Estação de Áudio Digital (DAW). O objetivo central deste sintetizador consiste em receber comandos MIDI e gerar sinais sonoros em conformidade com parâmetros como extensão, velocidade e notas, todos eles fornecidos pelo usuário.

Neste contexto, será realizada uma pesquisa bibliográfica para identificar e analisar as técnicas de programação de áudio e processamento digital de sinais (DSP) que serão úteis para a criação e desenvolvimento de um sintetizador. Adicionalmente, serão avaliadas as ferramentas mais adequadas para a implementação deste tipo de aplicação, incluindo considerações sobre a Interface Gráfica do Usuário (GUI).

Como desfecho deste trabalho de conclusão de curso, é antecipado que se alcance a produção de um aplicativo VST de sintetização sonora, desenvolvido com base em frameworks especializados. Este aplicativo estará capacitado a gerar uma ampla gama de timbres, com foco particular na criação de instrumentos musicais virtuais que possam ser incorporados em processos de produção musical.

2. Referencial Teórico

A revisão bibliográfica desempenha um papel importante no projeto de desenvolvimento de sintetizadores digitais. Nesta etapa, exploramos a fundo as práticas de engenharia de software e os algoritmos para o contexto de síntese sonora. A aplicação dessas técnicas é estratégica, visando proporcionar aos músicos e produtores uma experiência de sound design personalizada. Ao compreender os padrões de criação e preferências individuais dos usuários, os algoritmos desempenham um papel vital na geração de timbres únicos, aumentando a expressividade e a satisfação do usuário com o sintetizador digital.

O desenvolvimento de um sintetizador VST para produção musical envolve uma série de conceitos e técnicas fundamentais no campo da síntese sonora, processamento digital de sinais (DSP) e programação de áudio. Este referencial teórico visa fornecer uma base sólida para a compreensão destes princípios ao projeto proposto.

A síntese sonora é uma parte essencial para desenvolvimento de sintetizadores digitais, que visa a criação de sons por meio da manipulação das propriedades das ondas sonoras. Essa arte envolve a geração e modificação de formas de onda, frequentemente por meio de osciladores, a fim de produzir timbres específicos. Existem várias técnicas de síntese sonora, incluindo a subtrativa, aditiva e a modulação de frequência (FM). A escolha da técnica depende das necessidades artísticas e criativas do desenvolvedor do sintetizador e do usuário final.

No âmbito da produção musical digital, os programas integrados de áudio, conhecidos como VST (Virtual Studio Technology), desempenham o papel de aprimorar o ambiente de trabalho do produtor. Estes instrumentos digitais atuam como extensões integráveis em softwares de produção musical, tais como as Digital Audio Workstations (DAWs). Sua importância reside na ampla gama de possibilidades sonoras que oferecem, permitindo aos músicos e produtores explorar texturas, timbres, efeitos e experimentações.

A interação entre o músico e o sintetizador digital é uma parte do processo criativo na música digital, e essa interação é facilitada pelas entradas MIDI (Musical Instrument Digital Interface). O protocolo MIDI permite ao músico transmitir informações detalhadas sobre notas musicais, incluindo sua altura, intensidade e duração, ao sintetizador digital (Hewitt, 2015). Essas informações são transmitidas possibilitando ao músico expressar suas intenções musicais de maneira precisa. O sintetizador, por sua vez, interpreta esses dados e os converte em sons correspondentes, oferecendo um meio versátil para a criação musical. A integração das entradas MIDI é essencial para a criação de um sintetizador digital, permitindo um nível aprimorado de expressão e controle na geração de sons.

Na esfera do desenvolvimento de sintetizadores digitais, o processamento digital de sinais (DSP) surge como um elemento indispensável para garantir a produção de sons em termos de qualidade. Este campo, caracterizado por sua natureza multidisciplinar, encontra-se na encruzilhada entre engenharia elétrica, matemática aplicada e ciência da computação, desempenhando um papel importante na arte da síntese sonora. As técnicas de DSP são aplicadas às formas de onda geradas, abrindo um vasto horizonte de possibilidades para a criação de timbres complexos e expressivos.

Por fim, o design da Interface Gráfica do Usuário (GUI) é importante para a interação eficaz entre o músico e o sintetizador digital. Uma GUI bem projetada incorpora controles intuitivos que permitem ao músico ajustar parâmetros sonoros em tempo real, enriquecendo a experiência musical.

Em síntese, o desenvolvimento de sintetizadores digitais envolve a síntese sonora por meio de osciladores, a integração de entradas MIDI para controle e expressão musical, a aplicação de programação de áudio e DSP para criar timbres complexos e o design eficaz de interfaces gráficas para uma interação intuitiva. Esses elementos convergem para proporcionar aos sound designers as ferramentas necessárias para explorar diferentes sons e timbres.

