UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

IMPLEMENTAÇÃO DE UM PROTÓTIPO PARA GERAÇÃO DE 2ª VOZ MUSICAL UTILIZANDO REGRAS DE HARMONIA

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

ALESSANDRO KOTLINSKY

BLUMENAU, DEZEMBRO/2001

2001/2-23

IMPLEMENTAÇÃO DE UM PROTÓTIPO PARA GERAÇÃO DE 2ª VOZ MUSICAL UTILIZANDO REGRAS DE HARMONIA

ALESSANDRO KOTLINSKY

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Paulo Cesar Rodacki Gomes — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Paulo Cesar Rodacki Gomes

_____

Prof. Roberto Heinzle

_____

Prof. Miguel Wisintainer

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

AGRADECIMENTOS

A Deus por todo o meu talento, força de vontade e perseverança.

A meus pais e toda a minha família que sempre me apoiaram em minhas decisões.

Ao professor Paulo Rodacki pela sua sabedoria, paciência, dedicação e incentivo.

A banda Before Eden, formada por excelentes músicos e amigos que tornam possível a expansão do meu conhecimento.

A minha eterna namorada e complemento, Fernanda Makoski, por todo o seu carinho e incentivo.

Muito obrigado.

RESUMO

A finalidade deste trabalho de conclusão de curso é apresentar um estudo sobre a interface MIDI – *Musical Instrument Digital Interface*, com o objetivo de desenvolver um protótipo de software capaz de criar uma segunda voz musical, utilizando regras de harmonia. É feito um estudo em teoria musical, visando abranger da melhor forma possível todos os seus conceitos e especificações. A geração de segunda voz musical foi feita através de um método de parametrização dos modos musicais. A ferramenta utilizada para a construção do sistema e definições das regras foi a linguagem de programação *Delphi 5.0*.

ABSTRACT

The main purpose of this course conclusion work is to show a study about MIDI - *Musical Instrument Digital Interface,* with the intention of developing a software prototype able to create a second musical voice, using harmony rules. A deep study in musical theory is done, attempting to reach, in the best possible way, all its concepts and specifications. The second voice generation was made through a parametric method of the musical modes. The tool used for the construction of the prototype and rules definitions was the programming language *Delphi 5.0*.

1 INTRODUÇÃO

Música é a arte de combinar sons, estruturar determinadas notas musicais com o objetivo de criar melodias que podem expressar os mais diversos tipos de sentimentos existentes na mente humana. Para se alcançar tal objetivo, existe uma necessidade de combinar esse sons de uma forma inteligente, buscando assim uma ordem para a execução das notas, individualmente ou simultaneamente, para expressar o verdadeiro propósito do artista.

Para exprimir qualquer sentimento, ou descrever por meio da música qualquer quadro da natureza, torna-se imprescindível a participação em comum desses três elementos: melodia, ritmo e harmonia. Conforme (Maske, 2000), a melodia consiste na sucessão dos sons formando sentido musical. O ritmo é o movimento dos sons regulados pela sua maior ou menor duração e a harmonia consiste na execução de vários sons ouvidos ao mesmo tempo, observados as leis que regem os agrupamentos dos sons simultâneos, formando acordes. O conceito de acorde diz que três ou mais notas que soam simultaneamente formam um acorde (Maske, 2000).

Os sons musicais são representados graficamente por sinais chamados notas, e a escrita da música dá-se o nome de notação musical. As notas são sete: **Dó-Ré-Mi-Fá-Sol-Lá-Si**. Estas notas são escritas na pauta musical ou pentagrama, que é a reunião de cinco linhas horizontais, paralelas e eqüidistantes, formando entre si quatro espaços.

Por causa dessa fusão de elementos, essencial para os músicos estruturarem uma música e que muitas vezes complexa, veio à tona a possibilidade de implementar um software para melhorar o desempenho dos músicos em relação às suas composições e assim facilitar a estruturação de uma música e suas melodias.

Este software será encarregado de gerar uma segunda voz musical a partir de uma melodia musical pré-definida. Dada uma partitura musical com apenas uma voz melódica, o software se encarrega de gerar uma segunda voz, de acordo com regras de harmonia, facilitando assim, o trabalho de composição e harmonização de qualquer peça musical. A voz adicional (segunda voz) obedecerá os conceitos da teoria musical, podendo variar desde um

intervalo[1] de 2ª até um intervalo de 7ª, progressivamente. Um exemplo, seria tendo a nota **Sol** em uma partitura musical mais o acorde de **Dó** maior, solicitando-se a criação de uma 3ª no modo jônio[2], a segunda voz musical será a nota **Si**, pois esta é a terceira nota à partir da nota **Sol** que obedece os parâmetros colocados anteriormente.

Maske (2000) deixou como sugestão para futuras implementações de seu trabalho "um estudo mais aprofundado sobre harmonia musical para que fossem utilizadas todas as maneiras possíveis de notas cabíveis com a primeira voz musical". Para isto, seria necessária a adição de novas regras em um sistema especialista para a análise musical.

O Formato de arquivo utilizado para a leitura das melodias e a criação das demais vozes é o *Interface Digital de Instrumentos Musicais* (MIDI) . Segundo Gontijo (1998), MIDI é uma "linguagem eletrônica", composta de um grande conjunto de comandos musicais, com a qual instrumentos eletrônicos controlam uns aos outros. Da mesma forma que o *modem* passa bytes de dados entre o computador e o provedor Internet, usando um protocolo chamado TCP/IP, instrumentos musicais eletrônicos passam bytes de dados usando o protocolo MIDI.

Em relação ao trabalho de Maske (2000), aumentou-se a capacidade de análise musical para a criação de uma segunda voz. Eliminou-se o sistema especialista de seu protótipo, abandonando assim a premissa sobre a qual o seu trabalho foi construído, a da aplicabilidade dos sistemas especialistas na música. Em seu lugar foi implementado um método de parametrização para diversos tipos de melodias, isto é, cada uma podendo pertencer à um diferente campo harmônico. Em termos práticos, um campo harmônico representa a parte principal da definição do estilo de uma melodia. Hoje em dia, na música popular ocidental, tem-se diversos estilos musicais bem definidos, tais como o *Jazz*, o *Rock*, o *Funk*, entre outros. Cada um caracteriza-se pelo uso constante de determinados tipos de escalas e seus respectivos modos. Estas, por sua vez, pertencem à um determinado campo harmônico, caracterizando o estilo da música.

---

[1] A definição do conceito de intervalos musicais encontra-se no item 2.2

[2] Modo pertencente aos acordes maiores, item 2.4

O trabalho desenvolvido por Maske (2000) restringe-se à criação da segunda voz musical sem o uso do conceito de intervalos. Isto torna o seu trabalho não muito fiel às especificações da teoria musical, já que a segunda voz criada em seu protótipo sempre será uma das notas definidas na base do seu sistema especialista, onde as regras foram formuladas conforme o seu gosto musical e não seguindo alguma especificação teórica musical. Além disso, a criação da segunda voz torna-se extremamente lenta devido ao uso do sistema especialista.

Assim, surgiu um trabalho onde aplicou-se uma nova tecnologia ao trabalho de Maske (2000), eliminando o uso do sistema especialista. Em seu lugar é implementado um método de parametrização para a criação de uma segunda voz musical utilizando o conceito de intervalos, o que, em relação ao trabalho de Maske (2000), torna o presente trabalho muito mais fiel às definições da teoria musical. A implementação de um sistema baseado em uma formulação paramétrica para a geração de segunda voz musical também visa obter um desempenho computacional mais eficiente. A extensão deste trabalho compreende o aprofundamento das técnicas de criação automática de harmonia musical.

1.1 OBJETIVOS DO TRABALHO

O objetivo principal deste trabalho é desenvolver uma aplicação semelhante àquela do trabalho de Maske (2000), porém, eliminando o uso da tecnologia de sistemas especialistas. Como produto desta nova implementação, espera-se que o software se torne uma ferramenta mais poderosa, capaz de criar um arranjo musical com uma segunda voz musical através do conceito de intervalos, obedecendo ao campo harmônico da melodia e a partir de uma única linha melódica definida pelo usuário.

Os objetivos específicos do trabalho são:

a) implementação de um método de parametrização matemática para a implementação de uma segunda voz musical utilizando do conceito de intervalos, eliminando o sistema especialista;
b) Adição de uma opção que permitirá a escolha do tipo de segunda voz musical a ser criada, podendo variar de uma 2ª até uma 7ª, de forma ascendente.

1.2 ESTRUTURA DO TRABALHO

O trabalho está organizado em seis capítulos, descrevendo:

- Capítulo 1 – Introdução ao trabalho, com breve descrição do contexto desde, seus objetivos e sua organização;
- Capítulo 2 – Fundamentação teórica relativa a Música, abrangendo desde conceituação e harmonia;
- Capítulo 3 – Fundamentação teórica sobre MIDI (*Musical Instrument Digital Interface*), compreendendo conceituação, formato e especificação;
- Capítulo 4 – Fundamentação teórica relativa aos métodos desenvolvidos e técnicas aplicadas no protótipo, apresentação do protótipo desenvolvido;
- Capítulo 5 – Discussões e comparações do presente trabalho com trabalhos anteriores;
- Capítulo 6 – Conclusões e sugestões para futuros trabalhos.

2 MÚSICA

Conforme Chediak (1986), Música é a arte dos sons, constituída de melodia, ritmo e harmonia. A melodia corresponde à uma sucessão de sons musicais combinados. Já o ritmo é a duração, a acentuação dos sons e das pausas e a harmonia significa a combinação dos sons simultâneos.

A formação dos sons é feita através de movimentos de corpos vibratórios (Chediak, 1986). Para se produzir um som musical, necessita-se de uma fonte sonora, isto é, um corpo que produz sons ao vibrar. Os corpos vibrantes dos instrumentos são corda esticada (violão, violino, piano etc...), coluna de ar (flauta, trompete etc...) ou membrana (tamborim, cuíca etc...) Por exemplo, ao tocar uma corda do violão, observa-se que ela se movimenta de um lado para outro um determinado número de vezes por segundo, emitindo um som. A esse movimento é dado o nome de vibração, medida em Hz (Hertz) - Ciclos por segundo.

Quanto maior ou menor o número de vibrações por segundo, mais agudo ou grave será o som. E quanto maior a amplitude do movimento vibratório, maior será a intensidade do som produzido. O ouvido humano é capaz de perceber sons que vão aproximadamente de 20 Hertz a 18 mil Hertz. Os sons fundamentais se localizam numa faixa aproximada de 32 a 4 mil Hertz. Por exemplo, a nota **Lá** do diapasão tem 440 Hertz.

Acima de 4 mil Hertz encontram-se os harmônicos agudos que enriquecem o timbre do instrumento, dando mais brilho ao som gerado. Sem esse harmônicos os sons seriam opacos. Assim, sem os harmônicos não se tem os timbres característicos da cada instrumento.

Ainda seguindo Chediak (1986), as propriedades físicas dos sons são compostas por três elementos: altura, intensidade e timbre. A altura é a propriedade do som ser grave, médio ou agudo. Já a intensidade é a propriedade do som ser fraco ou forte. Caracteriza-se pela amplitude da vibração. Por exemplo, quando tocamos uma corda com mais força, a amplitude da vibração é maior e consequentemente o volume do som também será maior. O Timbre é a qualidade do som que nos permite reconhecer sua origem e é através dele que diferenciamos o som dos vários instrumentos. O timbre está relacionado com a série harmônica, produzida pelo som emitido.

2.1 ORIGEM DA MÚSICA

Conforme Plate (2000), como as primeiras manifestações musicais não deixaram vestígios, é praticamente impossível saber exatamente quando o homem descobriu a música. Alguns estudiosos nem tentam, outros enfrentam o problema com base naquilo que se sabe sobre a vida humana na Pré- história e preenchem as lacunas com certa dose de imaginação. Mas nenhuma hipótese comprova com exatidão o momento em que os primitivos começaram a fazer arte com os sons.

Tudo indica que o homem das cavernas dava à sua música algum sentido religioso. Provavelmente, interpretava-a como um presente dos deuses e atribuía-lhe funções mágicas. Junto com a dança, ela assumia para toda a tribo um caráter de ritual, e assim reverenciavam o desconhecido, agradecendo-lhe a fertilidade da terra, a abundância da caça e dos homens. Batendo as mãos e os pés, procuravam formar algum tipo de sincronia musical, que hoje chamamos de ritmo. Buscavam também celebrar fatos da sua realidade: vitórias na guerra, descobertas surpreendentes. Mais tarde, em vez de usar só as mãos e os pés, passaram a ritmar suas danças com pancadas na madeira, primeiro simples e depois trabalhadas para soarem de formas diferentes. Surgia, assim, o instrumento de percussão.

Os barulhos da natureza deviam fascinar o homem desses tempos, dando-lhe vontade de imitar o sopro do vento, o ruído das águas, o canto dos pássaros. Mas, para isto, o ritmo não bastava, e o artesanato ainda não permitia a invenção de instrumentos melódicos, de modo que estranhos sons tirados da garganta devem ter constituído uma forma rudimentar de canto. Juntamente com o ritmo, o canto resultou na mistura de palmas e roncos, pulos e uivos, batidas e berros. Era o que estava ao alcance do homem primitivo. E terá sido um estilo que resistiu a séculos.

Contudo, segundo os atuais conceitos de música, essas tentativas de expressão foram demasiadamente pobres para se enquadrarem na categoria de arte musical. Mas, do ponto de vista histórico, elas tiveram uma importância enorme. Porque a sua rítmica elementar acompanhou o homem à medida que este se espalhava sobre a Terra, formando culturas e civilizações. e evoluiu com ele, refletindo todas as transformações que a humanidade viveu até chegar a ser como é agora.

