Igor Bragio Persona

DIAPASON

(*versão pré-defesa, compilada em 18 de dezembro de 2018*)

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Bruno Müller Junior.

Curitiba PR
2018

Resumo

Este trabalho tem como finalidade apresentar o Diapason, uma ferramenta de auxilio ao aprendizado de canto. Pela comparação da voz do usuário com o som esperado, o sistema é capaz de dar *feedbacks* em tempo real para que o cantor afine sua voz de acordo com diferença calculada entre esses dois sons.

Implementado em um formato parecido a um jogo, o sistema torna-se mais amigável e intuitivo para quem o usa. A ferramenta foi desenvolvida para web, para que o maior número de pessoas que queiram usufruir da praticidade que ela dá sejam alcançadas.

O Diapason pode ser encontrado no GitHub sobre a licença Creative Commons em https://github.com/igorbpersona/diapason. **Palavras-chave:** Diapason, afinação vocal, feedback de canto.

Lista de Figuras

Sumário

1 Introdução

Existe no ser humano um fascínio pelos sons que se juntam em harmonia e passam por nossos ouvidos de maneira ritmada, o que chamamos de música. A música é uma sequência de sons que juntos nos remetem aos mais diversos tipos de sentimentos.

Quando dois sons produzem a mesma frequência dizemos que eles estão afinados entre si, portanto é possível afinar um som tendo como base outro. Se pegarmos um violão e tocarmos uma nota qualquer, conseguimos afinar nossa voz a ponto que fique semelhante a nota tocada, com algumas limitações.

Antes da era tecnológica em que vivemos, onde existem afinadores digitais que podem ser baixados diretamente em um celular e usados para afinar instrumentos musicais, os instrumentos muitas vezes eram afinados conforme o som produzido por um outro instrumento, como o diapasão, um instrumento metálico criado em 1711 por John Shore, que tem o formato de uma forquilha, e quando é golpeado contra alguma superfície, suas duas extremidades vibram em uma frequência específica produzindo uma nota musical, este som produzido é usado para a comparar e afinar instrumentos musicais ou a voz de uma pessoa. [1]

Deste instrumento tomei a inspiração para o nome do sistema descrito neste documento, “Diapason” nome da ferramenta em inglês.

1.1 Objetivo

Busco com este sistema dar a facilidade para um aspirante a cantor treinar a afinação de sua voz com feedbacks visuais de tempo real. O diapason é para ser uma ferramenta de aprendizado de canto, tanto para aquecer a voz, como para cantar músicas pré cadastradas no tom correto.

A ideia é comparar o canto do usuário com o canto esperado naquele momento, e mostrar graficamente a diferença entre os dois com o passar do tempo.

Pretendo neste documento descrever o sistema e como ele foi implementado, mostrando os resultados obtidos, desafios enfrentados e por fim possíveis melhoramentos.

1.2 Funcionamento do aplicativo

O método de treinamento utilizado na ferramenta é visto semelhantemente em algumas plataformas de jogos, como na série Guitar Hero, ou em máquinas de dança de shoppings, e em alguns aplicativos como o Smule para Android e o Vanido para iOS, que tem uma ideia muito parecida com a do Diapason.

Funciona da seguinte forma: o usuário pode escolher um desafio dentre os cadastrados anteriormente no sistema para fazer o treinamento. Após a escolha do desafio o usuário é levado para a tela do mesmo, um quadro retangular que podemos pensar como sendo um gráfico tempo x afinação, a linha do tempo fica em constante movimento da direita para a esquerda, onde passam as notas do desafio, que representam as diferentes afinações. No quadro constam uma linha vertical fixa representando o momento em que a nota passante deverá ser cantada; um círculo que se desloca por esta linha vertical indicando a afinação da voz do usuário lida pelo microfone; e retângulos que representam as notas, tendo como sua altura do eixo y a propriedade que define a afinação desta nota.

O desafio consiste em cantar as notas passantes do começo ao fim na afinação correta e no tempo certo, isso é passado para o sistema gerenciador da pontuação do Diapason, onde o usuário pode ver ao final uma estatística que pode ajudá-lo a saber se está melhorando no desafio com o passar do tempo.

Nesta primeira versão do Diapason ele conta com dois desafios para aquecer e treinar a voz, e também com uma música demo de exemplo, Man in The Box da banda Alice in Chains. A escolha desta música se deve ao fato de que na sua introdução são apenas gritos de vogais sem palavras, o que permite ver com bastante clareza a distinção entre as notas cantadas.

1.3 Motivação

Os aplicativos mais parecidos com a minha ideia que achei disponíveis foram o Smule [2] apenas para Android e iOS, e o Vanido [3] que é apenas para iOS. Minha intenção sempre foi criar esse aplicativo para web para ser suportado nos mais variados dispositivos, e deixá-lo gratuíto com o código aberto para quem quiser implementar algo a partir dele, contribuir para o projeto, ou apenas entender o seu funcionamento e poder usar isso como conhecimento em qualquer tipo de projeto que isso possa vir a ser útil.

O objetivo principal é poder ajudar o maior número de pessoas que queiram aprender ou praticar o canto, e ter uma plataforma que possa ser aperfeiçoada por qualquer um que queira contribuir para ela.

1.4 Estrutura do documento

Este documento está separado em seis capítulos principais, este é o primeiro, onde é dada uma breve pincelada sobre o que é o projeto, e algumas motivações para sua implementação. O segundo onde são descritos e definidos os termos utilizados durante a leitura do documento, termos da área da física responsável por estudar sons, e termos musicais necessários para o bom entendimento do funcionamento do sistema. Já no terceiro capítulo dou um breve histórico mundial sobre o tema abordado, e os relaciono com o Diapason. O quarto capítulo é a que traz uma ideia geral sobre a implementação do projeto. Não será visto nele nada de código de fato, mas sim algumas técnicas e métodos que foram necessários para o bom funcionamento da ferramenta. Também neste capítulo é abordado um pouco do histórico de tentativas, sucessos e fracassos no decorrer do projeto. O quinto, onde as técnicas e métodos exemplificados no capítulo 3 são relacionadas a implementação de fato. É dada a descrição das classes, métodos, e funções utilizadas, sendo uma espécie de documentação do código gerado. Documentado neste mesmo capítulo estão alguns resultados obtidos com o funcionamento do sistema. Por último, temos o capítulo 6, o de conclusão do projeto, onde os resultados são analisados e transformados em informação relevante para eventuais melhorias futuras e possíveis erros do sistema.

As referências bibliográficas utilizadas como fundamentos do texto deste documento se encontram ao fim do mesmo.

2 Fundamentação teórica

Neste capítulo estão descritos os fundamentos para o bom entendimento do texto deste documento. Para tal, o capítulo está dividido em 4 seções, que descrevem em ordem, os termos da ondulatória, parte da física que estuda ondas, como é o caso do som (seção 2.1), os termos musicais necessários para completo entendimento do que será proposto (seção 2.2), a representação de uma onda mecânica em memória (seção 2.3), e algumas funções usadas para manipulação de áudio na ferramenta (seção 2.4).