3. Desenvolvimento do software

Simplicidade e Personalização

Na concepção do desenvolvimento de sintetizadores digitais, garantir a simplicidade é uma das principais estratégias adotadas, sendo de grande valor em um cenário repleto de opções e complexidades. Um sintetizador digital simples é mais acessível, rápido de manipular e geralmente mais agradável. Ele elimina a complexidade do processo, proporcionando uma experiência de usuário suave e satisfatória.

Para alcançar essa simplicidade, é crucial focar na funcionalidade principal do sintetizador, priorizando recursos e elementos que se alinhem diretamente com os objetivos sonoros. Ao evitar a inclusão de funcionalidades desnecessárias, é possível oferecer aos músicos uma interface intuitiva e uma experiência eficiente e agradável. Além disso, a simplicidade do sintetizador digital pode resultar em uma curva de aprendizado mais suave para os usuários, facilitando a adoção do produto.

O sintetizador digital pode oferecer ao músico uma paleta sonora personalizada, alinhada com seus interesses musicais específicos. Essa abordagem não apenas aumenta a expressividade dos sound designers, mas também a satisfação

geral com o produto. Dessa forma, a personalização contribui significativamente para criar uma experiência de usuário mais envolvente e inspiradora no contexto da produção musical com sintetizadores digitais.

Ferramentas e Tecnologias

O projeto do plugin de sintetizador digital utilizará as seguintes ferramentas e tecnologias:

- IDE: Visual Studio
- Ambiente de execução: Ableton Live
- Sistema de controle de versão: Git
- Framework de desenvolvimento: JUCE
- Linguagem de programação: C++
- Interface de usuário: PluginGuiMagic

A seleção das ferramentas e tecnologias foi pautada na busca pela eficiência, produtividade e compatibilidade, com o objetivo principal de assegurar a qualidade e o desempenho do plugin. O IDE escolhido foi o Visual Studio na versão Community, devido a sua total integração com o desenvolvimento em JUCE. Para a execução do aplicativo VST (Virtual Studio Technology), será adotado o ambiente do Ableton Live, uma vez que a utilização de uma DAW (Digital Audio Workstation) é importante para analisarmos o comportamento do software em um ambiente real de produção. O controle de versão ficará a cargo do Git, proporcionando a habilidade do rastreamento das alterações realizadas. O framework JUCE desempenhará um papel fundamental no desenvolvimento do plugin, aproveitando suas diversas ferramentas especializadas em programação de áudio e criação de plugins VST. Além disso, o JUCE oferece um conjunto de ferramentas para a condução de testes de unidade, os quais serão implementados para garantir a robustez do software desenvolvido. Devido a natureza da aplicação, não se dá como necessário o desenvolvimento de banco de dados e servidores.

Requisitos

Levantamento de Requisitos

No processo de levantamento de requisitos para o desenvolvimento de um sintetizador digital, houve inspiração em plugins renomados de áudio e sintetizadores digitais(serum, vital, phase plant, pigments 2, synplant). A análise das funcionalidades, recursos e experiências oferecidos por esses plugins permitiu

identificar as melhores práticas e compreender as expectativas dos usuários nesse contexto específico.

Inspirado em referências estabelecidas no mundo da síntese sonora, foi estabelecida uma base sólida para a criação do sintetizador digital. A combinação dessa abordagem com as ferramentas selecionadas, como o JUCE, Visual Studio, Ableton Live e Git, promete um desenvolvimento eficiente e de alto desempenho para o projeto de sintetizador digital.

Análise de Requisitos

Abaixo estão os requisitos essenciais organizados em categorias de funcionais e não funcionais.

Requisitos Funcionais:

- Recepção de entradas MIDI: Os usuários devem poder, a partir de entradas MIDI, tocar melodias e acordes musicais.
- Integração com DAWs: Os usuários devem poder integrar o sintetizador com qualquer digital áudio workstation compatível com plugins VST (Virtual Studio Technology).
- Seleção de onda nos osciladores: Os usuários devem poder escolher em uma seleção de wavetables pré disponibilizadas a que se encaixa melhor com seus objetivos sonoros.
- Controle de envelope ADSR: Os usuários devem poder controlar a dinâmica (attack, decay, sustain, release) dos timbres criados a partir do sintetizador desenvolvido.
- Filtragem da saída de áudio: Os usuários devem poder filtrar, de forma personalizada para seus objetivos (highpass, lowpass, bandpass), os sons gerados pelos osciladores.
- Síntese FM: Os usuários devem poder realizar modulações de síntese FM a partir das ondas geradas pelos osciladores, a fim de aumentar a gama de possibilidades e timbres gerados pelo sintetizador
- Modulação do filtro: Os usuários devem poder modular os parâmetros do filtro a partir de um envelope ADSR, para expandir a paleta de timbres possíveis do sintetizador.
- Visualização e análise de espectro: deve ser possível ao usuário visualizar e analisar o espectro do áudio gerado, tornando possível uma interpretação visual das frequências presentes no som.
- Visualização no osciloscópio: deve ser possível ao usuário visualizar a forma de onda do áudio em um osciloscópio, permitindo uma análise visual completa da forma de onda gerada.
- Opções de áudio: o usuário deve poder selecionar várias opções relacionadas ao áudio, como diferentes dispositivos de saída e taxa de amostragem, de