2.1.1 OS PRIMEIROS ELEMENTOS

Ainda segundo Plate (2000), a noção que hoje se tem da música como "uma organização temporal de sons e silêncios" não é nova. Civilizações muito antigas já se aproximaram dela, descobrindo os elementos musicais e ordenando-os de maneira sistematizada. Os historiadores têm encontrado inscrições as quais indicam que um caráter nitidamente ritualístico impregnava a maior parte da criação musical da antiguidade.

Por muito tempo as formas instrumentais permaneceram sub-desenvolvidas. Predominava a música vocal. Essa forma, adicionando à música o reforço das palavras, era mais comunicativa e as pessoas assimilavam-na melhor. Assim se explica o grande desenvolvimento que atingiu entre os antigos.

Os povos de origem semita cultivavam a expressão musical, tornando-a bastante elaborada. Os que habitavam a Arábia, principalmente, distinguiram-se pela criatividade. Possuíam uma ampla variedade de instrumentos e dominavam diferentes escalas. Segundo parece, tocavam sobretudo para dançar, pois foi entre eles que surgiu a "Suíte de Danças", um gênero que sobrevive ainda hoje.

A Bíblia mostra que também os judeus tinham a música como hábito. Davi fala sobre ela nos "Salmos", e diversas outras passagens bíblicas contêm menções a respeito.

Na China, o peculiar era a própria música, devido à sua monumentalidade. Os chineses utilizavam nada menos que 84 escalas (o sistema tradicional da música ocidental dispunha de apenas 24) e a variedade da sua instrumentação era imensa. E já por volta do ano 2255 a.C. o domínio sobre a expressão musical atingia tal perfeição entre eles, que sua influência se estendia por todo o Oriente, moldando a música do Japão, da Birmânia, da Tailândia e de Java.

Mas indiscutivelmente, foram os gregos que estabeleceram as bases para a cultura musical do Ocidente. A própria palavra música nasceu na Grécia, onde "Mousikê" significa "A Arte das Musas", abrangendo também a poesia e a dança. O ritmo era o denominador comum das três artes, fundindo-as numa só. Dessa forma a Lírica era um gênero poético, mas seu traço principal era a melodia e até seu nome deriva de um instrumento musical - a Lira.

Como os demais povos antigos, os gregos atribuíam aos deuses sua música, definindo-a como uma criação integral do espírito, um meio de alcançar a perfeição.

Seu sistema musical apoiava-se numa escala elementar de quatro sons - o Tetracorde. Da união de dois tetracordes formaram-se escalas de oito notas, cuja riqueza sonora já permitia traçar linhas melódicas. Estas escalas mais amplas, os Modos, tornaram o sistema musical grego conhecido posteriormente como Modal.

O canto prendia-se a uma melodia simples, a Monodia, pois os músicos da Grécia ignoravam as combinações simultâneas de sons (harmonias). Mas nem por isso deixavam de caracterizar com seus Modos um sentido moral, o Ethos, tornando os ritmos sensuais, religiosos, guerreiros, e assim por diante.

Uma vez que os ritos religiosos quase não mudavam, conservando a tradição, com o tempo criaram-se melodias-padrão, muito fáceis e conhecidas de todos. Eram os Nomoi, cujo acompanhamento se fazia com a Cítara e o Aulos. A cítara descendia da lira e, como ela, tinha cordas. O aulos era um instrumento de sopro, ancestral do nosso oboé.

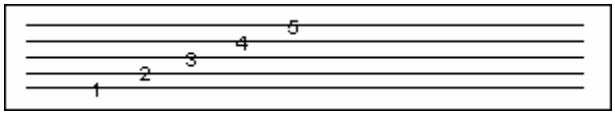
Partindo dos Nomoi, a música da Grécia evoluiu para a lírica solista, o canto conjunto e o solo instrumental. Depois, vieram as grandes tragédias inteiramente cantadas, que marcaram o apogeu da civilização helênica (do século VI ao século IV a.C.).

Daí por diante, a decadência do povo encaminhou a música da Grécia para o individualismo e o culto às aparências. Parecendo prever a dominação que lhes seria imposta pelos romanos, os gregos ironizavam a sua própria destruição.

2.2 TEORIA DA MÚSICA

De acordo com Maske (2000), a música é escrita em pautado de 5 linhas horizontais, eqüidistantes, formando entre si 4 espaços que se denomina pauta ou pentagrama. As linhas são contadas de baixo para cima. É nas linhas e nos espaços que se escrevem as notas representativas dos sons musicais (figura 1).

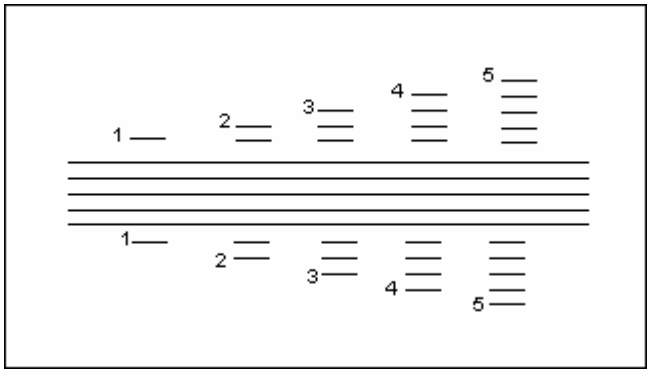
FIGURA 1 – PAUTA MUSICAL



Fonte: Maske (2000)

A pauta, entretanto, não é suficiente para conter todos os sons musicais. Por isso são usadas linhas e espaços acima ou abaixo da pauta para que se possa anotar todos os sons nas várias alturas. O número de linhas ou espaços suplementares não é limitado, contudo, não é comum empregar-se mais de 5 (figura 2).

FIGURA 2 – LINHAS E ESPAÇOS SUPLEMENTARES



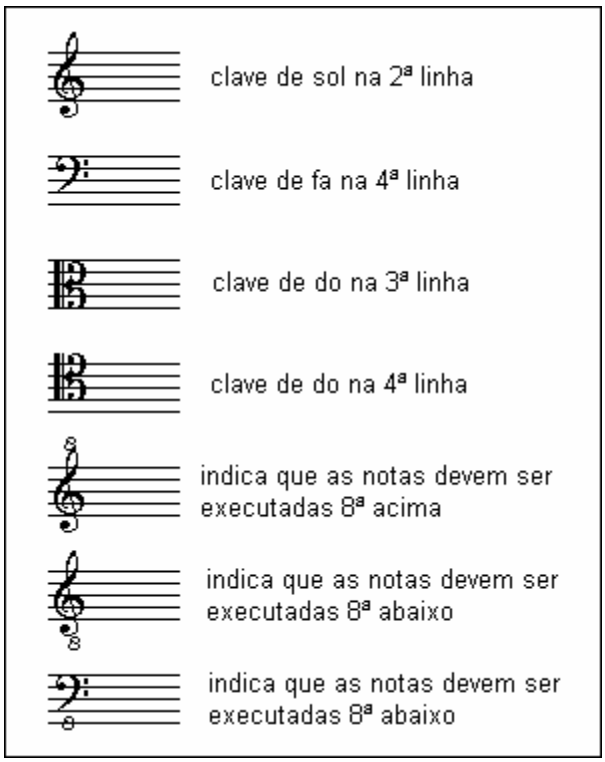
Fonte: Maske (2000)

Segundo Chediak (1986) **clave** é um sinal usado no início da pauta para determinar o nome e a altura das notas. São três os tipos de clave: **Sol**, **Fá** e **Dó**. A Clave de **Sol**, determina o local da nota "**Sol**", anotada na segunda linha. Já a Clave de **Fá** determina o local da nota **Fá**, anotada na Quarta e terceira linhas. Por fim, A Clave de **Dó** determina o local da nota **Dó**, podendo ser usada na primeira, segunda, terceira e quarta linhas.

Por ser o número de notas bem superior ao número de linhas e espaços da pauta (inclusive suplementares) faz-se necessário o uso das três claves. A clave de **Sol** é usada para os sons agudos (soprano). Por sua vez, a clave de **Fá** para os sons graves(barítono e baixo) enquanto que a de **Dó** para os médios (meio soprano, contralto e tenor). Modernamente a clave de **Fá** na terceira linha e a clave de **Dó** na primeira linha não são usadas. As claves mais usadas são as de **Sol**, de **Fá** na Quarta linha e a de **Dó** na terceira. Um exemplo de instrumentos que utilizam a clave de **Sol** são os de sons agudos como o violino, flauta,

trompete, oboé, gaita, violão, clarinete, cavaquinho e bandolim. Outros que utilizam a Clave de **Fá** são os instrumentos com sons graves, como o contra-baixo, trombone, violon-cello, fagote e tuba. Por último, a Clave de **Dó**, que é de pouco uso, é utilizada por instrumentos de sons médios como a viola.

FIGURA 3 – CLAVES MAIS USADAS



Fonte: Maske (2000)

Conforme Chediak (1986), **notação musical** é a representação gráfica da música. Existem sete notas naturais: **Dó**, **Ré**, **Mi**, **Fá**, **Sol**, **Lá**, **Si**, que correspondem aos monossílabos C, D, E, F, G, A, B, respectivamente.. Estas notas podem ser alteradas de forma ascendente ou descendente, tomando então, o lugar de uma de suas notas vizinhas, usando respectivamente, os sinais # (sustenido) e b (bemol), completando, assim, a série das doze notas que compõem a escala temperada ocidental, isto é, sete notas naturais e cinco alteradas.

Na verdade, todas as sete notas podem se alteradas, mas apenas cinco resultariam em novos sons. O **Mi#** e **Si#** têm som de **Fá** e **Dó**, respectivamente.

Os sons musicais têm durações diferentes. Essas durações são os valores, representados pelas figuras gráficas de notação musical. Temos ainda, para cada figura de som, uma correspondente, usada nos momentos de silêncio. São as pausas.

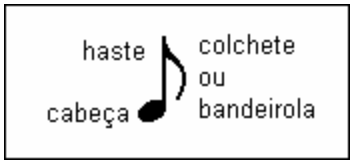
FIGURA 4 – DIVISÃO PROPORCIONAL DOS VALORES

Figura	Nota e pausa
Semibreve	
Mínima	
Semínima	
Colcheia	
Semicolcheia	
Fusa	
Semifusa	

Fonte: Chediak (2000)

Uma nota é dividida em: cabeça, haste e colchete (figura 5).

FIGURA 5 – PARTES DE UMA NOTA

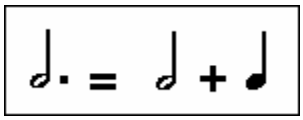


Fonte: Chediak (2000)

Quando se escrevem duas ou mais colcheias, semicolcheias, fusas ou semifusas consecutivas, usa-se também substituir os colchetes por barras horizontais, ficando as notas unidas em grupos.

Pode-se acrescentar um ponto numa nota (figura 6), chamado ponto de aumento, que prolongará a duração na metade de seu valor. Inclusive pode haver pontos duplos ou triplos. Cada um deles aumenta a duração na metade do valor anterior, razão pela qual, qualquer nota pode receber um número infinito de pontos (contudo, nunca chegará a duplicar o valor da nota original). Este ponto de aumento também pode ser usado para as pausas.

FIGURA 6 – PONTO DE AUMENTO



Fonte: Chediak (2000)

De acordo com Chediak (1986), **compasso** é a divisão de um trecho musical em pequenas partes de duração com séries regulares de tempos. Os tempos são agrupados em porções iguais, de dois em dois, de três em três ou de quatro em quatro, constituindo unidades métricas. Os compassos de 2 tempos são chamados **binários,** enquanto que os compassos de 3 tempos são chamados de **ternários**. Por fim, os compassos de 4 tempos são chamados de **quaternários**.

Conforme a figura 7, os compassos são separados por um traço vertical chamado de travessão ou barra simples.

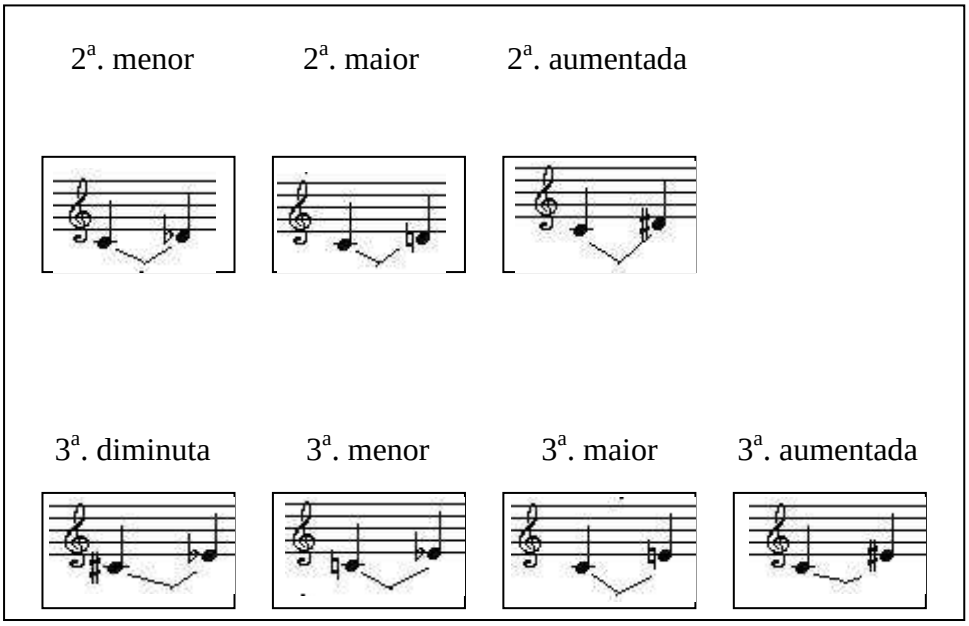
Figura 7 – Compassos separados por barra simples

Fonte: Chediak (2000)

Conforme Ribeiro (1965), **Intervalo** é a distância que há entre um e outro som. Há sete intervalos: 2ª. (segunda), 3ª. (terça), 4ª. (quarta), 5ª. (quinta), 6ª. (sexta), 7ª. (sétima) e 8ª. (oitava). Os intervalos da escala maior são todos maiores, menos a 4ª., a 5ª. e a 8ª., denominadas justas.

