# Desenvolvimento de *software* para auxílio no aprendizado de teoria musical por meio de visualização e interação

**Daniel de Lima Franceschetti [1], Leonardo Corsino Campello [1]**

[1] Instituto Federal do Sertão Pernambucano (IFSertãoPE)

daniel.source@tutanota.com, leonardo.campello@ifsertao-pe.edu.br

**Resumo.** O artigo trata do desenvolvimento de um *software* que permite interagir com um instrumento de teclado virtual, exibindo as notas, intervalos e acordes musicais executados pelo usuário com o teclado de computador ou mouse. Desta forma, é possível que este *software* sirva como uma ferramenta para entender a teoria musical dos acordes.

**Palavras-Chave:** teoria musical, visualização de acordes, programação em *C*, *raylib*.

***Abstract.*** *The article deals with the development of a software that allows interaction with a virtual keyboard instrument, which displays the notes, intervals or musical chords played by the user with the computer keyboard or mouse. In this way, it is possible that this software serves as a tool to understand the musical theory of chords.*

***Keywords:*** *music theory, chord visualization, C programming, raylib.*

## 1. Introdução

Teoria musical pode ser definida como a área de estudo da música. Conforme Lacerda (1967), a música é dividida em quatro propriedades: duração, o quão longo ou curto é a extensão de um som; intensidade, a sensação auditiva do quão quieto ou barulhento é um som; altura, o quão grave ou agudo é um som; e por fim o timbre, que compreende a qualidade de um som, aquilo que diferencia o som do piano com o do violino e da voz humana [12]. Sabendo disso, a melodia é a sucessão de sons que podem ter alturas e as outras propriedades com valores diferentes, que obedece alguma lógica, ao passo que a harmonia abrange os acordes, que são combinações de sons executados simultaneamente. Na música, a nota representa o menor elemento de um som musical. O que é entendido nesse contexto como teoria musical, serve como norma e princípio que rege esses conceitos expostos, mas que não se propõe a ser uma regra rigorosa.

Nesse artigo, é abordado principalmente a harmonia, demonstrando a concatenação de diferentes notas que formam os acordes. Nesse sentido, o objetivo deste trabalho é desenvolver um *software* para ajudar no aprendizado de teoria musical, possibilitando a visualização de intervalos e acordes no instrumento de teclado virtual, também sendo possível

usar esse *software* como uma ferramenta para quem já é músico. O *software* proposto terá os seguintes objetivos específicos: conseguir a entrada de dados pelo mouse e teclado; calcular quais notas, intervalos e acordes representar pela entrada de dados; mostrar o nome do acorde correspondente; e desenhar as teclas do teclado musical. Cada tecla do teclado emite uma nota musical diferente.

Com o avanço da tecnologia, é possível ter ferramentas virtuais que possam ajudar no aprendizado musical, como o Guitar Dashboard [10] que funciona como uma ferramenta para um guitarrista e não é exclusivamente voltada para o ensino (CONSTANTE; ARAÚJO; SCHIAVONI, 2019) [4].

Os tópicos seguintes deste artigo são divididos em: Referencial Teórico, onde consta a base teórica usada para desenvolver esse *software*; Metodologia, que apresenta a configuração do ambiente de desenvolvimento, o funcionamento do *software* e uma breve demonstração do código-fonte; e Considerações finais, que tira conclusões com o que foi desenvolvido e propõe possíveis melhorias.

## 2. Referencial Teórico

O objetivo da aprendizagem de teoria musical pode ser diverso, mas em geral são procedimentos estudados que têm como resultado a execução e composição de músicas. Se o estudante de música não se atentar em buscar conhecimentos teóricos seu ensino será limitado, porque o conhecimento não será retido, terá somente a mera memorização de uma única música, sem levar experiência adiante para aplicar novamente. Assim, é possível entender que:

> Para o músico, em algum momento, será necessário escrever ou registrar de alguma maneira um determinado exercício ou composição, um conhecimento que foge da simples imitação. É desejável, portanto, que o estudante aprenda não apenas a imitar um som registrado, mas também a executar a partir da notação escrita, bem como a registrar suas próprias composições ou exercícios. (VANZELA; OLIVEIRA; CARVALHO, 2016) [19].