modo a escolher a melhor qualidade de áudio para suas necessidades específicas.
- Teclado MIDI virtual: Os usuários devem poder utilizar um teclado MIDI virtual integrado ao sintetizador para tocar notas e acordes diretamente na interface do software, sem a necessidade de um teclado MIDI físico.

Requisitos não funcionais:

- Eficiência: Garantir que o aplicativo responda prontamente e tenha um desempenho rápido, assegurando o carregamento ágil e uma experiência de usuário suave.
- Facilidade de Uso: Projetar uma interface intuitiva para simplificar a navegação no aplicativo, tornando-o acessível aos usuários.
- Estabilidade: Construir um aplicativo robusto, reduzindo ao mínimo as falhas e erros, proporcionando uma experiência confiável e consistente aos usuários.
- Usabilidade: O sistema deve ser fácil de usar e entender.
- Compatibilidade: O sistema deve ser compatível com todas Digital Audio Workstations
- Manutenção: O sistema deve ser fácil de atualizar e manter.

Arquitetura

Ao abordar a arquitetura de projetos utilizando o JUCE, alguns aspectos essenciais merecem destaque:

- Componentização e Herança: A arquitetura do JUCE é fortemente baseada no conceito de componentes, que são unidades modulares de interface do usuário ou funcionalidades específicas. A herança é frequentemente empregada para criar componentes especializados, proporcionando uma estrutura flexível e reutilizável.
- Event-Driven Programming: O JUCE segue um paradigma de programação orientado a eventos. A interação do usuário e outros eventos são tratados por meio de callbacks, proporcionando uma abordagem eficiente para lidar com a entrada e saída em aplicações de áudio.
- Gestão de Áudio: O JUCE oferece classes e estruturas específicas para facilitar a manipulação e processamento de áudio. Sua arquitetura permite a implementação de sintetizadores, processadores de áudio e outros componentes relacionados de maneira eficiente.
- Cross-Platform Development: Uma característica fundamental do JUCE é sua capacidade de facilitar o desenvolvimento cross-platform. Isso significa que o

mesmo código pode ser utilizado para criar aplicações que funcionam em diferentes sistemas operacionais, proporcionando uma ampla compatibilidade.

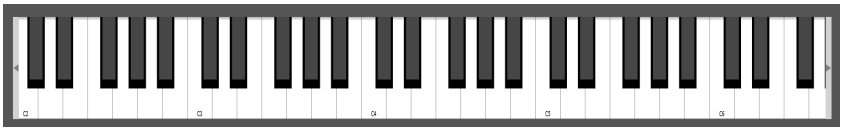
- Interface Gráfica (GUI): O JUCE inclui recursos robustos para a criação de interfaces gráficas de usuário. Seu sistema de desenho e manipulação de componentes gráficos facilita a criação de interfaces intuitivas e responsivas para sintetizadores digitais.
- Tratamento de Áudio em Tempo Real: A arquitetura do JUCE é otimizada para aplicações de áudio em tempo real, como sintetizadores. Ele fornece mecanismos eficientes para processamento de áudio em tempo real, garantindo baixa latência e desempenho confiável.
- Extensibilidade: O JUCE é altamente extensível, permitindo a adição de módulos e bibliotecas de terceiros. Isso facilita a incorporação de funcionalidades adicionais e a integração com outras ferramentas ou frameworks.
- Padrões de Projeto: A arquitetura do JUCE incorpora muitos padrões de projeto de software, como o padrão Observer para lidar com eventos e o padrão Composite para estruturas de componentes hierárquicos.

Recursos e funcionalidades

Ao considerarmos os requisitos funcionais específicos para o desenvolvimento do sintetizador digital, identificamos uma série de funcionalidades essenciais destinadas a atender às necessidades dos usuários e aos objetivos do aplicativo. Essas funcionalidades foram cuidadosamente delineadas para garantir uma experiência de usuário rica e versátil, explorando ao máximo as capacidades sonoras do sintetizador. Abaixo estão as funcionalidades principais detalhadas:

Recepção de Entradas MIDI:

- Os usuários terão a capacidade de expressar sua criatividade, tocando melodias e acordes musicais através de entradas MIDI, proporcionando uma interação musical mais intuitiva. A interface de usuário apresenta um componente de teclado para recepção de entrada MIDI

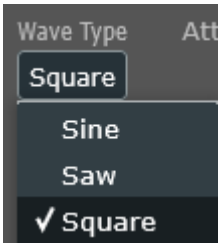


Integração com DAWs:

- O sintetizador será projetado para se integrar de maneira harmoniosa com qualquer estação de áudio digital (DAW) compatível com plugins VST, permitindo uma fluidez no fluxo de trabalho musical.