2.1 Som e as propriedades de uma onda mecânica

Primeiramente precisa-se entender o que é som. Som é a propagação de uma frente de compressão mecânica ou onda mecânica, é uma onda longitudinal, que se propaga de forma circuncêntrica, apenas em meios materiais (que têm massa e elasticidade), como os sólidos, líquidos ou gasosos.[4]

Uma onda mecânica é uma perturbação oscilante de alguma grandeza física no espaço e pode ser periódica no tempo.[5]

As características mais relevantes que extraímos olhando para a representação bidimensional de uma onda estão representadas na figura 2.1:

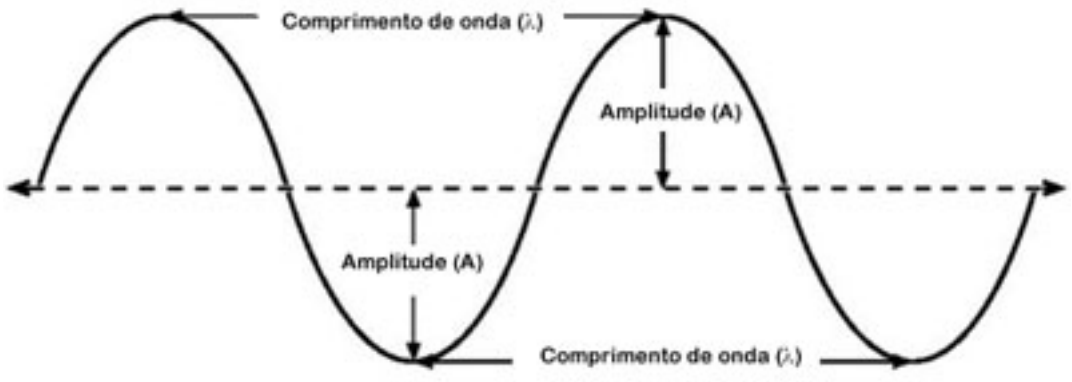

Figura 2.1: Representação bidimensional de uma onda mecânica.[19]

Comprimento de onda: Representado por lambda (λ), é a distância que separa dois pontos consecutivos que se encontram na mesma posição de vibração. Na figura apresentam-se

dois exemplos: a distância que vai de uma crista (ponto mais alto da onda) à outra e a distância que vai de um ventre (a posição mais baixa da onda) ao outro.

Frequência: A frequência (f) de uma onda representa o número de oscilações executadas pela fonte que produz a onda, em cada segundo. Se em 1 segundo a onda se repete 3 vezes então a frequência dela é de 3 Hz (Hertz).

Amplitude: A amplitude (A) representa o máximo afastamento, durante a oscilação, em relação à posição de equilíbrio. Na figura, a posição de equilíbrio está representada pela linha tracejada.

Período: O período (T) representa o intervalo de tempo correspondente a uma oscilação completa da fonte que produz a onda. T =1/f.

2.2 Termos musicais

Entrando agora na área musical, vamos mapear os termos da ondulatória para os termos usados na música. A principal propriedade de estudo para este trabalho é a frequência da onda, ela descreve a altura do som, vale lembrar que a altura não é o volume (intensidade) do som como estamos acostumados a falar, o volume é dado pela amplitude da onda, quanto maior a amplitude, maior a intensidade do som. A altura é o que diz se um som é grave (baixa frequência) ou agudo (alta frequência). Uma nota musical é definida por uma determinada frequência.

As notas musicais são uma convenção, um código estabelecido para os músicos lerem as partituras. Elas podem ser representadas de mais de uma maneira diferente. No Diapason elas são representadas por letras, uma linguagem universal para músicos. São elas: Dó – C; Ré – D; Mi – E; Fá – F; Sol – G; Lá – A; Si – B. Um destaque vai para o símbolo "#" chamado de **sustenido**, por exemplo, C# é o dó sustenido, o que quer dizer que é um Dó meio tom (um semitom) mais agudo. Dizemos que a distância entre duas notas consecutivas (no universo de todas as notas), é dada por um tom, e entre essa distância temos o sustenido, que é a distância que chamamos de um semitom. Isso é válido exceto para a distância do Mi até o Fá, e do Si até o Dó, que também são distâncias de um semitom, e portanto Mi e Si não tem sustenido. A figura 2.2 demonstra a diferença entre uma nota e seu sustenido, observe que as ondas começam no mesmo ponto, porém uma delas chega primeiro no topo onde y = 1, o que quer dizer que ela tem maior frequência que a outra, portanto o som produzido por ela é mais agudo. A onda de maior frequência representa o sustenido da outra onda sobreposta.

Tom é precisamente a distância entre dois sustenidos. Por exemplo, a distância entre dó e ré é de um tom, pois entre dó e ré há uma distância de dois sustenidos (de dó para dó sustenido e de dó sustenido para ré).[6]

É possível expressar a mesma nota com frequências diferentes, por exemplo um Lá pode ser da frequência 440 Hz, caso multipliquemos esta frequência por 2, teremos como resultado a mesma nota uma oitava acima. A oitava é justamente a nomenclatura dada para duas notas iguais porém com frequências diferentes. A nota Lá por exemplo pode ter as seguintes frequências, 27.5

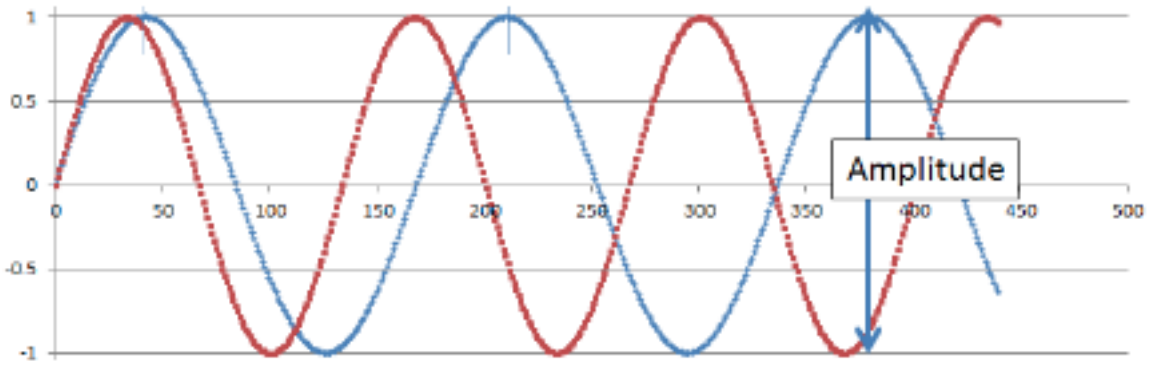

Figura 2.2: Sobreposição da representação bidimensional de uma nota e seu sustenido.[18]

Hz, 55 Hz, 110 Hz, 220 Hz, 440 Hz, 880 Hz, 1760 Hz, 3.520 Hz e assim por diante. Então se temos duas fontes de som, uma produzindo ondas de 440 Hz e outra de 880 Hz, elas descrevem a mesma nota porém em oitavas diferentes, a de 440 Hz sendo mais grave que a de 880 Hz. A notação desta característica pode ser feita com um número inteiro acompanhado da letra que representa a nota, então um Ré sustenido na terceira oitava pode ser representado como D#3. A figura 2.3 mostra a nota Lá representada em diferentes frequências.

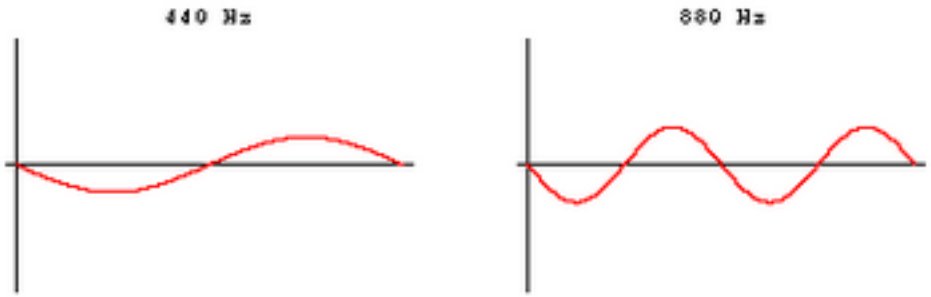

Figura 2.3: Nota Lá representada em diferentes frequências.[17]

Escalas musicais são sequências ordenadas de notas. Por exemplo: dó, ré, mi, fá, sol, lá, si, dó, repetindo esse ciclo.