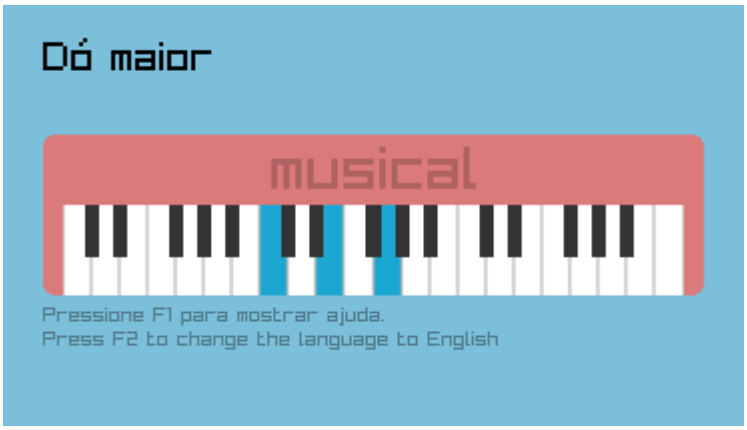
Para Constante et al. (2019), softwares livres parecem ser promissores para alunos, já que não costumam ter custo. Os Software livres são também conhecidos como FLOSS (Free Libre Open Source Software), e são distribuídos com seu código-fonte, ou seja, com a sua metodologia disponível para qualquer um e com as quatro liberdades básicas: a liberdade de executar o software para qualquer propósito; a liberdade de estudar como o software funciona internamente, possibilitando adaptá-lo; a liberdade de redistribuir cópias do software

indistintamente; e a liberdade de melhorar o software, e poder distribuir estas modificações para que pessoas de forma coletiva se beneficiem (CAMPOS, 2006b, apud CONSTANTE et al., 2019) [4]. O código-fonte do *software* desenvolvido neste trabalho é aberto e está disponível em [5].

## 3. Metodologia

O *software* tem o nome "musical" (estilizado com todas letras em caixa baixa), como é possível ver no teclado virtual musical da Figura 1.

**Figura 1 – Captura da janela de *musical***




Fonte: de autoria própria

O *software* é desenvolvido com a linguagem de programação *C*, dentro das especificações da norma ISO 9899:1999 [11], ordinariamente chamada de "C99". O processo de compilação é feito com o uso do GNU Make [8] e suas convenções básicas [9].

Na *Makefile* desse projeto também inclui o suporte de um simples *unit testing*, inspirado no MinUnit [2]. Os alvos (*targets)* das *Makefiles* fizeram o uso de comandos do pacote de utilitários GNU *Core Utilities* [7]. Os compiladores utilizados foram o GNU *Compiler Collection* (GCC) [6] e Clang [13], que conseguem gerar informações de depuração e alertas de correções para boas práticas de código através de opções de argumentos de linha de comando [15].

Como esse *software* também teve a motivação de ser disponível como FLOSS, a licença escolhida é a MIT License [3], que é considerada pequena e simples, o que é adequado com o escopo deste *software*.

O nome dos intervalos e acordes presentes no software estão de acordo com o livro

Teoria da Música, de Bohumil Med [14].

A biblioteca de programação de jogos eletrônicos *raylib* [18] é usada para a entrada de dados do teclado e mouse de computador e para a saída de dados por meio do desenho gráfico digital em uma janela do sistema operacional. Ela tem código aberto e usa a licença Zlib. Esta biblioteca possibilita interagir com a API OpenGL [16] de uma maneira simples através de funções como: "InitWindow", para inicializar uma janela no sistema operacional; "DrawRectangle", para desenhar um retângulo dentro dessa janela especificando posições x e y, largura, altura e cor; e "DrawText", para desenhar um texto com a fonte *bitmap* padrão, que é usada em todos os textos, como pode ser visto na Figura 1. A referência de todas as funções da *raylib* estão disponíveis em [17]. Todas as funções presentes no código-fonte de *musical* com o nome em *PascalCase* são funções da *raylib*, tal como "DrawRectangleRounded".