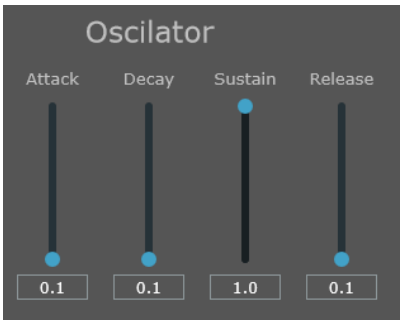
Seleção de Onda nos Osciladores:

- Uma variedade de wavetables pré-disponibilizadas oferecerá aos usuários a liberdade de escolher a textura sonora desejada, adaptando-se aos mais diversos estilos musicais.



Controle de Envelope ADSR:

- A dinâmica dos timbres será moldada de forma personalizada pelos usuários, que terão controle total sobre os parâmetros de ataque, decay, sustain e release para alcançar sonoridades distintas.



Filtragem da Saída de Áudio:

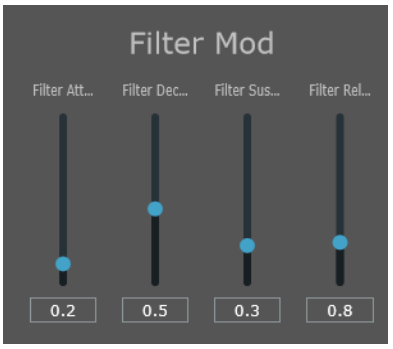
- Uma ampla gama de opções de filtragem, incluindo highpass, lowpass e bandpass, permitirá aos usuários esculpir o espectro sonoro conforme suas preferências individuais.

Síntese FM:

- A introdução da síntese de frequência (FM) proporcionará aos usuários a capacidade de explorar uma rica paleta de possibilidades sonoras, ampliando significativamente o potencial do sintetizador.

Modulação ADSR do filtro:

- A capacidade de modular o parâmetro de cutoff do filtro utilizando envelopes ADSR abrirá novas perspectivas para a criação de timbres dinâmicos e expressivos.



Essas funcionalidades representam uma visão abrangente e detalhada das características-chave do sintetizador digital desenvolvido.

Gerenciamento de Entradas MIDI

A arquitetura para gerenciar entradas MIDI em um aplicativo JUCE envolve a criação de uma classe específica para lidar com eventos MIDI. Esta classe deve herdar de juce::MidiInputCallback, uma interface que define o método handleIncomingMidiMessage. Este método é invocado sempre que uma mensagem MIDI é recebida, permitindo a implementação de lógica específica para o processamento dessas mensagens.

Inicialização e Abertura da Porta MIDI

A inicialização das portas MIDI é um passo crítico na configuração do aplicativo. O JUCE fornece métodos para listar todos os dispositivos MIDI disponíveis, permitindo que o desenvolvedor selecione e abra a porta desejada. A abertura de uma porta MIDI envolve associar o dispositivo a um callback, garantindo que todas as mensagens recebidas sejam encaminhadas para a classe apropriada para processamento.

Funcionamento do AudioProcessorValueTreeState

O AudioProcessorValueTreeState serve como uma ponte entre a interface de usuário e o núcleo de processamento de áudio, permitindo que os parâmetros sejam facilmente modificados e sincronizados. Ele utiliza um objeto ValueTree para armazenar os estados dos parâmetros, proporcionando uma forma eficiente de salvar e carregar presets e de sincronizar o estado dos parâmetros entre diferentes componentes do aplicativo.

O AudioProcessorValueTreeState facilita a integração desses parâmetros no fluxo de trabalho do sintetizador, permitindo a sincronização automática entre os controles da interface do usuário e o processamento de áudio. Isso significa que qualquer alteração feita nos controles da interface se reflete imediatamente no processamento de áudio.

Processamento de Mensagens MIDI

O processamento de mensagens MIDI é realizado dentro do método handleIncomingMidiMessage. Este método deve ser eficiente para garantir uma baixa latência e uma alta responsividade do aplicativo. A interpretação correta das mensagens MIDI, tais como Note On, Note Off e Control Change, é essencial para a funcionalidade precisa e previsível do aplicativo.

