Universidade Federal do Rio de Janeiro

Relatório do Projeto Final Cálculo Numérico - 2017.2

Interpolação linear em processamento de sinais digitais

Henrique V. de Toledo[1], João Antonio Recio da Paixão[2]

[1]DRE: 116076549

[2]Departamento de Ciência da Computação– UFRJ
Rio de Janeiro – RJ – Brazil

henriquevt98@gmail.com.br, jpaixao@dcc.ufrj.br

Resumo. *Esse documento é o relatório do estudo, técnicas e resultados observados no desenvolvimento de um aplicativo de áudio que emprega um dos métodos numéricos abrangidos na disciplina de Cálculo Numérico de 2017.2.*

Sumário

1. Introdução

Este projeto surgiu como uma expansão da ideia inicial de remover ruídos de arquivos de som. Durante as pesquisas iniciais de como coletar, armazenar e modificar sons no geral, foi consideravelmente aprofundado o conhecimento no ramo de processamento de sinais digitais (Digital Signal Processing - DSP), e uma série de problemas interessantes para análise numérica foram descobertos.

Um desses foi o surgimento de artefatos no ajuste de parâmetros em instrumentos digitais. Instrumentos digitais, disponibilizados como aplicativos stand-alone ou como arquivos binários em DLL chamados de VST ou em outros formatos como .au para Apple, são programas que podem ser rodados em DAWs (digital audio workstation) e usados em diversos workflows de produção musical.

Há inúmeras qualidades e belezas emaranhadas no mundo da programação de áudio, e vários métodos numéricos lecionados no curso de 2017.2, dentre outros mais complexos, aparecem no ramo para solucionarem as mais variadas necessidades. Sejam essas renderização de ondas de frequência em tempo real com precisão, efeitos diversos, técnicas de re-amostragem e renderização de áudio, há de se contar com a possibilidade de empregar um método numérico.

Como mencionado algumas vezes, o método abordado neste relatório e neste projeto é um que, embora simples, é comprovadamente funcional - a interpolação linear.

Terminologia

Em DSP, é imprescindível ter conhecimento de alguns termos para o entendimento pleno dos algoritmos que trazem tantas melodias aos nossos ouvidos. São eles, os principais mencionados aqui:

Período[1] - é o intervalo em que uma função periódica se repete. Para funções trigonométricas como a seno do Sinner, ela repete após 2π radianos. Iremos computar esse período inúmeras vezes durante o algoritmo.

Amplitude[2] - Embora haja várias definições para amplitude, ela é a diferença dos valores extremos de uma onda, ou o quão alto ou baixo ela chega. Para o Sinner, a amplitude irá definir justamente o volume das ondas que ouviremos, entre -1 e 1.

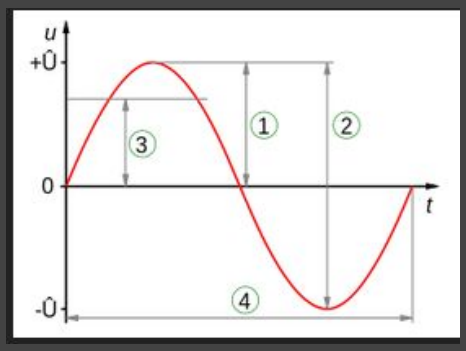

Figura 1 - Wikipedia

Vemos aqui alguns pontos de interesse:
1 representa a amplitude máxima, 2 a amplitude de topo pra topo, 3 é o valor eficaz de amplitude (que não nos concerne muito) e 4 é o período.

Sample (Amostra)[3] - Sons no meio ambiente ou ao serem sintetizados, quando chegam no computador (ou por meio de conversores analog - digital ou na própria síntese) são ondas contínuas de frequência que devem ser discretizadas, separadas em intervalos discretos para que possamos manipulá-los.
Uma sample ou amostra é um valor ou um conjunto de valores num ponto do tempo e espaço, e no nosso caso, samples serão cada pedaço audível das ondas de áudio que ouviremos.
O (Sin)ner, que sintetiza som à partir de ondas seno, por exemplo, terá todas as suas samples base entre -1 e 1, pois formaremos a onda com samples e função seno comporta valores entre -1 e 1. Mas uma onda seno com amplitude 1 é extremamente alta e raramente usada em produção musical.