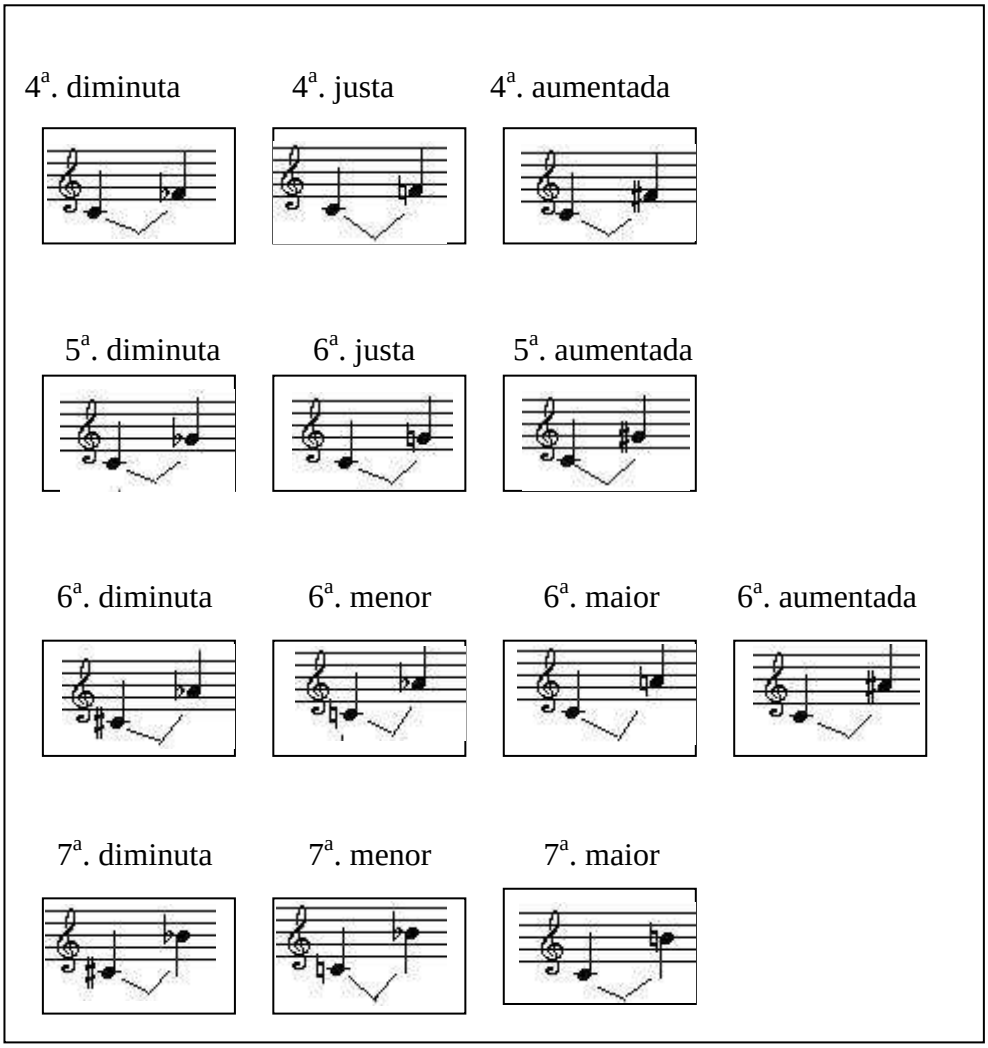
Os intervalos maiores elevados de meio tom tornam-se aumentados. Os intervalos maiores abaixados de meio tom tornam-se menores; abaixados de mais um semitom se tornam diminutos. Os intervalos justos (4ª., 5ª. e 8ª.) não podem ser maiores ou menores; se forem elevados de meio tom ficam aumentados; se forem abaixados de meio tom tornam-se diminutos.

QUADRO 1 – INTERVALOS DE 2A. ATÉ 3A.



Fonte: Ribeiro (1965)

QUADRO 2 – INTERVALOS DE 4A. ATÉ 7A.



Fonte: Ribeiro (1965)

Quadro, resumindo as alterações dos intervalos maiores:

QUADRO 3 – ALTERAÇÕES DOS INTERVALOS MAIORES

$2^a.$, $3^a.$, $6^a.$, $7^a.$ =	maiores
maior + semitom	aumentada
maior - semitom	menor
menor - semitom	diminuta
$4^a.$, $5^a.$, $8^a.$ =	justas
justa + semitom	aumentada
justa - semitom	diminuta

Ainda seguindo Ribeiro (1965), inverte-se um intervalo transportando o som grave à oitava superior, ou som agudo, à oitava inferior. Isso chama-se **Inversão dos Intervalos**, onde o número correspondente a dado intervalo, somado ao que lhe serve de inversão sempre será 9, invariavelmente.

QUADRO 4 - INVERSÃO DOS INTERVALOS

A inversão da $2^a.$ e a $7^a.$: 2 + 7 = 9
A inversão da $3^a.$ e a $6^a.$: 3 + 6 = 9
A inversão da $4^a.$ e a $5^a.$: 4 + 5 = 9
A inversão da $1^a.$ e a $8^a.$: 1 + 8 = 9

Já o número de semitons contidos num intervalo dado, somado ao que lhe serve de inversão, sempre corresponde a 12, conforme o quadro a seguir.

QUADRO 5 – SEMITONS CONTIDOS EM UM INTERVALO COM A SOMA DE SUA INVERSÃO

2ª. maior: 2 semitons
7ª. menor: 10 semitons
Total: 10 + 2 =12
3ª. maior: 4 semitons
6ª. menos: 8 semitons
Total: 4 + 8 =12
Etc.

Pela inversão, o intervalo diminuto se torna aumentado; o menor, maior e vice-versa. Os intervalos justos conservam a denominação de justos em suas inversões.

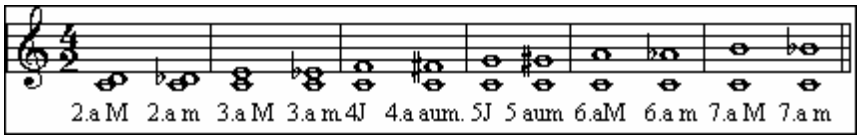
Conforme a direção em que seguem, os sons podem ser chamados de ascendentes, onde vão da parte grave para a aguda, ou de descendentes, que seguem da parte aguda para a grave.

2.3 HARMONIA

Segundo Maske (2000), harmonia é a relação vertical das notas que são executadas num mesmo momento. A harmonia pode ser ternária (sons formados pelo intervalo de terças, ex. **Dó**/**Mi**/**Sol** ou **Dó**/**Mib**/**Sol**), quaternária (formada por intervalos de quarta, ex. **Fá**/**Si**/**Mi** ou **Fá**/**Sib**/**Mi**), quinária (intervalo de quinta, inversão do de quarta, ex. **Si**/**Fá**/**Dó**), intervalo de segunda (ex. **Dó**/**Ré**/**Mi**) e assim por diante. É irrelevante se estes intervalos são maiores ou menores ou mesmo aumentados.

É básico para o estudo da Harmonia e para o estudo da Composição Musical, que se tenha em mente a seguinte tabela de intervalos (figura 8), correspondente aos graus dos tons com relação a uma nota fundamental, neste exemplo, esta nota é a de **Dó**.

FIGURA 8 – TABELA DE INTERVALOS



Fonte: Maske (2000)

Neste exemplo, a nota **Ré#** (ou **Mib**), forma com respeito à nota **Dó**, um intervalo de terça menor (3m).

2.4 ACORDES

Acordes são 3 ou mais notas tocadas ao mesmo tempo. Os acordes mais simples são formados por apenas 3 notas (tríades).

O acorde "Maior" é formado pelas notas dos graus 1, 3 e 5. Assim, o acorde de **Dó** Maior (CM) é formado pelas notas **Dó** (C), **Mi** (E) e **Sol** (G). O acorde "Menor" é semelhante ao Maior, porém a nota do grau 3 é reduzida em meio tom. Assim, o acorde de **Dó** Menor (Cm) é: **Dó** (C) , **Mib** (Eb) e **Sol** G. O acorde "Diminuto" é semelhante ao Menor, porém a nota do grau 5 também é reduzida em meio tom. Assim, o acorde **Dó** Diminuto (Cdim) é: **Dó** (C), **Mib** (Eb) e **Solb** (Gb). O acorde "Aumentado" é semelhante ao Maior, porém a nota do grau 5 é aumentada em meio tom. Assim, o acorde **Dó** Aumentado (C+) é: **Dó** (C), **Mi** (E) e **Sol#** (G#). O acorde com 7ª. (também chamado "Dominante") é obtido adicionando-se a nota do grau 7b ao acorde original. Assim o **Dó** com 7ª (C7) é: **Dó** (C), **Mi** (E), **Sol** (G) e **Sib** (Bb). Este tipo de acorde é o mais encontrado no Blues, Rock, Country e Pop.

Arpejos são as notas de um acorde tocadas separadamente, em seqüência, ao invés de todas juntas ao mesmo tempo. Conhecer os arpejos é muito útil para a improvisação e para manter a improvisação consistente com os acordes do acompanhamento.

Os acordes mais utilizados em uma canção são os do grau 1, 4 e 5. Ou seja, se o tom da música é C, os acordes mais utilizados serão os de C (C,E,G), F (F,A,C) e G (G,B,D). Freqüentemente os acordes são utilizados com acréscimo da 7ª. , como o acorde de C, por exemplo, que ficaria com as respectivas notas (C,E,G,Bb).

A tríade pode assumir 4 formas distintas (figura 9):

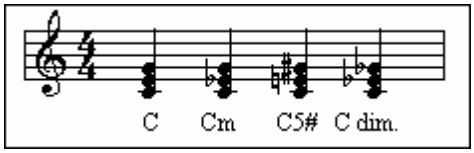
Maior;

Menor;

Diminuta;

Alterada.

FIGURA 9 – TIPOS DE TRÍADE



Fonte: Maske (2000)

2.4.1 ESCALAS DOS MODOS

As escalas dos modos, são: Iônico (também conhecido por Jônio), Dórico, Frígio, Mixolídio, Eólio e Lócrio. Esses modos, segundo Chediak (2000), têm nomes gregos porque a princípio se achava que correspondiam aos antigos modos da Grécia. Pesquisas mais recentes revelam que os modos gregos eram diferentes. Os modos Iônico e Eólio são os mesmos que os modos Maior e Menor natural respectivamente.

É importante ressaltar que os modos Iônico, Lídio e Mixolídio são maiores, porque o intervalo de terça em relação à tônica do acorde sempre será maior, enquanto que os modos Dórico, Frígio, Eólio e Lócrio são menores devido ao intervalo de terça que será sempre menor em relação à tônica do acorde.

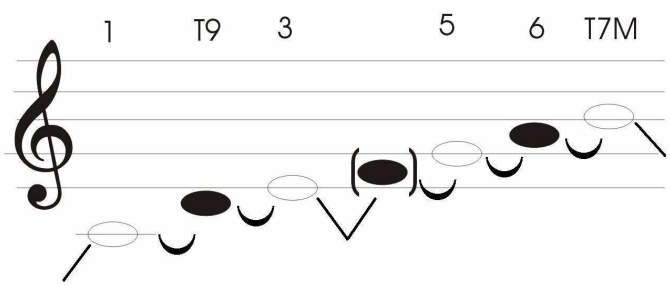
2.4.2 ESCALAS DOS ACORDES

Conforme Chediak (2000) as escalas dos acordes são o conjunto de notas disponíveis para formar harmonia ou linha de improviso. Os sete modos serão usados para associar a gama de notas das escalas dos acordes. Por exemplo, as notas disponíveis de um acorde de sétima da dominante (V7) coincidem com o modo Mixolídio e assim por diante.