Preparação e processamento de áudio

A função SynthVoice::prepareToPlay é utilizada para configurar a voz do sintetizador antes do início da reprodução. Ela recebe como parâmetros a taxa de amostragem (sampleRate), o tamanho dos blocos de amostras (samplesPerBlock) e o número de canais de saída (outputChannels). Dentro dessa função, é criado um objeto juce::dsp::ProcessSpec que armazena essas informações e as repassa para vários componentes do sintetizador, como osciladores, filtros e geradores de envelopes. A função prepara cada componente com as especificações apropriadas e define a configuração inicial do ganho. No final, a variável isPrepared é definida como verdadeira, indicando que a voz está pronta para processar o áudio.

Por outro lado, a função SynthVoice::renderNextBlock é responsável por gerar e processar o áudio para o próximo bloco de amostras. Ela recebe como parâmetros um buffer de áudio de saída (outputBuffer), a posição inicial da amostra (startSample) e o número de amostras (numSamples). Primeiramente, a função verifica se a voz está preparada e ativa; caso contrário, ela retorna imediatamente. Se estiver ativa, ela dimensiona um buffer temporário (synthBuffer) para corresponder ao número de canais e amostras do buffer de saída. O gerador de envelopes é aplicado ao buffer de

saída, e o buffer temporário é preenchido com as amostras geradas pelo oscilador. Em seguida, o gerador de envelopes e o filtro são aplicados ao buffer temporário. O ganho é processado e, finalmente, o conteúdo do buffer temporário é adicionado ao buffer de saída. Se o envelope não estiver ativo, a nota atual é limpa.

A função TapSynthAudioProcessor::processBlock é responsável pelo processamento do áudio e das mensagens MIDI em blocos. Primeiro, ela desabilita números denormais para otimizar a performance e limpa qualquer canal de saída extra. Em seguida, a função atualiza os parâmetros de cada voz do sintetizador com base nos valores armazenados no AudioProcessorValueTreeState (apvts), ajustando o tipo de onda do oscilador, os parâmetros de modulação de frequência, os envelopes de amplificação e filtro, e as configurações do filtro. Finalmente, a função synth.renderNextBlock é chamada para gerar o áudio processado, combinando as contribuições de todas as vozes ativas do sintetizador, resultando no buffer de áudio final que será reproduzido. Este processo garante que o sintetizador responda adequadamente às mudanças de parâmetros em tempo real, proporcionando um som dinâmico e ajustável.

Ondas implementadas

Onda Senoidal (Sine Wave)

Descrição: A onda senoidal é uma onda contínua que descreve uma oscilação repetitiva suave. É a forma de onda fundamental em muitas aplicações de som e sinal, sendo a base para a análise de Fourier e presente em várias funções de ondas em física e engenharia.

Fórmula: initialise([](float x) { return std::sin(x); });

Explicação da Fórmula: Nesta fórmula, a função std::sin(x) é utilizada para gerar a onda senoidal. A função sin da biblioteca padrão de C++ recebe um valor x e retorna o seno desse valor. O x geralmente representa o ângulo em radianos. À medida que x varia, a função sin produz valores que oscilam suavemente entre -1 e 1, criando a forma característica da onda senoidal.

Onda Dente de Serra (Sawtooth Wave)

Descrição: A onda dente de serra é uma onda que apresenta uma ascensão linear seguida de uma queda abrupta. Esta forma de onda é chamada assim devido à sua semelhança com os dentes de uma serra. É utilizada em síntese de áudio e em modulações de sinais.

Fórmula: initialise([](float x) { return x / juce::MathConstants<float>::pi; });

Explicação da Fórmula: Nesta fórmula, x é dividido pela constante π (pi) da biblioteca juce::MathConstants<float>. A constante π é aproximadamente 3.14159. Dividindo x por π, a função produz um valor que varia linearmente à medida que x aumenta. Este valor cresce continuamente, criando a rampa ascendente característica da onda dente de serra. A queda abrupta, ou reset, deve ser gerida pelo contexto em que a função é usada.

Onda Quadrada (Square Wave)

Descrição: A onda quadrada é uma forma de onda não senoidal que alterna entre dois níveis, geralmente -1 e 1, com transições abruptas entre esses níveis. É utilizada em eletrônica e em síntese de áudio para criar sons ásperos e clarinetes.

Fórmula: initialise([](float x) { return x < 0.0f ? -1.0f : 1.0f; });

Explicação da Fórmula: Nesta fórmula, a expressão x < 0.0f ? -1.0f : 1.0f utiliza um operador ternário para determinar o valor de retorno. Se x for menor que zero, a função retorna -1; caso contrário, retorna 1. Isso cria uma onda que alterna abruptamente entre -1 e 1, produzindo a forma característica da onda quadrada.