Sample Rate (taxa de amostragem)[4] - A sample rate é um valor em hertz que define quantas samples serão carregadas por segundo. Ela aparece diversas vezes na implementação do Sinner para saber, por exemplo, o espaço entre duas samples (fazendo 1/sampleRate), ou saber qualquer forma de incremento progressivo (como interpolação!). O Siner trabalha com a taxa padrão CD-ROM de 44100 hertz, um padrão na indústria de música por um tempo considerável.

Buffer Size - Quando falarmos em buffers para o Sinner, estaremos falando de quantas samples podem ser carregadas em uma determinada função que processa o áudio. Um buffer size maior garante maior fidelidade de áudio e precisão, mas demanda

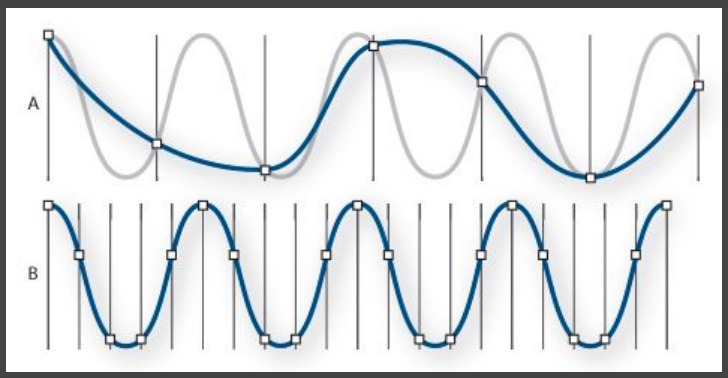

Figura 2 - Audacity

Nessa figura pode-se perceber duas sample rates distintas, A e B. Ambas possuem linhas verticais que demarcam intervalos na onda. Cada linha vertical demarca uma sample, e o total de linhas verticais daria a sample rate. O buffer size, em bits, dita quantas samples podemos processar de vez, ou seja, quantos blocos pegamos para realizar o processamento.

Frequência - Por fim, temos a frequência, que corresponde à qualidade musical do tom. Ajustando a frequência podemos chegar em todas as notas de todos os instrumentos musicais possíveis. Existem guias de quais frequências correspondem a cada cada nota no piano, por exemplo - inclusive, foram consultados no desenvolvimento do Sinner.

Isso conclui o básico de DSP para que seja possível compreender, plenamente, a implementação do Sinner.

2. Metodologia

Objetivo

O propósito se dá em um aplicativo stand-alone e multiplataforma, com controles para a frequência e o volume, em decibéis, que permite ao usuário modular uma onda seno sendo transmitida em dois canais (Stereo). Essa onda pode ser modulada de três maneiras:

2. Ingênua - representada no código como sampleFail.
Essa se baseia na simples fórmula $amplitude\ .\ sen\ (2\pi.frequência)$ **, e os valores são coletados diretos dos sliders. Isso cria uma descontinuidade na função seno que causa artefatos audíveis e indesejáveis em um instrumento digital contemporâneo.**

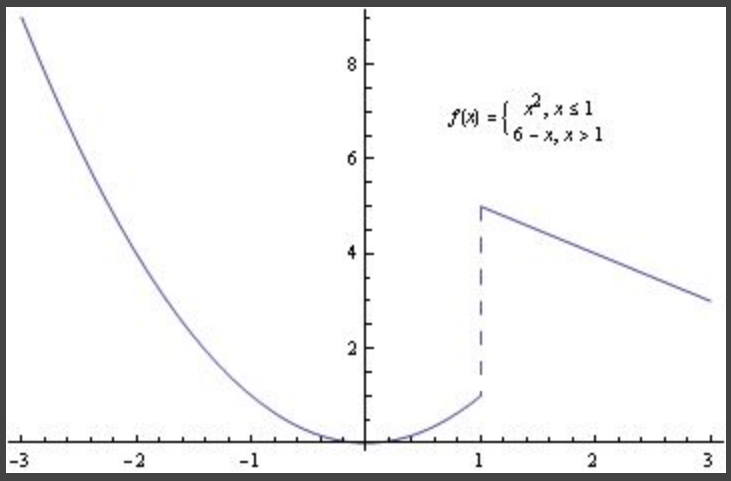

Figura 3 - Wikipedia
Uma descontinuidade qualquer (não especificamente da função seno) é ilustrada acima. Esse é o efeito que o ajuste de sliders como o da frequência em intervalos discretos, não contínuos causa. Saltos na função que causam artefatos.

