EDSON MELO FERREIRA DE MESQUITA

JONATHAN CHENG

PAULO RICARDO ROCHA VIANNA

"VIOLIN VILLAIN"
ANÁLISE DE ÁUDIO DE VIOLINO EM TEMPO REAL PARA JOGO MUSICAL

São Paulo

2010

EDSON MELO FERREIRA DE MESQUITA
JONATHAN CHENG
PAULO RICARDO ROCHA VIANNA

"VIOLIN VILLAIN" ANÁLISE DE ÁUDIO DE VIOLINO EM TEMPO REAL PARA JOGO MUSICAL

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para Projeto de Formatura.

Área de Concentração:
Engenharia Elétrica com Ênfase em Sistemas Eletrônicos
Orientador: Magno Teófilo Madeira da Silva
Co-orientador: Ricardo Nakamura

São Paulo
2010

Dedicamos este trabalho a nossas famílias, q
apoiaram no crescimento e na formação de c

EDSON MELO FERREIRA DE MESQUITA

JONATHAN CHENG

PAULO RICARDO ROCHA VIANNA

"VIOLIN VILLAIN" ANÁLISE DE ÁUDIO DE VIOLINO EM TEMPO REAL PARA JOGO MUSICAL

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para Projeto de Formatura.

São Paulo

2010

Agradecimentos

Paulo:

Ao meu orientador Magno, pela dedicação, incentivo e compreensão necessários para ajudar o projeto.

À Sérgio Carvalho, meu professor de violino, por me ensinar muito mais do que técnicas do instrumento.

À minha amiga artista plástica, Silvia Regina Aquilas Rodrigues, que desenvolveu a imagem do adesivo indicador da posição das notas.

À minha namorada, Débora Bentivegna Cesta, por me distrair do projeto diversas vezes e me lembrar do que é mais importante.

Jonathan e Edson:

Ao nosso orientador Ricardo, pela dedicação e apoio dado ao projeto.

Aos nossos colegas, que nos ajudaram muito nos vários anos de convivência.

À namorada do Edson, Ana Carla, por tirá-lo de São Paulo e deixá-lo feliz em momentos difíceis.

A Deus, por tornar todas as situações por que passamos possíveis de serem transpostas.

"A imaginação é mais importante que o conhecimento."

Albert Einstein

Resumo

"Violin Villain" é um jogo musical de computador que utiliza o violino como principal interface com o jogador através da captação e análise do som tocado em tempo real. O jogo funciona mostrando uma partitura simplificada na tela do computador, ao mesmo tempo em que capta o som tocado pelo violino e verifica se a frequência tocada é a esperada de acordo com a partitura, indicando imediatamente ao usuário se ele acertou ou errou e também qual a correção necessária para melhorar a afinação da nota. A partitura simplificada criada e utilizada no projeto apresenta os símbolos de cada nota coloridos com uma cor específica para cada semitom (Dó, Dó♯, Ré, Ré♯, Mi, Fá, Fá♯, Sol, Sol♯, Lá, Lá♯, Si), repetindo-se as cores apenas para a mesma nota oitavas acima ou abaixo. Outra simplificação é representar todos compassos com o mesmo comprimento e acrescentar em cada nota um rastro de comprimento equivalente à sua duração, facilitando a leitura do ritmo. No caso de violinos acústicos, a captação do som é feita através de um microfone de computador pessoal (PC) próximo ao violino. Para violinos elétricos conecta-se a saída de linha do violino à entrada de microfone do computador através de um cabo. Para a estimação da frequência fundamental do sinal, foi utilizado o algoritmo Fast Fourier Transform (FFT) que calcula a Transformada Discreta de Fourier (TFD) gerando o espectro do sinal, combinado com uma detecção de pico na região do espectro que é próxima a da frequência fundamental da nota esperada. Os sinais foram analisados utilizando-se a ferramenta computacional Matlab, mas o jogo foi implementado inteiramente na linguagem computacional Java, utilizando bibliotecas de uso livre e um repositório de subversão (SVN) para programação em equipe. A versão beta de Violin Villain estará disponível para download no site *www.violinvillain.com*.

Palavras-chave: análise de sinais musicais, obtenção da frequência fundamental, jogos de música, método de aprendizado de violino, violin hero.

Abstract

"Violin Villain" is a musical computer game which uses the violin as its main player interface through the capture and analyses of the sound played in real time. The game works showing a simplified music sheet on the computer screen, while at the same time capturing the sound played by the violin and checking if the frequency played is the same as it would be expected from the music sheet, imediately showing if the user scored or missed and which change is necessary to improve the note tunning. The simplified music sheet created and used in this project shows each note simbols colored with a especific color for each half tone (A, A♯, B, C, C♯, D, D♯, E, F, F♯, G, G♯), repeating only for the same note octaves above or below. Another simplification is to display every beat mark equidistantly and add a trail to every note with length equivalent to the note duration, therefore making rhythm reading easier. Using acoustic violins the sound capture is made through a personal computer microfone placed close to the instrument, while with electric violins the capture is made directly through a cable. The fundamental frequency estimation of the signal was achieved using the Fast Fourier Transform (FFT), witch applies the Discret Fourier Transform (DFT) to generate the signal spectrum, combined with a peak detection on the region of the spectrum where the fundamental frequency is expected to be. Signal Analyses tests were made using the computational tool Matlab, but the game was completely implemented using the computational language Java with open source libraries and a subversion repository (SVN) for team programming. A beta version of Violin Villain will be available for download at *www.violinvillain.com*.

Keywords: music signal analysis, fundamental frequency estimation, music games, violin learning method, violin hero.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

INTRODUÇÃO

Neste capítulo descrevemos o que nos levou a desenvolver Violin Villain como nosso projeto de formatura, quais os nossos objetivos e como este documento está estruturado. É importante deixar claro que que houve uma separação de enfoque do trabalho entre os integrantes do grupo, uma vez que estes pertencem a diferentes departamentos da Engenharia Elétrica. Nesta monografia se destacarão os aspectos referentes à análise do sinal do violino realizados pelo Paulo do Departamento de Engenharia de Sistemas Eletrônicos (PSI), enquanto que na monografia dos alunos Jonathan e Edson do Departamento de Engenharia da Computação (PCS) se destacará o desenvolvimento da programação do jogo.

1.1 Objetivo

O objetivo deste projeto é a elaboração de um software de jogo que utiliza o violino como interface principal, captando e processando o sinal sonoro do violino para interagir com a lógica do jogo.

Sendo assim, um objetivo do projeto está em descobrir quase imediatamente qual a nota tocada pelo violino. Para isso é preciso captar o sinal de áudio do violino através da amostragem da placa de som e descobrir qual a frequência fundamental deste sinal, já que cada nota corresponde a uma série harmônica definida pela sua frequência fundamental. Este medidor de frequências é implementado junto ao algoritmo do jogo, e permitirá a comparação entre a nota encontrada pelo medidor de frequências e a nota da partitura.

O outro objetivo do projeto é criar um jogo de computador que utiliza o violino e outros tipos de controle como interface, a fim de se ter um protótipo que funcione mesmo sem o analisador do sinal proposto pelo primeiro objetivo. Será utilizada uma interface que integre tanto o módulo de análise do som do violino quanto o joystick ou teclado, simulando

a entrada esperada do módulo, e integrando também o teclado com a função de seletor de opções do jogo. O jogo consistirá basicamente de um menu com diversas opções e telas de música. Essas telas de música apresentarão uma partitura simplificada e colorida se deslocando para esquerda conforme a música toca.

1.2 Motivação

Atualmente o mercado dos jogos relacionados com música está crescendo cada vez mais. Já existem jogos com Bongôs, Bateria, Guitarra, Microfone, Mesa de DJ (*disc jokey*), tapete de dança, jogos de educação musical e rítmica, etc. A maioria desses jogos entretêm, divertem e transmitem noções de musicalidade, mas a distância entre os instrumentos musicais reais e os controles utilizados neles limitam seu potencial educacional. Este projeto tem o intuito de permitir o real aprendizado de um instrumento musical - neste caso, o violino - ao mesmo tempo em que se diverte com o jogo.

O aprendizado de violino e música clássica é pouco disseminado em nosso país e em muitos outros do mundo. Este software ficará disponível para download como *open source*, permitindo uma divulgação do instrumento e de como aprender a tocá-lo de um jeito inovador. Com o sistema de captação de frequência do jogo proposto neste projeto, o aluno poderá verificar a afinação das notas mesmo sendo um iniciante, já que o software indicará se a nota está mais grave ou aguda do que a esperada. Para que o aluno saiba onde por os dedos, pretende-se disponibilizar um adesivo para ser colado no braço do violino com pequenas faixas de doze cores (próximas às do arco-íris), uma para cada semitom.

1.3 Organização do Documento

O presente documento está dividido nos seguintes capítulos:

- Capítulo 2 - Aspectos Conceituais: são definidos os conceitos básicos para compreender discuções ao longo de todo o documento;

- Capítulo 3 - Estudo sobre a Estimação de f_0: neste capítulo são descritos três métodos de estimação de f_0 que foram implementados e testados em matlab para definir qual deles utilizar no projeto;

- Capítulo 4 - Especificação do Projeto: neste capítulo são definidos todos os requisitos necessários para o projeto, bem como o impacto destes na definição do escopo do projeto. São levantadas as características técnicas dos equipamentos e padrões que serão estudados e utilizados ao longo do projeto;

- Capítulo 5 - Metodologia: este capítulo descreve a principal sequência de tarefas realizadas, levando em consideração as decisões tomadas sobre quais ferramentas e técnicas utilizar;

- Capítulo 6 - Implementação: este capítulo descreve todas as atividades, especificações técnicas e componentes de software que foram desenvolvidos para a implementação do projeto;

- Capítulo 7 - Testes e Avaliação: este capítulo descreve os principais testes executados para a validação das funcionalidades do jogo e do medidor de frequências;

- Capítulo 8 - Considerações Finais: este capítulo avalia os resultados globais do projeto, analisando as dificuldades e aspectos positivos do projeto, além de ressaltar os projetos futuros que poderão se basear no presente projeto;

- Capítulo 9 - Tarefas Adicionais: este capítulo descreve as tarefas que não são o foco do projeto de engenharia, mas são importantes para o funcionamento e divulgação do jogo.

Capítulo 2

ASPECTOS CONCEITUAIS

2.1 Jogos Musicais

Há uma grande variedade de jogos musicais para arcade, videogames (portáteis ou não), ou mesmo para computador. Todos eles trazem noções de ritmo e trazem uma noção superficial de tonalidade, alguns permitem um espaço para dinâmica ou composição musical. A maioria desses jogos funcionam com as "notas" (não são notas de verdade, mas sim algum botão representando a nota) ou instruções (como por exemplo, bater palmas ou pisar com o pé direito no canto superior do tapete) caminhando sobre uma esteira em direção a uma linha limite (chamam-se estas "notas" ou instruções de bolinhas, porque é assim que elas aparecem em muitos jogos). Quando as bolinhas chegam na linha limite, é o momento correto do jogador tocar a nota ou realizar a instrução. Veja o exemplo do jogo Donkey Konga (Nintendo Gamecube) na Figura 2.1-*b*). Este jogo da Nintendo tem como "instrumento" um tambor duplo chamado de bongô e também um sensor de palmas ilustrado na Figura 2.1-*a*). Quando a bolinha amarela chega ao círculo transparente no centro do lado esquerdo da tela (linha limite), o jogador deve bater no tambor esquerdo. Quando é a bolinha vermelha o jogador deve bater no tambor direito, e para a bolinha rosa bater nos dois tambores simultaneamente. Há ainda uma bolinha azul que indica que o jogador deve bater palmas, embora esta não apareça na figura. O jogo também tem um indicador de desempenho no canto superior direito, um contador de pontos no canto inferior direito, um contador de combos (notas acertadas em sequência) no canto inferior esquerdo e um macaco que imita o jogador no canto superior esquerdo (bate no tambor dele ao mesmo tempo em que o jogador toca no seu).

Figura 2.1: Propaganda *a*) e Tela do Jogo *b*) Donkey Konga.
Fonte: *http://www.video-games-magazine.com/images/donkeykonga.jpg*

Outro bom exemplo de jogo musical é o Jogo Guitar Hero®, ilustrado na Figura 2.2. Neste jogo, o controle utilizado pelo jogador é uma guitarra de plástico com cinco botões e uma barra para palhetar com os dedos, como ilustrado na Figura 2.3. A tela do jogo apresenta a esteira com bolinhas se deslocando na vertical, como se estivessem se aproximando do jogador. Quando as bolinhas chegam na linha limite é o momento do jogador palhetar sua guitarra, estando ao mesmo tempo pressionando o botão da cor da bolinha. Caso o jogador acerte, aparece uma chama acima da bolinha acertada (veja o círculo azul da linha limite na 2.2), caso ele erre a bolinha fica escura. Também há um contador de pontos, que aumenta para cada nota acertada, o quanto é aumentado depende de quantas notas em sequência o jogador já acertou ("combo" de notas). E há ainda uma barrinha de desempenho que fica verde se o jogador esta indo bem , ou vermelha se o jogador está indo mau. Caso o jogador esteja perdendo muitas notas, a música pára e o jogador é vaiado pelo jogo. Este jogo e seu nome foram um dos motivos da escolha do nome do projeto.

Figura 2.2: Tela do jogo Guitar Hero

Fonte:*http://databits.files.wordpress.com/2009/07/guitar-hero-5.jpg*

Figura 2.3: Paulo segurando a guitarra do jogo Guitar Hero

2.2 Aprendizado de violino

O violino é um instrumento de cordas que existe em vários tamanhos, desde violinos 4/4 (chamados violinos inteiros), com aproximadamente 60 cm, até violinos 1/16, com aproximadamente 30 cm [1]. Essa variação de tamanhos existe para que o instrumento possa ser tocado por crianças de qualquer idade. O instrumento apresenta quatro cordas (sol, ré, lá, mi) e uma caixa acústica como mostrado na Figura 2.4-*a*), que recebe o som através do cavalete que apoia as cordas e transmite este som para sua parte inferior através da alma do violino (pequeno pilar de madeira conectando a parte superior e inferior da caixa acús-

tica). Atualmente existem também violinos elétricos como mostrado na Figura 2.4-*b*), que tem as quatro cordas e um cavalete, mas ao invés de usar a caixa acústica, ele capta o som do cavalete e o transforma em sinal elétrico. Esse sinal normalmente é pré-amplificado no próprio violino e é passado para uma saída que deve ser conectada a algum amplificador externo ou a um fone de ouvido.

O aprendizado do violino é considerado difícil, entre outros motivos, pelo instrumento não possuir trastes indicando onde colocar os dedos e pela precisão necessária na colocação dos mesmos para obter notas afinadas. Após um tempo de estudo e dedicação considerável, o aluno decora a posição dos dedos e aprende a corrigir a afinação das notas apenas ouvindo.

a)  *b*) 

Figura 2.4: Violinos acústico *a*) e elétrico *b*) utilizados nos testes do projeto.

2.3 Teoria Musical

Nosso software foi feito de modo que mesmo um leigo em música consiga jogar. Para aqueles interessados em entender ou aprender um pouco sobre música e partituras musicais ele foi criado com características musicais reais, que são brevemente explicadas neste subcapítulo.

Todo som apresenta quatro propriedades: altura, intensidade, duração e timbre. De acordo com sua altura (frequência) eles foram classificados em sete notas musicais (Lá Si Dó Ré

Mi Fá Sol ou A B C D E F G). Estas notas musicais podem ser agrupadas entre si e com pausas, em qualquer ordem, com ou sem repetições, com ou sem sobreposições, tendo cada uma sua duração, timbre e intensidade. Estas sequências de notas ou combinações de notas (acordes) são exemplos de música.

Para este tipo de música, foi criado um método de descrever no papel o que deve ser tocado ou ouvido, conhecido como sistema clássico de partituras musicais. Neste sistema, toda nota é representada por uma bolinha num pentagrama (cinco linhas horizontais). De acordo com o símbolo desta bolinha sabe-se quanto tempo a nota dura. Ela pode durar um compasso inteiro ou frações de compasso, sendo chamada de semibreve, mínima, semínima, colcheia, semicolcheia, etc, como mostrado na Figura 2.5.

Sabe-se qual a nota que a bolinha representa de acordo com sua posição no pentagrama, se está em cima de qual linha ou entre quais das linhas da partitura, como ilustrado na Figura 2.6. Para cada clave (Fá, Sol e Dó), identificada pelo símbolo no início de toda linha da partitura, temos cinco linhas diferentes sendo apresentadas [2]. Como as partituras para violino usam sempre a clave de Sol, que é a mais aguda, e onde a segunda linha de baixo para cima representa um Sol, trataremos apenas desse tipo de partitura.

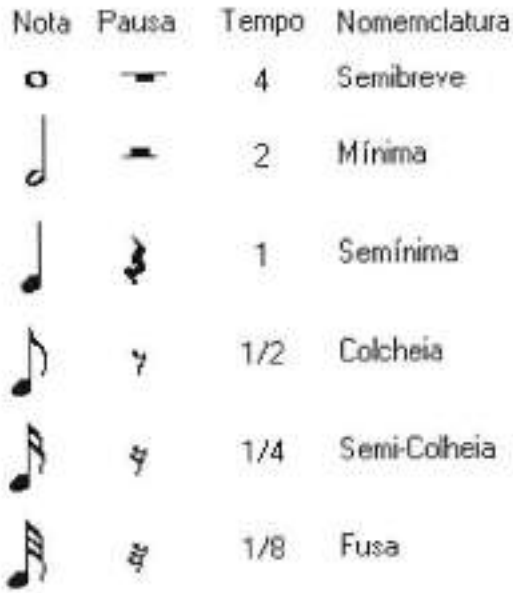

Figura 2.5: Símbolos que respresentam notas na partitura musical

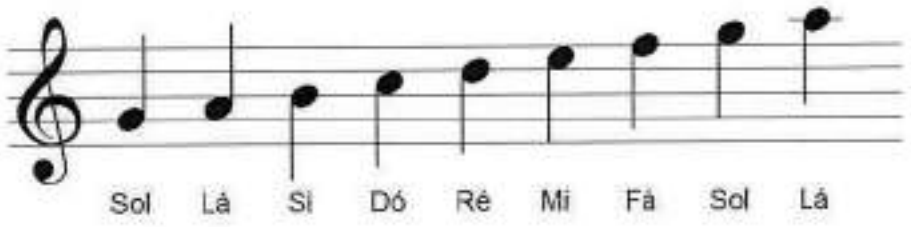

Figura 2.6: Exemplo de Partitura musical em Clave de Sol

2.4 Trabalhos relacionados

2.4.1 Área de Games

Jogos musicais já foram alvo de estudo para saber a real eficácia do aprendizado de um instrumento [3]. No jogo proposto neste projeto, pretende-se dar mais ênfase ao lado didático sem perder a diversão e a facilidade de controle do jogo. Pretende-se aproveitar o potencial dos computadores pessoais atuais ao invés de utilizar somente acionamentos de botões assim como em [4], em que comandos computacionais são acionados através de frases rítmicas.