Onda Triangular (Triangle Wave)

Descrição: A onda triangular é uma forma de onda não senoidal que tem um formato de dente de serra simétrico, ou seja, ela sobe e desce linearmente, formando um padrão triangular. É usada em síntese de áudio para produzir sons com um timbre mais suave em comparação à onda quadrada.

Fórmula: initialise([](float x) { return 2.0f * std::abs(2.0f * (x / (2.0f * juce::MathConstants<float>::pi) - std::floor(x / (2.0f * juce::MathConstants<float>::pi) + 0.5f))) - 1.0f; });

Explicação da Fórmula: Esta fórmula gera uma onda triangular normalizando x com 2π, usando a função std::abs para criar a subida e descida linear, e ajustando com std::floor para repetir o ciclo. A expressão cria um padrão de subida e descida que oscila entre -1 e 1.

Interface de usuário

Na criação da interface do usuário para o sintetizador VST, utilizou-se o PluginGuiMagic, uma ferramenta que permite a construção de GUIs sem a necessidade de codificação. Este módulo, desenvolvido com um modelo DOM, fornece uma informação hierárquica e uma folha de estilo CSS para definir as regras de aparência da interface. O PluginGuiMagic inclui um editor de arrastar e soltar para adicionar elementos de GUI e conectar parâmetros do AudioProcessor, além de um editor no estilo FireBug para investigar propriedades individuais e como foram calculadas. O módulo pode ser adicionado ao projeto através do Projucer ou CMake, facilitando a configuração e a gestão da interface de usuário. A utilização desta ferramenta não só acelerou o processo de desenvolvimento, mas também garantiu uma interface intuitiva e responsiva, aprimorando a experiência ao utilizar o sintetizador.

Na criação da interface do usuário, além do uso do PluginGuiMagic, foram integrados diversos componentes visuais essenciais para uma experiência de sound design completa. Entre os componentes utilizados, destacam-se o espectrograma e o osciloscópio, que permitem aos usuários visualizar as características e a evolução dos sinais sonoros em tempo real. Além disso, a interface inclui um teclado MIDI virtual, proporcionando um meio interativo para tocar e testar os sons gerados pelo sintetizador diretamente na interface. Também foram implementadas análises de pico (peak) e RMS (Root Mean Square), oferecendo informações detalhadas sobre os níveis de volume e a dinâmica dos sinais sonoros. Esses componentes, combinados, proporcionam uma interface rica e informativa, essencial para músicos e produtores que buscam controle e precisão na criação e manipulação de timbres.

Tipos de Plugins de Áudio

Formatos de Plugins de DAW

Estes plugins são utilizados principalmente dentro de Digital Audio Workstations (DAWs):

1. VST3 (Virtual Studio Technology 3)
2. VST2 (Virtual Studio Technology 2)
3. AAX (Avid Audio eXtension)
4. AU (Audio Units)
5. AUv3 (Audio Units v3)

Aplicações Independentes

Plugins ou aplicações que funcionam de forma independente, sem a necessidade de uma DAW:

1. **Standalone**

Plugins para Jogos e Aplicações Interativas

Plugins utilizados dentro de motores de jogos e ambientes interativos:

1. **Unity Plugin**

VST3 (Virtual Studio Technology 3)

Descrição: VST3 é a terceira geração da tecnologia de plugins VST desenvolvida pela Steinberg. É um formato de plugin de áudio amplamente utilizado para adicionar efeitos e instrumentos virtuais em DAWs (Digital Audio Workstations).

Compilação: VST3 plugins são compilados usando frameworks como JUCE, que suporta várias plataformas, incluindo Windows, macOS e Linux. O processo envolve o uso de compiladores como MSVC (Microsoft Visual C++) para Windows e Xcode para macOS.

Compatibilidade:

- **DAWs compatíveis:** Cubase, Nuendo, Ableton Live, FL Studio, Studio One, Reaper, entre outros.
- **Sistemas operacionais:** Windows, macOS, Linux.

Diferenças:

- Suporta recursos avançados como múltiplos pinos de entrada/saída, MIDI Polyphonic Expression (MPE), e processamento de áudio mais eficiente.
- A separação de processamento de áudio e interface gráfica permite uma melhor performance e flexibilidade.

VST2 (Virtual Studio Technology 2)

Descrição: VST2 é a segunda geração da tecnologia VST. Embora ainda amplamente utilizada, está sendo gradualmente substituída pelo VST3.

Compilação: Semelhante ao VST3, utilizando frameworks como JUCE e compiladores apropriados para cada plataforma.

Compatibilidade:

- **DAWs compatíveis:** Cubase, Ableton Live, FL Studio, Pro Tools (com wrapper), Logic Pro (com wrapper), entre outros.
- **Sistemas operacionais:** Windows, macOS, Linux.