> [A biblioteca] *raylib* não fornece a documentação típica de API ou um grande conjunto de tutoriais. A biblioteca foi projetada para ser minimalista e ser aprendida apenas a partir de uma folha de dicas com todas as funcionalidades necessárias e uma grande coleção de exemplos para ver como usar essa funcionalidade. A melhor maneira de aprender a programar é lendo o código. (RAYLIB, 2022, tradução nossa) [18].

O código-fonte escrito em *C* (dentro da norma "C99" [11]) foi dividido em 9 arquivos que estão no diretório "src/", sendo estes: "chord_finder.c" e "chord_finder.h", que incluem código relacionado com a identificação de acordes formados a partir das teclas do teclado musical; "config.h", que inclui variáveis configuráveis no momento da compilação; "language.h", que inclui constantes de enumeração relacionadas com o idioma; "mouse_click.h", que inclui código com a lógica da entrada de dados do mouse (para clicar nas teclas do teclado musical); "musical.c", que inclui o código principal relacionado com a chamada de funções da *raylib* e o ponto de entrada pela função "main"; "timer.c" e "timer.h", que incluem código relacionado com temporizadores (p. ex., usados para calcular quanto tempo uma nota se mantém ativa); e "util.h", que inclui algumas variáveis e funções de pré-processamento (chamadas de macro) usadas por mais de um arquivo de código-fonte.

### 3.1. Configuração do ambiente de desenvolvimento

O projeto do *software* está em um repositório *git* hospedado no GitHub [5]. Ele foi desenvolvido em uma distribuição do sistema operacional Linux, usando o GNU Make [8] e utilitários GNU *Core Utilities* [7]. A *Makefile* foi escrita para sistemas "Unix-like" (sistemas similares ao Unix, p. ex., Linux e macOS) e deve ser modificada para cada caso específico,

como a compilação do executável para o Microsoft Windows. O uso de compilador cruzado também é possível, embora não tenha sido testado nesse projeto.

A biblioteca *raylib* [18] é gratuita e está disponível para baixar no site oficial e no GitHub (na página *releases* do repositório). Essa biblioteca deve estar instalada, com o caminho dos cabeçalhos (*headers*) e o caminho dos objetos compartilhados (*shared objects*) ou do arquivo de biblioteca estática (*static library*) sendo referenciados nos comandos da *Makefile*. No caso desse projeto, a distribuição Linux (Arch Linux) utilizada fornecia um pacote para a *raylib* [1], o que dispensou configurações adicionais, pois o compilador e ligador (*linker*) conseguiram encontrar os arquivos da biblioteca nos diretórios padrões do sistema, que nessa distribuição são "/usr/lib" para os objetos compartilhados e "/usr/include" para os cabeçalhos.

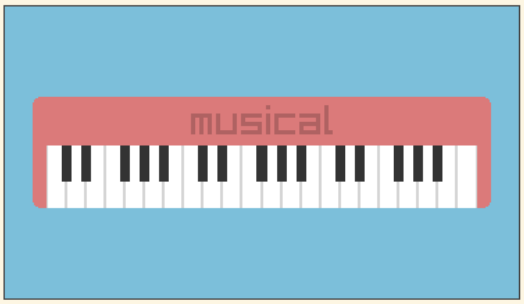
Na *Makefile*, para a variável do compilador ("CC") foi atribuído o comando "cc". Em alguns sistemas "Unix-like", "cc" é um atalho simbólico ("*symlink*") para o compilador padrão. O compilador GCC [6] e o Clang [13] foram utilizados e podem substituir o valor da variável "CC". Teoricamente, qualquer compilador compatível com a norma "C99" [11] pode compilar esse *software*.