2. Interpolada - No código, é o que a flag interpolating ativa.
Essa modulação é feita com o auxílio de uma função externa updateAngle() que atualiza o ângulo fora do processamento principal, e na função de processamento, computamos um incremento pequeno que permite preencher os pontosde descontinuidade na função com pontos novos (frequências) interpolados. O frequencyDelta é a variável de incremento que interpola novos pontos e remove os artefatos.

3. Effect - flag effect no código. Durante o desenvolvimento, foi percebido que, sem o auxílio da função updateAngle(), o som produzia um efeito interessante musicalmente. Um som relativamente metálico, sem artefatos como o da implementação ingênua e amplificado. Considerando que historicamente, foi por meio de acidentes que uma variedade de efeitos extremamente populares como distorção ou compressão fizeram seu marco na indústria de música, resolveu-se deixar o effect como um easter egg, uma função extra que pode ser usada no futuro em outras iterações do Sinner.

Criaram-se botões que permitem mostrar o efeito da interpolação, todos descritos na implementação. (2 - Interpolate, 3 - Mystery Effect).

Além disso, algumas teclas ajustam a frequência para frequências de uma oitava de dó maior de um piano qualquer, permitindo ao usuário controlar e tocar algumas músicas breves canções, conforme a imagem abaixo:

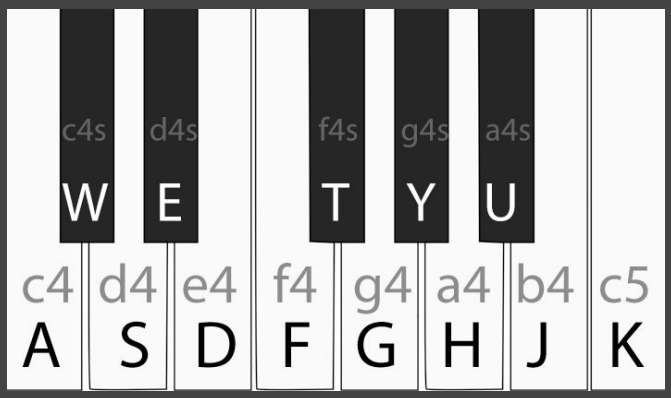

Figura 4 - Autoria própria
(c4, c4s ... etc são as notas da tecla, a letra embaixo é a tecla para apertar e ouví-la)

Implementação

Para tal fim, foi utilizada a biblioteca semi open-source JUCE, escrita em C++. Por conseguinte, o Sinner é escrito em C++, fazendo frequente uso da orientação em objeto para modularizar os problemas enfrentados ao programar uma interface funcional. Um dos obstáculos foi aprender do zero a linguagem nova, cheia de suas peculiaridades.

Há uma diversidade de padrões programáticos em processamento de sinais digitais. Nenhum deles é trivial. É comum utilizar-se, por exemplo, de funções de síntese de áudio que rodam 100 ou mais vezes por segundo dependendo da sample rate, fazendo da otimização uma estratégia indispensável.

Para criar um método interpolador, não basta criar uma função isolada que realiza a interpolação entre uma sample e a próxima. Precisa-se ou gerir uma classe que processe o intervalo entre as samples durante as iterações principais do buffer, ou emaranhar o código com condicionais meticulosamente colocados fora dos loops principais das samples. O segundo caso é o implementado na interpolação ao longo do código do Sinner.

3. Resultados

Dentro da pasta Resources do repositório, podem ser encontradas amostras do Sinner em funcionamento, devidamente nomeadas.

Foi observado um comportamento, como descrito anteriormente, de baixa fidelidade e alta interferência / artefatos na opção sem interpolação. É um cenário observado em plug-ins VST de baixa qualidade que são facilmente descartados no mercado.