Diferenças:

- Menos flexível em termos de roteamento de sinal comparado ao VST3.
- Não suporta alguns recursos avançados que o VST3 oferece, como MPE.

Standalone

Descrição: Standalone refere-se a aplicações de áudio que funcionam de forma independente, sem necessidade de uma DAW. Esses programas geralmente oferecem uma interface de usuário completa para controlar e processar áudio.

Compilação: Compilado como uma aplicação independente usando frameworks como JUCE ou Qt, que fornecem suporte para interfaces gráficas e processamento de áudio.

Compatibilidade:

- **Sistemas operacionais:** Windows.
- **Programas:** Funciona como uma aplicação independente, não depende de DAWs.

Diferenças:

- Não requer uma DAW para funcionar.
- Útil para apresentações ao vivo ou para usuários que preferem trabalhar fora de um ambiente de DAW.

AAX (Avid Audio eXtension)

Descrição: AAX é o formato de plugin de áudio desenvolvido pela Avid para ser usado em suas estações de trabalho de áudio, especialmente o Pro Tools.

Compilação: Compilado usando o SDK AAX da Avid, juntamente com ferramentas como MSVC para Windows e Xcode para macOS.

Compatibilidade:

- **DAWs compatíveis:** Pro Tools.
- **Sistemas operacionais:** Windows, macOS.

Diferenças:

- Integrado de forma otimizada com o Pro Tools.
- Suporta processamento de áudio em tempo real com latência mínima, essencial para produção profissional de áudio.

Unity Plugin

Descrição: Plugins de áudio para Unity são utilizados dentro do motor de jogos Unity para adicionar processamento de áudio e efeitos em jogos e outras aplicações interativas.

Compilação: Compilado usando o SDK de áudio do Unity, e pode utilizar frameworks como JUCE para simplificar o desenvolvimento. Geralmente, usa C# para integração com Unity e C++ para processamento de áudio de alta performance.

Compatibilidade:

- **Motores de jogos:** Unity.
- **Sistemas operacionais:** Windows, macOS, Linux (dependendo da plataforma alvo do jogo).

Diferenças:

- Específico para uso em ambientes de jogos e aplicações interativas.
- Integra-se diretamente com o ambiente de desenvolvimento do Unity, permitindo acesso e controle precisos do áudio dentro do jogo.

Conclusão

Cada tipo de plugin de áudio serve a um propósito específico e é compatível com diferentes ambientes e programas. Para este trabalho o programa foi compilado em todos os formatos abaixo:

- **VST3:** Moderno, flexível, usado em várias DAWs.
- **VST2:** Amplamente utilizado, mas gradualmente substituído pelo VST3.
- **Standalone:** Funciona de forma independente, útil para apresentações ao vivo.
- **AAX:** Otimizado para Pro Tools, utilizado em produção profissional.
- **Unity Plugin:** Integrado em jogos, utilizado no Unity.

Instalação e Execução

Instalar um sintetizador de áudio pode variar dependendo do formato. Aqui estão as instruções para cada um dos formatos:

VST3 (Virtual Studio Technology 3)

1. Download: Baixe o arquivo VST3 do sintetizador desejado.
2. Instalação: Extraia o conteúdo baixado.
3. Caminho de Instalação: Copie o arquivo .vst3 para o diretório padrão de plugins VST3 no seu sistema:
    - Windows: C:\Program Files\Common Files\VST3
    - Mac: /Library/Audio/Plug-Ins/VST3
4. DAW: Abra sua DAW (Digital Audio Workstation) e escaneie os novos plugins VST3. A maioria das DAWs detecta automaticamente os novos plugins instalados.

VST2 (Virtual Studio Technology 2)

1. Download: Baixe o arquivo VST2 do sintetizador desejado.
2. Instalação: Extraia o conteúdo baixado.
3. Caminho de Instalação: Copie o arquivo .dll (Windows) ou .vst (Mac) para o diretório padrão de plugins VST2 no seu sistema:
    - Windows: C:\Program Files\VSTPlugins ou C:\Program Files (x86)\VSTPlugins
    - Mac: /Library/Audio/Plug-Ins/VST
4. DAW: Abra sua DAW e escaneie os novos plugins VST2. Pode ser necessário configurar a DAW para procurar o diretório onde você colocou o plugin.

Standalone

1. Download: Baixe o arquivo executável standalone do sintetizador desejado.
2. Instalação: Execute o arquivo .exe (Windows) ou .dmg (Mac).
3. Simplicidade: Esta é a forma mais fácil de instalar, pois não requer nenhuma outra aplicação. Apenas siga as instruções do instalador. Uma vez instalado,

você pode executar o sintetizador diretamente como um programa independente.