A seguir serão mostradas as escalas dos acordes tomando como base a tonalidade de **Dó** maior.

a) Escala do modo Iônico (figura 10)

FIGURA 10 – ESCALA DO MODO IÔNICO

Grau	Acorde	IÔNICO	Outras possibilidades de acorde
17M	C7M		C6 C 6 9 C7M(9) C7M(6)

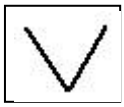
Fonte: Chediak (1986)

Este modo tem a mesma formação que a escala modelo do modo maior, isto é, intervalos de semitom entre os graus 3-4 e 7-8, e os de tom, entre os demais graus. Os números sobre as notas das escalas representam os intervalos a partir da nota fundamental do acorde. As notas brancas representam as notas básicas do acorde, enquanto que as notas entre parênteses não devem entrar na formação dos acordes (situação vertical) e no improviso (situação horizontal) devem ser usadas de passagem, isto é, sem que se pare nelas. A letra "T" significa nota de tensão (dissonante). Por último, para representar os intervalos de tom e semitom (meio-tom) entre as notas foram usados os símbolos das figuras abaixo:

FIGURA 11 – REPRESENTAÇÃO DO INTERVALO DE TOM

Fonte: Chediak (1986)

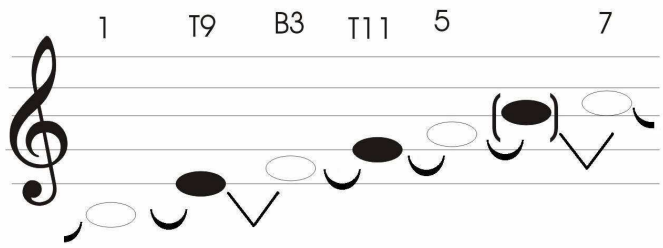
FIGURA 12 – REPRESENTAÇÃO DO INTERVALO DE SEMITON



Fonte: Chediak (1986)

b) Escala do modo Dórico (figura 13)

FIGURA 13 – ESCALA DO MODO DÓRICO

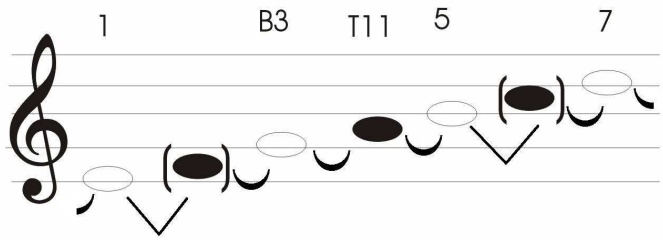
Grau	Acorde	DÓRICO	Outras possibilidades de acorde
Iim7	Dm7		Dm7(9) Dm7(11) Dm7(9 11)

Fonte: Chediak (1986)

No modo Dórico os intervalos de semitom ficam entre os graus 2-3 e 6-7, e de tom, entre os demais graus.

c) Escala do modo Frígio (figura 14)

FIGURA 14 – ESCALA DO MODO FRÍGIO

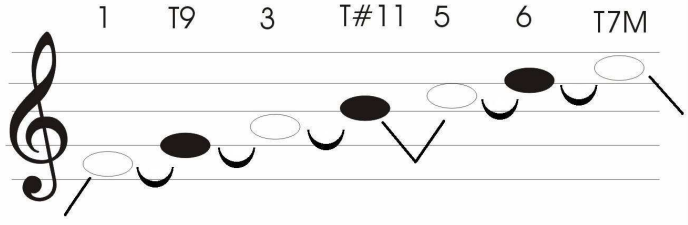
Grau	Acorde	FRÍGIO	Outras possibilidades de acorde
IIIm7	Em7		Em7(11)

Fonte: Chediak (1986)

No modo Frígio os intervalos de semitom ficam entre os graus 1-2 e 5-6, e de tom, entre os demais graus. Neste modo, a nona (não diatônica) é algumas vezes aceitável.

d) Escala do modo Lídio (figura 15)

FIGURA 15 – ESCALA DO MODO LÍDIO

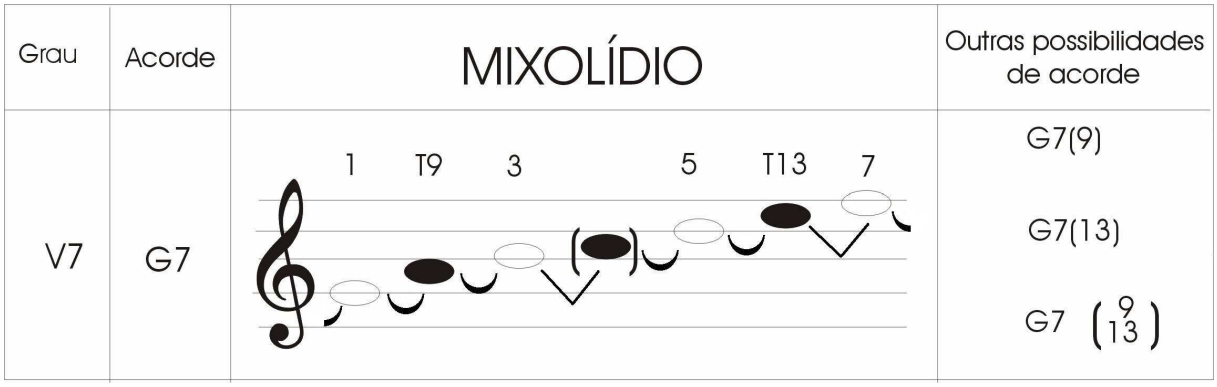
Grau	Acorde	LÍDIO	Outras possibilidades de acorde
IV7M	F7M		F7M(6) F7M(9) F7M(6 9) F7M(#11) F7M(9 #11) F6 F 6 9 F 6 9 (#11)

Fonte: Chediak (1986)

No modo Lídio os intervalos de semitom ficam entre os graus 4-5 e 7-8, e de tom entre os demais graus.

e) Escala do modo Mixolídio (figura 16)

FIGURA 16 – ESCALA DO MODO MIXOLÍDIO

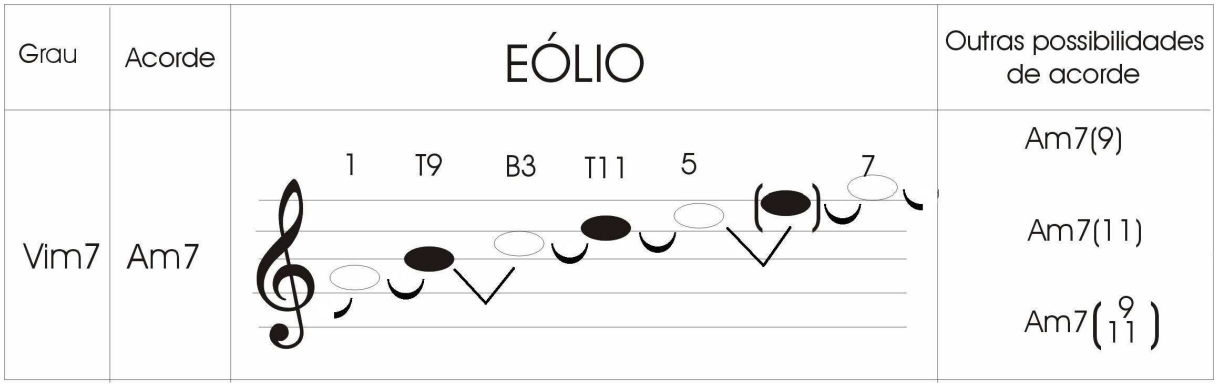


Fonte: Chediak (1986)

No modo Mixolídio os intervalos de semitom ficam entre os graus 3-4 e 6-7, e de tom entre os demais graus.

f) Escala do modo Eólio (figura 17)

FIGURA 17 – ESCALA DO MODO EÓLIO

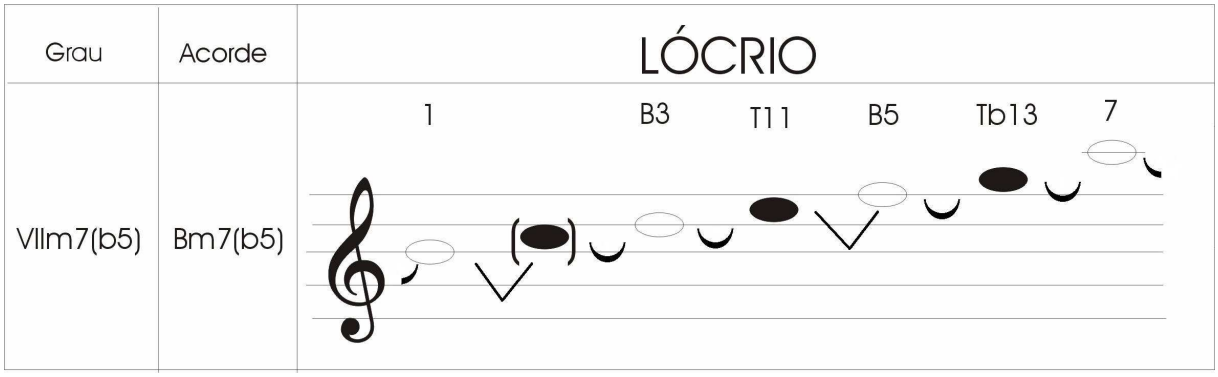


Fonte: Chediak (1986)

Este modo tem a mesma formação que a escala do modo menor natural, isto é, intervalos de semitom entre os graus 2-3 e 5-6, e de tom, entre os demais graus.

g) Escala do modo Lócrio (figura 18)

FIGURA 18 – ESCALA DO MODO LÓCRIO



Fonte: Chediak (1986)

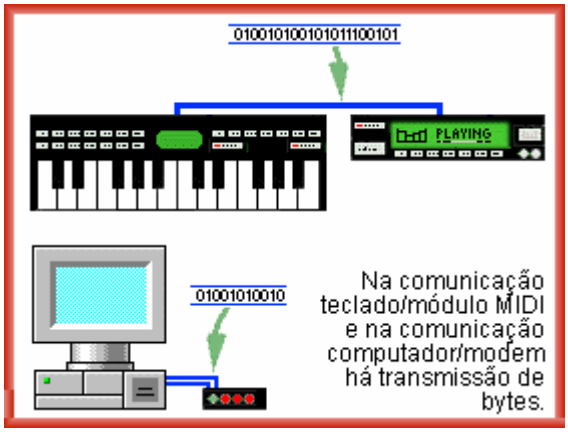
No modo Lócrio os intervalos de semitom ficam entre os graus 1-2 e 4-5, e de tom, entre os demais graus.

3 MIDI

Conforme Gontijo (2000) MIDI, que é sigla para "Musical Instrument Digital Interface", ou *Interface Digital para Instrumentos Musicais*, é uma "linguagem eletrônica", composta de um grande conjunto de comandos musicais, com a qual instrumentos eletrônicos controlam uns aos outros. Foi criado por alguns fabricantes de instrumentos musicais (inicialmente a Roland e Sequential Circuits, seguidos pela Yamaha, Korg e outros) que se reuniram em 1983 com o objetivo de criar esta "linguagem digital" onde todos os instrumentos musicais de teclado que fabricassem, a partir de então, pudessem trocar informações entre si.

MIDI permite que dispositivos eletrônicos (usualmente sintetizadores, mas também computadores, gravadores multipista, e até mesmo controladores de luzes para shows, videocassetes etc.) interajam e trabalhem em sincronia com outros dispositivos compatíveis com MIDI. Usando um controlador mestre, como um teclado, um deles pode reproduzir ou enviar sons para outro instrumento conectado à ele remotamente. Isto elimina a necessidade de um tecladista ter nove ou dez teclados à sua volta. Ele pode tocar todos os teclados usando um só, simplesmente conectando-os via MIDI. Os outros teclados não precisam estar próximos, ele não precisa nem encostar a mão neles, apesar de poder tocá-los. Da mesma forma que o *modem* passa bytes de dados entre o computador e o provedor Internet, usando um protocolo chamado TCP/IP, instrumentos eletrônicos passam bytes de dados usando o protocolo MIDI, conforme mostra a figura 19.

Figura 19 – Transmissão de bytes



Fonte: Gontijo (2000)

Este protocolo define várias séries diferentes de bytes. MIDI é, assim, um protocolo para transferência de informações (informações musicais em sua maioria). Estas informações tomam a forma de sinais eletrônicos que os instrumentos passam entre si. O formato MIDI padrão (*Standard MIDI File*, ou SMF) é um formato de arquivos especificamente projetado para armazenar os dados que um seqüenciador (software ou hardware) grava e reproduz. Este formato armazena as mensagens MIDI padrão (bytes de status, com os bytes de dados apropriados) mais um "*timestamp*" para cada mensagem (isto é, uma série de bytes que representam quantos pulsos de relógio aguardar antes de "reproduzir" o evento). O formato permite salvar informações sobre tempo, número de pulsos em resolução de semínima (ou resolução expressa em divisões de segundo, chamada *configuração SMPTE*), duração e tom da música, e nomes de trilhas e padrões. Pode armazenar múltiplos padrões e trilhas, de forma que qualquer aplicativo pode preservar estas estruturas quando carregar o arquivo.

O formato foi projetado para ser genérico, de forma que qualquer seqüenciador poderia ler ou escrever tal arquivo, sem perder os dados mais importantes, e flexível o suficiente para um determinado aplicativo armazenar seus dados próprios ("extras") de forma que outro aplicativo poderia carregá-lo sem problemas, e poderia ignorar com segurança esses dados extras que não precisasse, podendo analisá-los, carregá-los, saltá-los etc. Assim, pode ser facilmente estendido para incluir informações proprietárias de um programa.

Dados são sempre salvos dentro de um *bloco*. Pode haver vários blocos dentro de um arquivo MIDI. Cada bloco pode ter um tamanho diferente (onde o tamanho refere-se a quantos bytes o bloco contém). Os bytes de dados em um bloco estão relacionados de certa forma. Um bloco é simplesmente um grupo de bytes relacionados.

Cada bloco começa com um *identificador* de 4 bytes ASCII, que diz que "tipo" de bloco é. Os próximos 4 bytes (assumindo que um byte é igual a 8 bits), indicam o tamanho em bytes do bloco (este valor é expresso em um número de 32 bits). **Todos os blocos devem começar com estes dois campos** (estes 8 bytes), os quais são referidos como *cabeçalho do bloco*. O *tamanho* não inclui o cabeçalho de 8 bytes. Simplesmente lhe diz quantos bytes de dados estão no bloco *seguindo o cabeçalho*.

Aqui temos um exemplo de um cabeçalho (com bytes expressados em hexadecimal):

4D 54 68 64 00 00 00 06

Note que os primeiros 4 bytes formam o identificador ASCII "**MThd**", isto é, os primeiros 4 bytes são valores ASCII para "M", "T", "h" e "d". Os próximos 4 bytes nos dizem que deve haver mais 6 bytes de dados no bloco (e após isso encontrados o próximo *cabeçalho de bloco* ou o fim do arquivo). Todo arquivo MIDI começa com esse *cabeçalho MThd* (e é assim que se sabe que este arquivo é um arquivo MIDI).

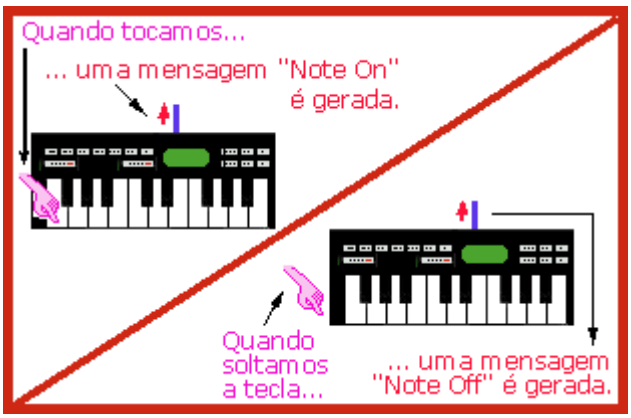
Há certas coisas que todo instrumento musical faz. Em qualquer hora, o músico pode fazer um instrumento começar a fazer um som. Por exemplo, um músico pode pressionar uma tecla de um piano para iniciar um som, ou beliscar uma corda de um violão etc. A ação de iniciar um som denomina-se "*Note On*".