2.4.2 Análise de Áudio de Violino

Já foram realizados trabalhos com o intuito de se ensinar a tocar violino pelo computador (veja, por exemplo, [5] e [6]). Além de verificar a correta afinação do som tocado como no jogo proposto, esses programas utilizam videos ou *avatares* para ajudar a ensinar como segurar o instrumento e tocar corretamente.

2.4.3 Estimação de f_0

O problema de se encontrar a frequência fundamental (f_0) de sinais é bem conhecido e tem sido abordado por diversos métodos ao longo dos anos. Um breve resumo da evolução do problema pode ser encontrado em [7]. Cercando um pouco mais o problema, sabemos que temos que encontrar a f_0 de sinais musicais, mas mesmo se tratando de um sinal musical [8] ainda há muitas abordagens possíveis. Para ter uma base consolidada das ferramentas a nossa disposição, foi feito um estudo das soluções contidas na literatura,

que abordaram problemas similares.

Os métodos de se encontrar f_0 podem ser divididos em 4 aproximações [9]:

1. encontrar f_0 através de reconhecimento de padrões;
2. encontrar o período fundamental do sinal no tempo;
3. encontrar a diferença entre dois harmônicos do espectro do sinal;
4. utilizar métodos que simulam a audição humana.

Pode-se pensar em outros métodos, como simplesmente calcular o espectro do sinal e verificar onde se encontra o primeiro pico de amplitude que não tenha sido criado por rebatimento ou interferência, a frequência deste será f_0.

Para encontrar o período do sinal (segunda abordagem), normalmente se usa a autocorrelação, que pode ser calculada usando convolução. Devido às propriedades desta operação matemática, a autocorrelação de um sinal é máxima em zero [10] e tem máximos locais para valores de intervalo t iguais a múltiplos do período do sinal. Portanto, basta encontrar o primeiro ponto de máximo da autocorrelação diferente de zero que temos o período fundamental do sinal. Um exemplo de sucesso deste método é descrito em [8].

A terceira abordagem, que trata de encontrar a diferença entre dois harmônicos do sinal, pode ser interpretada como descobrir o "período" do espectro do sinal. Portanto se torna um problema parecido com o descrito anteriormente, sendo que a autocorrelação pode ser utilizada. A Figura 2.7 retirada de [9] exemplifica esta abordagem.
Outra possível abordagem (quarta na lista anterior) trata de simular a audição humana. Para simular o sistema auditivo, normalmente os algoritmos realizam uma sequência de transformadas. Primeiramente, a forma de onda é retificada (retira-se a parte negativa do sinal), o que faz aparecer a diferença entre os harmônicos (f_0) na área de baixas frequências do espectro do sinal. Como se deseja apenas as frequências próximas do zero, aplica-se um filtro passa-baixa ao sinal. Assim, restam apenas as frequências baixas, e a primeira destas é f_0. Esta abordagem é exemplificada na Figura 2.8 retirada de [9].

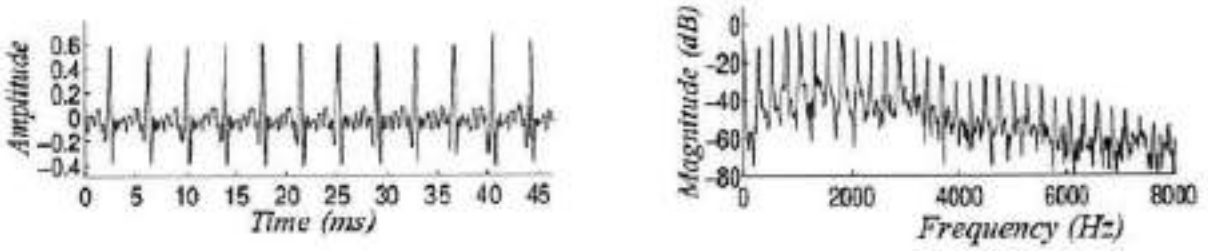

Figura 2.7: Estimação de f_0 por diferença entre picos do espectro

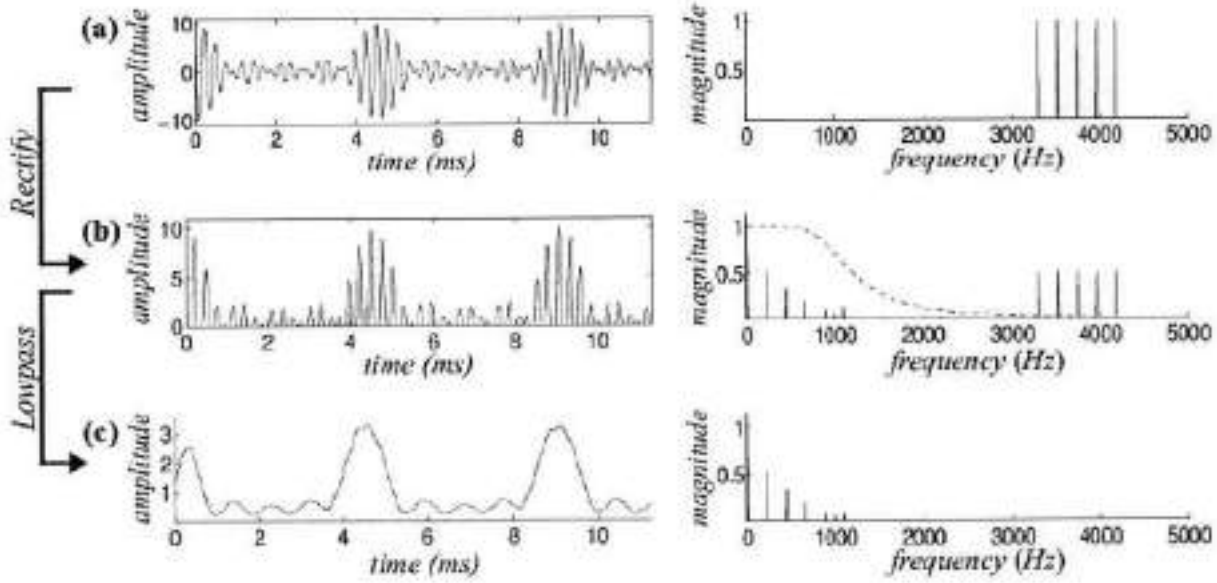

Figura 2.8: Estimação de f_0 por simulação de audição

As vantagens e desvantagens de cada método estão ligadas às características das formas de onda no tempo ou na frequência. Em alguns casos, é interessante utilizar dois métodos e comparar os resultados, uma vez que a periodicidade do período pode indicar uma frequência que é o dobro da desejada e a periodicidade da frequência pode indicar uma frequência que é metade da desejada.

Capítulo 3

ESTUDO SOBRE ESTIMAÇÃO DE f_0

Devido às muitas opções possíveis para o método de estimar a frequência fundamental foi necessário um estudo detalhado para escolher o melhor método para nosso projeto. Primeiramente, gravamos sinais do violino acústico e elétrico para nos familiarizarmos com o tipo de sinal a ser estudado. Testes iniciais feitos nestas gravações revelaram características do som do violino que ajudaram na escolha de três métodos indicados para resolver o problema. Estes três métodos foram implementados e testados no Matlab, para finalmente chegar a uma conclusão de qual o melhor método no nosso caso.

3.1 Testes Iniciais

Antes de gravar os sinais, pesquisamos programas de afinação instântanea de instrumentos como o descrito abaixo, cuja versão demo é de uso livre.

3.1.1 Wtune

Este programa mostra o espectro do sinal captado pelo microfone do computador como mostrado na Figura 3.1, dando opção de gravar o sinal como ".wav", MIDI ou num relatório ".txt". Como padrão, ele utiliza o algoritmo FFT com 8192 pontos e janela de Hamming. Porém, é possível escolher outra janela ou menos pontos. Ele foi projetado para ser utilizado com qualquer instrumento e não tem um desempenho muito bom com violinos. Foi observada dificuldade de identificação da frequência nas transições de uma corda para outra. Além disso, quando a amplitude de algum harmônico é maior do que a da frequência fundamental, ele erroneamente indica a frequência do harmônico como a que esta sendo tocada. Isto não ocorreu para os testes com o violino elétrico, reforçando a nossa escolha pelo instrumento. Os dois espectros mostrados na Figura 3.2 são muito parecidos, porém fica visível que, enquanto para o violino elétrico a frequência medida é a esperada f_0 (podemos localizar a frequência medida pelo pequeno "x" vermelho no espectro), para o violino acústico a frequência indicada é um dos harmônicos, não a fundamental. Foi interessante

utilizar este programa para ter uma ideia dos problemas que poderíamos encontrar ao utilizar nosso método para encontrar a frequência fundamental.

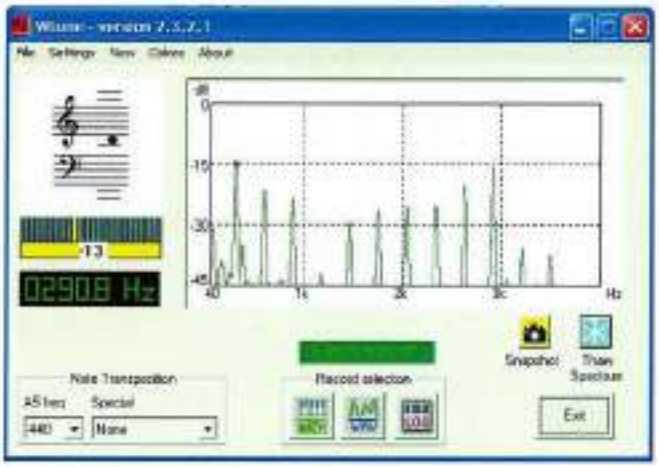

Figura 3.1: Exemplo de espectro da nota Re (293.66Hz) captado pelo programa Wtune

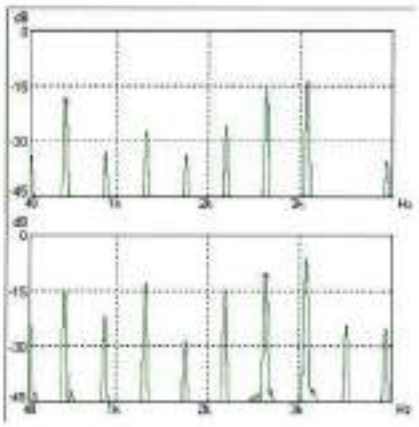

Figura 3.2: Espectro da nota La com programa Wtune.
Acima violino Elétrico, abaixo violino Acústico

3.1.2 Sinais Gravados

Foram captados sinais considerados interessantes para análise do som do violino elétrico, estando este conectado diretamente ao computador através da porta de microfone. Foi utilizado o gravador de áudio convencional do Windows Xp e o formato ".wav" (PCM 22,05 kHz ; 16 bit ; Mono). A seguir, estão listados os principais sinais gravados.

Sinais para familiarização com o violino:

escalaCmaior - Escala Dó maior indo desde o C4 (261.63Hz) até o C6 (1046.50Hz) subindo nota por nota e depois descendo pelas mesmas notas.

E-sobedesce - "Escala"da corda Mi inteira sem separação nota à nota (glissando).

Extremos - Notas mais grave(Sol- G3 196.00Hz), mais aguda(aprox. 2kHz) e intermediária do instrumento.

TwinkleStar - Canção Infantil bem conhecida.

Sinais gravados para testar os métodos:

Lá-acus / **Lá-elec** - Apenas a nota Lá 440Hz, cada vez com um dos violinos(acústico e elétrico).

Si-acus / **Si-elec** - Apenas a nota Si 493.88Hz, cada vez com um dos violinos.

Ré-acus / **Ré-elec** - Apenas a nota Lá 293.66Hz, cada vez com um dos violinos.

3.1.3 Espectograma

Utilizando um trecho de código ".mat" com a função espectograma (spectogram) do MatLab foi possível gerar o espectograma dos sinais descritos no item anterior, cujo código está apresentado no Apêndice A. Espectograma é um gráfico que traz a informação do espectro de um sinal que esta sendo captado ao longo do tempo. Nas figuras 3.3 e 3.4 o eixo x (horizontal) representa o tempo, eixo y (vertical) representa a frequência, e a cor representa a amplitude do espectro do sinal.

Não foi feito nenhum ajuste no tamanho das janelas ou em outros parâmetros, para melhorar a qualidade do espectograma (diminuir a intensidade do ruído entre os harmônicos), mas mesmo assim estão visíveis características interessantes. Como esperado, podemos observar uma grande quantidade de harmônicos no sinal, e estes tem intensidades de mesma ordem de grandeza que a fundamental. Apenas com a Figura 3.3 não é possível saber se as frequências estão sendo obtidas com uma precisão razoável, já que a diferença entre as notas é muito pequena na escala utilizada.

Comparando o espectograma da Figura 3.4 com a partitura musical da Figura 3.5 fica clara a relação entre o espectograma e a partitura musical tocada. Cada bolinha da partitura representa uma nota e também um patamar do espectograma. Dependendo da posição da bolinha no pentagrama ela representa uma nota diferente **??**. No caso a primeira bolinha é um Dó, seguida das bolinhas Ré, Mi, Fá, Sol, Lá, Si, Dó (uma escala acima do primeiro, tem o dobro da frequência) Re Mi Fa Sol La Si Dó (duas escalas acima do primeiro, tem quatro vazes a frequência dele) e voltando pelas mesmas notas. No espectograma, observamos as frequências subindo de degrau em degrau (nota em nota), e posteriormente descendo, também nota à nota.

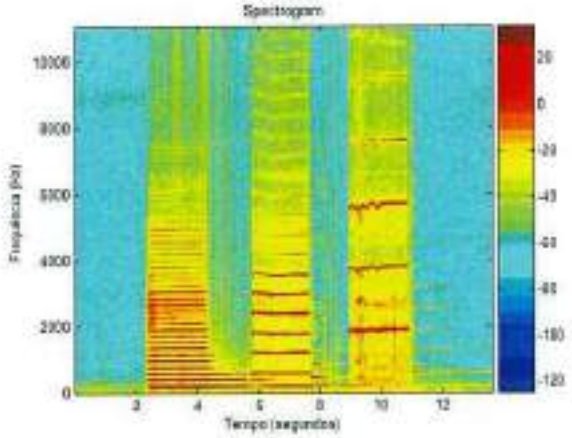

Figura 3.3: Espectograma do sinal Extremos.wav

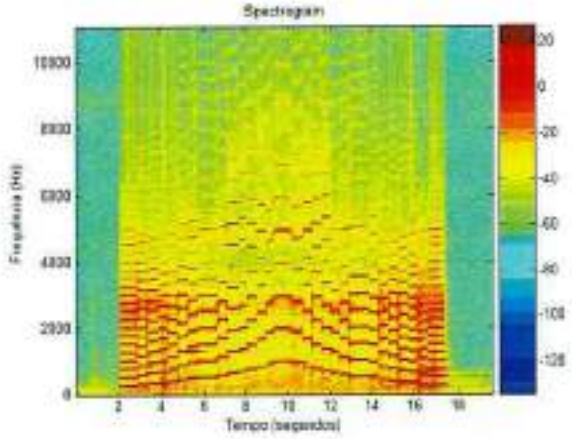

Figura 3.4: Espectograma do sinal escalaCmaior.wav

Figura 3.5: Escala C maior em partitura musical

"Brilha Brilha Estrelinha" é uma música de ninar bem famosa, ela é muito simples, pois quase todas as notas tocadas tem mesma duração e ela repete os mesmos trechos diversas vezes. No seu espectograma mostrado na Figura 3.6, fica evidente essa repetição e também a regularidade de duração das notas, que esta representada nos patamares de mesmo comprimento.

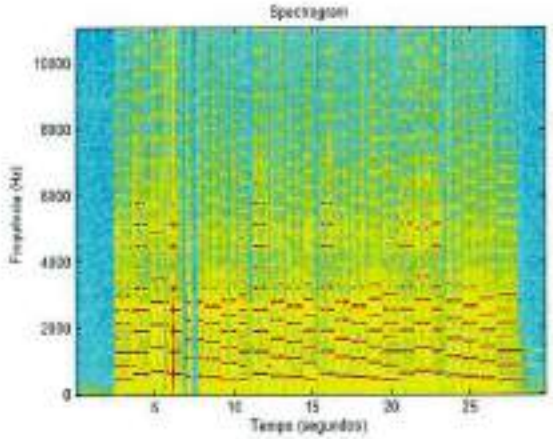

Figura 3.6: Espectograma do sinal TwinkleStar.wav

3.2 Métodos Implementados no Matlab

A seguir, descrevemos brevemente os três métodos de estimação de f_0 implementados no Matlab.