A interpolação, ao apertar o botão, mostra uma transição suave e contínua, que é o comportamento esperado por padrão em instrumentos virtuais.

Quanto ao efeito, observou-se uma amplificação metálica e constante do som.

4. Discussão e conclusões

Em suma, há de se convir: os métodos numéricos de Cálculo Numérico são imprescindíveis em aplicações no mundo todo, e música não é exceção. A interpolação se mostra claramente beneficial para a qualidade e fidelidade de qualquer instrumento virtual, sendo praticamente uma técnica indispensável em alguns casos. A própria biblioteca JUCE implementa interpoladores de lagrange e lineares para o desenvolvedor (não utilizadas no projeto)

Deve-se discutir também, sobre a existência de outros métodos, fora do escopo que foi visto neste relatório. No decorrer da pesquisa sobre o tema do projeto, descobriu-se em diversos fórums outras maneiras de realizar o mesmo efeito, até mesmo outros interpoladores.

Por exemplo, como curiosidade, a interpolação hermite()[4], de maior precisão e maior complexidade para implementar, usa 4 variáveis de controle (e não só o vizinho mais próximo como feito no Sinner) para interpolar pontos entre dois pontos conhecidos. (descrita nas referências).

Além disso, também é possível realizar o procedimento com outras técnicas até, porém também não contempladas por cálculo numérico - como um filtro low-pass descrito nas referências.[5]

Ao fim do projeto, a jornada de aprendizado para conseguir uma familiaridade com a biblioteca JUCE e C++ foi um ganho considerável acadêmico, mostrando um caminho escalável para outras aplicações comerciais em processamento de áudio digital.

5. Referências

Interpolação [4]

Xoxox - PDF com interpolação hermite dentre outras
http://www.xoxos.net/sem/dsp2public.pdf

KVRaudio - Sugestões de parameter smoothing
https://www.kvraudio.com/forum/viewtopic.php?p=6548432

Leitura sobre splines
https://www.mathworks.com/help/curvefit/smoothing-splines.html

JUCE forum - Algoritmos de smoothing para paratemers
https://forum.juce.com/t/parameter-smoothing-methods-algorithms/14773

JUCE forum - Mais sobre smoothing
https://forum.juce.com/t/how-to-do-continuous-parameter-automation-with-juce/24842

JUCE forum - Eficácia de smoothing dentre mais algoritmos
https://forum.juce.com/t/efficient-parameter-smoothing/16077

JUCE forum - Low Pass como alternativa a interpolação
https://forum.juce.com/t/dsp-question-implementing-simple-low-pass-in-processblock/7727

Wikipedia - Artigo geral sobre smoothing
https://en.wikipedia.org/wiki/Smoothing

Wikipedia - Linear Interpolation
https://en.wikipedia.org/wiki/Linear_interpolation

Wikipedia - Interpolation
https://en.wikipedia.org/wiki/Interpolation

Foi referenciada a pasta examples do JUCE.

Documentação JUCE

https://juce.com/doc/classes
https://juce.com/learn/coding-standards
https://forum.juce.com/t/how-to-use-getnextaudioblock-method/15580

Para teclas do sintetizador

Sengpiel Audio - Frequências de Teclas
http://www.sengpielaudio.com/calculator-notenames.htm
http://www.sengpielaudio.com/KeyboardAndFrequencies.gif

DSP (processamento)[1],[2],[3]

Processing.org - Sound
https://processing.org:8443/tutorials/sound/

Audacity - Sample Rate
http://wiki.audacityteam.org/wiki/Sample_Rates

Wikipedia - Feedback de Áudio
https://en.wikipedia.org/wiki/Audio_feedback

Wikipedia - Filtros de Áudio
https://en.wikipedia.org/wiki/Audio_filter

Wikipedia - Amplitude de ondas
https://en.wikipedia.org/wiki/Amplitude

Wikipedia - Frequência
https://en.wikipedia.org/wiki/Audio_frequency

Wikipedia - Profundidade de bit em áudio
https://en.wikipedia.org/wiki/Audio_bit_depth

Wikipedia - Taxa de amostragem
https://en.wikipedia.org/wiki/Sampling_(signal_processing)#Sampling_rate