Muitos instrumentos permitem que o músico "pare" o som em determinado tempo. Por exemplo, um músico pode soltar uma tecla de um piano, e interromper seu som. Esta ação de interromper o som denomina-se "*Note Off*".

As ações acima são ações musicais. E MIDI define, para cada ação musical (como as duas mencionadas acima), uma certa série de bytes chamadas de mensagens.

Assim, por exemplo, MIDI define como um instrumento deve iniciar um som ou seja, executar a ação "*Note On*" com uma mensagem chamada "*Note On*". Da mesma forma, MIDI diz ao instrumento para interromper o som usando uma mensagem chamada "*Note Off*" (figura 20).

Figura 20 – Mensagens "*Note On*" e "*Note Off*".



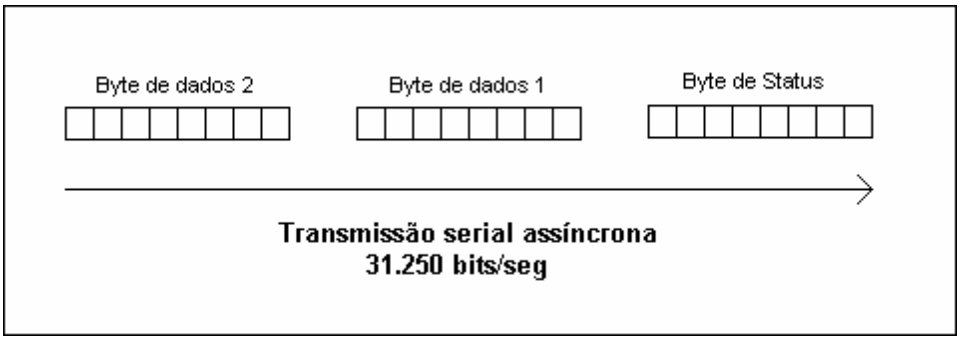
Fonte: Gontijo (2000)

Mas MIDI é mais do que mensagens "*Note On*" e "*Note Off*". Há muitas mensagens diferentes, uma para cada ação musical.

Há uma mensagem que diz a um instrumento que mova sua Roda de Modulação ("*Pitch Wheel*") e quanto deve movê-la. Há outra mensagem que diz ao instrumento pressionar ou soltar o pedal de sustentação; outra que diz ao instrumento que deve mudar de volume e o quanto fazê-lo; outra que diz para o instrumento mudar seu timbre (órgão para violão, por exemplo); etc. Pode-se dizer que MIDI pode fazer com um instrumento eletrônico tudo que um humano é capaz de fazer, e mesmo algumas coisas que humanos não podem.

Segundo Ratton (1995), para transferir as informações através do cabo MIDI, o instrumento codifica essas informações sob a forma de números binários, que são enviados serialmente como mostra a figura 21. A transmissão ocorre a uma velocidade de 31.250 bits por segundo. Como a maioria das mensagens MIDI possui apenas um, dois, ou três bytes de tamanho, estas mensagens MIDI "voam" pelo cabo MIDI rapidamente fazendo com que o ouvido humano pense que os dois instrumentos estão realmente tocando em uníssono.

Figura 21 – Formato da mensagem MIDI



Fonte: Maske (2000)

Pode-se conectar um cabo MIDI na porta *MIDI OUT* (ou *MIDI THRU*) do segundo instrumento à uma porta *MIDI IN* de um terceiro, e então o segundo instrumento repassaria para o terceiro as mensagens que recebeu do primeiro. Assim, todos os três instrumentos podem tocar em uníssono. Poderia conectar um quarto, quinto, sexto, ..., instrumentos. Chama-se isso de ligar instrumentos "em série" (ou *daisy-chaining*).

Entretanto, instrumentos ligados em série não precisam, necessariamente, tocar em uníssono sempre. Cada um pode tocar sua própria parte musical mesmo que todas as mensagens MIDI que controlam todos os instrumentos passe por cada um deles.

Existem, em MIDI, 16 "canais". Todos eles existem em um só cabo MIDI, que liga dois ou mais instrumentos (e talvez um computador) em conjunto. Por exemplo, a mensagem MIDI para o **Dó** central poderia ser enviada no "canal 1", ou no "canal 2", etc.

A maioria dos instrumentos MIDI permite ao músico selecionar a que canal(is) o instrumento deve responder e qual(is) ignorar. Por exemplo, para configurar um instrumento para responder apenas às mensagens MIDI no canal 1, e enviar uma mensagem MIDI "*Note On*" pelo canal 2, então o instrumento não tocaria a nota. Assim, se um músico possui diversos instrumentos ligados em série (o *daisy-chaining* citado acima), ele poderia configurar todos os instrumentos para responder a diferentes canais cada um. Assim, o músico pode ter diversos instrumentos tocando em diversos canais, e pode controlar o que cada um deles vai tocar e com qual timbre sonoro irá tocar, a partir de um só instrumento.

Quando se quer gravar uma só nota, há só duas mensagens envolvidas: uma *Note On*, para iniciar o som da nota, de tamanho igual a 3 bytes, e uma *Note Off* (outros 3 bytes), para terminá-la após algum tempo. Esta segunda mensagem, *Note Off*, não é gerada até que se solte a tecla da nota. Isto soma 6 bytes. Na verdade, pode-se pressionar a nota por uma hora, que ainda teria apenas 6 bytes: uma mensagem *Note On* e outra *Note Off*. Pois na mensagem *Note On* existe: "a nota foi tocada no tempo x". Em *Note Off*, existe: "a nota foi solta no tempo y".

3.1 PROTOCOLO

A seguir são abordadas as principais características do protocolo MIDI, composto basicamente pelas Mensagens MIDI, pelo Running Status e pelos Eventos MIDI.

3.1.1 MENSAGENS

O protocolo MIDI é composto de mensagens. Uma mensagem consiste em uma *string* (isto é, uma série) de bytes de 8 bits. MIDI tem muitas mensagens definidas. Algumas mensagens consistem em somente 1 byte. Outras mensagens tem 2 bytes. Outras tem 3 bytes. Um certo tipo de mensagem MIDI pode ter um número ilimitado de bytes.

Uma coisa que todas as mensagens têm em comum é que o primeiro byte da mensagem é o byte de *status*. Este é um byte "especial", porque é o único byte que tem o bit 7 ligado.

Assim, sempre que for encontrado um byte destes, será detectado o começo de uma mensagem MIDI. Este será um byte de *status* entre 0x80 a 0xFF. Os bytes restantes da mensagem (isto é, os bytes de dados, se existirem) estarão na escala 0x00 a 0x7F (para indicar o uso do sistema numérico hexadecimal, de base 16, usou-se a convenção da linguagem C de iniciar um valor com o 0x).

Os bytes de *status* de 0x80 a 0xEF são para as mensagens que podem ser transmitidas em qualquer dos 16 canais MIDI. Por causa disto, estes são chamados "*Voice Messages*" (ou "Mensagens de Voz").

Para estes bytes de *status*, quebra-se o byte de 8 bits em 2 conjuntos de 4 bits. Por exemplo, um byte de *status* de valor 0x92 pode ser quebrado em 2 conjuntos, com valor de 9 (conjunto superior) e de 2 (conjunto inferior). O conjunto superior diz-lhe que tipo é a mensagem MIDI.

Estão aqui os valores possíveis para o conjunto superior, e a que tipo de *Voice Messages* cada um representa:

8 = Note Off,

9 = Note On,

A = *AfterTouch* (também conhecida como "*Key Pressure*"),

B = Control Change,

C = Program Change,

D = Channel Pressure,

E = Pitch Wheel.

Assim, o exemplo de 0x92, mostra que seu tipo de mensagem é "*Note On*" (já que o conjunto superior é 9). E o que é o 2 do conjunto baixo? Isto significa que a mensagem está no canal MIDI 2. Há 16 canais (lógicos) possíveis em MIDI, sendo 0 o primeiro. Assim, esta mensagem é um "*Note On*" no canal 2.

O byte de *status* para especificar uma mudança do programa no canal 0, o conjunto superior necessitaria ser C, para um *Program Change*, e o conjunto baixo necessitaria ser 0 para o canal 0. Assim, o byte de *status* seria 0xC0. Embora o byte de *Status* MIDI contém os 16 canais em números 0 a F (15), todo equipamento MIDI (incluindo softwares de computador) indicam um número de canal ao músico como 1 a 16. Assim, um byte de *status* emitido no canal 0 MIDI é considerado estar no "canal 1" para o músico.

Portanto, bytes de *status* de 0x80 a 0xEF são para as mensagens de voz. Os bytes de *status* de 0xF0 a 0xFF são para as mensagens que não estão em nenhum canal particular. Estes bytes de *status* são usados para as mensagens que carregam uma informação que interessa a todos os dispositivos MIDI, tais como sincronização de todos os dispositivos de *playback* em determinado momento. Estes bytes de *status* são divididos em duas categorias. Os bytes de *status* de 0xF0 a 0xF7 são chamados de "Mensagens Comuns Ao Sistema" (ou "*System Common Messages*"). Os bytes de *status* de 0xF8 a 0xFF são chamados de "Mensagens de Tempo Real do Sistema" (ou "*Realtime System Messages*").

Determinados bytes de *status* dentro desta escala não são definidos pela especificação MIDI, e são reservadas para uso futuro. Por exemplo, os bytes de *status* de 0xF4, 0xF5, e 0xFD não são usados. Se um dispositivo de MIDI receber tal byte de *status*, deve ignorar essa mensagem.

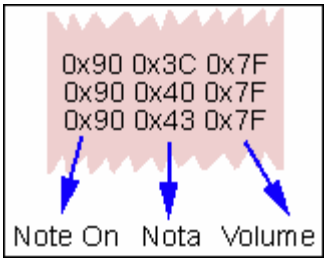
3.1.2 RUNNING STATUS

A especificação MIDI permite que uma mensagem MIDI seja emitida sem seu byte de *status* (ou seja, apenas seus bytes de dados serão transmitidos), desde que a mensagem anteriormente transmitida tenha o mesmo *status*. Isto é chamado de "*running status*", que é simplesmente um esquema inteligente para maximizar a eficiência da transmissão MIDI (através da remoção de bytes de *status* redundantes). A idéia básica do *running status* é que um dispositivo deve sempre se lembrar do último byte de *status* que recebeu (à exceção de

RealTime), e, se não receber um byte de *status* em mensagens subseqüentes, deve supor que está tratando de uma situação "*running status*". Um dispositivo que gera mensagens MIDI deve sempre se lembrar do último byte de *status* que emitiu (à exceção de *RealTime*), e se precisar emitir outra mensagem que possua o mesmo *status*, o byte de *status* pode ser omitido.

Abaixo encontra-se um exemplo de um dispositivo que cria um fluxo de mensagens MIDI; o dispositivo emite 3 mensagens *Note On* no canal 0 (figura 22).

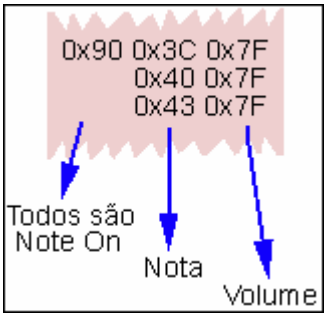
Figura 22 – Três bytes de *status*.



Fonte: Gontijo (2000)

Observa-se que os bytes de *status* de todas as 3 mensagens são os mesmos no caso, *Note On*, canal 0. Conseqüentemente, o dispositivo podia executar um *running status* para as últimas 2 mensagens, emitindo os seguintes bytes (figura 23):

FIGURA 23 – UM BYTE DE *STATUS*.



Fonte: Gontijo (2000)

Isto permite que o dispositivo poupe tempo, já que há 2 bytes a menos para transmitir. Claro, se a mensagem que o dispositivo enviou antes destas 3 também é um *Note On* no canal 0, então ele poderia ter omitido o *status* da primeira mensagem acima, também.

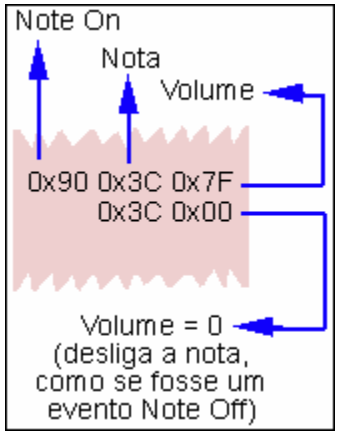
Ele recebe o *status* da primeira mensagem (0x90) e mantém este *status* para os próximos 2 bytes de dados que virão a seguir. Após receber os 2 bytes de dados, ele, então, recebe o byte de dados da segunda mensagem (0x40).

Depois disso, como o dispositivo não recebe outro byte de *status*, ele usa o *running status* anterior. O último byte de *status* recebido era 0x90, então supõe-se que esta nova mensagem tenha este mesmo *status*. Portanto, este 0x40 é o primeiro byte de dados de uma outra mensagem *Note On* no canal 0.

Uma mensagem *Note On* com uma velocidade 0 é considerada como sendo uma *Note Off*. Sendo assim, pode-se emitir um fluxo inteiro de mensagens de nota desligando ou ligando notas, sem precisar enviar um novo byte de *status*, além do presente na primeira mensagem. Todas as mensagens terão o mesmo *status* de *Note On*, mas as mensagens que desligam notas terão a velocidade zero.

O exemplo abaixo mostra como tocar e abafar uma nota, utilizando este esquema de *running status* (figura 24).