3.2.1 Fast Fourier Transform (FFT):

É um algoritmo eficiente para o cálculo da transformada de Fourier discreta, que por sua vez é calculada como

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}, \quad k = 0, \ldots, N-1, \tag{3.1}$$

sendo $x(n)$, $n = 0, 1, \ldots, N-1$ uma sequência formada por N amostras de um sinal de violino e $X(k)$ amostras do espectro desse sinal. Para se obter a frequência fundamental, é necessário encontrar onde a magnitude máxima de $X(k)$ ocorre. Supondo que o máximo de $|X(k)|$ ocorra em $k = k_{max}$, pode-se calcular $f_0 = (k_{max} f_a)/N$, sendo f_a a frequência de amostragem do sinal. Devido a alta intensidade dos harmônicos presentes no sinal do violino, a amplitude máxima nem sempre ocorre em f_0. Esse problema foi contornado em **??** transpondo-se o espectro para um eixo contendo apenas as frequências das possíveis notas do violino. Somando-se as fundamentais com seus cinco primeiros harmônicos, a f_0 da nota tocada corresponderá à maior soma obtida.

3.2.2 Autocorrelação:

Dada uma sequência $x(n)$ com N amostras, a autocorrelação de $x(n)$ pode ser estimada como

$$r(\ell) = \frac{1}{N} \sum_{n=0}^{N-n-1} x(n)x(n-\ell). \tag{3.2}$$

Sabe-se que a autocorrelação é máxima em $\ell = 0$ e seu segundo máximo ocorre em $\ell = T_0 = 1/f_0$ no caso de sinais periódicos. Dessa forma, basta encontrar o segundo valor máximo para se estimar a frequência fundamental. As figuras 3.7 e 3.8 ilustram um sinal representado no tempo e sua autocorrelação, respectivamente. A autocorrelação ainda pode ser implementada de forma eficiente, utilizando-se o algoritmo FFT.

3.2.3 Cepstrum picth:

É similar ao método da autocorrelação, mas neste caso calcula-se

$$c(\ell) = \text{IFFT}[\log(|\text{FFT}[x(n)]|)], \tag{3.3}$$

sendo IFFT$[\cdot]$ a FFT inversa. Como na autocorrelação, o segundo máximo ocorre em

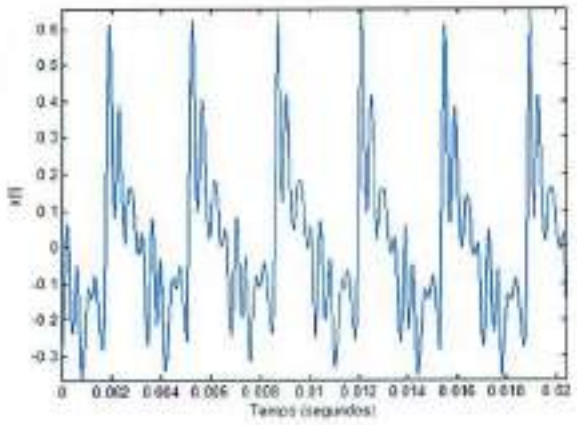

Figura 3.7: Sinal no Tempo da nota Ré (293,66Hz).

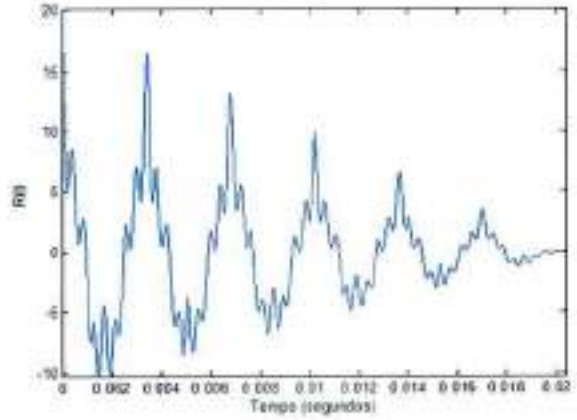

Figura 3.8: Autocorrelação da nota Ré (293,66Hz).

$\ell = T_0 = 1/f_0$. Esse método tem a vantagem de ser eficiente devido à utilização do algoritmo de FFT e de enfatizar os picos do espectro em relação ao ruído e demais harmônicos.

3.3 Teste dos Métodos

Os três métodos estudados foram implementados no Matlab (versão 7.6.0.304) e testados com três sinais de cada violino (acústico e elétrico) gravados no formato PCM, 16 bits, Mono e $f_a = 22,050$ kHz. Os trechos de análise foam compostos por 1024 amostras. No primeiro método, foi feita uma FFT com 8192 pontos (a sequência é completada com 7168 zeros). Foram avaliados o erro percentual da estimativa de f_0 e o tempo de processamento, mostrados na Tabela 3.1 no formato erro[%] / tempo[ms]. O tempo de processamento foi calculado no Matlab instalado num microcomputador com processador Athlon XP 2GHz e 500 MB de memória RAM.

3.3.1 Análise dos Resultados

O método da autocorrelação não se mostrou eficiente em termos de tempo de processamento quando comparado aos outros dois. O método de detecção do cepstrum picth é eficiente em termos de tempo de processamento, porém sua precisão variou mais que a do método da FFT. Além disso, apresentou um erro percentual elevado para a nota Lá do violino acústico (La-acus). Através destes testes, a FFT foi considerada o método mais indicado para o medidor de frequência dos sinais de violino. Testes com gravações diferentes, que levem em conta diversos trechos do sinal podem privilegiar o método da detecção por cepstrum picth.

Tabela 3.1: Resultados dos testes no Matlab

Sinal	FFT (8192)	Autocorr.	Cepstrum
La-acus	0,33 / 1,5	1,85 / 71	16,54 / 1,5
Si-acus	0,83 / 2,1	1,49 / 74	2,92 / 1,3
Re-acus	0,82 / 1,5	1,90 / 73	0,12 / 1,4
La-elec	0,94 / 1,7	1,85 / 72	0,23 / 1,3
Si-elec	1,37 / 1,9	1,49 / 70	1,47 / 1,3
Re-elec	0,82 / 1,8	1,90 / 72	1,20 / 1,3

Capítulo 4

ESPECIFICAÇÕES

4.1 Especificações do projeto

4.1.1 Recursos

Utilizou-se a sala com isolamento acústico do LPS (Laboratório de Processamento de Sinais do PSI) para fazer captações mais precisas do violino acústico. Fora isso, os principais recursos utilizados são programas de computador. Abaixo, estão listados os programas e sua finalidade no projeto:

- **Wtune** - mostar espectro do som do violino.
- **Gravador de som do Windows Xp** - captar sinais do violino e grava-los em ".wav"
- **Matlab** (versão 7.6.0.324) - testar algoritmos com funções de análise espectral já implementadas ou facilmente implementáveis.
- **Paint** - desenhar protótipos de como será a tela do jogo.
- **RFF** - desenhar diagrama de fluxo dos menus do jogo.
- **Gantt Project** - elaborar o cronograma em diagrama de Gantt.
- **Eclipse** - plataforma de programação em Java.
- **xp-dev** - repositório SVN (para programação em grupo).
- **Ilustrator CS4** - design do adesivo.
- **Encore** (versão5.0) - editar e criar partituras em formato MIDI.
- **EaseMIDIConverter** - passar as músicas do formato .mid para .mp3.
- **Photo Filtre** - editar imagens do violino e das partituras.

4.1.2 Custos

Por se tratar de um software, foi possível evitar custos diretos na produção do produto final utilizando plataformas livres para a programação e criação do mesmo. Houve o custo da inscrição no Simpósio Brasileiro de Games (SBGames), onde foi apresentado o artigo que está no Apendice E e pequenos custos como a impressão do adesivo, das monografias e dos pôsteres.

4.1.3 Riscos / Potenciais problemas

O maior risco de nosso software seria o jogo não funcionar em tempo real. Isso tiraria muito do atrativo do jogo e também prejudicaria a eficiência do aprendizado do instrumento. Para que isto não ocorresse, foi necessário que a frequência fundamental fosse obtida instantaneamente (para padrões de reflexos humanos) e que a resposta do jogo para um acerto ou erro também fosse imediata.

4.1.4 Produtos

O principal produto final do projeto é o software nomeado "Violin Villain". Outro produto foi um *short-paper* sobre o jogo em sua fase de desenvolvimento, apresentado no SBGames realizado em Florianópolis de 8 a 10 de novembro de 2010 (cópia do artigo está no Apêndice E). Também foi criado um site do jogo e o adesivo a ser grudado no braço do violino. Foram escritas duas monografias, uma para o PSI e outra para o PCS. As monografias são idênticas em sua introdução, conclusão e grande parte do desenvolvimento, de modo que cada uma delas seja suficiente para entender o projeto e software completamente. Há diferenças apenas na parte mais especializada das tarefas dos alunos de cada departamento.

4.1.5 Disciplinas relacionadas

DO CURSO DE SISTEMAS ELETRÔNICOS (PSI): Durante o curso de graduação em Engenharia Elétrica com ênfase em Sistemas Eletrônicos cursei diversas disciplinas que me prepararam para realizar este projeto. As principais estão listadas a seguir:

- *PTC 2307- Sistemas e Sinais I*
- *PTC 2308- Sistemas e Sinais II*
- *PSI 2432- Projeto e Implementação de Filtros Digitais*
- *PSI 2533- Modelagem em Processamento de Sinais*

As quatro disciplinas acima tratam de ferramentas ou conceitos necessários para a análise de sinais. É importante ter uma base para poder entender o problema, imaginar soluções e interpretar com facilidade as referências nesta área.

- *MAC 0122- Princípio em Desenvolvimento de Algoritmos*
- *PCS 2478- Tópicos de Programação*
- *PSI 2653- Meios Eletrônicos Interativos I*

Estas três disciplinas de programação, em C e C++, servem como base para aprender a programar em outras linguagens e implementar algoritmos de análise de sinais em código de linguagens de alto nível.

DO CURSO DE COMPUTAÇÃO (PCS): Durante o curso de graduação em Engenharia Elétrica com ênfase em Computação, os alunos Edson e Jonathan cursaram diversa disciplinas que ajudaram na realização do projeto. As principais estão listadas a seguir:

- *MAC2301 - Laboratório de Programação*
- *PCS2309 - Engenharia de Software I*
- *PCS2310 - Engenharia de Software II*
- *PCS2419 - Laboratório de Engenharia de Software I*
- *PCS2420 - Laboratório de Engenharia de Software II*
- *PCS2405 - Arquitetura de Computadores*

- *PCS2453 - Sistemas Operacionais*

Essas disciplinas proporcionaram uma compreensão acerca de lógica de programação e de boas práticas, além de aspectos secundários que impactam no funcionamento de um software, como por exemplo, a necessidade de programação por múltiplas *threads*, implicando na necessidade de tratamento de eventos por filas.

- *PCS2510 - Hipermídia e multimídia*
- *PCS2520 - Tecnologia de Computação Gráfica*
- *PCS2530 - Design e Programação de Games*

Essas disciplinas deram os conhecimentos para a programação de jogos, envolvendo a fundamentação de computação gráfica, o uso de ferramentas como o *Processing*, utilizado no projeto, o aprendizado em design de jogos e aspectos como a interface com usuário.

- *PTC2359 - Engenharia de Comunicações*

A fundamentação necessária para o tratamento do requisito não-funcional de tempo de resposta advém dos conhecimentos adquiridos por meio dessa disciplina, que trouxe conceitos de amostragem e conversão analógico-digital, proporcionando, ao aplicá-los no projeto, maior entendimento sobre a transformação do sinal de violino em estados de variáveis do programa.

Além dessas muitas outras disciplinas deram o embasamento teórico em aspectos de eletrônica, elétrica e software, porém dá-se o destaque para os mencionados acima, por maior impacto no projeto.

4.2 Especificações do Medidor de Frequências

Devido ao estudo descrito no capítulo anterior, o método especificado para o medidor de frequências é o algoritmo FFT, seguido da detecção de pico local próximo da frequência

esperada. A frequência de amostragem utilizada foi 22.05kHz, e o número de amostras para se fazer uma medição foi igual a 1024. Desse modo, o tempo de amostragem ficou igual a 46ms, que é um tempo suficientemente baixo para manter a resposta rápida do jogo. Porém, neste caso a precisão é igual a apenas 21Hz, que não é suficiente para permitir a correção da afinação das notas. Este problema foi resolvido implementando-se a FFT no vetor de 1024 pontos completado com zeros até ter 4096 pontos. Com isso, a precisão foi melhorada para 5.25Hz, sem aumentar muito o tempo de processamento, devido à eficiência do algoritmo da FFT. Mas 5.25Hz de precisão ainda não é suficiente para corrigir precisamente a afinação do instrumento. Foi necessário acrescentar uma interpolação por polinômio, feita através do ponto máximo do espectro e seus vizinhos. O valor máximo desse polinômio indica f_0, com precisão de 0.5Hz, que é uma variação normalmente imperceptível para o ouvido humano.

4.3 Especificações do jogo

Violin Villain terá a mesma estrutura que muitos jogos de música. Primeiramente, haverá um conjunto de menus com diversas opções de escolha e em seguida será apresentada uma tela contendo a partitura da música escolhida. Terminada a música mostra-se o desempenho do jogador e volta-se ao menu. Para tornar o jogo atrativo mesmo para pessoas que não tenham um violino, será possível jogar com um *joystick* convencional ou mesmo com o teclado. Nestes casos, serão utilizados botões ou teclas para representar as notas, do mesmo modo que em outros jogos musicais.

4.3.1 Menus e Modos de Jogo

Ao ser iniciado, o jogo pergunta qual a interface a ser usada (violino, teclado ou joystick). Em seguida, tem-se uma seção de menus como descrita no fluxograma da Figura 4.1. Primeiramente, o jogador escolhe entre o **Options** (afinar seu violino, ajustar a configuração do teclado ou joystick), **Study** (praticar) ou **Game** (apenas se divertir). Escolhendo **Game**, o jogador então escolherá qual música ele vai tocar e qual a dificuldade em que a música vai ser apresentada (*Easy Mode*, *Normal Mode*, *Real Mode*). O *Easy Mode* apresentará apenas as notas principais da partitura da música e não se preocupará em

seguir fielmente as distâncias tonais entre as notas. Este terá apenas quatro notas, todas na corda La (corda solta-La, primeiro dedo-Si, segundo dedo-Do e terceiro dedo-Re). O *Normal Mode* já terá uma relação mais próxima com a partitura real, mas ainda não trará nenhum trecho difícil. Este terá oito notas, quatro na corda La e quatro na corda Mi. O *Real Mode* apresentará a partitura completa e usa todas as notas do instrumento. Tanto o *Easy* quanto o *Normal Mode* não serão agradáveis sonoramente quando jogados com violinos acústicos, já que nesses modos o som que o violino produz é diferente do som tocado pelo computador e isso atrapalha o jogador. Com o violino elétrico este problema é menor já que o som produzido pelo instrumento sem amplificadores externos é muito baixo. Escolhendo **Study**, o jogador poderá optar entre músicas e exercícios. Além de escolher a dificuldade, o jogador também poderá escolher a velocidade em que a partitura será passada e se quer tocar a música inteira ou apenas um trecho da música.

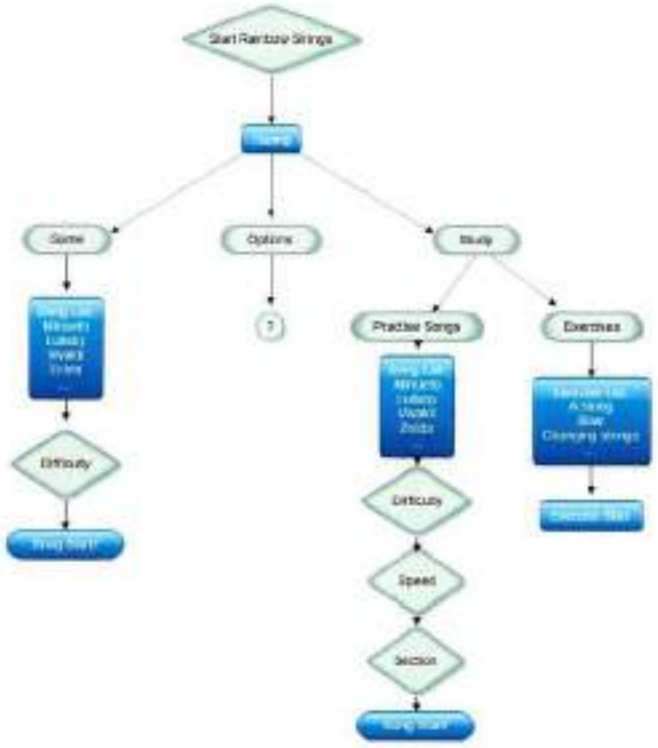

Figura 4.1: Esquema de menus de Rainbow Strings.

4.3.2 Proposta de Interface

Pretendia-se que a tela do jogo no momento de se tocar as músicas fosse apresentada como na Figura 4.2. Este protótipo foi desenhado para o modo **Easy**, que por ter

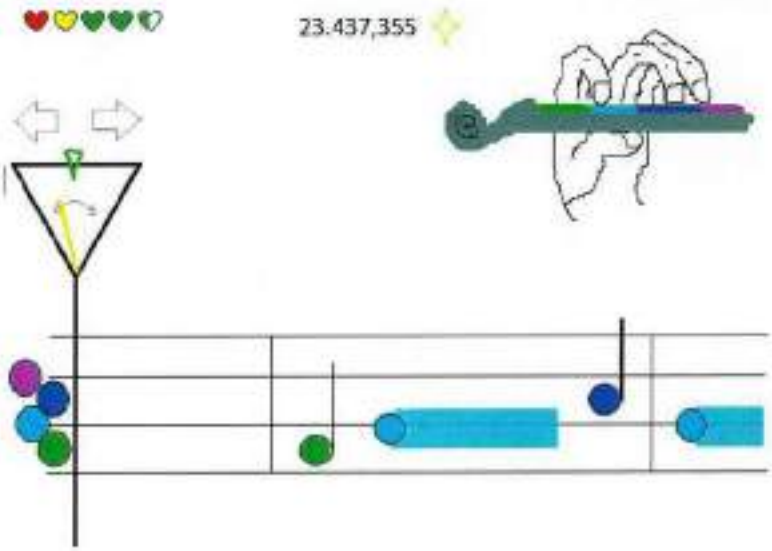

Figura 4.2: Protótipo da tela do jogo

apenas quatro notas, precisa de apenas quatro linhas na partitura. Para os outros níveis de dificuldade aparecerão cinco linhas, como numa partitura normal. O triângulo invertido acima da linha onde as notas devem ser tocadas indicará a afinação da nota. A seta para direita acenderá se a nota estiver grave demais e a seta para esquerda acenderá se a nota estiver aguda demais. Além disso, no triângulo haverá um ponteiro que indicará a mesma coisa, porém com uma precisão maior do que com as setas. No canto superior esquerdo haverá uma figura para indicar a colocação dos dedos (quais dedos deverão estar juntos e a distância entre eles). Isso é importante porque para cada escala, os dedos ficam em uma configuração específica. No centro da parte superior da tela, haverá um indicador de pontos que aumenta sua contagem de acordo com o desempenho do jogador na música escolhida. No canto superior esquerdo, haverá corações para indicar se o jogador está tocando bem ou mal, caso o jogador perca muitas notas em sequência os corações começarão a desaparecer. Caso sumam todos os corações, a música parará porque o jogador

não conseguiu acompanhá-la adequadamente. Deve-se observar que a proposta atual de interface pode ser modificada durante a implementação e teste do jogo com usuários.