A tessitura, muitas vezes confundida com extensão vocal, é o conjunto de notas que um cantor consegue alcançar sem fazer esforço. A extensão vocal já é o conjunto total de notas que o cantor pode propagar com sua voz. Então é possível articular notas além de sua tessitura porém nunca além de sua extensão vocal.[7]

A linha de voz de uma música, é a escala que a voz do cantor percorre no decorrer do tempo, ou seja, são notas com intervalo de tempo bem definido.

Resumindo o mapeamento das características de uma onda para as propriedades de um som, de maior relevância para o tema abordado, temos que: a frequência implica na altura do som, é o que define a nota escutada, graves e agudos. A amplitude da onda é a intensidade do som, o que chamamos de volume, som forte e som fraco.

2.3 Representação de uma onda em memória

Tendo essa base podemos descrever como um som é armazenado na memória de um computador. Vemos que a onda é desenhada bidimensionalmente em um gráfico de tempo x amplitude, o que podemos facilmente representar com um vetor, onde o índice será o tempo e o valor contido neste índice é a amplitude naquele dado momento. Como o som é uma onda contínua e precisamos armazenar isso de maneira discreta, quanto mais pontos tivermos para descrever o mesmo espaço de tempo de um som, mais fiel esta representação será a realidade, melhor a qualidade do som.

2.4 Funções usadas para manipulação de áudio

Algumas técnicas são usadas para trabalhar sobre a entrada de áudio obtida do microfone. Além da facilidade fornecida pelas interfaces de som da biblioteca p5.js usada no sistema, contamos com diversas funções responsáveis por transformar o dado de entrada em informação relevante.

Esta seção se divide em subseções que explicam cada uma destas funções que serão utilizadas pelo Diapason para o correto cálculo da diferença das afinações de entrada e esperada.

2.4.1 Transformada rápida de Fourier (FFT)

A "Transformada Rápida de Fourier"(FFT) é um importante método na ciência de medição de áudio e acústica. Ele converte um sinal em componentes espectrais individuais e, portanto, fornece informações de freqüência sobre o sinal.

Estritamente falando, o FFT é um algoritmo otimizado para a implementação da "Transformação Discreta de Fourier"(DFT). Um sinal é amostrado durante um período de tempo e dividido em seus componentes de freqüência. Esses componentes são oscilações sinusoidais únicas em freqüências distintas, cada uma com sua própria amplitude e fase. Essa transformação é ilustrada na figura 2.3, durante o período de tempo medido, o sinal contém 3 frequências dominantes distintas.[13]

O FFT extrai do vetor que contém a onda discutido na seção 2.2, um vetor de frequências que compõem a onda representada em questão. Este algoritmo será utilizado posteriormente para identificar corretamente o tom recebido pela entrada do microfone.

2.4.2 Filtro Low Pass (Passa baixa)

Low Pass é um filtro onde passam baixas frequências a atenuam-se as altas frequências. Por exemplo, uma parede ou um carro agem como um filtro Low Pass pois o som mais grave atravessa sem muitos problemas e os sons agudos são atenuados, ficando quase ou completamente imperceptíveis.

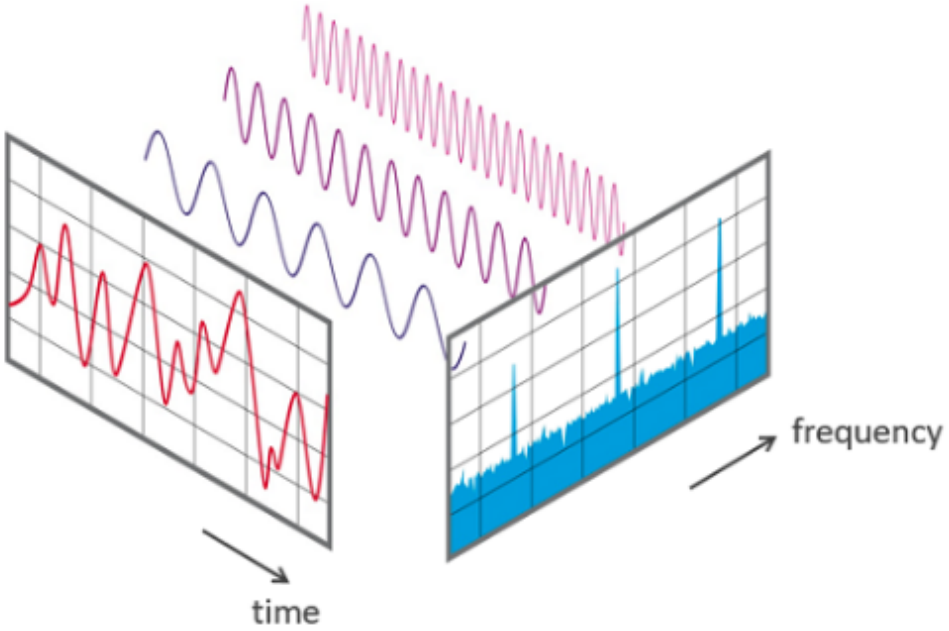

Figura 2.4: Destrinchamento de um sinal nas diferentes frequências que o compõe.[16]

2.4.3 Normalize (Normalização)

A função Normalize é usada para normalizar um buffer, o valor mais alto encontrado é alterado para 1, e na sequência todos os valores do buffer são divididos por esse valor encontrado para ficarem na mesma escala.

2.4.4 Autocorrelation (Autocorrelação)

A função de autocorrelação utiliza a normalização antes e depois de seu algoritmo, e é usada para facilitar a detecção de um tom em um som. Autocorrelação significa a correlação de valores de uma mesma variável ordenados no tempo com dados de séries temporais, como é o caso do vetor descrito na seção 2.3, utilizado como parâmetro para a função de autocorrelação.

3 Revisão bibliográfica

Alguns sistemas já foram criados para instruir, ajudar, ou simplesmente entreter um cantor para que use sua voz.

3.1 Karaokê

Em 1940 o japonês Daisuke Inoue inventou o karaokê, aparelho no qual as pessoas cantam acompanhando versões instrumentais de músicas famosas.

A música tocada é gravada sem voz. Existem CDs de karaokê especiais, geralmente no formato CD+G (formato que conta com texto de legenda para letra da música, além do áudio). Estes CDs incluem a versão instrumental da música e as informações de texto. Ao tocar o CD, o cantor e o público podem ouvir a música e o cantor ler o texto na tela ao cantar a música. A maioria usam para orientar o trecho a ser cantado marcado em cores ou com uma animação.[9]

O objetivo do karaokê é o entretenimento, após cantar a música o usuário é avaliado pela sua performance com uma nota de 0 a 100 na tela do aparelho, porém esta avaliação não tem relação com o tom da voz, diferindo do Diapason.

3.2 Guitar Hero

Guitar Hero é um jogo eletrônico musical desenvolvido pela Harmonix Music Systems e publicado pela RedOctane para o console de videogame Playstation 2. É o primeiro jogo da série Guitar Hero que foi lançado em 8 de novembro de 2005.