FIGURA 24 – TOCANDO E ABAFANDO UMA NOTA.



Fonte: Gontijo (2000)

As mensagens da categoria de *RealTime* (*status* de 0xF8 a 0xFF) não efetuam *running status* de forma nenhuma. Pois uma mensagem de *RealTime* consiste em somente 1 byte, e pode ser recebida a qualquer momento (inclusive dentro de outra mensagem), desta maneira, deve ser tratada de forma "transparente".

As mensagens da categoria Comuns ao Sistema (de *status* 0xF0 a 0xF7) cancelam todo *running status*. Ou seja, a mensagem após uma Mensagem Comum Ao Sistema deve começar com um byte de *status*.

O *running status* é executado somente para mensagens da Categoria de Voz (cujo *status* é 0x80 a 0xEF).

Os primeiros dois bytes de dados (que seguem os 8 bytes do cabeçalho), nos dizem o formato ou tipo. Há hoje 3 tipos diferentes de arquivos MIDI.

O **tipo 0** significa que o arquivo contém uma só trilha contendo dados MIDI em talvez todos os 16 canais MIDI. Um arquivo de **tipo 1** significa que o arquivo contém uma ou mais trilhas simultâneas (isto é, todas começam de um tempo assumido como zero), talvez cada uma em um canal MIDI. Juntas, todas estas trilhas são consideradas uma seqüência ou padrão. Já um arquivo do **tipo 2** significa que o arquivo contém um ou mais padrões de uma só trilha, mas que são seqüencialmente independentes.

Os dois próximos bytes (chamados Número de Trilhas, ou "*NumTracks*") dizem quantas trilhas estão armazenadas no arquivo. Para arquivos de formato 0, ele é sempre igual a 1. Para os outros dois formatos, pode haver diversas trilhas. Os dois últimos bytes indicam em quantos Pulsos por Semínima (ou "*Pulses per Quarter Note*", abreviado como PPQN) a resolução das batidas de tempo são baseados, chamado de Divisão ("*Division*").

Aqui está um exemplo de um bloco MThd completo (com o cabeçalho):

4D 54 68 64 Identificador MThd,

00 00 00 06 O tamanho de um bloco MThd é sempre 6,

00 01 O Formato é 1,

00 02 Há 2 blocos MTrk neste arquivo,

00 F0 Cada incremento de tempo representa 1ms.

Após o bloco MThd, deve-se encontrar um Bloco MTrk, já que este é o único outro bloco definido atualmente (se encontrar outro identificador de bloco, pode ser proprietário de

algum outro programa, então deve-se saltá-lo, ignorando o número de bytes de dados indicados no campo Tamanho).

Um bloco MTrk contém todos os dados MIDI (com bytes de tempo), mais alguns dados não-MIDI opcionais para uma trilha. Deve-se encontrar tantos blocos MTrk existentes no arquivo, quantos o byte *NumTracks* do bloco MThd indicar.

O cabeçalho MTrk inicia com o identificador MTrk, seguido pelo tamanho (número de bytes de dados a serem lidos para esta trilha). O tamanho será provavelmente diferente para cada trilha.

Nos arquivos de formato 0, alterações nos sinais de compasso e tempo são passados através de um MTrk. No formato 1, o primeiro MTrk deve consistir apenas de eventos sinais de compasso e tempo de forma que possa ser lido por um dispositivo capaz de gerar um "mapa de tempos". No formato 2, cada MTrk deve iniciar com ao menos um evento de tempo e compasso iniciais. Se não há eventos de sinal de tempo e compasso no arquivo MIDI, é assumido 120 BPM (batidas por minuto) e compasso 4/4.

3.1.3 EVENTOS

O primeiro evento na trilha pode ser soar uma nota **Dó** central. O segundo pode ser tocar o **Mi** acima do **Dó** central. Estes dois eventos podem acontecer ao mesmo tempo. O terceiro evento pode ser liberar a nota **Dó** central, sendo que, pode ocorrer alguns compassos após os primeiros dois eventos (isto é, o **Dó** central é mantido pressionado por alguns compassos). Cada evento possui um "tempo" para ocorrer, e os eventos são arranjados dentro de "blocos" de memória na ordem em que ocorrem.

Em um arquivo MIDI, o "tempo" de um evento precede os bytes de dados que criam o evento. Em outras palavras, os bytes que formam a "batida do tempo" vêm primeiro. Uma dada batida de tempo é referenciada pelo evento prévio. Por exemplo, se um evento ocorre 4 tempos após o início da reprodução, seu "tempo delta" é 04. Se o próximo evento ocorre simultaneamente com este primeiro evento, seu tempo é zero. Assim, um "tempo delta" é a duração (em tempos) entre um evento e o evento precedente.

Um tempo delta é armazenado como uma série de bytes que são chamados de quantidade de tamanho variável. Apenas os primeiros 7 bits de cada byte são significantes (alinhados à direita, como um byte ASCII). Então, com um tempo-delta de 32 bits, deve-se desempacotá-lo em uma série de bytes de 7 bits, o que fará com que se tenha um número variável de bytes dependendo do seu tempo-delta. Para indicar qual é o último byte da série, deixa o bit 7 apagado. Em todos os bytes precedentes, liga-se o bit 7. Assim, se um tempo-delta está entre 0 e 127, ele pode ser representado por um byte. O maior tempo-delta é 0FFFFFFF, que se traduz em um tamanho variável de 4 bytes. Aqui estão alguns exemplos de tempo-delta como valores de 32 bits, e as quantidades de tempo variáveis para a qual eles são traduzidos (figura 25).

FIGURA 25 – QUADRO DE QUANTIDADE DE TEMPO TRADUZIDOS PARA VALORES DE 32 BITS

Número	Quantidade
00000000	00
00000040	40
0000007F	7F
00000080	81 00
00002000	C0 00
00003FFF	FF 7F
00004000	81 80 00
00100000	C0 80 00
001FFFFF	FF FF 7F
00200000	81 80 80 00
08000000	C0 80 80 00
0FFFFFFF	FF FF FF 7F

Fonte: Gontijo (2000)

Os primeiros (1 a 4) byte(s) em um MTrk serão os primeiros eventos de tempo-delta de quantidade de tamanho variável. O próximo byte de dados é atualmente o primeiro byte do evento em si. O chamaremos de *Status* do evento. Para eventos MIDI, este será o byte de *Status* MIDI atual (ou o primeiro byte de dados MIDI em caso de "*running status*"). Por exemplo, se o byte vale 90 hexa, então este evento é um "*Note On*" no canal 0 MIDI; mas já se por exemplo, o byte vale 23 hexa, precisa-se verificar o *status* do evento anterior ("*MIDI*

running status"). Obviamente, o primeiro evento MIDI em MTrk deve ter um byte de *status*. Após um byte de *status* temos um ou 2 bytes de dados (dependendo do *status* - algumas mensagens MIDI possuem apenas um byte de dados subseqüente). Após isso tem-se o tempo-delta do próximo evento (de quantidade variável) e o início do processo de leitura do próximo evento.

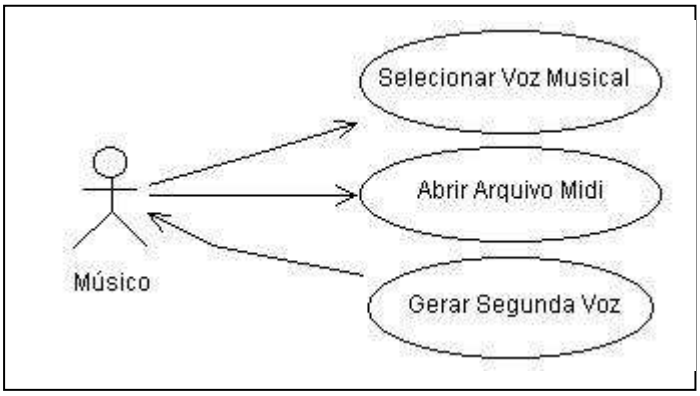
4 DESENVOLVIMENTO DO PROTÓTIPO

4.1 ESPECIFICAÇÃO

Esta seção segue com as especificações utilizadas para o desenvolvimento do protótipo. Para criar o diagrama foi utilizada a ferramenta Rational Rose e para a criação do fluxograma a ferramenta Microsoft Word.

A figura 26 apresenta o diagrama de caso de uso do protótipo.

FIGURA 26 – DIAGRAMA DE CASO DE USO

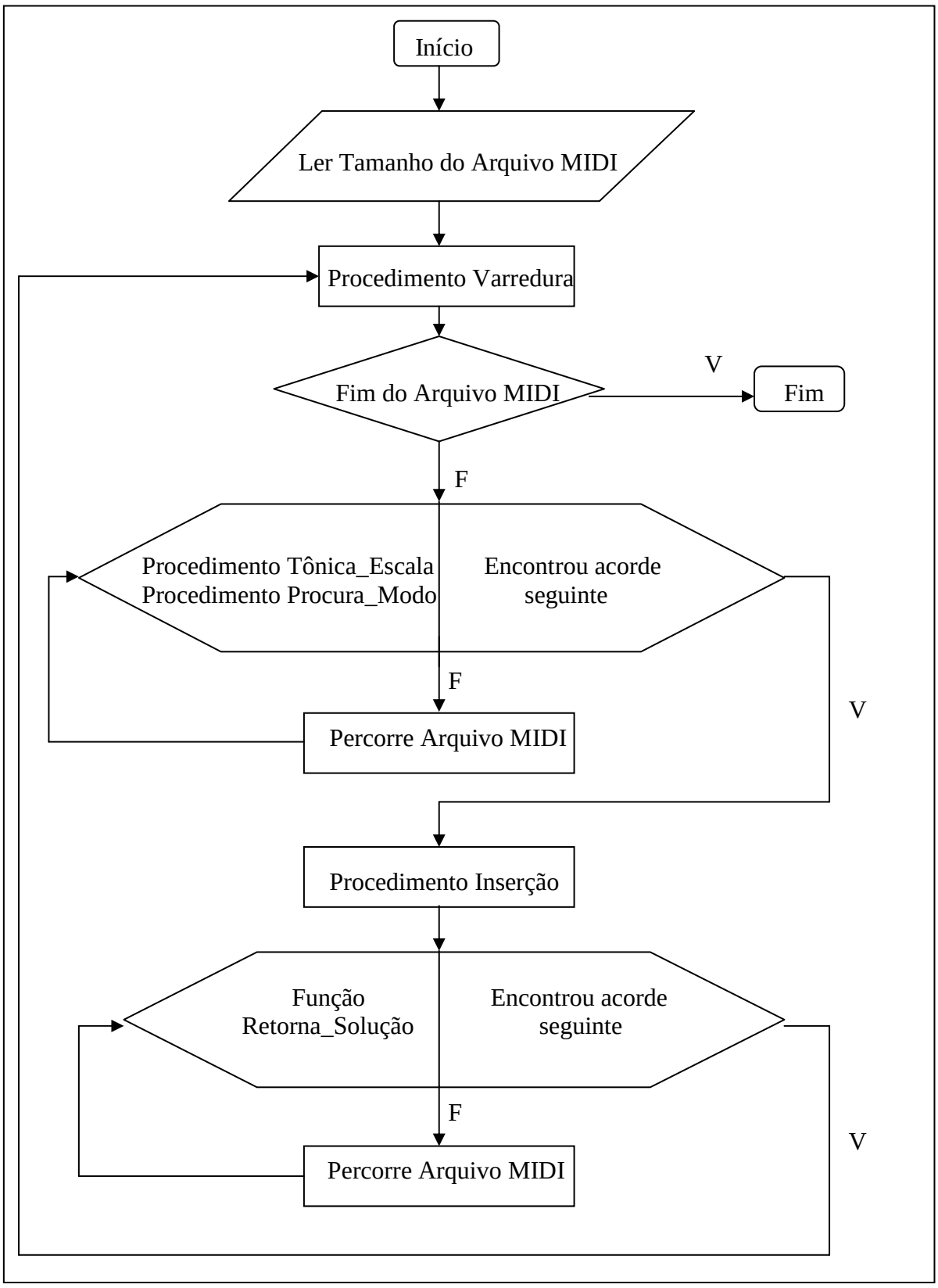


Para um melhor entendimento, seguem descritos abaixo os eventos:

a) Selecionar Voz Musical: O músico seleciona o tipo de voz musical à ser criada pelo protótipo, que poderá ser desde uma 2ª até uma 7ª (conceito de intervalos), respectivamente;
b) Abrir Arquivo MIDI: Aqui o músico seleciona o arquivo MIDI que terá a adição de uma segunda voz musical pelo protótipo conforme o tipo de voz selecionada;
c) Gerar segunda voz: O Protótipo gera um arquivo MIDI com uma segunda voz adicional para o usuário. Durante este processo são executadas várias rotinas para a criação da segunda voz musical.

A figura 27 apresenta o fluxograma dos principais eventos, funções e procedimentos do protótipo.

FIGURA 27 – FLUXOGRAMA DO PROTÓTIPO



À seguir estão descritas de forma reduzida as funcionalidades dos procedimentos e funções presentes no fluxograma:

Procedimento Varredura: Após ser lido o tamanho do arquivo MIDI, esta procedimento fica percorrendo toda a sua extensão. Enquanto não chega ao fim, dentro dele é realizado uma verificação de acorde até acorde, isto é, o primeiro acorde da melodia é lido e enquanto não encontra o próximo acorde definido nesta, são chamados outros dois procedimentos:

a) Procedimentos Tônica_Escala : Aqui é verificado qual a tônica do acorde lido, e a partir desta tônica é montada a escala da melodia em questão no modo 'Iônico' (modo default);
b) Procedimentos Procura_Modo : Todas as notas da melodia presente são analisadas com a escala montada no procedimento anterior, procurando por notas que possam descaracterizar a melodia do modo default 'Iônico', visando assim descobrir qual o modo correto da melodia.