4.3.3 Reprodução de Áudio

Assim como outros jogos semelhantes ao proposto neste artigo, como *Performous* [11] e *Frets on Fire* [12], cada música será dividida em três arquivos: um arquivo de áudio com a música a ser tocada sem o som do violino, outro arquivo de áudio somente com o som do violino e por último um arquivo com as informações das notas que devem ser tocadas, informando o tipo da nota e o tempo em que elas devem ser tocadas. O jogador poderá escolher se quer que o som do seu violino seja tocado ou se o segundo arquivo de áudio seja usado como padrão. Assim, durante uma partida, quando a nota tocada estiver correta, o som do violino ou do arquivo de áudio com violino irá para a saída de som. Se a nota tocada estiver errada, sons de erro serão tocados. Se o jogador perder uma nota, o som de violino não será tocado.

4.3.4 Tecnologia

Para a implementação do projeto será utilizada a linguagem java. Quanto à captação dos sinais de áudio, o próprio pacote javax.sound do jdk será usado. Para a interface gráfica será usada a biblioteca OpenGL para Java, mais especificamente JOGL junto com a biblioteca Processing. Já o controle do jogo via Joystick será feito através da biblioteca JXinput, parte do projeto Distributed Realtime Simulation *(http://sourceforge.net/projects/drts/)*.

4.3.5 Game Design para Aprendizado

A primeira dificuldade em se aprender a tocar violino é conseguir segurar o instrumento de forma confortável. O jogo proposto não verificará se o jogador está segurando o instrumento corretamente. No entanto, serão apresentadas fotos e instruções que descreverão como isso deve ser feito (mantenha o pulso esquerdo reto, segure o instrumento com o peso do queixo, não mover o antebraço para tocar, mantenha-se relaxado para tocar, etc).

Outra dificuldade é a leitura de partituras musicais, onde cada símbolo (colcheia, semicolcheia, semínima, etc) representa uma duração e a posição da bolinha indica qual nota deve ser tocada. No jogo proposto, cada nota será mantida com seu símbolo, porém será acrescentado um rastro na mesma para indicar sua duração (como no jogo Guitar Hero ou em outros jogos). Para saber qual a nota a ser tocada, as bolinhas terão a cor da nota que representam. Pretende-se colocar um adesivo no braço do violino para indicar onde colocar os dedos. No entanto, como o mesmo dedo pode ficar em mais de uma posição, será mostrado um desenho ou uma foto de como os dedos devem ser posicionados em cada música (no modo *Easy*, todas as músicas usarão a mesma posição para facilitar).

Para jogadores que queiram se tornar alunos, o jogo terá uma seção de treino e aprendizado, onde o aluno poderá tocar exercícios de violino, necessários para o aprendizado do instrumento (exercícios de arco, escalas, mudança de corda, afinação, velocidade, etc).

4.3.6 Requisitos Funcionais e Não-Funcionais

Abaixo listamos os requisitos funcionais elaborados na etapa de especificação.

1. *O programa receberá o sinal de um violino elétrico como entrada, e cada nota será um comando. Como se o violino fosse um Joystick.*
2. *O programa receberá o sinal padrão de entrada do teclado do computador com funções atreladas a teclas previamente configuradas.*
3. *O programa receberá o sinal de um joystick como entrada.*
4. *Durante uma partida haverá uma barra na esquerda da tela indicando o momento de uma nota ser tocada. Quando uma nota chega à barra se for tocada a nota é acertada, senão o jogador perde a nota. Essa barra será chamada de barra de sincronia.*
5. *O jogador pode escolher no menu Options se ele quer que o som do violino seja reproduzido na saída de áudio ou não.*

6. *Se o jogador optar por não reproduzir o som de violino na saída ou estiver usando o Teclado ou Joystick, os sons correspondentes às notas que devem ser tocadas serão reproduzidos automaticamente pelo jogo.*

7. *Quando uma nota é errada (foi tocada uma nota mas não é a correta) será reproduzido um som de erro e o som do violino não é jogado na saída.*

8. *Quando uma nota é perdida (deveria ter sido tocada mas não foi) o som do violino não é jogado na saída.*

9. *Durante uma partida haverá uma música de fundo sincronizada com as notas que deslocam na tela sem o som do violino.*

10. *Durante uma partida haverá um sistema de pontos.*

11. *Durante uma partida haverá um sistema que avisa o quão bem o jogador está indo na partida, quando ele acerta notas o indicador aumenta e quando ele erra as notas o indicador diminui. Esse sistema será chamado de indicador de sucesso.*

12. *Durante uma partida se o indicador de sucesso chegar a níveis baixos por certo tempo o jogador perde a partida.*

13. *Se até o final de uma música o jogador não perder por nível baixo de indicador de sucesso o jogador ganha a partida.*

14. *O programa terá um menu para que o jogador possa ir para telas diferentes.*

15. *O menu terá uma entrada Game para o jogador atingir a tela de escolha de música.*

16. *O jogador pode escolher a música que quer jogar em uma partida.*

17. *O jogador pode escolher a dificuldade da música que quer jogar em uma partida.*

18. *O menu terá uma entrada Options para o jogador escolher as opções de jogo.*

19. *Assim que o programa inicia entra em uma tela de calibração, para que o jogador toque notas específicas do violino a fim de ajustar o software de análise de sinal.*

20. *Arquivos contendo as músicas sem o violino, só com o violino e com os momentos que cada nota deve ser tocada devem ser fornecidos para a execução do jogo.*

21. *O Teclado e o Joystick terão um botão que durante a partida se for pressionado faz o jogo voltar para o menu inicial.*

22. *O menu terá uma entrada Study para o jogador poder praticar músicas e fazer exercícios*

23. *Modo Practice. No modo de praticar músicas o jogador não perde por nível baixo de indicador de sucesso.*

24. *No modo de praticar músicas o jogador poderá escolher a velocidade da música.*

25. *No modo de praticar músicas o jogador poderá escolher a seção da música.*

26. *Modo Exercises. No modo Exercises o jogador fará exercícios, modo ainda a ser melhor especificado.*

27. *Durante uma partida haverá um indicador de o quão afinado foi uma nota. Se a entrada for violino esse indicador efetivamente mostrará se a nota tocada foi mais grave ou aguda que a nota esperada. Se a entrada for joystick o indicador mostrará o qual sincronizado a nota foi tocada em relação a barra de sincronia.*

Os requisitos não-funcionais não foram formalizadas na etapa de especificação, mas basicamente consistem do tempo de resposta, pelo fato do jogo funcionar com captação de áudio em tempo real, além de usabilidade e bom design para a maior atratividade do jogo.

Capítulo 5

METODOLOGIA

5.1 Foco do PSI

Grande parte da metodologia inicial desenvolvida pelo aluno do PSI foi detalhada no Capítulo 3. Esta buscou resolver o problema da estimação da frequência fundamental do som do violino, chegando à conclusão de qual método utilizar. Com esta informação passou-se então para a fase avançada, que foi a integração com os alunos do PCS e aprimoramento do algoritmo. Para realizar este aprimoramento, foi considerado a plataforma utilizada pelos alunos do PCS, a linguagem Java no caso. Primeiramente testamos e entendemos as funções de análise de sinais já existentes na plataforma escolhida. Estando com os métodos de análise de sinais prontos nesta linguagem, implementamos o algoritmo e o combinamos com o resto do jogo criado pelo grupo do PCS.

5.2 Foco do PCS

Inicialmente foi construída uma especificação simplificada do projeto com base em conhecimentos prévios acerca de jogos. Focamos nos estudos de viabilidade dos requisitos principais do projeto, a captação de áudio, o tempo de resposta do jogo e a renderização de jogo. Fazendo testes com alguns algoritmos, como o algoritmo de FFT, tomamos a decisão de utilizar a linguagem Java, devido à familiaridade dos integrantes. Além disso, a decisão de utilizar um violino elétrico ao invés do acústico para simplificar a análise do áudio, visto que o acústico exige maiores complexidades de análise. Adotou-se a visualização do jogo em 2D, porém a renderização em 3D, para facilitação de futuras animações. Tendo a especificação básica, escolhemos a biblioteca OpenGL por meio do Processing para a renderização, além da biblioteca de Java Sound para a captação do áudio. Para tanto, foram necessários estudos e testes com tais bibliotecas para a familiarização. Dentre algumas extensões de áudio, escolhemos focar em MIDI para ser utilizado como o áudio da música, além de ser mais fácil de obter a partitura a partir dela. Na segunda

etapa de especificação do projeto, decidimos pela utilização do Latex para a redação da monografia, o que exigiu algum tempo para compreendê-la. Porém a sua utilização se mostrou bastante vantajosa principalmente por formatar alguns itens importantes com relativa facilidade, como as referências bibliográficas e os índices, sem exigir esforço maior para ajustes. Sendo assim, definimos os requisitos funcionais que o projeto deveria contemplar. Nesse interim, foi necessário bastante tempo de discussão e pesquisa a respeito de boas práticas (da área de desenvolvimento de jogos), a fim de se produzir uma especificação que se adequasse à expectativa final do produto a ser desenvolvido. Para melhorar a visualização, definimos uma arquitetura simplificada, num nível de abstração da interface humana, para melhorar os aspectos de modularização e controle de andamento do projeto. Como alguns integrantes do grupo jamais haviam tido contato (ou muito pouco) com o jogo Guitar Hero®, achamos necessário fazer uma rodada de apresentação do jogo, testando e observando suas funcionalidades. Isso contribuiu para refinar os requisitos do projeto e também aumentou o entusiamo pelo mesmo. Além disso, testamos os jogos semelhantes implementados para usar o teclado do computador como interface, no caso o Freats On Fire e o Performous. Além disso, existia possibilidade de olhar o código para tomar como referência. Uma outra referência utilizada foi um artigo sobre o Rock Band, que trouxe noções de como projetar melhor a interface de usuário tanto para o violino quanto para o joystick considerando mais de um nível de dificuldade. Seguindo o cronograma, criamos o projeto no Eclipse, colocando-o no repositório do site Xp-Dev, a fim de sincronizar o projeto entre nós por meio do Subversion (controle de versão com SVN). Já antes de iniciar as implementação adicionamos a biblioteca da ferramenta Processing, baseada em Java OpenGL, iniciando as atividades de prototipação das telas e de alguma implementação inicial. Acrescentamos as bibliotecas JInput e JXInput para usar o joystick.

Capítulo 6

IMPLEMENTAÇÃO

6.1 Protótipos

A fim de facilitar o desenvolvimento posterior da interface, foram feitos os protótipos das telas do jogo (figuras 6.1, 6.2 e 6.3). Esses protótipos servem para a melhor visualização da lógica de jogo, facilitando decisões que tenham de ser feitas em etapas anteriores da implementação do jogo e percepção de requisitos não-vistos a tempo de correções ou adaptações cabíveis.

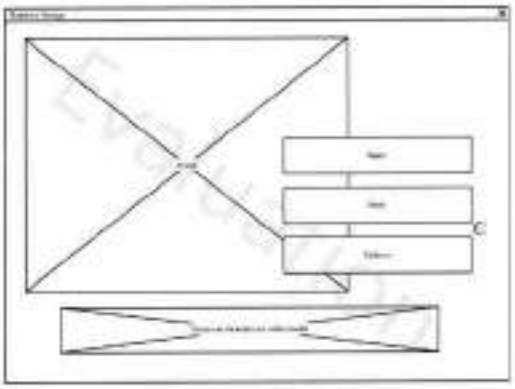

Figura 6.1: Menu Principal

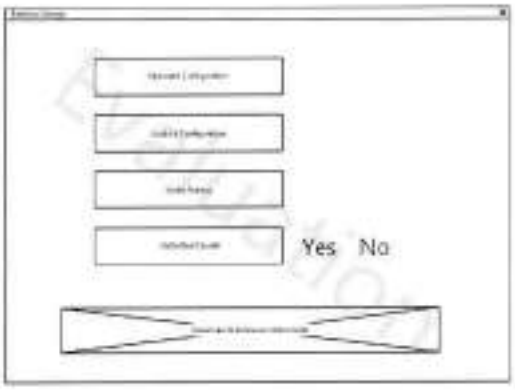

Figura 6.2: Protótipo da tela de opções: Options

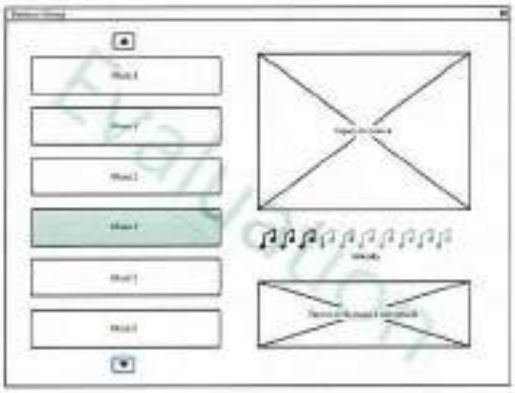

Figura 6.3: Protótipo da tela de escolha de músicas

6.2 Implementação do jogo

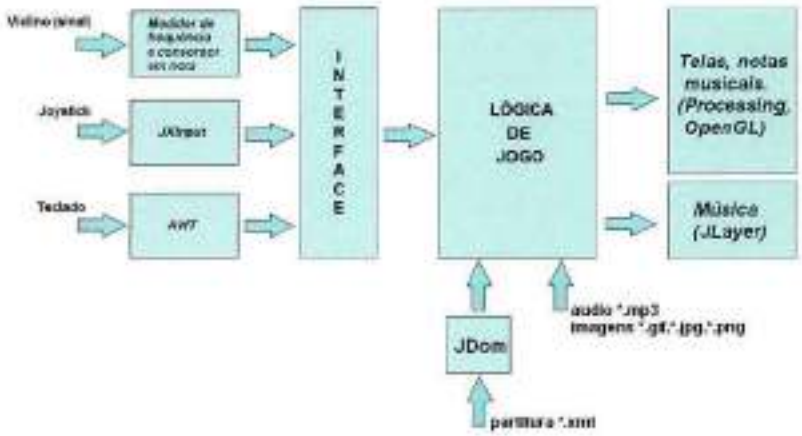

Figura 6.4: Diagrama de entradas do jogo

O diagrama da Figura 6.4 ilustra o funcionamento do jogo em alto nível de abstração. O Violin Villain possui três tipos de entrada: violino, joystick e teclado. Para a entrada por violino foi desenvolvido como parte do projeto o medidor de frequência e o conversor frequência nota. Para a entrada via joystick foi utilizada a biblioteca JXInput. Finalmente, para a entrada via teclado o próprio JDK foi utilizado, mais especificamente Abstract Window Toolkit (AWT). Os dispositivos de entrada então criados são conectados à lógica do jogo por uma interface que permite tanto o tratamento da entrada através de eventos nas telas de menu (joystick e teclado apenas) quanto verificação procedural do estado de cada tipo de entrada para as tela de jogo. Além disso, o jogo tem dados de entrada como

imagens utilizadas na renderização das telas e a própria música. Em especial deve-se comentar o arquivo xml de partitura que contém as informações sobre as músicas do ponto de vista do jogo, como a dificuldade da música e principalmente o instante e duração de cada nota musical. Vale ressaltar que existe um arquivo de partitura para cada dificuldade da mesma música. O arquivo de partitura é interpretado a partir de um parser criado com a biblioteca JDom e os dados obtidos são introduzido à lógica de jogo. Finalmente, através das informações de entrada e da lógica de jogo, pode-se renderizar as telas com a biblioteca Processing e com os componentes criados a partir da biblioteca ControlP5 e tocar as músicas com o player, a partir da biblioteca JLayer.

6.2.1 Detalhamento do software do Jogo

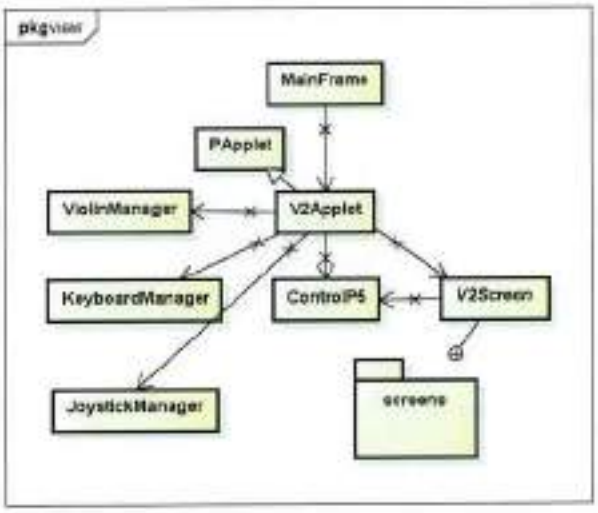

Figura 6.5: Estrutura de inputs e visualização

Como descrito na Figura 6.5, o jogo é renderizado em um applet java, mais especificamente na classe V2Applet que é filha da classe PApplet (biblioteca processing Processing). Esse applet está contido em uma JFrame (classe MainFrame) e por sua vez carrega as telas filhas da classe abstrata V2Screen com o auxilio dos componentes criados. As interfaces dos dispositivos de entrada são as classes ViolinManager, KeyboardManager, JoystickManager dos dispositivos violino, teclado e joystick respectivamente.