### 3.2. Demonstração do *software*

Após os procedimentos do tópico "3.1", com o ambiente de desenvolvimento configurado e com a biblioteca *raylib* instalada, é possível compilar (usando comando "make") e executar o binário "musical", que estará na raiz do repositório, como é possível ver na Figura 2.

**Figura 2 – Processo de clonagem do repositório, compilação e execução de *musical* no Linux**

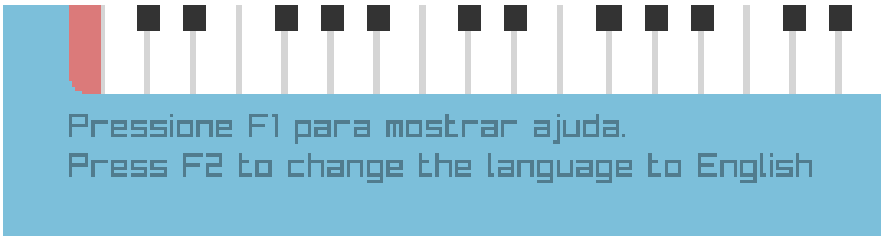
```
$ git clone https://github.com/danielsource/musical
Cloning into 'musical'...
remote: Enumerating objects: 246, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 246 (delta 4), reused 4 (delta 4), pack-reused 228
Receiving objects: 100% (246/246), 61.26 KiB | 950.00 KiB/s, done.
Resolving deltas: 100% (136/136), done.
$ cd musical
$ make clean all
rm -f musical src/chord_finder.o src/musical.o src/timer.o test/test_chor
est_chord_finder.o
cc -o src/chord_finder.o -c -std=c99 -g -Wall -Wextra -Wpedantic src/chor
cc -o src/musical.o -c -std=c99 -g -Wall -Wextra -Wpedantic src/musical.c
cc -o src/timer.o -c -std=c99 -g -Wall -Wextra -Wpedantic src/timer.c
cc -o musical src/chord_finder.o src/musical.o src/timer.o -lraylib -lm
$ ./musical
Made by daniel.source (https://github.com/danielsource/musical) with
raylib. See more about raylib here: https://www.raylib.com
```




Fonte: de autoria própria

Ao executar o programa, é iniciada uma janela que mostra um teclado virtual musical e uma mensagem que dura cinco segundos mostrando que existe um breve tutorial sobre o funcionamento de *musical* que pode ser acessado ao apertar a tecla “F1” e que o idioma pode ser trocado ao apertar a tecla “F2”. Os idiomas português e inglês são suportados, tendo os nomes dos acordes traduzidos nas duas línguas. A mensagem pode ser vista de forma amplificada na Figura 3.

**Figura 3 – Amplificação de captura de tela com ênfase à mensagem**




Fonte: de autoria própria

Para descobrir os atalhos de teclado de computador padrões, a Figura 4 mostra o vetor (*array*) “keybindings” do arquivo “src/config.h”, que possui a configuração do teclado de computador com as teclas que ativam suas respectivas notas no teclado musical ou que executam outras funcionalidades. Esse vetor funciona como um vetor associativo, do tipo de estrutura (*struct*) “Keybinding”. Essa estrutura, declarada em “src/musical.c”, mapeia cada tecla do teclado do computador a alguma função. O atributo da estrutura “key” armazena o valor de uma tecla, e ao apertar a tecla, o programa chama a função armazenada no atributo “pressed_func” com o argumento vindo do atributo “arg”. As estruturas do tipo “Keybinding” estão definidas nesse vetor, como é possível ver na Figura 4.