O jogo apresenta um controlador em forma de guitarra (semelhante a uma miniatura Gibson SG) que o jogador usa para simular a reprodução de uma música de rock.

A jogabilidade é semelhante a do GuitarFreaks, em que o jogador pressiona os botões do controlador na hora em que as notas musicais se deslocam no ecrã do jogo. O jogo apresenta covers de canções populares de rock, além de faixas bônus de artistas independentes.

Guitar Hero tornou-se uma surpresa positiva, ganhando críticas e elogios e também ganhando muitos prêmios por parte das principais publicações de videogames, e foi considerado um dos jogos mais influentes da primeira década do século XXI. [10]

A jogabilidade do Diapason foi inspirada no Guitar Hero - que foi inspirado em um outro jogo chamado Guitar Freaks - e é de fato muito intuitiva. A diferenciação dos dois está em seu propósito. O Diapason lê a entrada a partir de um microfone, e apesar de que o usuário possa usar um instrumento musical como um violão para percorrer o desafio, o intuito é que use sua voz para validar sua afinação, já os primeiros Guitar Hero eram jogos para entretenimento, posteriormente lançando versões que permitiram o usuário a de fato aprender à tocar guitarra participando dos desafios do jogo.

3.3 Smule

O Smule é um aplicativo para Android com uma ideia bem parecida com o Diapason, já uma plataforma bem evoluída que conta com milhares de usuários ao redor do mundo. No aplicativo é possível cantar junto com músicas sozinho, em dueto ou até mesmo em grupo.

O funcionamento dele segue o mesmo princípio do Diapason, notas vão passando da direita para a esquerda e são cantadas em determinado momento aonde indicado.

Depois de cantar uma musica o usuário pode aplicar filtros a sua voz e escutar como ficou a gravação do seu canto com a música.

3.4 Vanido

Vanido é o aplicativo treinador de canto desenvolvido para iOS, ele segue o mesmo princípio do Diapason e do Smule em estilo de funcionamento, mas o diferencial dele são os treinamentos diários, existem desafios para treinar tipos de voz, este tema não foi abordado aqui, mas existe a voz da cabeça e a voz do peito, que são treinadas separadamente no Vanido. Existem também treinamentos para agilidade e fundamentos.

A interface visual dele é realmente muito bem elaborada, o que deixa o aplicativo muito intuitivo e completo, e ele ainda conta com a reprodução da voz esperada no momento em que ela deve ser cantada, o que facilita muito a encontrar a nota com a voz.

4 Descrição conceitual

Tendo a revisão bibliográfica como base, o próximo passo é planejar um sistema que auxilie um cantor a articular sua voz corretamente conforme o tom definido. A princípio minha ideia era de sobrepor a representação bidimensional da onda definida pela nota desejada, com a onda produzida pela voz cantada pelo usuário, mas isto exigiria um conhecimento da parte do cantor de entender as propriedades de uma onda para que ajustasse sua voz apenas observando esta sobreposição, a necessidade de ter um sistema mais amigável para o usuário ficou clara. Após as pesquisas realizadas buscando ferramentas que abordam o mesmo tema, notei que este formato estilo jogo parece ser de uma experiência muito mais agradável para o usuário, além de ser muito mais intuitivo, e decidi então seguir este modelo.

Com essa ideia definida, precisava de uma forma de animar objetos em tela com conhecimentos em programação. Com uma noção de HTML, CSS, e JavaScript, peças fundamentais do desenvolvimento web, procurei alguma biblioteca ou framework que poderia me ajudar a fazer o esperado. Por mais que meu conhecimento e gosto por JavaScript não eram dos maiores, resolvi fazer este trabalho no lado do cliente ao invés de fazer do lado do servidor pois como a ideia é pegar a entrada do microfone e dar feedback em tempo real para o usuário, enviar esses dados para o servidor processar resultaria numa latência muito maior, e consequentemente andaria na contramão do propósito do aplicativo.

Como o sistema lida com objetos desenhados na tela com interação por parte do usuário, e também lida com o áudio imposto no microfone, além da reprodução de áudio pelo próprio sistema, foi difícil encontrar uma biblioteca que trabalharia com todas estas vertentes, talvez fosse necessário usar mais de uma. Depois de muita procura encontrei uma biblioteca que parecia ter exatamente o que precisava, o p5.js.

O p5.js é uma biblioteca JavaScript que começa com o objetivo original do Processing[11] - tornar a codificação acessível para artistas, designers, educadores e iniciantes - e reinterpreta isso para a web de hoje. Processing é uma linguagem de programação e ambiente de desenvolvimento integrado, inserida no contexto das artes visuais.

Usando a metáfora original de um caderno de esboços de software, o p5.js possui um conjunto completo de funcionalidades de desenho. No entanto, como não está limitado a tela de desenho, é possível pensar em toda a página do navegador como seu esboço. Para isso, o p5.js possui bibliotecas adicionais que facilitam a interação com outros objetos HTML5, incluindo texto, entrada, vídeo, webcam, e o que não poderia faltar, o som.[12]

Neste capítulo é descrita a ideia geral da implementação na seção 4.1, na seção 4.2 estão descritos os elementos visuais do sistema, e por fim na seção 4.3 está uma revisão breve do fluxo do sistema.

4.1 Ideia geral da implementação

Os desafios cadastrados no sistema seguem um formato de descrição específico que será demonstrado no decorrer deste documento, mas é possível qualquer pessoa escrever um arquivo nos padrões de leitura do sistema de gerenciamento de desafios do Diapason, para que este possa ser escolhido do catálogo de desafios da ferramenta.

A biblioteca p5.js conta com algumas funções já implementadas, como a setup() e a draw(). A função de setup é executada antes de tudo, é a função onde são carregados todos os elementos necessário para o desafio, como a música a ser tocada, a posição dos elementos na tela, a disposição das notas conforme o arquivo de descrição do desafio escolhido pelo usuário. Já a função draw é o que faz as animações e a interação com o usuário conforme o passar do tempo possível. Ela é chamada dentro de um laço até que o desafio encerre-se ou seja pausado. Esta função é responsável por desenhar os elementos na tela. Como ela é chamada incessantemente até que o desafio acabe, a modificação do posicionamento dos elementos na tela a cada chamada da função, dá a impressão de movimento dos objetos, pois a cada chamada da função os parâmetros internos do Diapason são atualizados.

Descrevo aqui esse elementos do desafio que são alterados conforme a entrada lida pelo microfone e o passar do tempo (passar das chamadas da função draw).

4.2 Elementos visuais do sistema

Diversos elementos visuais compõem o desafio de maneira a fornecer interação com o usuário, de forma a tornar sua experiência mais prática, intuitiva, e produtiva. A biblioteca p5.js provê uma interface simples para gerar elementos no canvas, como retângulos, círculos, linhas e outras formas geométricas. Com essas ferramentas foram elaborados e criados os elementos visuais do sistema, que serão descritos nas subseções a seguir.

4.2.1 Canvas

O espaço, ou quadro, chamado de canvas, é o ambiente onde os elementos usados podem cumprir seus papéis. O canvas é um retângulo, e o posicionamento e tamanho dele e dos elementos nele inseridos são referidos conforme o plano cartesiano. Todos os outros elementos desenhados na tela são desenhados dentro do canvas, em relação à suas coordenadas. Os elementos tem seu posicionamento descrito em termos de x e y. Um pouco diferente do que estamos acostumados a ver, o eixo y “cresce” para baixo, já o eixo x é como geralmente aparece,

aumentando da esquerda para a direita. No canvas não existem x e y negativos, então o ponto mais alto e mais a esquerda, x e y valem 0, já no ponto mais abaixo e mais a direita, x vale a largura do canvas, e y vale a altura do canvas, medidas em pixels.