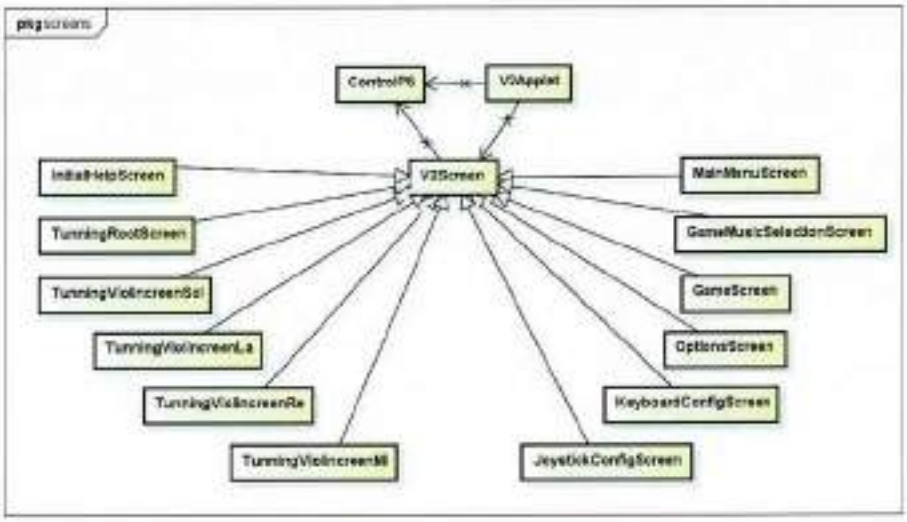

Figura 6.6: Estrutura de telas

A estrutura de telas pode ser vista pelo diagrama da Figura 6.6. Todas as telas são filhas da classe abstrata V2Screen. InitialHelpScreen é a tela de informações iniciais do jogo. As telas de Tunning são as telas de afinação do violino. MainMenuScreen é a tela de menu principal. Também está presente a tela de seleção de música, de opções, de configuração de teclado e joystick e finalmente a tela de jogo.

Os componentes utilizados no projeto foram criados com o auxilio das bibliotecas Processing e ControlP5, ilustrados na Figura 6.7. Alguns componentes foram transformados em componentes ControlP5 extendendo da Classe Controller outros são combinações de componentes criados ou componentes do próprio ControlP5. Como componentes originários do ControlP5 temos: TextLabel, Button e ControlFont (tipo de font, tamanho e cor). Já PImage é uma classe do Processing que representa uma imagem. CenterTextLabel é uma label com o texto centralizado. MenuButton é um botão com uma label no meio. MusicDifficultMeter é o medidor de dificuldade da música. ImageLabel é uma label com imagem. LifeBar é o medidor de vida no jogo. SlindingNote é o componente que contém a nota que desliza. Carousel é o seletor de músicas. E finalmente MultilineTextLabel é uma label com mais de uma linha.

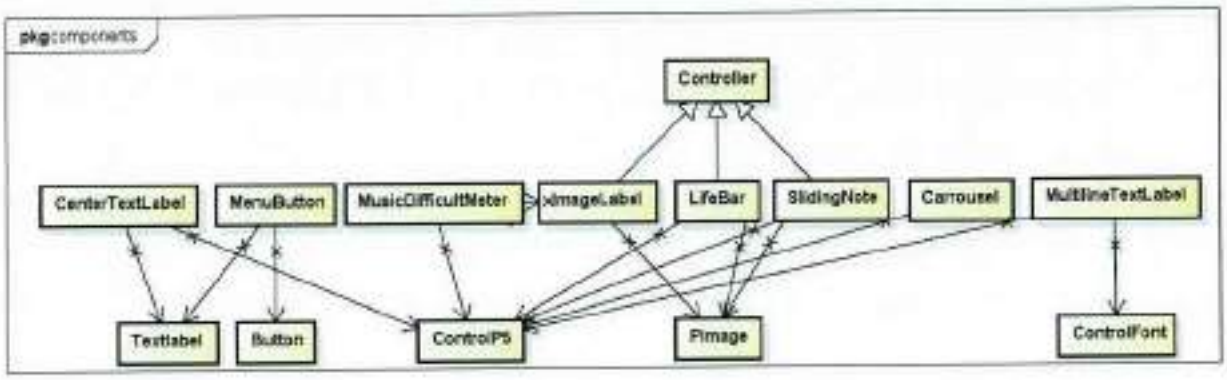

Figura 6.7: Estrutura de componentes

A leitura das partituras já em formato .xml é feita como descrito na figura 6.8. A classe MusicaParser usa a classe de Leitura implementada com o uso da biblioteca JDom para fazer o parser do xml e criar uma classe que representa a partitura da música (classe PartituraStructure).

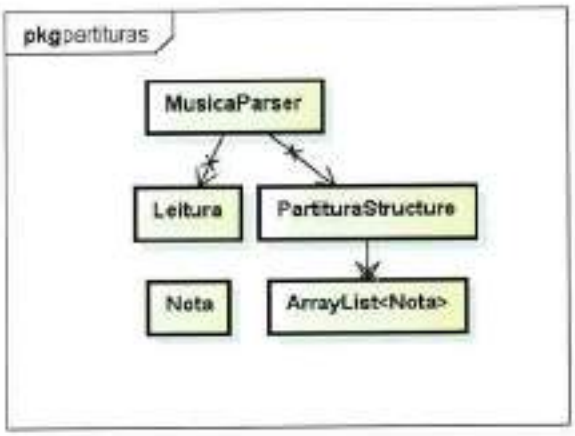

Figura 6.8: Estrutura do parser de partitura

6.3 Medidor de frequência

Como uma classe pertencente ao projeto, foi implementado um medidor de frequência que utiliza o algoritmo Fast Fourier Transform (FFT) e detecção de pico local do espectro. O método carrega no buffer N amostras do sinal captado pela entrada de microfone do computador, com uma frequência de amostragem (f_a) específica. Então, esses N pontos são

passados para um vetor no qual se realiza a FFT através do método fft disponível na biblioteca *commonsmath*. O método devolve um vetor de números complexos que representa o espectro do sinal. Em seguida, calcula-se o módulo desse espectro e encontra-se o máximo desse módulo que esteja abaixo de 1.5 vezes a frequência fundamental esperada, para evitar que seja identificado um harmônico ao invés da fundamental. Esse máximo e seus dois vizinhos são interpolados por uma parábola ($y = a + b * x + c * x^2$), utilizando funções de resolução de sistemas de equações lineares. Em seguida encontra-se o máximo desta parábola, que é $-b/2c$. Devolve-se o índice desse máximo, que ao ser multiplicado por (f_a) e dividido por N corresponde à frequência do sinal. O medidor de frequência esta ligado ao jogo como uma thread que utiliza a fft da biblioteca CommonsMath. O medidor por sua vez é utilizado pelo ViolinManager, como ilustrado na Figura 6.9.

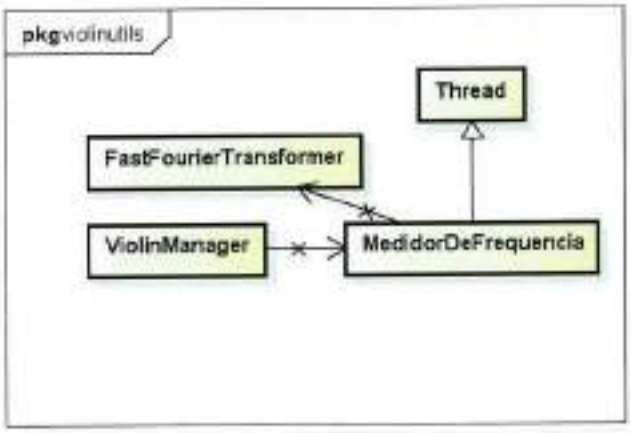

Figura 6.9: Estrutura do medidor de frequência

6.4 Sequência de Desenvolvimento do Código

Como ocorreram imprevistos e mudanças nos pré-requisitos durante a realização do projeto, o código implementado não seguiu rigidamente o cronograma ou metodologia especificada. Abaixo, estão listadas as tarefas realizadas em ordem cronológica.

- Port do Processing para o eclipse. Utilização da biblioteca gráfica Processing no ambiente de desenvolvimento Eclipse.
- Criação dos protótipos das telas iniciais com o WireframeSketcher.

- Incorporação da biblioteca Jakarta CommonsMath, que possui funções matemáticas avançadas (fft).
- Mais protótipos de tela: menu inicial, tela de opções e de seleção de música.
- Criação de componentes de UI(User Interface) posteriormente descartados.
- Versão inicial do Medidor de Frequência com o uso do CommonsMath.
- Alteração do modo de renderização do Processing para OpenGL e adição das bibliotecas do JOGL e OpenGL.
- Reestruturação da UI com o uso da biblioteca ControlP5. Componentes ImageLabel (imagem na tela), MultilineLabel (Label com mais de uma linha) e MenuButton (botão de menu).
- Tela de menu principal.
- Tela inicial de ajuda.
- Tela raiz de afinação de Violino (somente para a escolha de uso do violino ou não).
- Medidor de frequência com entrada pelo microfone.
- Tela de configuração de botões de teclado.
- Adição da biblioteca JXInput (para entrada por joystick).
- Melhorias na entrada por Joystick e entrada por teclado completa.
- Tela de configuração de botões de joystick.
- Atualização do medidor de frequência para resultado em Hz.
- Primeira mudança de imagens do projeto.
- Protótipo da Tela de Música (tela de jogo).
- Protótipo da Tela de Música. Easy Mode , Normal Mode e Tela Anterior as Músicas.
- Adição da biblioteca JDom (para leitura de xml).

- Leitura de arquivo xml.
- Implementação da conversão de arquivo xml de partitura para objeto java.
- Tela de seleção de músicas sem botões estáticos, componente MusicDifficultMeter (medidor de dificuldade de uma música).
- Componente Carrousel para botões dinâmicos na tela de seleção de músicas.
- Correção do componente Carrousel para situações de menos de sete músicas.
- Medição de desempenho do medidor de frequência.
- Alterações para melhoria de desempenho do medidor de frequência.
- Medidor de frequência melhorado com possibilidade de receber a frequência esperada pelo jogo.
- Arquivo xml de partitura da música Ode To Joy.
- Tela de seleção com musicas, descrição e imagens reais.
- Arquivo xml de partitura da música brilha brilha estrelinha.
- Conversor de frequência para número de nota.
- Tela de seleção de músicas completa.
- Adição da biblioteca JLayer (player de mp3).
- Componente player de mp3.
- Início da ferramenta de criação de xml de partitura a partir da música mid.
- Componente SlidingNote (Nota que anda na tela).
- Conexão do medidor de frequência com o jogo.
- Verificação de acerto ou erro de nota para violino.
- Telas de afinação de violino.

- Segunda mudança de imagens do projeto.
- Apresentação do projeto no SBGames 2010.
- Compenente LifeBar(medidor de vida).
- Melhoria de precisão do medidor de frequência.
- Imagens dos diferentes tipos de notas.
- Conversor nota frequência.
- Inclusão de velocidade de música na partitura.
- Melhoria da tela de seleção de músicas.
- Passagem da frequência de afinação para o conversor.
- Tela de jogo (easy mode).
- Novo modo de jogo (bow mode).
- Medium mode.
- Afinação salva em arquivo.
- Real mode.

6.5 Aspectos técnicos

Aqui se falará sobre os aspectos técnicos envolvidos no desenvolvimento do software.

6.5.1 Amostragem de sinal sonoro

Como mencionado diversas vezes ao longo do texto, a transformação de um sinal sonoro (analógico) em um sinal digital representa o diferencial do jogo desenvolvido no projeto, pois, diferente de jogos com simuladores de instrumentos, este permite a utilização de um instrumento real.

Numa abordagem mais computacional, a medição de frequência em si não é o que de fato

interessa para a lógica do jogo, uma vez que numa partitura musical não vem informações de frequência, numericamente falando. O que interessa para as comparações de acerto ou erro são as detecções de qual nota está sendo tocada pelo instrumento, que pode estar produzindo um som com variação tanto devido às vibrações físicas quanto devido ao posicionamento não-perfeito dos dedos. Por exemplo, caso o jogador olhe na tela do jogo uma nota Si (da corda La) para ser tocada, ele posicionará o dedo na segunda posição (uns 2cm da ponta) e deslizará o arco para produzir o som. No entanto, o som produzido (e, no caso, detectado pelo jogo) não será exatamente de 493.883 Hz, mas sim uma variação em torno desse valor. Sendo assim, foi implementada a classe 'ConversorNotaFreq', que, para ser instanciada, recebe como parâmetros as 4 frequências das cordas soltas calibradas (próximas mas não idênticas aos valores teóricos) e com isso transforma qualquer valor das frequências de um certo intervalo para valores discretizados de notas, segundo a classificação MIDI (55 a 96, vide Tabela 6.1), representando os semitons aceitos no jogo. Esse conversor garante que não haja valores de frequências desse intervalo que não tenham uma correspondente numeração MIDI.

Quanto ao medidor de frequência, uma das principais preocupações durante as especificações do projeto, como já mencionado, era que a obtenção do valor de frequência instantânea pudesse estar bastante atrasada, ao ponto de dessincronizar grosseiramente com a visualização de notas na tela de jogo, causando por exemplo a detecção de erro da batida quando na verdade o jogador tocou no momento certo e a nota certa. Para tanto, a classe de medição foi implementada de modo a funcionar paralelamente aos mecanismos do jogo, sendo ela uma *thread* criada no momento em que a *Applet* do jogo é instanciada. Dessa forma, em qualquer momento e em qualquer tela pode-se fazer uma medição do valor de frequência (vide as telas de afinação e a tela principal do jogo) e essa medição não interfere significativamente no processamento gráfico e lógico que a tela de jogo exige.

6.5.2 Formato de áudio

Em se tratando de um jogo, é comum que se tenha uma música acompanhando o ritmo do jogo, com os objetivos e desafios propostos ao jogador. Ainda mais, sendo um jogo musical, é imprescindível haver as músicas que correspondam à partitura que estará sendo tocada pelo jogador. Sendo assim, a inclusão de arquivos de áudio foi uma preocupação do projeto a partir de um certo ponto do desenvolvimento, uma vez que seria necessária o tratamento da lógica de acerto ou erro do jogador por meio de dados de referência, exatamente correspondente ao áudio tocado.

Foi necessária a decisão do formato de áudio a ser utilizado no jogo. A pesquisa de jogos semelhantes (Performous, Freats On Fire) mostrou algumas possibilidades de áudio, como o '.ogg' ([13],[14]), '.mod' ([15],[16],[17]) e o '.mid' ([18],[19],[20]). Além disso, alguns requisitos de projeto no subsistema referente a carregamento foram inspirados nesses. Como exemplo, existe uma estrutura de arquivos dentro do jogo de Freats On Fire para permitir o tal carregamento. Há um arquivo '.ogg', que representa uma música em si, e outro '.ogg', que representa somente o som da guitarra, permitindo que o instrumento tenha um destaque no jogo e possivelmente seja cortado quando o jogador erra. Há também um arquivo '.png' para efeito de atração visual de catálogo, um '.ini' contendo a descrição básica da música, e por último, não menos importante, o arquivo '.mid' correspondente a uma sequência de notas, contendo apenas as informações do que deve ser tocado (o correspondente à nossa partitura).

Tendo em vista essas informações, decidimos por utilizar diretamente arquivos de áudio já no formato '.mid' (ou MIDI), ainda que com qualidade inferior. O fato de ser mais difícil encontrar músicas de alta qualidade e com "arquivos de partitura", a fim de extrairmos uma sequência de notas correspondente à música para a lógica do jogo, justificou tal escolha que fizemos. Para melhorar a qualidade do áudio, utilizamos de um conversor de áudio (EaseMIDIConverter) para obter uma música no formato '.mp3' , com um som ligeiramente mais agradável de se ouvir.

Tabela 6.1: Nota, frequência e valor no protocolo MIDI

Nota	Frequência	MIDI		Nota	Frequência	MIDI
G 3	195.997718	55		E 5	659.255114	76
G♯3	207.652349	56		F 5	698.456463	77
A 3	220.000000	57		F♯5	739.988845	78
A♯3	233.081881	58		G 5	783.990872	79
B 3	246.941651	59		G♯5	830.609395	80
C 4	261.625565	60		A 5	880.000000	81
C♯4	277.182631	61		A♯5	932.327523	82
D 4	293.664768	62		B 5	987.766603	83
D♯4	311.126984	63		C 6	1046.502261	84
E 4	329.627557	64		C♯6	1108.730524	85
F 4	349.228231	65		D 6	1174.659072	86
F♯4	369.994423	66		D♯6	1244.507935	87
G 4	391.995436	67		E 6	1318.510228	88
G♯4	415.304698	68		F 6	1396.912926	89
A 4	440.000000	69		F♯6	1479.977691	90
A♯4	466.163762	70		G 6	1567.981744	91
B 4	493.883301	71		G♯6	1661.218790	92
C 5	523.251131	72		A 6	1760.000000	93
C♯5	554.365262	73		A♯6	1864.655046	94
D 5	587.329536	74		B 6	1975.533205	95
D♯5	622.253967	75		C 7	2093.004522	96

MIDI

O formato MIDI ([21],[22],[23]) possui como características principais a presença de dados que definem a música como se fosse um arranjo de instrumentos e notas por eles tocados, sequências de eventos temporizados que podem ou não alterar as velocidades de cada canal (instrumento) e uma sintaxe extensível. A sua criação visa permitir uma padronização de diferentes usuários e fabricantes de instrumentos musicais, uma vez que arquivos de áudio sintetizados por meios eletrônicos em geral são volumosos, diminuindo sua usabilidade.