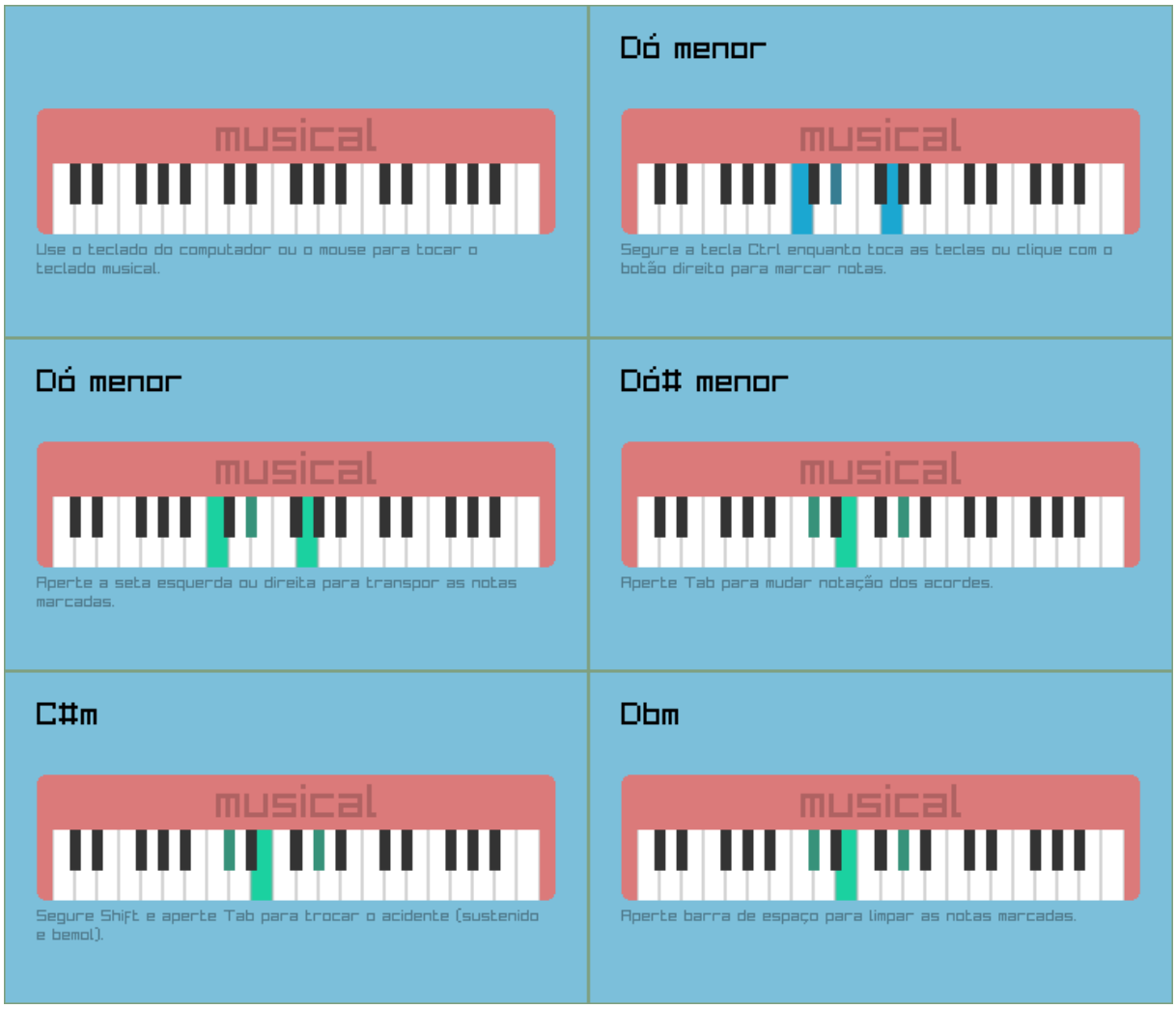
**Figura 4 – Vetor “keybindings”**

```
Keybinding keybindings[] = {
  {.key = KEY_F1,    .pressed_func = show_tutorial, .arg = {.v=NULL}},
  {.key = KEY_F2,    .pressed_func = cycle_language, .arg = {+1}},
  {.key = KEY_TAB,   .pressed_func = toggle_chord_notation, .arg = {.v=NULL}},
  {.key = KEY_INSERT,.pressed_func = print_screen, .arg = {.v=NULL}},
  {.key = KEY_LEFT,  .pressed_func = transpose_notes, .arg = {-1}},
  {.key = KEY_RIGHT, .pressed_func = transpose_notes, .arg = {+1}},
  {.key = KEY_SPACE, .pressed_func = clear_marked_notes, .arg = {.v=NULL}},
  {.key = KEY_Q,     .pressed_func = play_note, .arg = {24}},
  {.key = KEY_TWO,   .pressed_func = play_note, .arg = {25}},
  {.key = KEY_W,     .pressed_func = play_note, .arg = {26}},
  {.key = KEY_THREE, .pressed_func = play_note, .arg = {27}},
  {.key = KEY_E,     .pressed_func = play_note, .arg = {28}},
  {.key = KEY_R,     .pressed_func = play_note, .arg = {29}},
  {.key = KEY_FIVE,  .pressed_func = play_note, .arg = {30}},
  {.key = KEY_T,     .pressed_func = play_note, .arg = {31}},
  {.key = KEY_SIX,   .pressed_func = play_note, .arg = {32}},
  {.key = KEY_Y,     .pressed_func = play_note, .arg = {33}},