Imagine o canvas como sendo uma pilha de folhas de papel onde os elementos estão desenhados, essas folhas vão sendo removidas do topo uma a uma, dando a impressão de que os objetos estão se movendo.

4.2.2 Linha de canto

Temos uma linha vertical denominada “linha de canto”, posicionada mais à esquerda da tela, ela representa o tempo presente, momento pelo qual o usuário deverá dar seu canto de entrada ao microfone, para que ele e o sistema comparem a tonalidade da voz com a nota indicada na nota da tela.

4.2.3 Notas de voz

Essas notas que passarão pela linha de canto, chamarei de notas de voz, elas podem se deslocar em qualquer altura do eixo y, quanto maior a altura (menor o valor de y) mais aguda a voz terá que ser para atingir a nota de voz, pelo mesmo raciocínio, quanto mais baixa a altura da nota com relação ao eixo y (maior o valor de y), mais grave a voz terá que ser para atingir a nota com sucesso. As notas atravessam o canvas percorrendo o eixo x indo da direita para a esquerda, e ficam fixadas em uma altura y. Cada nota tem sua altura em y, nenhuma outra nota pode ter a mesma altura, visualmente falando a altura em que uma nota aparece no canvas é justamente o que define a frequência (nota) que ela representa.

4.2.4 Ponto de voz

O indicador da tonalidade da voz atual é uma circunferência denominada “ponto de voz”, que passeia pelo eixo y, e fica fixa no eixo x junto da linha de canto. Como explicado nas notas, o ponto de voz sobe e desce pela linha conforme a entrada do microfone muda sua tonalidade, vai pra cima se a voz fica mais aguda e vai pra baixo se a voz fica mais grave. Quando o ponto de voz atinge a nota com certa precisão -calculada usando uma constante definida no programa-, sua altura, posição no eixo y, fica igual a altura da nota passante, o ponto de voz tem sua cor alterada para amarelo, para ficar visualmente mais fácil de saber que naquele momento o usuário está cantando afinadamente.

4.2.5 Barra de notas

Na extrema esquerda onde x = 0, fica uma coluna com várias linhas, cada linha indica a nota que passa naquela altura do eixo y. Isto é apenas um facilitador para o usuário, cada nota passante tem uma cor que corresponde a mesma cor da linha dessa coluna. Esta coluna referencio

por “barra de notas”, ela é baseada em um piano, então podemos pensar nela como sendo um piano virado na vertical.

4.2.6 Histórico da voz

O histórico de voz é desenhado entre o ponto de voz e a barra de notas, e é apenas um registro de onde o ponto de voz esteve nos últimos momentos. Ele é uma linha constituída da junção dos pontos onde esteve o ponto de voz. Também é apenas um facilitador para o usuário, já que pode ver o histórico andando junto a uma nota de voz caso tenha cantado ela corretamente.

4.3 Revisão do sistema

A figura 4.1 mostra o aplicativo em uso para melhor exemplificar os elementos descritos anteriormente.

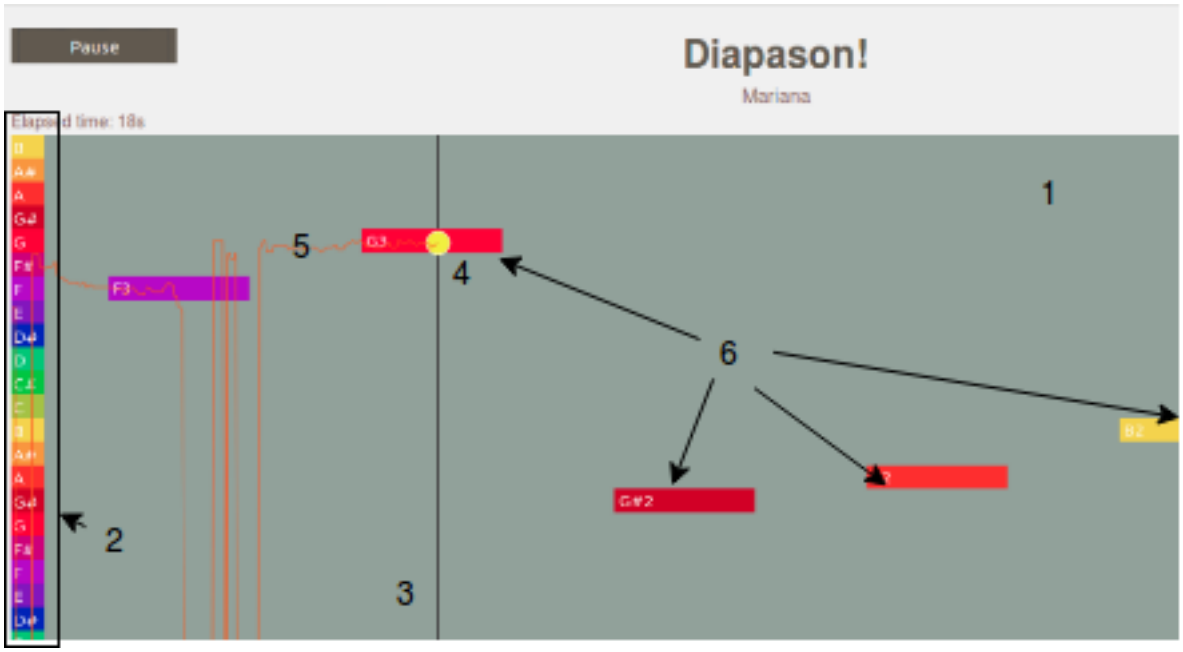

Figura 4.1: Momento em que a voz atinge um G3 (Sol na terceira oitava) no Diapason.

Na figura 4.1, os elementos estão representados pelos seguintes números:

1. Canvas: Todo retângulo onde os outros elementos se encontram.
2. NotesBar: Barra de notas indicativas de altura.
3. SingingLine: A linha de canto logo ao lado do número 3, representando o momento em que a nota passante deverá ser cantada.
4. VoiceDot: O ponto de voz no meio da nota G3, o que quer dizer que neste momento o usuário está cantando afinadamente.

5. VoiceHistory: Linha que mostra onde o ponto de voz esteve num breve período do passado.

6. VoiceNote: Notas de voz movimentando-se da direita para a esquerda.

Para resumir o sistema, no menu principal o usuário terá a opção de cadastrar sua tessitura vocal, escolher um desafio dentre os previamente cadastrados, ou ler as instruções do Diapason.

Quando ele escolhe um desafio é levado para a tela do mesmo para cantar as notas no tempo certo, como descrito no arquivo de configuração do desafio.

Após percorrer o desafio, é dado ao usuário o número de pontos conquistados na partida, para que ele possa comparar com suas partida prévias e saber se está indo no caminho certo para melhorar seu canto.

5 Implementação

O programa foi desenvolvido utilizando HTML, CSS, e JavaScript com o auxílio das bibliotecas p5.js e jQuery. O jQuery é um facilitador na hora de mexer com a árvore de objetos dos documentos HTML.

Separei as funcionalidades e os elementos do sistema em modelos, e existem arquivos com funções que permeiam todo o projeto. A lib p5 traz diversas funções para facilitar a criação e manipulação de objetos HTML5 no canvas bem como facilitar a manipulação de áudio.