Dito isso, o que o tornou atrativo para o uso no projeto foi o fato de poder ser usado como uma fonte tangível de informação de partitura, ou mais especificamente, notas, como mencionado acima. A partir dos arquivos '.mid' seria possível extrair apenas um canal do instrumento de violino, e pelas informações de temporização, obter a correta sequência para comparação de acerto ou erro do jogador. No protocolo MIDI cada nota tem um valor entre 0 e 127, as notas usadas pelo violino estão mostradas na Tabela 6.1.

Entretanto, uma dificuldade encontrada no seu uso foi o fato de se ter por definição uma série de *chunks*, "pedaços" (mesma idéia de pacotes), que permitem uma variada quantidade de informações, classificadas pelo seu respectivo cabeçalho. Isso dificulta a filtragem de dados para obter as notas de violino, pois as inúmeras meta-informações precisariam ser completamente entendidas para garantir que as notas filtradas estão de fato no instante e na velocidade certa dentro da música.

Para facilitar o nosso trabalho, encontramos disponíveis alguns conversores de MIDI para XML, ou ainda, XMID ([24]), um formato de XML que corresponde exatamente a um arquivo MIDI, sendo eles intercambiáveis.

6.5.3 Partitura em arquivo

Conforme descrito acima, é necessário haver uma fonte de dados que permitam as comparações de acerto ou erro no jogo, sejam eles obtidos dinamicamente ou estaticamente. De início optamos por definir um formato XML próprio (Figura 6.10), utilizando-se dos atributos de instante, duração e nota (semi-tom na classificação MIDI, um inteiro de 55

a 96, vide Tabela 6.1).

Para o escopo do projeto esses atributos são o suficiente para definir cada nota. Além desses, para cada partitura há também os atributos de velocidade, que poderia ser variável devido a mudança de ritmo de compasso (períodos musicais), dificuldade (por haver vários níveis, sendo que cada nível corresponde a um arquivo de partitura) e posição de mão (como segurar o braço do violino), que poderia afetar nas numerações de notas (semitons).

A fim de facilitar a implementação, utilizou-se a biblioteca JDOM ([25]), bastante estável

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- TEM que ter 'propriedades' e 'notas'... apenas para estruturar melhor.
     As notas podem conter um tag qualquer (diferente de 'nota') -->
<raiz>
    <propriedades modo="EAST" dificuldade="02" posicao="01"
        velocidade="15.333333333333" />
    <notas>
        <nota numero="71" duracao="8" instante="0" />
        <nota numero="71" duracao="8" instante="8" />
        <nota numero="71" duracao="8" instante="16" />
        <nota numero="74" duracao="8" instante="24" />
        <nota numero="74" duracao="8" instante="32" />
        <nota numero="71" duracao="8" instante="40" />
        <nota numero="71" duracao="8" instante="48" />
        <nota numero="69" duracao="8" instante="56" />
        <nota numero="69" duracao="8" instante="64" />
        <nota numero="69" duracao="8" instante="72" />
```

Figura 6.10: Arquivo XML que representa a partitura da música

e eficiente, para fazer o *parsing* dos arquivos XML e obter uma representação de dados simples de usar (em outras palavras, um objeto de partitura, na figura 6.8). A grande vantagem de se utilizar tal biblioteca é o fato de que combina com o XML, razão pela qual se escolheu XML e não '.txt' ou qualquer outro formato. A edição de um arquivo XML é fácil e intuitiva, pois obedece a regras rígidas de formato. Dessa forma, os arquivos de partitura também têm uma fácil visualização e a edição dessas é trivial, uma vez tendo um arquivo de exemplo em mãos.

Entretanto, durante o desenvolvimento do projeto, percebeu-se que a utilização do formato do XML que definimos seria inviável para músicas comuns que contêm violino, mesmo os de média duração (4 minutos), uma vez que seria necessário editar uma a uma os arquivos de partitura para poder jogar com a música correspondente. Disso surgiu, por acaso, a possibilidade de utilizar o formato XMID ([24]), mencionado mais acima, como a entrada

de partitura. A transformação de um arquivo XMID ([24]), padronizado, no objeto-partitura do jogo seria bem mais fácil e racional do que editar cada XML (figura 6.10) ou utilizar diretamente o arquivo MIDI, com sequências de bits não-triviais de se decodificar.
A partir do objeto-partitura pode-se fazer facilmente as comparações de acerto ou erro, bem como a partir dele obter a nota que deve aparecer na tela de jogo e fazê-la deslocar até a linha de batida.

6.5.4 Processing

Processing é uma linguagem de programação que tem por objetivo o ensinamento do básico de computação gráfica. A linguagem é construída a partir da linguagem java e é integrada com a biblioteca JOGL (Java OpenGL), simplificando suas funcionalidades e criando algumas novas. No projeto o Processing não foi usado como uma linguagem de programação, mas como uma biblioteca para a fácil utilização do OpenGL. Como biblioteca, o Processing no projeto é resumido à classe Papplet, que é filha da classe java Applet. No ViolinVillain é representado pela classe V2Applet entre alguns componentes de tela que reusam componentes do Processing.

6.5.5 ControlP5

O ControlP5 é uma biblioteca de interface gráfica do usuário feita para Processing . A biblioteca implementa componentes básicos para criação de menus e permite a criação de novos componentes. No projeto o ControlP5 foi amplamente utilizado, sendo a base da maioria dos componentes criados tando na renderização das telas de menu, quanto da tela do jogo.

6.5.6 JXInput

O JXInput é uma biblioteca java para controle de interfaces como teclado , joysticks e gamepads. No projeto o JXinput foi utilizado para possibilitar o Joystick como interface do jogo. Como desafio técnico essa gerência é feita em uma *thread* separada e seus eventos são colocados em uma fila e coletados pelo *applet* Processing para não interferir na renderização do jogo (sistema produtor - consumidor).

Capítulo 7

TESTES E AVALIAÇÃO

7.1 Medidor de Frequências

Primeiramente a classe MedidorDeFrequencias foi testada separadamente do jogo. O Medidor ficava amostrando pontos da entrada de microfone, analisando sua frequência e escrevendo a frequência medida na tela continuamente. Com este teste, já foi possível verificar o correto cálculo da frequência, porém era percebido um atraso entre a mudança da nota tocada e a mudança da frequência medida. Foram incluídos medidores de tempo no meio do código para descobrir qual trecho estava tomando mais tempo. Verificamos então que a amostragem do sinal era muito mais demorada do que todos cálculos realizados posteriormente a essa. Essa constatação ajudou a definir qual frequência de amostragem e quantos pontos utilizar.

O primeiro teste da classe MedidorDeFrequencias junto com o jogo, foi na implementação da tela de afinação. Nesta tela, o jogador toca a nota natural da corda, isto é, toca a corda sem pressioná-la com nenhum dedo (corda solta). A frequência obtida é armazenada para ser utilizada como padrão de frequência correta pelo jogo. Esta frequência também é usada para verificar quais são as frequências de todas outras notas possíveis nesta corda através do cálculo:

$$F(i) = F_0 * 2^{i/12}, \quad i = 0, \ldots, 12, \tag{7.1}$$

F_0 corresponde à frequência da corda solta e i representa o número do semitons de distância entre F_0 e F_i. i vai até 12 porque na escala cromática existem 12 semitons dentro de uma oitava.

Nesta tela, o medidor de frequências recebe como frequência esperada o valor padrão da nota da corda solta, Tabela 7.1. Para evitar resultados errados, a frequência medida deve se repetir por 10 medições em sequência para ser considerada uma afinação válida. Neste teste, a classe do medidor se mostrou funcional em tempo real conforme esperado. A tela

Tabela 7.1: Corda (Nota) x Frequência Esperada

Corda	Frequência(Hz)
Mi	659.26
Lá	440.00
Ré	293.67
Sol	196.00

de afinação tem a aparência da Figura 7.1. Ela apresenta a imagem do violino, com a corda a ser afinada em realçe e a partitura com a nota a ser afinada. Esta partitura fica se repetindo até que o jogador toque uma frequência próxima da esperada no momento em que a nota se aproxima da barra vertical do lado esquerdo.

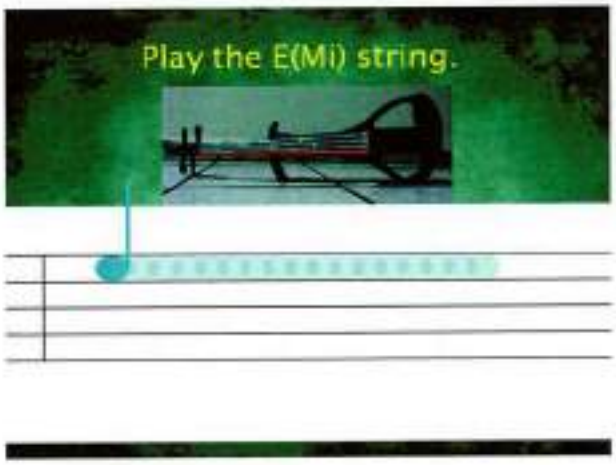

Figura 7.1: Tela de Afinação (Corda Mi)

7.2 Telas do Jogo

Utilizando as bibliotecas e classes descritas no Capítulo 6, foram geradas as telas do jogo. As funcionalidades do jogo foram testadas através destas telas. Primeiramente foi implementada a tela do menu principal, ilustrada na Figura 7.2. Esta já foi suficiente para testar o desenho da tela e a funcionalidade do teclado como controle.

Em seguida testou-se a configuração do teclado ou joystick na tela da Figura 7.3. Nesta também foi testada a transição entre telas e possíveis problemas de vazamento de memória.

Na tela de seleção de música, Figura 7.4 testou-se o tocador de mp3 e a leitura das pastas com arquivos de cada música.

Finalmente, na tela de jogo, Figura 7.5, testou-se a síncronia entre a música e a partitura e o processamento de desenhar muitas notas na tela ao mesmo tempo.

Figura 7.2: Tela de Menu Principal

Figura 7.3: Tela de ConFiguração de Teclado.

Figura 7.4: Tela de Seleção de Músicas.

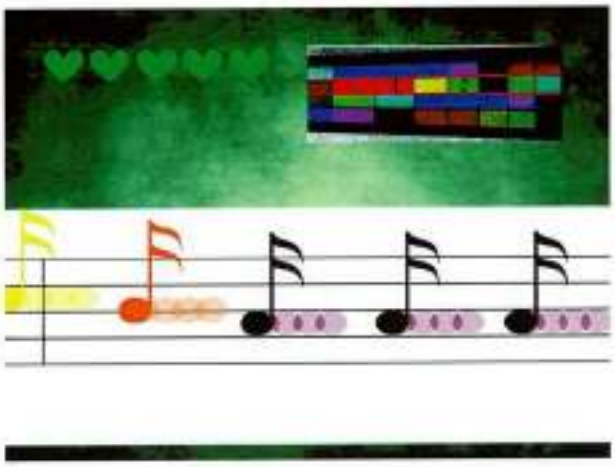

Figura 7.5: Tela Principal do Jogo.

Capítulo 8

CONSIDERAÇÕES FINAIS

8.1 Análise dos Resultados

Após os testes realizados com as telas do jogo já integradas com o medidor de frequências, concluímos que o jogo está funcionando de forma satisfatória. A plataforma escolhida (Java) se mostrou adequada para suportar o processamento do jogo, desde que haja o cuidado de se evitar vasamento de memória. O método de estimação de f_0 foi implementado de modo que a frequência é estimada rapidamente e com precisão razoável para níveis de audição humanos. Para obter estes resultados, foi preciso sincronizar as classes do jogo, otimizar parâmetros de análise de sinais e simplificar alguns dos requisitos planejados na disciplina Projeto de Formatura *I*.

No decorrer do projeto, durante a fase de implementação, ocorreram algumas alterações de escopo. Foram feitas algumas simplificações que removeram alguns requisitos funcionais ou modificaram-nos, porém sem retirar a proposta central do jogo. A exemplo disso, deixamos de separar o áudio do violino do áudio da música, uma vez que não encontramos com facilidade músicas dessa forma, e ainda com um MIDI associado para extrair a partitura. Outro motivo é que para o aprendizado do instrumento é fundamental ouvir a música para saber onde se está errando.

Foi acrescentado o modo "BOW" aos níveis de dificuldade. Neste modo o jogador toca apenas com cordas soltas. Isso, para um usuário leigo (sem prática de violino) já é um desafio, evidenciando maior necessidade de integrar ao projeto outros indivíduos que possuam uma visão mais próxima à do provável jogador.

8.2 Outras Considerações

Percebeu-se que muitas das necessidades do projeto foram supridas por bibliotecas já existentes. Foi aprendido na prática, já que um dos erros no desenvolvimento aconteceu

logo no começo do projeto, quando se tentou criar componentes de GUI sem conhecer bibliotecas já existentes. Mesmo que incompleta, a biblioteca ControlP5 passou a ser utilizada no projeto e os componentes necessários foram criados sobre ela.

O projeto, da forma como foi conduzido e no prazo dado, produziu um programa com funcionalidades demonstráveis e atingindo a expectativa inicial. Houve uma preocupação muito grande no tempo de resposta do medidor de frequência, o que acabou se mostrando bastante eficiente. Porém ao final do projeto percebeu-se que a renderização é que se tornou o gargalo de processamento, bem mais preocupante do que o medidor de frequência. Isso poderia ter sido contornado caso houvesse um esforço mais concentrado, desde a fase de especificação, em definir precisamente os requisitos e a arquitetura necessários para a alta eficiência da parte gráfica do jogo.

Depois de desenvolver o projeto e ver o resultado, percebemos que se tratando de um jogo o projeto ficou prejudicado por não ter um designer. A interface com usuário (parte gráfica) ficou muito pobre e o jogo ficou sem animações.

8.3 Trabalhos Futuros

Um objetivo que pode ser buscado por trabalhos posteriores é tornar o jogo mais atrativo e divertido. Pode-se desenvolver uma opção do jogador compor suas próprias músicas. Pode-se acrescentar um segundo jogador (modo Multiplayer), acrescentar músicas de diversos estilos, acrescentar exercícios mais eficazes para o aprendizado do instrumento, e melhorar a apresentação e interface do jogo. Além de aprimoramentos sobre o software desenvolvido, trabalhos futuros podem incluir estudos sobre o uso do jogo como ferramenta didática no aprendizado de violino.

No artigo enviado ao SBGAMES 2010 foi apresentada apenas a idéia do jogo. Agora, como houve uma evolução e o jogo foi finalizado, há a possibilidade de enviar um novo artigo ou mesmo o jogo em si.

Capítulo 9

TAREFAS ADICIONAIS

9.1 SBGames 2010

O Simpósio Brasileiro de Jogos e Entretenimento Digital de 2010 ocorreu de 08 a 10 de Novembro em FLorianópolis - SC. Nele, apresentamos um *short-paper* sobre o jogo em fase de desenvolvimento, já que na data de submissão ainda não existia nem um protótipo do jogo para permitir a participação do projeto como jogo independente. O artigo que foi publicado nos anais do simpósio está no Apêndice E.

9.2 Simplificação das Músicas

Para a criação das músicas nos modos mais fáceis, foi necessário criar versões simplificadas das músicas verdadeiras. Estas versões simplificadas só poderiam utilizar as notas reservadas para cada modo de jogo e também teriam que soar bem quando tocadas juntamente com a música original. A criação dos arquivos .mid das músicas simplificadas forão criados utilizando o programa Encore (versão5.0) que reproduz e edita partituras em formato MIDI.

9.3 Adesivo

O adesivo a ser colado no braço do violin para indicar ao jogador aonde colocar os dedos foi confeccionado pela artista plástica Silvia Aquilas. A Figura 9.1 mostra como ficou o adesivo. Conforme proposto, o adesivo tem 12 cores, uma para cada semitom. A relação de qual nota é representada por qual cor esta na Tabela 9.1.

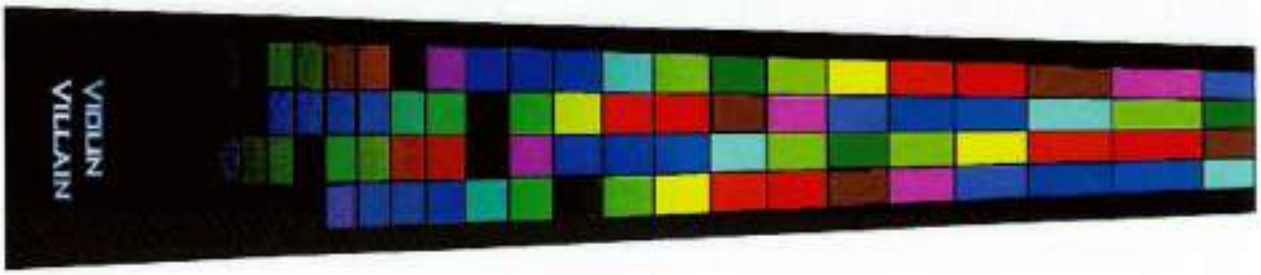

Figura 9.1: Adesivo para o braço do violino.

Tabela 9.1: Relação Cor x Semitom

Nota	Cor	R,G,B
Dó	Amarelo	(249, 226, 12)
Dó♯	Verde Claro	(127, 182, 27)
Ré	Verde Escuro	(34, 102, 39)
Ré♯	Verde Água	(81,178,129)
Mi	Azul Claro	(11, 182, 221)
Fá	Azul Escuro	(51, 71, 149)
Fá♯	Roxo	(83, 36, 127)
Sol	Violeta	(152, 79, 150)
Sol♯	Rosa	(229, 62, 140)
Lá	Vinho	(153, 17, 53)
Lá♯	Vermelho	(228, 43, 48)
Si	Laranja	(235, 110, 27)

9.4 Site do Projeto

Os domínios www.violinvillain.com e www.violinvillain.net foram comprados em 5/11/2010. Estes estão hospedados num servidor pago. Primeiramente o site apresenta apenas um arquivo .html com as mesmas informações do pôster apresentado no SBGames. Também há uma indicação de que o site esta em construção. Posteriormente o jogo será disponibilizado para download neste site.

Referências Bibliográficas

[1] DONOSO, J. P. A fisica do violino. In: . [S.l.]: IFSC - USP.

[2] BONA, P. *Método Completo de Divisão Musical*. [S.l.]: Ricordi, 2008.

[3] TANENBAUM, J.; BIZZOCCHI, J. Rock band: a case study in the design of embodied interface experience. In: *Proceedings of the ACM SIGGRAPH Symposium on Video Games*. New York, NY, USA: ACM, 2009. p. 127–134. ISBN 978-1-60558-514-7.

[4] KRAKOWSKI, S.; VELHO, L.; PACHET, F. Pandeiro funk: experiments on rhythm-based interaction. In: *Proceedings of SIGGRAPH '09*. New York, NY, USA: ACM, 2009.

[5] YIN, J.; WANG, Y.; HSU, D. Digital violin tutor: an integrated system for beginning violin learners. In: *Proceedings of the 13th Annual ACM International Conference on Multimedia*. New York, NY, USA: ACM, 2005. p. 976–985. ISBN 1-59593-044-2.

[6] WANG, Y.; ZHANG, B.; SCHLEUSING, O. Educational violin transcription by fusing multimedia streams. In: *Proceedings of the International Workshop on Educational Multimedia and Multimedia Education*. New York, NY, USA: ACM, 2007. p. 57–66. ISBN 978-1-59593-783-4.

[7] MAHER, R. C.; BEAUCHAMP, J. W. Fundamental frequency estimation of musical signals using a two-way mismatch procedure. *The Journal of the Acoustical Society of America*, ASA, v. 95, n. 4, p. 2254–2263, 1994.

[8] CHEVEIGNE, A. de; KAWAHARA, H. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, ASA, v. 111, n. 4, p. 1917–1930, 2002.

[9] RYYNÄNEN, M. Fundamental frequency estimation. In: . [S.l.]: ISMIR Graduate School, 2004.

[10] OPPENHEIM, W. *Signals and Systems*. [S.l.]: Prentice Hall, 1996.

[11] PERFORMOUS. *What is it?* Acesso em: 18 out. 2010. <http://performous.org/about.html>.

[12] SOURCEFORGE. *About the game.* Acesso em: 18 out. 2010.
<http://fretsonfire.sourceforge.net/about/>.

[13] XIPH.ORG. *Vorbis audio compression.* 2008. Acesso em: 04 dez. 2010.
<http://xiph.org/vorbis/>.

[14] OGG. Jul 2010. Acesso em: 04 dez. 2010.
<http://pt.wikipedia.org/wiki/Ogg>.

[15] MOD (file format). Acesso em: 04 dez. 2010.
<http://en.wikipedia.org/wiki/MOD (file format)>.

[16] JVC. *MOD file extension.* Acesso em: 04 dez. 2010.
<http://dotwhat.net/mod/2483/>.

[17] FILEINFO.COM. *.MOD File Extension.* Nov 2009. Acesso em: 04 dez. 2010.
<http://www.fileinfo.com/extension/mod>.

[18] FILEXT. *File Extension MID.* Acesso em: 04 dez. 2010.
<http://filext.com/file-extension/MID>.

[19] FILE-EXTENSIONS.ORG. *File extension MIDI.* Acesso em: 04 dez. 2010.
<http://www.file-extensions.org/midi-file-extension>.

[20] FILEINFO.COM. *.MID File Extension.* Feb 2010. Acesso em: 04 dez. 2010.
<http://www.fileinfo.com/extension/mod>.

[21] STANSIFER, R. *The MIDI File Format.* Feb 2005. Acesso em: 29 out. 2010.
<http://cs.fit.edu/ ryan/cse4051/projects/midi/midi.html>.

[22] MATTHEW. *The MIDI File Format.* 2005. Acesso em: 29 out. 2010.
<http://midi.mathewvp.com/aboutMidi.htm>.

[23] LAZZARO J. WAWRZYNEK, U. B. J. *RTP Payload Format for MIDI.* Jan 2006. Acesso em: 29 out. 2010. [small]
<http://tools.ietf.org/html/draft-ietf-avt-rtp-midi-format-15#page-37>.

[24] RECORDARE.COM. *MusicXML 2.0 Specification*. 2010. Acesso em: 18 out. 2010. <http://www.recordare.com/musicxml/specification>.

[25] JDOM.ORG. *JDOM*. Jul 2009. Acesso em: 18 out. 2010. <http://www.jdom.org/>.

[26] BIGHAM, J. *A super-sweet article*. May 2006. <http://violinovermelho.wordpress.com/violino/tamanhos/>.

[27] KLAPURI, M. A. et al. Probabilistic modelling of note events in the transcription of monophonic melodies. 2004.

[28] LEROY, N.; FLéTY, E.; BEVILACQUA, F. Reflective optical pickup for violin. In: *NIME '06: Proceedings of the 2006 conference on New interfaces for musical expression*. Paris, France, France: IRCAM — Centre Pompidou, 2006. p. 204–207. ISBN 2-84426-314-3.

[29] RABINER, L. R.; SCHAFER, R. W. *Digital Processing of Speech Signals*. [S.l.]: Prentice Hall, 1978.

[30] MITRA, S. K. *Digital Signal Processing: a computer-based approach*. 3rd. ed. [S.l.]: McGraw-Hill, 2006.

[31] RUIZ-REYES, N. et al. New speech/music discrimination approach based on fundamental frequency estimation. *Multimedia Tools Appl.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 41, n. 2, p. 253–286, 2009. ISSN 1380-7501.

[32] ZHOU, R. et al. A computationally efficient method for polyphonic pitch estimation. *Journal of Adv. Signal Process.*, Hindawi Publishing Corp., New York, NY, United States, v. 2009, p. 1–11, 2009. ISSN 1110-8657.

[33] PROCESSING. *Overview. A short introduction to the Processing software and projects from the community*. Acesso em: 20 set. 2010. <http://processing.org/about/>.

[34] SCHLEGEL, A. *About*. 2010. Acesso em: 20 set. 2010. <http://www.sojamo.de/libraries/controlP5/#about>.

[35] ORACLE. *Java at a Glance*. Acesso em: 05 dez. 2010.
<http://www.oracle.com/technetwork/java/javase/overview/index.html>

[36] HARDCODE. *JXInput - Input Devices for Java*. Acesso em: 28 set. 2010.
<http://www.hardcode.de/jxinput/>.

[37] OPENGL. *OpenGL Overview*. Acesso em: 05 dez. 2010.
<http://www.opengl.org/about/overview/>.

[38] ORACLE. *Abstract Window Toolkit (AWT)*. Acesso em: 05 dez. 2010.
<http://download.oracle.com/javase/1.5.0/docs/guide/awt/index.html>

[39] PUBLISHING, I. A. *Guitar Hero*. Acesso em: 25 nov. 2010.
<http://hub.guitarhero.com/>.

[40] FOUNDATION, T. E. *Eclipse Newcomers FAQ*. 2010. Acesso em: 25 set. 2010.
<http://www.eclipse.org/home/newcomers.php>.

[41] FOUNDATION, T. A. S. *Apache Subversion FAQ*. 2010. Acesso em: 05 dez. 2010.
<http://subversion.apache.org/faq.html>.

[42] SOLUTIONS, E. *XP-Dev.com Enterprise Subversion, Git & Mercurial Hosting for Peanuts*. Acesso em: 15 jul. 2010.
<http://www.xp-dev.com/>.

Apêndice A

Código Matlab com função espectograma