  {.key = KEY_SEVEN, .pressed_func = play_note, .arg = {34}},
  {.key = KEY_U,     .pressed_func = play_note, .arg = {35}},
  {.key = KEY_I,     .pressed_func = play_note, .arg = {36}},
  {.key = KEY_Z,     .pressed_func = play_note, .arg = {36}},
  {.key = KEY_NINE,  .pressed_func = play_note, .arg = {37}},
  {.key = KEY_S,     .pressed_func = play_note, .arg = {37}},
  {.key = KEY_O,     .pressed_func = play_note, .arg = {38}},
  {.key = KEY_X,     .pressed_func = play_note, .arg = {38}},
  {.key = KEY_ZERO,  .pressed_func = play_note, .arg = {39}},
  {.key = KEY_D,     .pressed_func = play_note, .arg = {39}},
  {.key = KEY_P,     .pressed_func = play_note, .arg = {40}},
  {.key = KEY_C,     .pressed_func = play_note, .arg = {40}},
  {.key = KEY_V,     .pressed_func = play_note, .arg = {41}},
  {.key = KEY_G,     .pressed_func = play_note, .arg = {42}},
  {.key = KEY_B,     .pressed_func = play_note, .arg = {43}},
  {.key = KEY_H,     .pressed_func = play_note, .arg = {44}},
  {.key = KEY_N,     .pressed_func = play_note, .arg = {45}},
  {.key = KEY_J,     .pressed_func = play_note, .arg = {46}},
  {.key = KEY_M,     .pressed_func = play_note, .arg = {47}},
  {.key = KEY_COMMA, .pressed_func = play_note, .arg = {48}},
};
```