Procedimento Inserção: Após ser encontrado o acorde seguinte no procedimento varredura, toda a melodia referente ao acorde anterior agora será analisada para a criação da segunda voz musical. A condição de saída deste procedimento é encontrar o acorde seguinte, fazendo exatamente a mesma busca do procedimento varredura, mas desta vez chamando uma função:

a) Função Retorna_Solução: Analisa cada nota presente na melodia e cria a sua segunda voz musical conforme o tipo selecionado pelo usuário. Através do modo encontrado para a presente melodia é criada a escala referente a tônica do acorde em questão, que serve como base para a criação da segunda voz musical através do método da parametrização, descrito posteriormente nesta monografia.

Encontrando o próximo acorde, ele será passado como parâmetro para o procedimento varredura para que ele volte a pesquisar o arquivo MIDI a partir deste. Este mesmo procedimento se repetirá até o fim do arquivo, de uma forma recursiva.

4.2 IMPLEMENTAÇÃO

Informações sobre a implementação do trabalho são apresentadas nesta seção. As principais técnicas e ferramentas são discutidas.

4.2.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

O trabalho proposto por Maske (2000) tem como objetivo a especificação e implementação de um protótipo de um sistema especialista utilizando a VCL (Biblioteca de Componentes Visuais) *Expert SINTA*, desenvolvida para a linguagem de programação Delphi em sua versão 3.0. Seu protótipo tem como objetivo principal auxiliar e facilitar o trabalho de músicos na adição de uma segunda voz musical na escrita de partituras.

Um Sistema Especialista (SE) de acordo com Feiko (1983), é um programa inteligente de computador que usa conhecimentos e procedimentos inferenciais, para resolver problemas que são bastante difíceis, de forma a requererem para sua solução, muita perícia humana. O conhecimento de um sistema especialista consiste em fatos e heurísticas, sendo assim projetado e desenvolvido para atender a uma aplicação determinada e limitada do conhecimento humano. É capaz de emitir uma decisão, apoiado em conhecimento justificado, a partir de uma base de informações, tal qual um especialista de determinada área do conhecimento humano (Fávero, 2000).

O *Expert SINTA* é uma ferramenta computacional que utiliza técnicas de inteligência artificial para gerar de forma automática sistemas especialistas. Esta ferramenta utiliza um modelo de representação do conhecimento baseado em regras de produção e probabilidades, tendo como objetivo principal simplificar o trabalho de implementação de sistemas especialistas através do uso de uma máquina de inferência compartilhada, do tratamento probabilístico das regras de produção e da utilização de explicações sensíveis ao contexto da base de conhecimento modelada.

Dada uma partitura musical com apenas uma voz melódica, o protótipo se encarrega de gerar uma segunda voz, de acordo com regras de harmonia definidas em uma base de um sistema especialista. A ordem dos eventos do protótipo de Maske são:

1. Músico entra com um arquivo MIDI gerado num software do tipo seqüenciador.

2. Sistema Especialista pesquisa dados na base de conhecimento.

3. Sistema Especialista analisa dados da base de conhecimento.

4. Base fornece a solução (conhecimento) para o sistema especialista.

No momento em que o arquivo estiver aberto, o protótipo se encarrega de consultar a base de conhecimento que foi gerada no *Expert Sinta Shell* e automaticamente verifica todas as notas que compõem a melodia e inclui de acordo com a harmonia, a nota que mais se enquadra para formar a segunda voz musical.

Na figura 28 tem-se uma partitura musical editada no Encore 4.21, referente à música "Noite Feliz". Nota-se que a primeira nota da música noite feliz é a nota "**Sol**", que equivale ao número "67" em decimal. Convertendo em hexadecimal o número é "43". Após ter sido entrado com o número da nota em decimal, o sistema percorre toda a base de conhecimento, através da ferramenta *Expert Sinta Shell*, e retorna a solução de acordo com as regras. Este exemplo retorna como solução a nota de número "40" em hexadecimal, que representa a nota "**Mi**" Figura 29.

FIGURA 28 – NOITE FELIZ EDITADA NO ENCORE

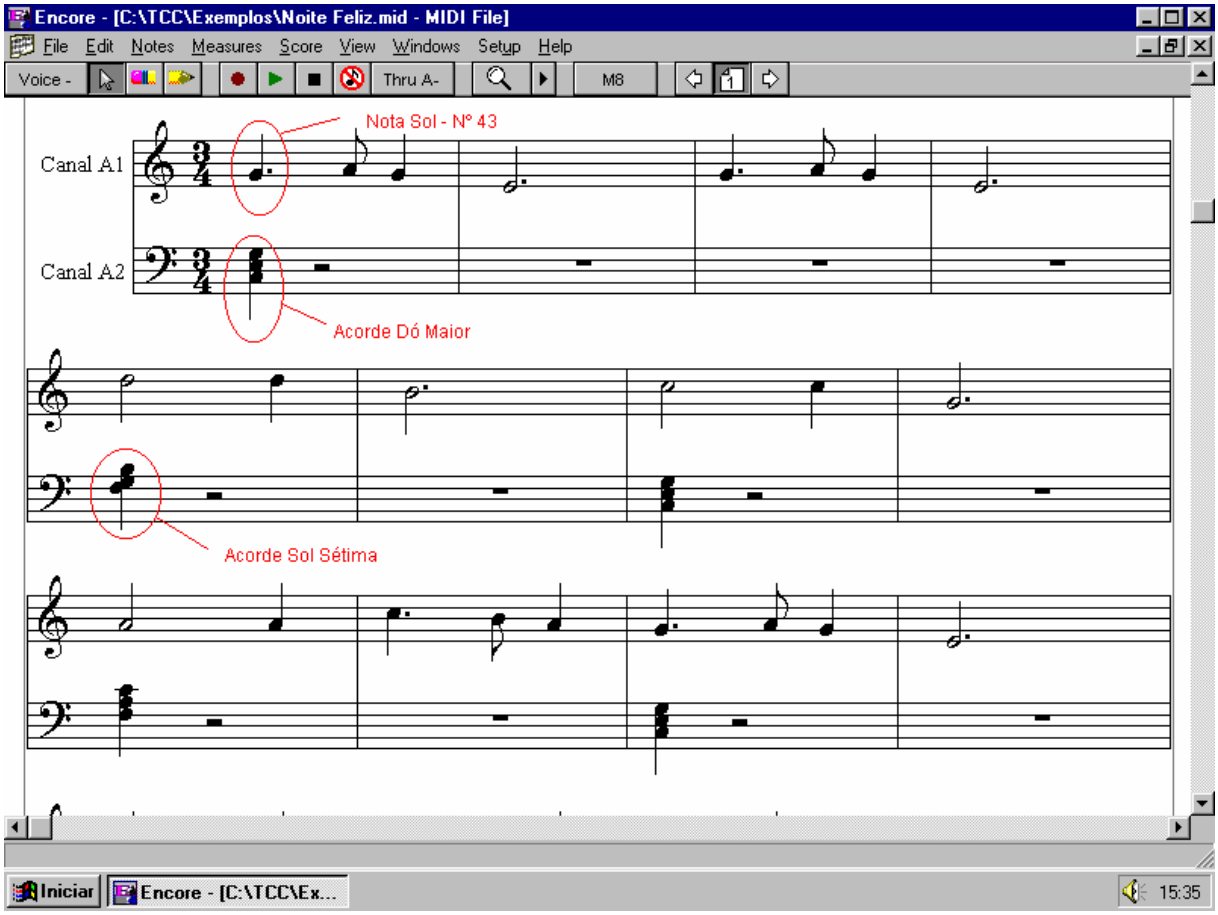
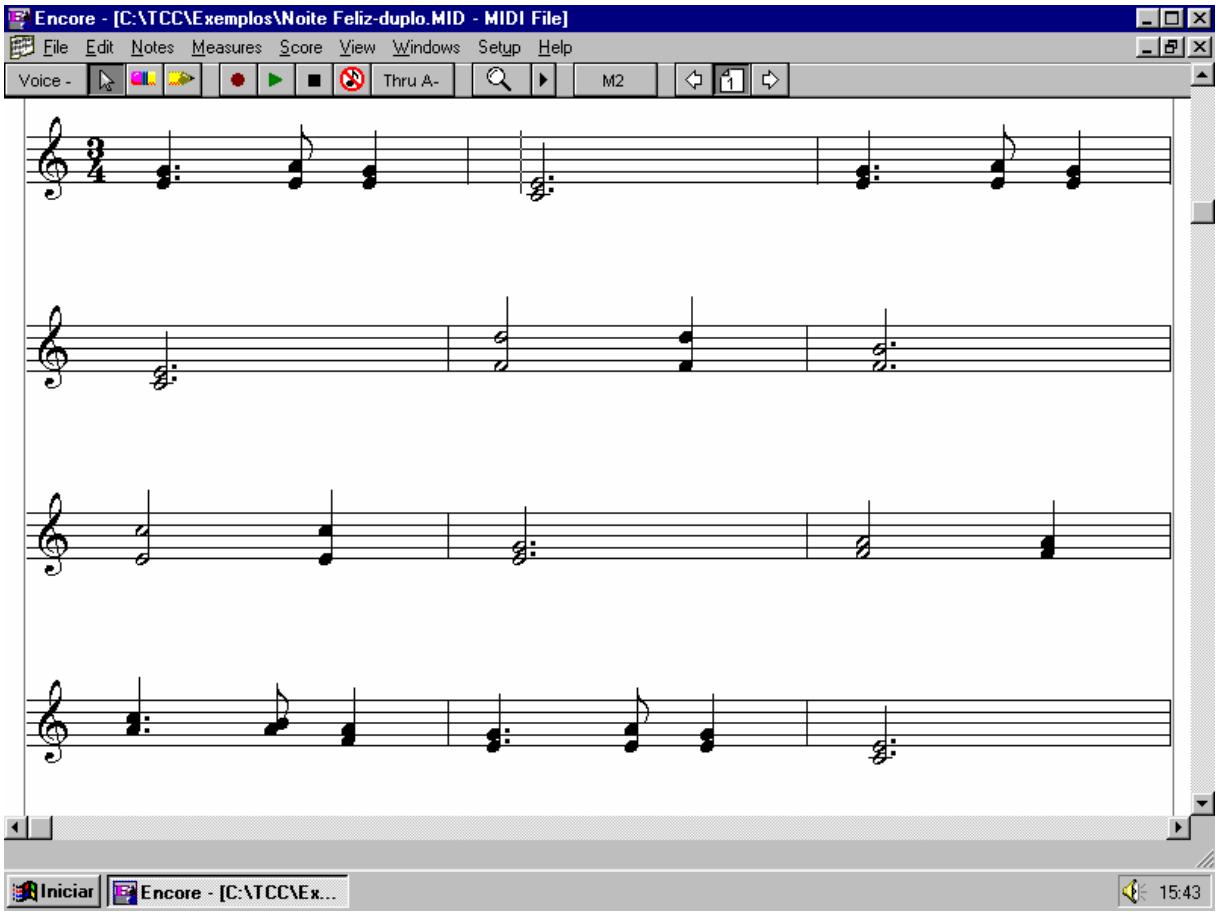


FIGURA 29 – NOITE FELIZ COM SEGUNDA VOZ GERADA NO PROTÓTIPO DE MASKE (2000)



A solução encontrada pelo sistema especialista de Maske (2000), neste caso, conforme figura 29, refere-se à regra de número 1 implementada na base de conhecimento de seu sistema. Esta regra diz que sempre que houver uma nota **Sol** (canal 1 do arquivo MIDI) juntamente com o acorde de **Dó** Maior (canal 2), a solução será, isto é, a segunda voz musical à ser adicionada, a nota **Mi**. Descrita no quadro abaixo está a regra, especificada na sua base de conhecimento. Nota-se que a nota é analisada em seu formato decimal e a solução retorna uma nota no formato hexadecimal.

QUADRO 6 – REGRA DE NÚMERO 1 DESENVOLVIDA POR MASKE (2000)

```
Regra 1

SE Nota = 67

E  Acorde = Dó Maior

ENTÃO Solução = 40 CNF 100%
```

Existem no total 167 regras definidas em seu sistema especialista, onde todas elas seguem basicamente o mesmo raciocínio aplicado para a regra descrita no quadro acima.

O protótipo de Maske (2000) possui algumas limitações. Dentre elas, pode-se citar:

a) foram utilizados somente acordes maiores, menores e de sétima, e, acordes sem acidente, devido ao aumento excessivo de regras;
b) limitou-se a entrada de arquivo MIDI ao uso do formato 1, pois este formato separa os canais em blocos diferentes;
c) neste protótipo foi utilizado somente a nota de um acorde, que melhor soava juntamente com a nota da primeira voz musical;
d) A solução referente à uma nota e seu respectivo acorde sempre será uma mesma nota, como se pode verificar no exemplo descrito acima baseado na regra de número 1, além de ser independente do modo usado na melodia (Jônio, Lídio, etc...). Desta forma, a segunda voz não poderá soar muito bem em determinadas melodias. Sendo assim, o seu protótipo não é muito fiel às especificações da teoria musical, onde existem conceitos como o de intervalos para a criação da segunda voz sem prejudicar as melodias presentes em uma música, pois este conceito baseia-se nos modos musicais para a criação de uma segunda voz musical.

4.2.2 O MÉTODO DA PARAMETRIZAÇÃO

Inicialmente, o objetivo principal proposto para o presente trabalho era estender o trabalho, desenvolvido por Maske (2000) aumentando o número de regras na base de seu sistema especialista. Dessa forma, esperava-se que o protótipo desenvolvido pudesse ser capaz de gerar vozes musicais para diversos estilos de músicas. Tais estilos poderiam conter diferentes tipos de campos harmônicos ou melodias. Os diferentes campos harmônicos,

juntamente com diferentes estruturas melódicas causam ao ouvinte a impressão subjetiva de se estar ouvindo determinado estilo de música. Como um exemplo, o ouvinte poderia ter a sensação de estar apreciando uma música de origem oriental, ou um *blues*, ou um chorinho, dentre outros.