AAX (Avid Audio eXtension)

1. Download: Baixe o arquivo AAX do sintetizador desejado.
2. Instalação: Extraia o conteúdo baixado.
3. Caminho de Instalação: Copie o arquivo .aaxplugin para o diretório padrão de plugins AAX no seu sistema:
    - Windows: C:\Program Files\Common Files\Avid\Audio\Plug-Ins
    - Mac: /Library/Application Support/Avid/Audio/Plug-Ins
4. Pro Tools: Abra o Pro Tools e o plugin AAX deve estar disponível para uso.

Unity Plugin

1. Download: Baixe o arquivo do plugin para Unity.
2. Instalação: Extraia o conteúdo baixado.
3. Importar no Unity:
    - Abra o Unity e o seu projeto.
    - Vá para Assets -> Import Package -> Custom Package.
    - Selecione o arquivo baixado e clique em Import.
4. Configuração: Configure o plugin dentro do Unity conforme necessário, usando os componentes e scripts fornecidos.

Nota Adicional

Para quem busca a forma mais simples de instalação, o formato Standalone é o mais indicado, pois requer apenas a execução do arquivo de instalação sem a necessidade de configuração adicional dentro de uma DAW ou outro software.

link do executável para instalação:
https://github.com/giogbraga/TCC-Synth/tree/main/Standalone%20Plugin

Plano de Implementação

Etapa 1: Construção do protótipo e estruturação de arquitetura

- Desenvolvimento da estrutura base para o sintetizador sonoro utilizando C++ e JUCE.
- Implementação dos módulos de osciladores e envelope ADSR para o protótipo.
- Desenvolvimento da lógica de processamento para receber entradas MIDI e roteamento de áudio para os canais de saída.
- Configurar a compilação em VST3 e testar compatibilidade com Ableton Live e outros digital audio workstation.
- Configurar a compilação no formato standalone para rodar a aplicação sem a necessidade de uma DAW.
- Configuração da arquitetura do plugin, dividindo-o em camadas e módulos.

Etapa 2: Construção do Aplicativo

- Desenvolvimento do módulo de UI para criar a interface do usuário.
- Desenvolvimento do módulo Data para gerenciar a árvore de valores e estados do aplicativo e as ações do sistema.
- Desenvolvimento dos módulos funcionais como filtros, sintese FM, entre outros.
- Desenvolvimento de componentes visuais como teclado MIDI, Analisador de espectro e osciloscópio.
- Criação do instalador da aplicação.

Etapa 3: Testes e Validação

- Realização de testes manuais.
- Validação do correto funcionamento do roteamento de áudio e recepção de entradas MIDI.
- Validação do aplicativo em diferentes Digital Audio Workstation (Ableton Live, Reaper, Fruit Loops, Cubase FE) para garantir usabilidade e compatibilidade.
- Validação do aplicativo em diferentes dispositivos para garantir usabilidade e compatibilidade.
- Validação do aplicativo no modo standalone

4. Considerações Finais

Com a conclusão deste trabalho de desenvolvimento de um sintetizador VST, atingimos todos os objetivos propostos, desde a recepção de entradas MIDI até a integração com diversas DAWs. O sintetizador desenvolvido oferece uma ampla gama de funcionalidades, permitindo aos usuários explorar diferentes técnicas de síntese sonora e criar timbres personalizados.

O uso do framework JUCE foi fundamental para a construção eficiente e modular do projeto, suas funcionalidades e a extensibilidade de módulos auxilia muito no processo de desenvolvimento de áudio. A estrutura modular adotada, que separa componentes de interface e dados, facilita a manutenção e futuras expansões do sintetizador.

O Foley's GUI Magic foi essencial para criar uma interface gráfica eficiente e intuitiva para o projeto. Com suas ferramentas de design visual e suporte para componentes avançados como osciloscópios e analisadores de espectro, ele simplificou o desenvolvimento e a personalização da interface do sintetizador. A abordagem permitiu uma rápida interação e ajustes, garantindo um workflow mais ágil e flexível.

A realização dos testes de compatibilidade e estabilidade em diferentes ambientes assegurou a qualidade do sintetizador final. Com uma interface amigável e uma resposta rápida, o sintetizador não só cumpre com os requisitos técnicos estabelecidos, mas também proporciona uma experiência interessante de síntese sonora.

5. Referências Bibliográficas

link para repositorio git: https://github.com/giogbraga/TCC-Synth

https://juce.com/learn/documentation/

https://foleysfinest.com/developer/pluginguimagic/

https://foleysfinest.com/foleys_gui_magic/index.html

https://thewolfsound.com/sound-synthesis/wavetable-synth-plugin-in-juce/

Bjarne Stroustrup (2013), The C++ programming language

Steven W. Smith (1997), The Scientist and Engineer's Guide to Digital Signal Processing

Will C. Pirkle (2019), Designing Audio Effect Plugins in C++