Fonte: de autoria própria

Ao apertar “F1”, inicia-se o tutorial. O tutorial funciona por meio de frases que aparecem embaixo do teclado musical que dão dicas sobre o que é possível fazer no programa. O tutorial mostra-se interessante de ser abordado por mostrar a maior parte das funcionalidades desse programa. A Figura 5 retrata as etapas do tutorial.

**Figura 5 – Etapas do tutorial (ordem das imagens: da esquerda para direita, de cima para baixo)**




Fonte: de autoria própria

A etapa 1 exibe uma mensagem que pede para o usuário utilizar o teclado do computador ou o mouse para tocar as teclas do teclado musical.

Após o usuário interagir com o teclado musical, percebe-se que as teclas apertadas são reproduzidas em azul e o intervalo ou acorde é exibido no canto superior esquerdo da janela do programa. A figura da etapa 2 mostra o usuário apertando três notas que formam o dó menor.

Além de poder tocar o teclado musical também é possível marcar as teclas do teclado musical para que permaneçam ativas. Isso se dá por meio de segurar a tecla “Ctrl” (*Control*) enquanto toca as teclas ou pelo clique do botão direito do mouse em cima das teclas do teclado musical.

As teclas marcadas pelo modo citado são representadas em verde e podem ser transpostas, isto é, podem mudar de posição em um semitom à esquerda ou à direita com as setas esquerda e direita do teclado de computador, respectivamente. A figura da etapa 3 mostra as três notas que tinham sido simplesmente pressionadas na figura da etapa anterior agora estão marcadas, formando o dó menor com teclas verdes.

A figura da etapa 4 mostra que o usuário aperta a seta direita, pois o acorde dó menor, que estava com as notas marcadas, agora foi transposto e o programa exibe dó sustenido menor (notas a um semitom de distância do dó menor. Por padrão, os acordes são representados em uma notação em português ou inglês. Na etapa 4, a mensagem pede para o usuário trocar a notação dos acordes com a tecla “Tab”, que nesse caso, seria trocar para a notação em inglês abreviada.

O dó sustenido menor que antes estava em uma notação em português, na figura da etapa 5 foi representado como “C#m”, que significa o acorde em inglês “C sharp minor”. A mensagem da etapa 5 pede para o usuário trocar a representação das notas acidentais entre sustenido e bemol (“♯” e “♭”, respectivamente).

O acorde anterior agora se transformou em “D♭m”, equivalente a “Ré bemol menor”. Este acorde é composto pelas mesmas notas do acorde da etapa anterior, tendo como a única diferença a representação da notação do acorde. Por fim, a mensagem da etapa 6 pede para o usuário usar a tecla barra de espaço para limpar todos os acordes marcados. Depois disso, o tutorial acaba.

### 3.3. Limitações

Esse *software* possibilita a visualização de intervalos e acordes no instrumento de teclado virtual, o que pode ajudar no aprendizado de teoria musical. Entretanto, para exibir o nome dos acordes e intervalos, o programa procura esses em um vetor associativo, no qual é registrado os acordes e intervalos, com os seus nomes pré-computados. Esse vetor tem o nome “chords” e é definido em “src/chord_finder.c”. Isso significa que o programa não pode exibir virtualmente qualquer acorde ou intervalo, mas somente aqueles que estão armazenados nesse vetor.

## 4. Considerações finais

Conclui-se, então, que o objetivo do artigo foi atingido, mas de forma limitada, como foi dito no tópico “3.3”. Por isso, uma possível melhoria para esse projeto seria a

implementação de um algoritmo que, por meio de princípios, possa descrever qualquer concatenação de notas, mas não somente as pré-definidas no programa.

Outra possível melhoria seria a implementação da separação de conceitos (do inglês *separation of concerns, SoC*). Na programação, a separação de conceitos é importante porque ajuda a manter o código limpo e fácil de entender. Quando um programa é escrito, pode-se ter várias coisas acontecendo ao mesmo tempo, como entrada de dados, processamento, saída de dados, etc. Se tudo isso ficar misturado, o código fica confuso e difícil de ler. A separação de conceitos ajuda a dividir o código em partes menores e mais fáceis de entender. Por exemplo, pode-se ter uma parte do código que lê os dados de entrada, outra parte que processa esses dados e uma terceira parte que exibe os resultados. Assim, fica mais fácil entender o que cada parte do código está fazendo e como elas se relacionam.

O fornecimento do *software* pré-compilado para os sistemas operacionais e a inclusão de uma documentação são outras possíveis melhorias. O *software* desenvolvido tem o seu código-fonte aberto [5] e deixa em aberto as possíveis melhorias.

## 5. Referências


[1] ARCH LINUX. **raylib package**. 4.2.0-2. [S. l.], 2022. Disponível em: https://archlinux.org/packages/community/x86_64/raylib/. Acesso em: 25 jan. 2023.