```
%%% Código que gera espectograma do sinal lido.
    Janela retângular ou hamming %%%

     close all
     clear all

     [x,fs,Nbits]=wavread('TwinkleStar.wav');
         % Lê ammostras do arquivo TwinkleStar.wav
         % fs é a frequência de amostragem
     Nx=length(x);
     n=0:Nx-1;
     t=n/fs;

     figure(1), clf
     plot(t,x)
     xlabel('Tempo (segundos)')
     ylabel('x(t)')
     axis([0 t(end) min(x) max(x)])

     % Parâmetros
     Nb = 1024;                % Nb: tamanho do bloco
     janela = hamming(Nb);     % janela de Hamming de tamanho Nb
     %janela=boxcar(Nb);       % janela retangular de tamanho Nb
     Ns=Nb/4;                  % comprimento da sobreposição
     Nfft=1024;                % comprimento da fft
```

```
% Cálculo da STFT (Short-Time Fourier Transform)
[S,Freq,Tempo,P] = spectrogram(x,janela,Ns,Nfft,fs);

figure(2), clf
imagesc(Tempo,Freq,20*log10(abs(S)));

xlabel('Tempo (segundos)');
ylabel('Frequência (Hz)');
colormap('jet')
axis xy
title('Spectrogram'),
     %colorbar- barra de intensidade opcional
```

Apêndice B

Código Matlab de estimação de f_0 com FFT

```
% Código lê o sinal e pega trecho especificado por número de períodos
% Aplica transformada de fourier neste trecho e encontra o ponto de
% maior amplitude do espectro.
clear ; close all

fbuscada = 293.66; %Ré
  % fbuscada = 493.88; %Si
  % fbuscada = 440.00; %Lá
Tbuscado = 1/fbuscada;
margemf =  fbuscada/2;     %margem para desconsiderar o
                           %pico no zero da autocorrelação
N = 1024;

[Y,fa,NBITS]=wavread('reelec.wav');   figure(1); plot(Y)
  % pega um trecho regular e plota  %%trecho minimo de dois periodos

x = Y(1.55e5 : 1.55e5 + N-1 );
iii = 8;
n = 0 : iii*length(x)-1;
t = n/fa;
f = n*fa/(iii*length(x));
Nfft = iii*length(x);

tic
X = fft(x,Nfft);
[ampli,picoR] = max( X(1:3*margemf*Nfft/fa));
```

```
picoR*fa/Nfft
fmedida = picoR*fa/Nfft ;
Tmedido = 1/fmedida;
Erro = abs(fmedida - fbuscada);
ErroPercentual = Erro*100/fbuscada;
toc

fprintf('Tbuscado = %f Tmedido = %f \n',Tbuscado,Tmedido);
fprintf('fbuscada = %f fmedida = %f \n',fbuscada,fmedida);
fprintf('Erro = %f ErroPercentual = %f \n',Erro,ErroPercentual);
```

Apêndice C

Código Matlab de estimação de f_0 com Autocorrelação

```
% Código lê o sinal e pega trecho especificado por número de períodos
% Cálcula a autocorrelação deste trecho e encontra o pto máximo desta
% Este ponto indica qual o período do sinal, e portanto sua f_0.
clear all; close all

fbuscada = 293.66; %Ré
  % fbuscada = 493.88; %Si
  % fbuscada = 440.00; %Lá
Tbuscado = 1/fbuscada;
margemf =  fbuscada/2;     %margem para desconsiderar o
                           %pico no zero da autocorrelação
N = 1024;

[Y,fa,NBITS]=wavread('reelec.wav');   figure(1); plot(Y)
  % pega um trecho regular e plota  %%trecho minimo de dois periodos

x = Y(1.55e5 : 1.55e5 + N-1 );
n = 0 : length(x)-1;
t = n/fa;

tic
R = xcorr(x);
[ampli,picoR] = max( R (end/2+margemT*fa : end));
Tmedido = picoR/fa + margemT;
fmedida = 1/Tmedido;
```

```
Erro = abs(fmedida - fbuscada);
ErroPercentual = Erro*100/fbuscada;
toc

fprintf('Tbuscado = %f Tmedido = %f \n',Tbuscado,Tmedido);
fprintf('fbuscada = %f fmedida = %f \n',fbuscada,fmedida);
fprintf('Erro = %f ErroPercentual = %f \n',Erro,ErroPercentual);
```

Apêndice D

Código Matlab de estimação de f_0 com Cepstrum Pitch

```
% Código lê o sinal e pega trecho especificado por número de períodos
% Cálcula o periodo do sinal por meio do Ceptrum pitch.
clear all; close all

fbuscada = 293.66; %Ré
  % fbuscada = 493.88; %Si
  % fbuscada = 440.00; %Lá
Tbuscado = 1/fbuscada;
margemf =  fbuscada/2;     %margem para desconsiderar o
                           %pico no zero da autocorrelação
N = 1024;

[Y,fa,NBITS]=wavread('reelec.wav');   figure(1); plot(Y)
  % pega um trecho regular e plota  %%trecho minimo de dois periodos

x = Y(1.55e5 : 1.55e5 + N-1 );
iii = 1;
n = 0 : iii*length(x)-1;
t = n/fa;
f = n*fa/(iii*length(x));
Nfft = iii*length(x);

jjj = 2;
n2 = 0 : jjj*iii*length(x)-1;
t2 = n2/fa;
```

```
tic
b = fft(x,Nfft);
B = log(b);
R = ifft(B, Nfft*jjj);
[ampli,picoR] = max( R (margemT*fa +2 : 2*margemT*fa));
Tmedido = picoR/fa + floor(margemT*fa)/fa;
fmedida = 1/Tmedido;
Erro = abs(fmedida - fbuscada);
ErroPercentual = Erro*100/fbuscada;
toc