Cada elemento visual do sistema descrito na seção anterior, tem um modelo (classe) definido para atuar como um objeto independente. Todos eles contam com o método draw, responsável pela lógica necessária para posicionar e desenhar este elemento no canvas.

Além dos arquivos de funções, temos um arquivo com constantes que são incluídos em todo o programa. As constantes definidas e as funções usadas a parte, serão explicadas no momento em que surgirem na explicação de alguma funcionalidade do sistema que as use.

Para explicar este capítulo, divido-o em quatro seções, a seção 5.1 descreve o que esperar nas seções 5.2 de elementos visuais, 5.3 de ferramentas de áudio, e 5.4 de gerenciamento de partituras, que chamo de subpartes do sistema que compõem o Diapason.

5.1 Elementos do sistema

Para ter uma organização mais clara do sistema, resolvi criar modelos encarregados de diferentes tarefas. Apesar de não ser um sistema 100% orientado a objetos, desta forma foi possível fazer uma melhor abstração do sistema em geral.

Existe um modelo principal chamado Diapason que é o gerenciador geral do desafio escolhido. Por ele ser encarregado de lidar com todos os outros modelos do sistema, o acoplamento dele é muito alto, não foi um sistema muito elaborado para oferecer micro serviços neste aspecto, a separação em modelos foi mais por uma questão de organização mesmo.

O Diapason pode ser separado os elementos em três grupos, o primeiro é o grupo de elementos visuais, o segundo é o grupo de ferramentas de áudio, e o terceiro é o grupo de gerenciamento de partituras.

As informações pertinentes apenas ao Diapason descritas nas propriedades do modelo são, id do desafio atual, pontos conquistados no desafio, e contador de iterações. Além destes temos as propriedades pertinentes ao grupo de ferramentas de áudio, como nível de suavização,

que veremos para que serve mais para frente nesta seção, vetor de frequências, e frequência atual. Suas necessidades serão explicadas a seguir. Em um primeiro nível de abstração podemos ver o Diapason como na figura 5.1.

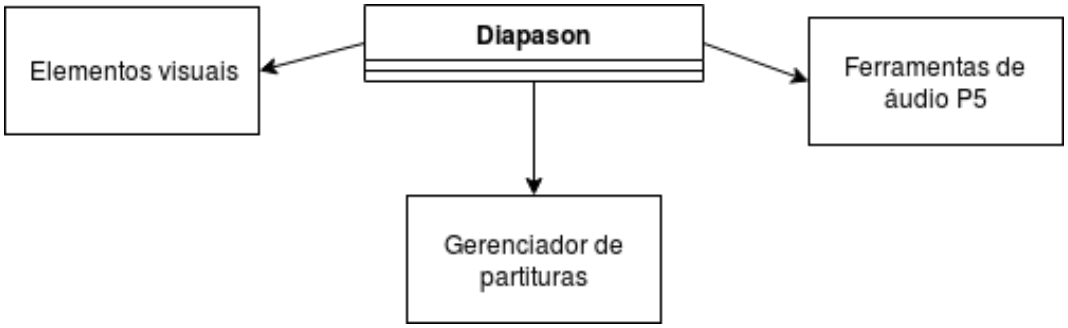

Figura 5.1: Abstração da classe Diapason.

Existe um ponto de entrada da classe que é método setup(), responsável por carregar e instanciar todas os atributos e relações do Diapason, nele instanciamos os elementos visuais descritos na seção anterior controlados pelo Diapason, como a SingingLine (linha de canto), o VoiceDot (ponto de voz), e a NotesBar (barra de notas). Instanciamos também as ferramentas de áudio e nosso gerenciador de partituras, ambos veremos mais profundamente na sequência.

Os métodos play() e pause() são utilizados para iniciar e pausar o desafio respectivamente. O que isso faz na prática é continuar e parar o loop realizado para chamar a função draw() do p5.js.

O método iterate() é o método executado a cada chamada da função draw. Nele está toda a lógica que faz os elementos moverem-se de acordo com a iteração atual e a entrada do microfone. O Diapason chama o método draw de todos os elementos que ele controla, além de iterar o gerenciador de partituras, após atualizar seu controle de áudio p5.

Dito isso explico a seguir mais detalhadamente as subpartes que contemplam esta classe principal.

5.2 Subparte: Elementos visuais

Exceto pelas notas de voz que são controladas pelo gerenciador de partituras, os elementos visuais do sistema são instanciados nesta classe.

Todos estes elementos são desenháveis e portanto implementam seu próprio método draw, que fica responsável por desenhar o elemento em questão no canvas.

5.2.1 NotesBar (Barra de notas)

Um dos elementos mais simples é a barra de notas, classe NotesBar, nela constam apenas um vetor de notas (atributo notes), e um valor inteiro que diz o tamanho em pixels que as notas devem ser desenhadas na tela (atributo noteHeight). O único método que esta classe

implementa é o draw, neste caso a função draw imprime na tela cada elemento do vetor notes, usando uma cor definida nas constantes do sistema para a nota em questão, usando o noteHeight como tamanho.

5.2.2 SingingLine (Linha de canto)

Outro elemento simples é o SingingLine que representa a linha de canto, ele tem como atributos x, y, w, h, e color, respectivamente descrevendo a posição no eixo x do canvas, a posição no eixo y do canvas, a largura da linha, a altura da linha e a cor da linha. O método draw implementado imprime na tela uma linha, usando como parâmetro todas as propriedades definidas no objeto, e este é o único método da classe.

5.2.3 VoiceDot (Ponto de voz)

Por último temos um elemento mais complexo, o VoiceDot (ponto de voz) é afetado pela entrada dada no microfone. Este elemento tem como atributos x, y, radius, e color, propriedades que definem sua posição em relação ao eixo x e y no canvas, o raio da circunferência, e a cor do círculo desenhado respectivamente. Além de conter um vetor de posições prévias do onde o ponto de voz esteve no eixo y, chamado py. Ele guarda também em maxPy o tamanho máximo que o vetor py pode ter. Por fim é guardado uma flag para saber se o ponto está atingindo a nota pretendida naquele momento, chamada hittingNote.

Esta classe implementa o método draw da seguinte forma, o método recebe como parâmetro uma frequência sonora, um vetor com as oitavas usadas no desafio, e um volume. O volume é usado como filtro, caso a entrada do microfone esteja captando um som muito fraco, o voiceDot vai simplesmente ignorar essa chamada da função draw, caso passe o limiar de volume aceitável como tentativa de canto, sua lógica de atualização é executada. A frequência recebida como parâmetro é o que vai ditar a posição do ponto de voz no eixo y, que é o que define uma nota. Pensar em uma solução para mapear uma frequência recebida em uma posição no eixo y não foi uma tarefa fácil, não existe uma fórmula em função da frquência que retorne uma altura no eixo y para minha realidade, após dispender um certo tempo pensando sobre isso, optei por fazer uma **aproximação** que usa a frequência e as oitavas passadas como parâmetro.

A aproximação da frequência em posição no eixo y se dá da seguinte forma, no arquivo de definição de constantes do sistema, temos uma matriz chamada de FREQUENCY_NOTES_MAP, esta matriz tem suas linhas representadas por notas musicais, e suas colunas representadas pelas oitavas dessas notas, o conteúdo de cada posição é portanto a frequência que está uma determinada nota numa determinada oitava. A matriz vai de Si a Dó e da oitava 0 até a 8, o que mapeia frequências de 16.35 Hz até 7902.13 Hz. Com a frequência passada no método draw, encontramos na matriz as duas notas que ficam mais próximas do que foi cantado, sabendo quais são estas duas notas posicionamos o ponto de voz ponderadamente entre as posições do eixo y definidas para elas.