[2] BREWER, John. **MinUnit:** a minimal unit testing framework for C. [S. l.]: Jera Design LLC, [20--]. Disponível em: https://jera.com/techinfo/jtns/jtn002. Acesso em: 5 jan. 2023.

[3] **CHOOSEALICENSE.COM**. MIT License. [S. l.]: GitHub, 2023. Disponível em: https://choosealicense.com/licenses/mit. Acesso em: 05 jan. 2023.

[4] CONSTANTE, Rogério Tavares; ARAÚJO, João Teixeira; SCHIAVONI, Flávio Luiz. **Harmonia:** uma ferramenta de aprendizagem musical. Pelotas: XXIX Congresso da Associação Nacional de Pesquisa e Pós-Graduação em Música, 2019.

[5] FRANCESCHETTI, Daniel de Lima. **musical**. [S. l.], 2022. Repositório GitHub; Licença MIT. Disponível em: https://github.com/danielsource/musical. Acesso em: 5 jan. 2023.

[6] **GNU Compiler Collection**. [S. l.]: Free Software Foundation, 2022. Licença GPL3. Disponível em: https://www.gnu.org/software/gcc/. Acesso em: 5 jan. 2023.

[7] **GNU Coreutils**. [S. l.]: Free Software Foundation, 2022. Licença GPL3. Disponível em: https://www.gnu.org/software/coreutils. Acesso em: 5 jan. 2023.

[8] **GNU Make**. [S. l.]: Free Software Foundation, 2022. Licença GPL3. Disponível em: https://www.gnu.org/software/make. Acesso em: 5 jan. 2023.

[9] GNU Project. GNU Coding Standards, 7.2 **Makefile Conventions**. [S. l.]: Free Software Foundation, 2022. Licença GNU Free Documentation. Disponível em: https://www.gnu.org/prep/standards/html_node/Makefile-Conventions.html. Acesso em: 5 jan. 2023.

[10] HADLOW, Mike. **Guitar Dashboard**. [S. l.], 2021. Disponível em: https://guitardashboard.com. Acesso em: 05 jan. 2023.

[11] ISO/IEC JTC 1/SC 22. **ISO/IEC 9899:1999**, Programming languages — C. [S. l.]. 1999.

[12] LACERDA, Osvaldo Costa de. **Compêndio de Teoria Elementar da Música**. São Paulo: Ricordi, 1967.

[13] LLVM Developer Group. **Clang**. [S. l.], 2022. Licença Apache License 2.0 com exceções da LLVM. Disponível em: https://clang.llvm.org/. Acesso em: 5 jan. 2023.

[14] MED, Bohumil. **Teoria da Música**. 4. ed. Brasília: MusiMed, 1996.

[15] **Man Page Lookup**. [S. l.], 2023. Manual do grupo Clang / LLVM. Disponível em: https://manpage.me/?q=cc. Acesso em: 05 jan. 2023.

[16] **OPENGL Wiki - FAQ**. [S. l.]: Khronos Group, 2023. Disponível em: https://www.khronos.org/opengl/wiki/FAQ#What_is_OpenGL?. Acesso em: 05 jan. 2023.

[17] RAYLIB. **v4.2 quick reference card.** [S. l.], 2023. Disponível em: https://www.raylib.com/cheatsheet/cheatsheet.html. Acesso em: 5 jan. 2023.

[18] **RAYLIB:** A simple and easy-to-use library to enjoy videogames programming. [S. l.], 2023. Licença Zlib. Disponível em: https://www.raylib.com. Acesso em: 5 jan. 2023.

[19] VANZELA, Alexsander Vanzela; OLIVEIRA, Leida Calegário de; CARVALHO, Marivaldo Aparecido de. **Música, tecnologia e educação musical:** a guitarra em foco. Curitiba: Música em Perspectiva, dez. 2016. v.9 n.2, p. 121-133.