fprintf('Tbuscado = %f Tmedido = %f \n',Tbuscado,Tmedido);
fprintf('fbuscada = %f fmedida = %f \n',fbuscada,fmedida);
fprintf('Erro = %f ErroPercentual = %f \n',Erro,ErroPercentual);
```

Apêndice E

Short Paper apresentado no SBGames2010

Título do artigo:

Rainbow Strings: Jogo para aprendizado de violino com processamento de áudio

Autores:

Paulo R. R. Vianna Edson M. F. Mesquita Jonathan Cheng

Ricardo Nakamura Magno T. M. Silva

IX Símposio Brasileiro de Jogos e entretenimento Digital

08 a 10 de Novembro de 2010 | Florianópolis - SC

Rainbow Strings: Jogo para aprendizado de violino com processamento de áudio

Paulo R. R. Vianna*
Edson M. F. Mesquita
Jonathan Cheng
Ricardo Nakamura
Magno T. M. Silva
Escola Politécnica da Universidade de São Paulo†

Resumo

Neste artigo discute-se o desenvolvimento do jogo "Rainbow Strings", com finalidades lúdica e didática. Trata-se de um jogo para computadores que apresenta uma partitura musical simplificada, ao mesmo tempo em que o som de um violino tocado pelo jogador é captado e analisado. Erros cometidos pelo jogador são detectados pelo jogo, que indica também correções necessárias. Por se tratar de um trabalho em andamento, são apresentadas a especificação do jogo e formas de interação com o jogador. Os desafios provenientes da captação e análise do som do violino são discutidos, bem como as soluções que se pretende adotar no projeto.

Keywords:: jogos de música, método de aprendizado de violino, violin hero, análise de sinais em games

Author's Contact:

{paulo.rrv, edson.mesquita}@usp.br, jcheng.entrep@hotmail.com, ricardo.nakamura@poli.usp.br, magno@lps.usp.br

1 Introdução

Atualmente o mercado dos jogos relacionados com música está crescendo cada vez mais. Já existem jogos com Bongôs, Bateria, Guitarra, Microfone, Mesa de DJ (*disc jokey*), tapete de dança, jogos de educação musical e rítmica, etc. A maioria desses jogos entretém, divertem e transmitem noções de musicalidade, mas a distância entre os instrumentos musicais reais e os controles utilizados neles limitam seu potencial educacional. Este projeto tem o intuito de permitir o real aprendizado de um instrumento musical - neste caso, o violino - ao mesmo tempo em que se diverte com o jogo.

O aprendizado de violino é considerado difícil, entre outros motivos, pelo instrumento não possuir trastes indicando onde colocar os dedos e pela precisão necessária na colocação dos mesmos para obter notas afinadas. Após um tempo de estudo e dedicação considerável, o aluno decora a posição dos dedos e aprende a corrigir a afinação das notas apenas ouvindo. Com o sistema de captação de freqüência do jogo proposto neste artigo, o aluno poderá verificar a afinação das notas mesmo sendo um iniciante, já que o software indicará se a nota está mais grave ou aguda do que a esperada. Para que o aluno saiba onde por os dedos, pretende-se disponibilizar um adesivo para ser colado no braço do violino com pequenas faixas de sete cores (próximas às do arco-íris), uma para cada nota musical, trazendo então o nome do jogo.

*Paulo faz iniciação científica neste tema com bolsa PIBIC-CNPq (processo:127651/2010-1) sob orientação do prof. Magno.

†O jogo apresentado neste artigo decorre do trabalho de formatura dos autores Paulo, Edson e Jonathan, sob orientação dos profs. Ricardo e Magno, que será apresentado ao curso de Engenharia Elétrica da Escola Politécnica da USP no final de 2010.

2 Trabalhos Relacionados

Jogos de música funcionam com "notas", que não são notas de verdade, mas sim algum botão representando a nota como nos jogos *Guitar Hero*, *Rock Band* ou *Guitar Freaks*. Ou mesmo instruções, como por exemplo, bater palmas no *Donkey Konga* ou pisar em um tapete como no *Dance Dance Revolution* ou no *Pump It Up*. Essas "notas" caminham sobre uma esteira em direção a uma linha limite e quando chegam, é o momento correto do jogador tocar a nota ou realizar a instrução. Pretende-se seguir o mesmo princípio no jogo proposto, mas com a execução de notas reais ao invés de botões, permitindo o aprendizado do instrumento.

Além de jogos comerciais como os citados acima, já existem diversos softwares livres (*open source*) de sucesso como *Performous* e *Frets on Fire*. Esses serviram de motivação e como referência para a criação do jogo proposto.

Jogos musicais já foram alvo de estudo para saber a real eficácia do aprendizado de um instrumento [Tanenbaum and Bizzocchi 2009]. O jogo proposto neste artigo pretende-se dar mais ênfase no lado didático sem perder a diversão e a facilidade de controle do jogo. Pretende-se aproveitar o potencial dos computadores pessoais atuais ao invés de utilizar somente acionamentos de botões assim como em [Krakowski et al. 2009], em que comandos computacionais são acionados através de frases rítmicas.

Já foram realizados trabalhos com o intuito de se ensinar a tocar violino pelo computador (veja, por exemplo, [Yin et al. 2005] e [Wang et al. 2007]). Além de verificar a correta afinação do som tocado como no jogo proposto, esses os jogos existentes utilizavam vídeos ou *avatares* para ajudar a ensinar como segurar o instrumento e tocar corretamente.

3 Especificação do jogo

Rainbow Strings terá a mesma estrutura que muitos jogos de música. Primeiramente, haverá um conjunto de menus com diversas opções de escolha e em seguida será apresentada uma tela contendo a partitura da música escolhida. Terminada a música mostra-se o desempenho do jogador e volta-se ao menu. Para tornar o jogo atrativo mesmo para pessoas que não tenham um violino, será possível jogar com um *joystick* convencional ou mesmo o teclado. Nestes casos, serão utilizados botões ou teclas para representar as notas, do mesmo modo que em outros jogos musicais.

3.1 Menus e Modos de Jogo

Ao ser iniciado, Rainbow Strings pergunta qual a interface a ser usada (violino, teclado ou joystick). Em seguida, tem-se uma seção de menus como descrita no fluxograma da Figura 1. Primeiramente, o jogador escolhe entre o **Options** (afinar seu violino, ajustar a configuração do teclado ou joystick), **Study** (praticar) ou **Game** (apenas se divertir). Escolhendo **Game**, o jogador então escolherá qual música ele vai tocar e qual a dificuldade em que a música vai ser apresentada (*Easy Mode*, *Normal Mode*, *Real Mode*). O *Easy Mode* apresentará apenas as notas principais da partitura da música

e não se preocupará em seguir fielmente as distâncias tonais entre as notas. Este terá apenas quatro notas, todas na corda La (corda solta-La, primeiro dedo-Si, segundo dedo-Do e terceiro dedo-Re). O *Normal Mode* já terá uma relação mais próxima com a partitura real, mas ainda não trará nenhum trecho difícil. Este terá oito notas, quatro na corda La e quatro na corda Mi. O *Real Mode* apresentará a partitura completa e usa todas as notas do instrumento. Tanto o *Easy* quanto o *Normal Mode* não poderão ser jogados com violinos acústicos, já que nesses modos o som que o violino produz é diferente do som tocado pelo computador e isso atrapalha o jogador. Com o violino elétrico isso é possível já que o som produzido pelo instrumento sem amplificadores externos é muito baixo. Escolhendo **Study**, o jogador poderá optar entre músicas e exercícios. Além de escolher a dificuldade, o jogador também poderá escolher a velocidade em que a partitura será passada e se quer tocar a música inteira ou apenas um trecho da música.

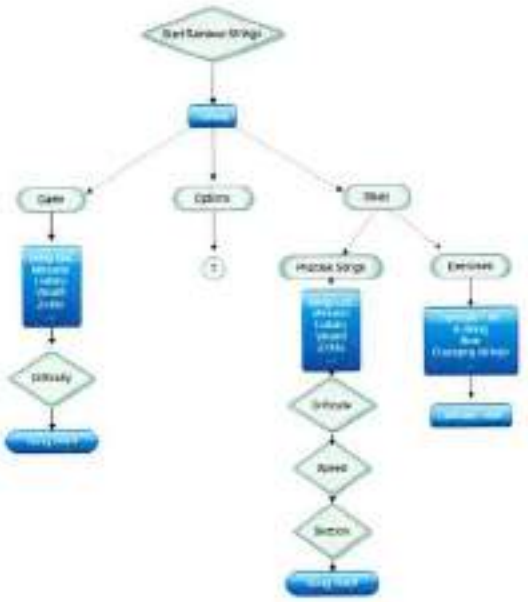

Figura 1: *Esquema de menus de Rainbow Strings.*

3.2 Proposta de Interface

Pretende-se que a tela do jogo no momento de se tocar as músicas seja apresentada como na Figura 2. Este protótipo foi desenhado para o modo **Easy**, que por ter apenas quatro notas, precisa de apenas quatro linhas na partitura. Para os outros níveis de dificuldade aparecerão cinco linhas, como numa partitura normal. O triângulo invertido acima da linha onde as notas devem ser tocadas indicará a afinação da nota. A seta para direita acenderá se a nota estiver grave demais e a seta para esquerda acenderá se a nota estiver aguda demais. Além disso, no triângulo haverá um ponteiro que indicará a mesma coisa, porém com uma precisão maior do que com as setas. No canto superior esquerdo haverá uma figura para indicar a colocação dos dedos (quais dedos deverão estar juntos e a distância entre eles). Isso é importante porque para cada escala, os dedos ficam em uma configuração específica. No centro da parte superior da tela, haverá um indicador de pontos que aumenta sua contagem de acordo com o desempenho do jogador na música escolhida. No canto superior esquerdo, haverá corações para indicar se o jogador está tocando bem ou mal, caso o jogador perca muitas notas em sequência os corações começarão a desaparecer. Caso sumam todos os corações, a música parará porque o jogador não conseguiu acompanhá-la adequadamente. Deve-se observar que a proposta atual de interface pode ser modificada durante a implementação e teste do jogo com usuários.

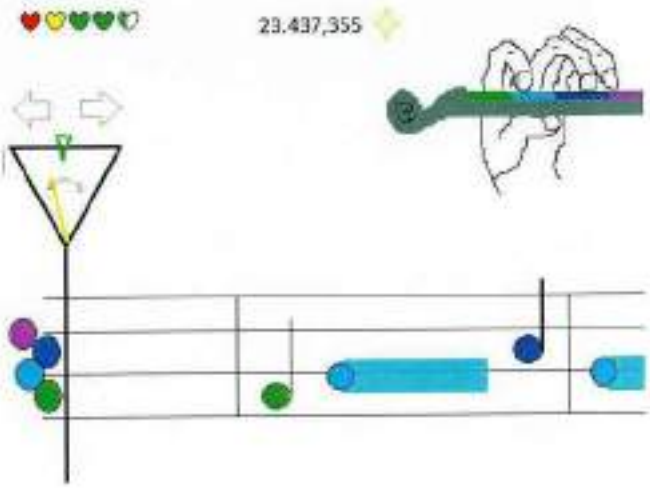

Figura 2: *Protótipo da tela do jogo*

3.3 Reprodução de Áudio

Assim como outros jogos semelhantes ao proposto neste artigo, como *Performous* e *Frets on Fire*, cada música será dividida em três arquivos: um arquivo de áudio com a música a ser tocada sem o som do violino, outro arquivo de áudio somente com o som do violino e por último um arquivo com as informações das notas que devem ser tocadas, informando o tipo da nota e o tempo em que elas devem ser tocadas. O jogador poderá escolher se quer que o som do seu violino seja tocado ou se o segundo arquivo de áudio seja usado como padrão. Assim, durante uma partida, quando a nota tocada estiver correta, o som do violino ou do arquivo de áudio com violino irá para a saída de som. Se a nota tocada estiver errada, sons de erro serão tocados. Se o jogador perder uma nota, o som de violino não será tocado.

3.4 Tecnologia

Para a implementação do projeto será utilizada a linguagem java. Quanto à captação dos sinais de áudio, o próprio pacote javax.sound do jdk será usado. Para a interface gráfica será usada a biblioteca OpenGL para Java, mais especificamente JOGL junto com a biblioteca Processing. Já o controle do jogo via Joystick será feito através da biblioteca JXinput, parte do projeto Distributed Realtime Simulation (http://sourceforge.net/projects/drts/).

3.5 Game Design para Aprendizado

A primeira dificuldade em se aprender a tocar violino é conseguir segurar o instrumento de forma confortável. O jogo proposto não verificará se o jogador está segurando o instrumento corretamente. No entanto, serão apresentadas fotos e instruções que descreverão como isso deve ser feito (mantenha o pulso esquerdo reto, segure o instrumento com o peso do queixo, não mover o antebraço para tocar, mantenha-se relaxado para tocar, etc). Outra dificuldade é a leitura de partituras musicais, onde cada símbolo (colcheia, semicolcheia, semínima, etc) representa uma duração e a posição da bolinha indica qual nota deve ser tocada. No jogo proposto, cada nota

será mantida com seu símbolo, porém será acrescentado um rastro na mesma para indicar sua duração (como no jogo Guitar Hero ou em outros jogos). Para saber qual a nota a ser tocada, as bolinhas terão a cor da nota que representam. Pretende-se colocar um adesivo no braço do violino para indicar onde colocar os dedos. No entanto, como o mesmo dedo pode ficar em mais de uma posição, será mostrado um desenho ou uma foto de como os dedos devem ser posicionados em cada música (no modo *Easy*, todas as músicas usarão a mesma posição para facilitar). Para jogadores que queiram se tornar alunos, o jogo terá uma seção de treino e aprendizado, onde o aluno poderá tocar exercícios de violino, necessários para o aprendizado do instrumento (exercícios de arco, escalas, mudança de corda, afinação, velocidade, etc).

4 Análise do Sinal do Violino

Utilizar um instrumento musical real como controle e ainda permitir que seja possível aprender jogando trazem novas necessidades para um jogo de música. Para ser mais interessante e melhorar o aprendizado de violino, Rainbow Strings precisa fazer uma medição quase imediata da frequência tocada. Para isso, será necessário captar o sinal de áudio do violino, utilizando a amostragem feita pela placa de som. Em seguida, deve-se fazer uma análise desse sinal para estimar sua frequência fundamental. Como o violino acústico produz um som muito rico em harmônicos e necessita de um microfone acoplado para captá-lo, será analisado o áudio de um violino elétrico numa etapa inicial. Neste caso, a captação do som é mais simples já que a saída de áudio do violino elétrico é igual a de microfones, o que elimina ruídos externos e evita a inconveniência de acoplar um microfone ao violino. Além disso, como o som próprio do violino elétrico é muito baixo, o jogador poderá tocar uma versão simplificada da música e ouvir o som da versão original, tocado pelo computador. O sinal do violino elétrico será captado pela entrada de microfone da placa de som de um computador convencional e será manuseado através da biblioteca de áudio da linguagem do software. Então, nessa mesma linguagem, será feita a análise do sinal e a frequência fundamental obtida será devolvida para o jogo, para que este a compare com a freqüência esperada.

Com o intuito de familiarizar-se com o tipo de sinal que será tratado foram feitas captações do sinal do violino e alguns testes com ferramentas matemáticas. Com o gravador de som do Windows (utilizando especificações: PCM 22,05 kHz; 16 bits; Mono) foram gravados desde sinais simples com apenas uma nota durante toda sua duração até sinais mais complexos com trechos de músicas. Estes sinais foram brevemente estudados, utilizando-se o software Matlab (versão 7.6.0.324) e suas funções de análise de sinais.

Devido à não estacionariedade de um sinal musical, é mais adequado usar a transformada de Fourier de curto prazo (STFT - short-time Fourier transform) para analisar seu espectro. Entretanto, um segmento de um sinal musical num curto intervalo de tempo pode ser considerado como um sinal estacionário e consequentemente a transformada de Fourier discreta desse segmento pode prover uma representação razoável do espectro nesse intervalo [Rabiner and Schafer 1978; Mitra 2006]. A função **spectrogram.m** do Matlab foi usada, assumindo uma frequência de amostragem de 22,05 kHz combinada com uma janela de Hamming e uma FFT (fast Fourier Transform) com comprimentos iguais a 256 e uma sobreposição de 50%. Com esses parâmetros foram gerados gráficos do espectrograma dos sinais. Na Figura 3, é mostrado o espectrograma de uma escala de Dó maior. Quanto mais vermelho, maior a amplitude do espectro de frequência naquele instante. A escala de Dó Maior, mostrada na Figura 4, começa com a nota Dó, passa por notas intermediárias, chega ao Dó que tem o dobro da frequência do Dó inicial e depois volta passando pelas mesmas notas. No caso do espectrograma da Figura 3, subiu-se na escala até o Dó que tem quatro vezes a frequência do Dó inicial.

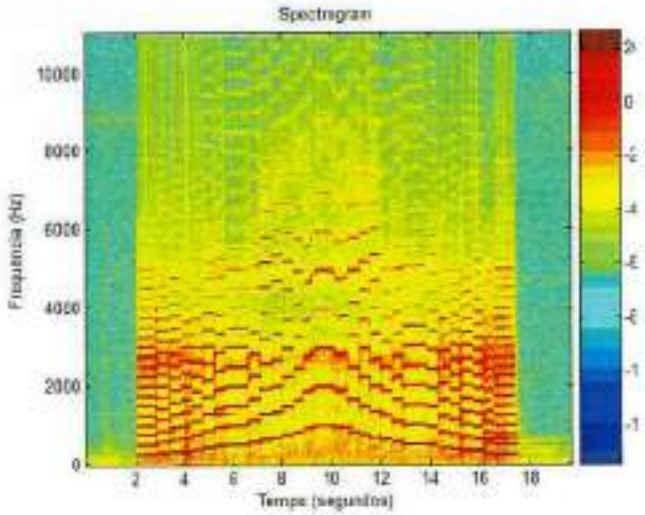

Figura 3: *Espectograma gerado pelo Matlab*

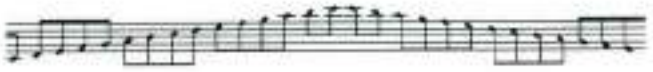

Figura 4: *Partitura Musical de Escala em Dó Maior*

4.1 Obtenção da frequência fundamental do som do violino

O problema de se estimar a frequência fundamental (f_0) de sinais musicais é bem conhecido e tem sido abordado por diversos métodos ao longo dos anos. Para ter uma base consolidada das ferramentas disponíveis, foi feito um estudo das soluções utilizadas em outros projetos, que abordaram problemas similares, como por exemplo, em [Ruiz-Reyes et al. 2009] e [Zhou et al. 2009].

No caso do jogo proposto, a frequência esperada é conhecida, mas é necessário obter sua estimativa quase instantâneamente. A informação da frequência esperada abre possibilidades de utilizar métodos novos e mais precisos do que os utilizados em outros estudos, porém a necessidade de realizar este processamento em tempo real restringe as possíveis soluções.

Devido ao conflito entre precisão e velocidade normalmente encontrado nas análises estudadas, optou-se por utilizar mais de um método em paralelo para inserir diversidade e garantir precisão nos resultados. No início de cada nota da partitura, pretende-se utilizar um método pouco preciso, porém bem rápido e que necessite de um trecho curto do sinal para funcionar (chamado de método rápido ou método 1). Este método é suficiente para indicar se a nota foi tocada no instante correto e se está próxima da esperada, mas não tem a precisão necessária para corrigir a afinação milimétrica do instrumento. Dessa forma, pretende-se utilizar o segundo método que começará a captar pontos no mesmo instante que o primeiro, mas utiliza um trecho maior do sinal (chamado de método lento ou método 2). Com um tempo de cálculo ligeiramente maior, o segundo método indicará a freqüência tocada com maior precisão, permitindo que o jogador possa corrigir erros finos na posição de seus dedos.

O método rápido será implementado utilizando a autocorrelação do sinal no tempo. Dada uma sequência $x(n)$ com N amostras,

a autocorrelação de $x(n)$ pode ser estimada como

$$r(\tau) = \frac{1}{N} \sum_{n=0}^{N-\tau-1} x(n)x(n-\tau), \quad (1)$$

Sabe-se que a autocorrelação é máxima em $\tau = 0$ e seu segundo máximo ocorre em $\tau = T_0 = 1/f_0$ no caso de sinais periódicos. Dessa forma, basta encontrar o segundo valor máximo para se estimar a frequência fundamental [Yin et al. 2005]. A autocorrelação ainda pode ser implementada de forma eficiente, utilizando-se o algoritmo FFT [Mitra 2006]. Para que o método funcione, deve-se garantir que N seja grande o suficiente para que $x(n)$ tenha pelo menos dois períodos do sinal. O segundo método será implementado utilizando-se o algoritmo FFT para o cálculo do espectro e verificando-se no espectro gerado a freqüência que tem pico de amplitude nos arredores da freqüência esperada.

5 Conclusão

Foram apresentadas a proposta e especificação de Rainbow Strings, um jogo musical que pode ser utilizado no aprendizado de violino. O uso de violinos reais como dispositivos de entrada para o jogo foi discutido, incluindo-se os desafios relacionados ao processamento do som produzido por tais instrumentos. Por se tratar de um trabalho em andamento, não existem ainda resultados conclusivos sobre as propostas apresentadas. Uma vez que o jogo tenha sido desenvolvido, pretende-se disponibilizá-lo na forma de software livre, incluindo-se uma página com instruções sobre como utilizar violinos acústicos e elétricos com o mesmo. Além de aprimoramentos sobre o software desenvolvido, trabalhos futuros podem incluir estudos sobre o uso do jogo como ferramenta didática no aprendizado de violino.

Referências

KRAKOWSKI, S., VELHO, L., AND PACHET, F. 2009. Pandeiro funk: experiments on rhythm-based interaction. In *Proceedings of SIGGRAPH '09*, ACM, New York, NY, USA.

MITRA, S. K. 2006. *Digital Signal Processing: a computer-based approach*, 3rd ed. McGraw-Hill.

RABINER, L. R., AND SCHAFER, R. W. 1978. *Digital Processing of Speech Signals*. Prentice Hall.

RUIZ-REYES, N., VERA-CANDEAS, P., MU NOZ, J. E., GARCÍA-GALÁN, S., AND CA NADAS, F. J. 2009. New speech/music discrimination approach based on fundamental frequency estimation. *Multimedia Tools Appl. 41*, 2, 253–286.

TANENBAUM, J., AND BIZZOCCHI, J. 2009. Rock band: a case study in the design of embodied interface experience. In *Proceedings of the ACM SIGGRAPH Symposium on Video Games*, ACM, New York, NY, USA, 127–134.

WANG, Y., ZHANG, B., AND SCHLEUSING, O. 2007. Educational violin transcription by fusing multimedia streams. In *Proceedings of the International Workshop on Educational Multimedia and Multimedia Education*, ACM, New York, NY, USA, 57–66.

YIN, J., WANG, Y., AND HSU, D. 2005. Digital violin tutor: an integrated system for beginning violin learners. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, ACM, New York, NY, USA, 976–985.

ZHOU, R., REISS, J. D., MATTAVELLI, M., AND ZOIA, G. 2009. A computationally efficient method for polyphonic pitch estimation. *Journal of Adv. Signal Process. 2009*, 1–11.