Finalmente checamos se a posição do ponto de voz acrescido e decrescido de uma constante de aceitação de tom definida no código do programa, é plausível de se dizer que o canto está afinado com relação à alguma nota. Esta constante pode ser alterada caso a aceitação de afinação precise ser outra. Caso esteja nos níveis de aceitação, a flag hittingNote é alterada para o valor true, e o atributo color do ponto de voz é alterado para demonstrar ao usuário que ele está dentro da afinação aceitável dos padrões definidos. O vetor py é atualizado, e caso este tenha a quantidade limite definida por maxPy, o valor mais antigo adicionado ao vetor é removido para que um novo possa ser inserido.

Existe nesta classe também o método para desenhar o histórico do objeto. O método drawVoiceHistory simplesmente itera sobre o vetor py, criando uma espécie de gráfico entre a SingingLine e a NotesBar que representa onde o ponto de voz esteve em relação ao eixo y.

5.3 Subparte: Ferramentas de áudio

As ferramentas de áudio são definidas pela biblioteca p5.sound, ela provê uma interface simples para trabalhar com os mais variados propósitos sobre áudio. Neste sistema utilizamos as classes p5.Amplitude, p5.AudioIn, p5.FFT, e p5.LowPass, explicadas com mais detalhes na subseção seguinte.

Além dos objetos de áudio instanciados pelas classes do p5.sound, são usadas algumas funções para trabalhar com os dados obtidos e os transformar em informação de relevância para o tema abordado.

Para entender melhor como funciona o tratamento de áudio no Diapason, explico na subseção 5.3.1 as classes da p5.sound, e na subseção 5.3.2 como funciona uma iteração do Diapason com relação à captura e tratamento de áudio.

5.3.1 Classes do p5.sound

É instanciado primeiramente o objeto que representará o microfone, a classe p5.AudioIn é a responsável por captar áudio de uma entrada. Usando o método start(), o navegador é solicitado a ativar um microfone, que por sua vez faz a requisição para o usuário.

Em seguida instanciamos um objeto da classe p5.LowPass que age como um filtro de frequências para o microfone, usando o método connect() do p5.AudioIn, conectamos o p5.LowPass a entrada de áudio.

Em seguida instanciamos o objeto p5.FFT, ele é quem fica encarregado de transformar a entrada escutada em um vetor de frequências, que será usado a cada iteração para recalcular a frequência fundamental, e assim decifrar a nota atingida pela voz do usuário naquele dado momento. O objeto FFT lê dados do microfone ao definirmos como entrada o objeto p5.AudioIn, passado como parâmetro ao método setInput().

Por fim é instanciado o objeto da amplitude da onda, o p5.Amplitude também utiliza o método setInput para receber dados do objeto do microfone. Ele é o encarregado de informar o volume do som recebido como entrada.

5.3.2 Iteração

Ao começo da execução do método de iteração do Diapason, todos os objetos são atualizados. A primeira linha de código do método o objeto FFT utiliza o método waveform(), ele retorna o vetor de frequências discutido na seção 2.3 deste documento. O tamanho deste vetor é passado como argumento para o método, no Diapason este valor é 1024. Este vetor nada mais é do que a amplitude da onda em 1024 unidades de tempo sequenciais, o valor da amplitude tem seu domínio entre os números reais de -1 a 1. Este vetor é associado a uma variável chamada timeDomain.

Na sequência a variável time domain é usada na função autoCorrelate(), que retorna um buffer da correlação, como discutido na seção 2.4.4 do documento.

O buffer é usado em seguida na função findFrequency(), o que esta função faz é, percorrer este vetor representante da onda observando qual é o maior pico encontrado, para então ver qual a distância entre dois desses picos. Dividindo a quantidade de amostras por segundo por essa distância encontrada, obtemos a frequência fundamental gerada.

Existe em seguida uma função de suavização, ela nada mais é do que a moda das últimas n frequências detectadas, este valor n é definido pela constante SMOOTHNESS_LEVEL, e tem atualmente o valor 4. Isso é feito para suavizar as oscilações da entrada, bem como suavizar a movimentação do ponto de voz ao passar das iterações, trazendo uma experiência melhor para o usuário.

A frequência e a amplitude obtidas na iteração, são passadas como argumento a função draw do ponto de voz para que ele possa ser posicionado de acordo com a entrada do microfone, como visto na seção 5.2.3.

5.4 Subparte: Gerenciador de partituras

O último componente restante do Diapason é o gerenciador de partituras, ou Music Sheet Manager como é nomeado no sistema. Esta classe fica principalmente responsável por ler, interpretar e carregar em memória o desafio escolhido pelo usuário a partir de um arquivo em formato JSON, também é responsável por manter as notas de voz, descritas na seção 4.2.3.

JSON (JavaScript Object Notation, Notação de objeto do JavaScript), é um formato de dados comum e independente de linguagem que fornece uma representação de texto simples de estruturas de dados arbitrárias.

5.4.1 Formato da partitura

Pensando em todo dado necessário para fazer a lógica de posicionamento das notas de voz no canvas, escrevi um padrão de descrição de desafios em formato JSON. Em cada arquivo de desafio, escala ou música, o arquivo conta com os seguintes campos. “Name”, campo descritivo identificador do desafio. “Type” é o campo que define o tipo do desafio, pode ter o valor “music” ou “scale”. “Author”, campo identificador do compositor da música em questão, esse campo é vazio caso o desafio seja uma escala. “Sheet” é o que chamo de partitura, um vetor que contém os vetores descritivos das notas do desafio, cada nota é representada por quatro valores, os dois primeiros referenciam a quantidade de milissegundos que deverão passar após o início do desafio para que esta nota comece e termine de ser cantada respectivamente. O terceiro valor é a nota em si, pode assumir qualquer valor como os descritos na notação da seção 2.2, variando de “C”, “C#”, “D”, . . . “B”. O quarto e último valor é a oitava que a nota deverá ser cantada, é um número inteiro que pode assumir valores de 0 a 8, respeitando a quantidade de frequências que foram mapeadas na constante FREQUENCY_NOTES_MAP descrita na seção 5.2.3.

Um exemplo de JSON corretamente escrito para ser usado de desafio do sistema pode ser visto na figura 5.2 com o trecho de código.

```
{
  "name": "Estamos errados",
  "type": "scale",
  "author": "",
  "sheet": [
    [3000, 5000, "E", 2],
    [7000, 9000, "F#", 2],
    [11000, 13000, "G", 2],
    [15000, 17000, "A", 2],
    [19000, 21000, "B", 2],
    [23000, 25000, "C", 3],
    [27000, 29000, "D", 3],
    [31000, 33000, "E", 2],
    [35000, 37000, "F#", 2],
    [39000, 41000, "G", 2],
    [43000, 45000, "A", 2],
    [47000, 49000, "B", 2],
    [51000, 53000, "C", 3],
    [55000, 57000, "D", 3],
    [59000, 61000, "C", 3]
  ]
}
```

Figura 5.2: Escala “Estamos Errados” descrita no padrão de leitura de desafios do Diapason.

Os arquivos de partituras contidos no Diapason foram manualmente escritos e estão em tamanho reduzido, usados apenas para testes de validação de funcionamento.

5.4.2 Métodos da classe

A classe loadChallenge() recebe como parâmetro o id de um desafio, internamente este id leva o fluxo do código para chamar a função loadJsonSheet() para o arquivo específico do desafio em questão, função que busca usando uma requisição AJAX o arquivo requisitado, e retorna o JSON lido para uma variável, que será destrinchada entre os atributos da classe MusicSheetManager, que tem mapeamento direto chave-valor com o formato padrão de desafio definido na seção 5.4.1. O método loadChallenge() enfim retorna uma referência para um arquivo de som que será executado junto ao desafio para que o usuário possa cantar escutando a música desejada.

5.4.3 Elementos visuais

Como dito anteriormente, é o gerenciador de partituras que fica responsável por manter as notas de voz do desafio. Após o arquivo de descrição do desafio ser lido, o gerenciador de partituras instancia um objeto da classe VoiceNotes, que mantém um vetor de objetos da classe VoiceNote, além de guardar as informações da partitura, as oitavas usadas pelas notas do desafio, ele também guarda a posição da linha de canto (SingingLine) em relação ao eixo x, para saber o momento em que as notas deverão ser cantadas pelo usuário.

O método draw() desta classe chama o método draw() dos objetos do vetor de notas de voz um de cada vez, e, após isso, ele checa se alguma nota já cumpriu seu papel no desafio para então removê-la do vetor.

A classe VoiceNote fica responsável por toda a lógica necessária para posicionar as notas no canvas de acordo com a informação passada pelo gerenciador de partituras (lida do arquivo descritor do desafio). Isto é feito da seguinte forma, o posicionamento inicial da nota é mapeado pelo valor do tempo em milissegundos do início da nota, que pode variar de 0 ao tempo total do desafio, para um valor que varia da posição do eixo x da linha de canto, até uma conversão do tempo total do desafio para pixels.

A duração de uma nota, Tf - Ti, tempo final menos o tempo inicial, representa a largura da nota de voz no canvas, esse valor resultante é dividido pelo tempo de cada iteração do Diapason, para chegar a um valor condizente ao esperado pelo arquivo descritor do desafio.

O método draw() desta classe a cada chamada diminui em uma unidade a distância da nota ao começo do canvas, isto faz o movimento das notas ser descrito da direita para a esquerda. Após a atualização da posição da nota, é feita uma validação para checar se a nota está atravessando a linha de canto, caso positivo isto quer dizer que o usuário deve cantar a nota, com esta validação é possível atribuir pontos ao canto do usuário, podemos validar se ele está cantando a nota no momento certo.

6 Conclusão

Neste capítulo irei dar um resumo geral com comentários sobre as tecnologias usadas, plataforma do app e melhorias do projeto.

6.1 Notas gerais

O Diapason é muito versátil pelo seu sistema de gerenciamento de partituras, é possível qualquer pessoa escrever um desafio no formato interpretado pelo sistema. A grande vantagem desse sistema é que ele foi desenvolvido para a web, o que permite que seu uso possa ser feito através de múltiplas plataformas e sistemas operacionais, abrangendo assim o maior número de possíveis usuários. O aplicativo usa uma forma intuitiva, já de conhecimento amplo de apreciadores de jogos, para executar o desafio proposto, o que o torna realmente simples de ser usado.

6.2 Melhorias

Infelizmente não pude implementar todas as funcionalidades que gostaria de ter no sistema no prazo definido, portanto já tenho em mente diversas melhorias para o sistema, que listo a seguir.

A tessitura vocal do usuário poderia ser usada para melhorar a experiência dele no desafio. As oitavas cadastradas no desafio seriam alteradas para se enquadrar pela extensão da tessitura do usuário, para que o mapeamento de frequências seja mais real, e para que a saúde vocal do cantor não seja prejudicada, cantar além da sua tessitura pode gerar desconforto e problemas para as cordas vocais.

Como melhoria, o sistema deve aceitar que um usuário cadastre um arquivo descritor de desafio, para que este possa ser interpretado pelo gerenciador de partituras do sistema, e disponível no catálogo de desafios da página inicial. Um cenário ainda melhor para este problema, seria se o Diapason pudesse gerar o arquivo descritor do desafio a partir de uma música, mas isto é de uma complexidade que supera todo o sistema feito.

O cálculo da diferença entre o som esperado e o falado poderia ser diferente, como por exemplo usar a representação bidimensional da onda esperada como curva normal, e usar desvios padrões para encontrar a diferença com a onda dada de entrada pela voz do usuário.

No sistema de gerenciamento de partituras, o tempo descrito para cada nota em um arquivo descritor de desafio não está 100% preciso, mas sim muito próximo disso. Para escalas musicais isto não tem importância, mas para desafios usando músicas isto é essencial que funcione o mais corretamente possível.

A suavização de posicionamento do ponto de voz deve ser melhorada, a suavização atual prejudica a performance do sistema a cada nível incrementado na constante de suavização de voz.

6.3 Contribua

Este sistema ficará disponível para ser usado por qualquer um que assim o queira, e suscetível à requisições de melhoria de código no GitHub, o projeto pode ser encontrado em https://github.com/igorbpersona/diapason, e estará sobre a licesa Creative Commons Não Comercial (NC), que dá o direito de copiar, distribuir, exibir e executar a obra e fazer trabalhos derivados dela, desde que sejam para fins não-comerciais.

Referências

[1] https://pt.wikipedia.org/wiki/Diapas%C3%A3o, acessado em Outubro de 2018.

[2] https://www.smule.com/, acessado em Outubro de 2018.

[3] https://vanido.io/, acessado em Outubro de 2018.

[4] https://pt.wikipedia.org/wiki/Som, acessado em Outubro de 2018.

[5] https://pt.wikipedia.org/wiki/Onda, acessado em Outubro de 2018.

[6] http://www.descomplicandoamusica.com/tom-semitom/, acessado em Outubro de 2018.

[7] https://pt.wikipedia.org/wiki/Extens%C3%A3o_vocal, acessado em Outubro de 2018.

[8] https://pt.wikipedia.org/wiki/Sustenta%C3%A7%C3%A3o_(som), acessado em Outubro de 2018.

[9] https://pt.wikipedia.org/wiki/Karaok%C3%AA, acessado em Novembro de 2018.

[10] https://pt.wikipedia.org/wiki/Guitar_Hero_(s%C3%A9rie), acessado em Novembro de 2018.

[11] https://processing.org/, acessado em Novembro de 2018.

[12] https://github.com/processing/p5.js/wiki/p5.js-overview, acessado em Novembro de 2018.

[13] https://developers.google.com, acessado em Novembro de 2018.

[14] http://archive.oreilly.com/oreillyschool/courses/data-structures-algorithms/soundFiles.html, acessado em Dezembro de 2018.

[15] https://pt.wikipedia.org/wiki/Licen%C3%A7as_Creative_Commons, acessado em Dezembro de 2018.

[16] https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft, acessado em Dezembro de 2018.

[17] https://faraday.physics.utoronto.ca/PVB/Harrison/Vibrations/Vibrations.html, acessado em Dezembro de 2018.

[18] https://www.chegg.com/homework-help/consider-two-waves-shown-fig-12-31-wave-thought-superpositio-chapter-12-problem-14q-solution-9780321736994-exc, acessado em Dezembro de 2018.

[19] https://www.researchgate.net/figure/Figura-1-Imagem-da-oscilacao-senoidal-de-uma-onda-mecanica-gerada-pelo-equipamento-com_fig1_273992387, acessado em Dezembro de 2018.