Caso este caminho tivesse sido tomado, o protótipo continuaria vinculado ao sistema especialista e limitado a seu processo de análise. Neste sistema, regras de harmonia estariam definidas de forma a criarem uma segunda voz conforme o gosto musical da pessoa que implementou as regras, e não à verdadeira formação de segundas vozes musicais, que utiliza o conceito de intervalos. Tal conceito é mais universal no sentido de ser completamente fiel às especificações da teoria musical.

Tendo em vista esta limitação da arquitetura proposta por Maske (2000), buscou-se alguma outra forma de resolver o problema de geração de segunda voz musical, possivelmente, um que não fosse tão dependente de um conjunto de regras formais.

Como resultado, partiu-se em busca de uma solução que fosse fiel aos conceitos da teoria musical, fazendo com que o protótipo fosse capaz de gerar uma segunda voz musical utilizando do conceito de intervalos e não mais do sistema especialista.

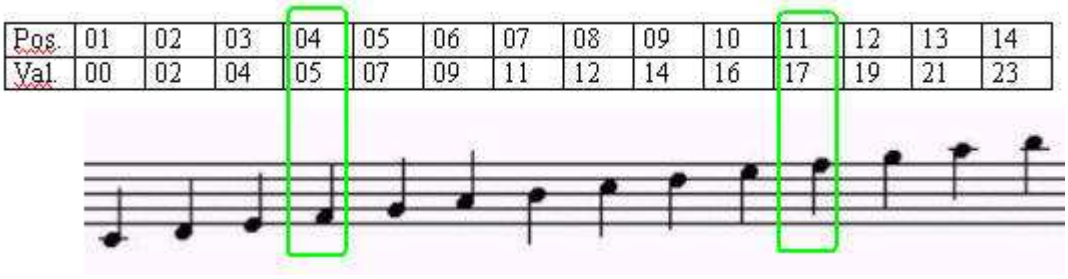
Após um estudo sobre os modos musicais, que são responsáveis pela formação de outras vozes musicais para uma determinada melodia através do conceito de intervalos, encontrou-se uma maneira de transformar a sua especificação teórica em uma parametrização matemática. Esta parametrização poderia então ser implementada através de qualquer linguagem de programação seqüencial e imperativa, e não dependeria mais de uma linguagem declarativa usada em sistemas especialistas baseados em lógica formal.

No exemplo a seguir são apresentados os 3 diferentes modos (em forma de tabela) que podem ser utilizados para qualquer acorde maior. Estas tabelas representam vetores que foram implementados neste protótipo. Na variável “Pos.” estão apresentados os índices de cada posição do vetor e logo abaixo na variável “Val.” os seus respectivos valores de intervalos, onde cada unidade representa um semitom. Note que são representadas duas oitavas, isto porque foi necessária a adição de mais uma oitava nos vetores para a criação da segunda voz pelo protótipo.

Abaixo das tabelas, escrito na pauta musical, estão as notas correspondentes à tonalidade de **Dó** Maior. No modo Iônico, a escala é natural, isto é, sem acidentes, enquanto que nos outros dois modos, o modo Lídio e o modo Mixolídio há uma pequena diferença quanto aos intervalos, alterando-se uma das notas. Cada nota representada na pauta é a soma do valor do intervalo em relação à tônica, neste caso, como já foi dito, a nota **Dó**, refere-se à posição zero do vetor. Para um melhor entendimento, abaixo das notas estão seus valores em decimal, conforme estão definidos na especificação MIDI. Somando-se o valor de intervalo de cada nota ao valor em decimal da tônica da escala, tem-se todas as notas pertencentes a ela.

a) Modo Iônico (figura 30)

FIGURA 30 – TABELA MODO IÔNICO

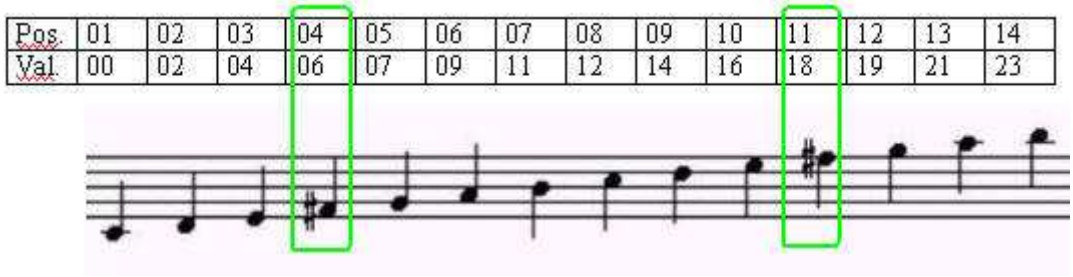


Pos.	01	02	03	04	05	06	07	08	09	10	11	12	13	14
Val.	00	02	04	05	07	09	11	12	14	16	17	19	21	23

	DÓ	RÉ	MI	FÁ	SOL	LÁ	SI	DÓ	RÉ	MI	FÁ	SOL	LÁ	SI
Dec.:	36	38	40	41	43	45	47	48	50	52	53	55	57	59

b) Modo Lídio (figura 31)

FIGURA 31 – TABELA MODO LÍDIO

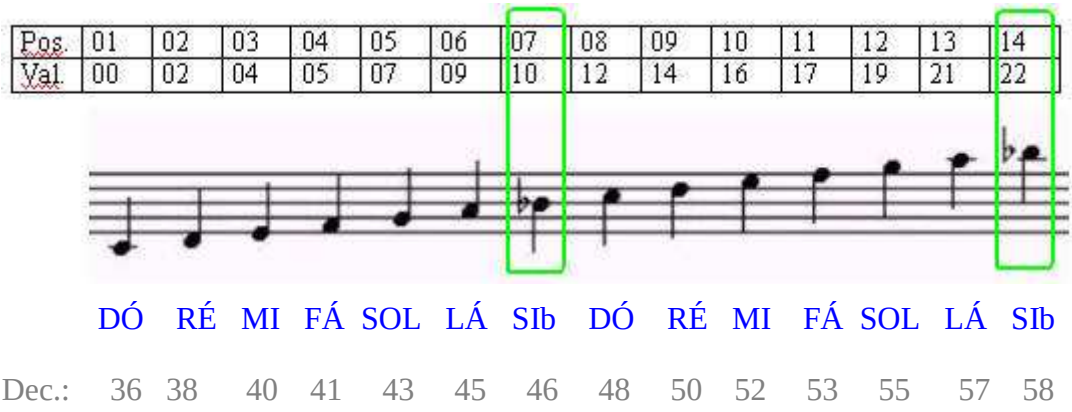


Pos.	01	02	03	04	05	06	07	08	09	10	11	12	13	14
Val.	00	02	04	06	07	09	11	12	14	16	18	19	21	23

	DÓ	RÉ	MI	FÁ#	SOL	LÁ	SI	DÓ	RÉ	MI	FÁ#	SOL	LÁ	SI
Dec.:	36	38	40	42	43	45	47	48	50	52	54	55	57	59

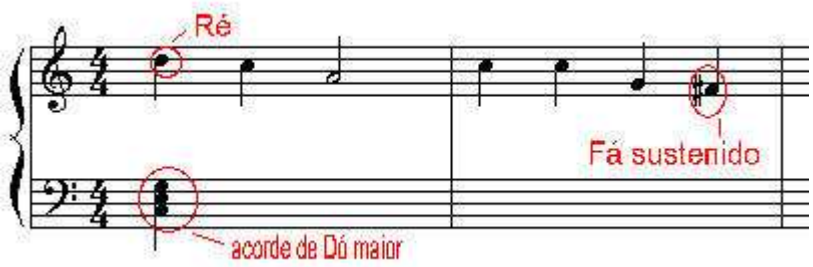
c) Modo Mixolídio (figura 32)

FIGURA 32 – TABELA MODO MIXOLÍDIO



Para a criação de uma segunda voz musical, deve-se somente saber qual o campo harmônico e o modo de uma determinada melodia. Seguindo o exemplo, em uma melodia com campo harmônico em **Dó** maior (Figura 33), desejando-se criar uma segunda voz musical como uma 3ª (intervalo de terça), o cálculo será o seguinte:

Figura 33 – Melodia para criação de uma segunda voz musical



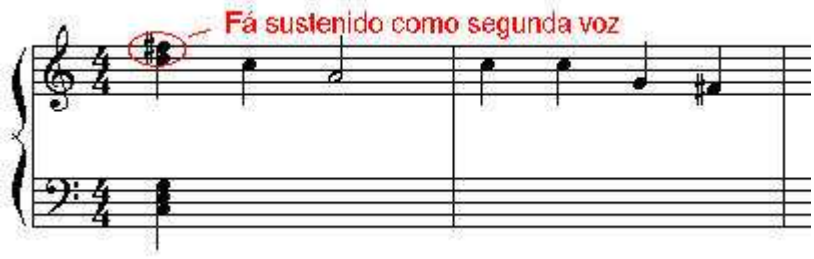
A primeira nota da melodia à ser analisada é a de **Ré**, que em decimal corresponde ao valor 38. Contando-se 3 notas acima da nota **Ré** (incluindo esta) tem-se a nota **Fá**, que equivale ao intervalo de terça, a segunda voz à ser calculada. Mas, dependendo do modo da melodia, este **Fá** poderia não ser o **Fá** natural, mas sim o **Fá** sustenido. Para o modo Jônio e Mixolídio tem-se o **Fá** natural, já no modo Lídio, ocorre o **Fá** sustenido. A melodia representada na figura 33 tem um **Fá** sustenido no final, caracterizando-a na tonalidade de **Dó**

maior no modo Lídio. Então, para o cálculo da segunda voz, seria utilizado o vetor de intervalos definidos no modo lídio.

A nota **Ré** refere-se ao índice 2 do vetor de intervalos e a este valor deve-se somar o número de notas para o intervalo de terça, neste caso, 2 notas, pois a própria nota não precisa ser contada. Então, índice = 2 + 2 = 4.

No índice 4 do vetor de intervalos, tem-se o valor 6. Subtraindo-se o valor 2 do índice 2 do valor 6 do índice 4, chega-se ao valor 4. Este é o número que deve ser somado à nota **Ré** (em decimal) para se encontrar a segunda voz musical. Como **Ré** em decimal é igual a 38, a segunda voz musical neste exemplo é o valor decimal 42. Este valor equivale à nota **Fá** sustenido, conforme a segunda voz musical representada na Figura 34. Este mesmo método é aplicado para as outras notas da melodia.

FIGURA 34 – RESULTADO ATRAVÉS DO MÉTODO DE PARAMETRIZAÇÃO

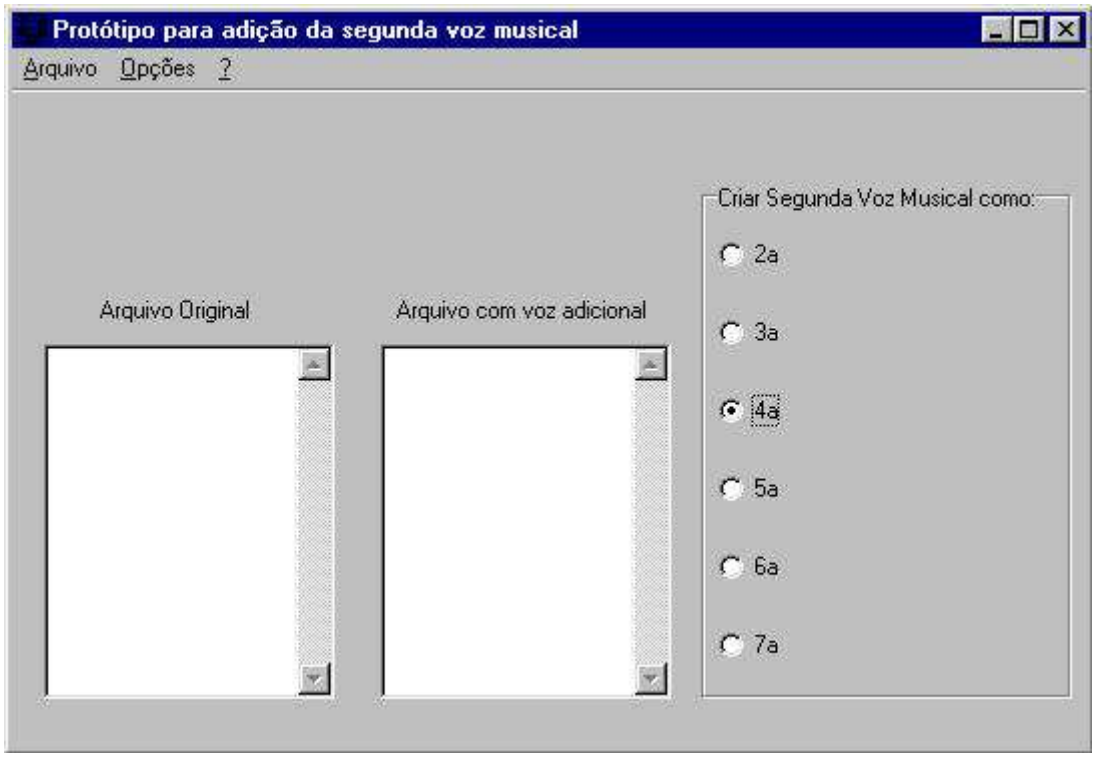


4.2.3 OPERACIONALIDADE DA IMPLEMENTAÇÃO

A seguir, será apresentado o funcionamento do protótipo. Nesta apresentação, serão mostradas as telas do protótipo, bem como as características de cada uma delas.

A figura 35 é a primeira tela, inspirada no protótipo de Maske (2000), sendo a abertura do sistema. Neste momento já se deve especificar qual o tipo de voz musical à ser criada, escolhendo-se uma das opções existentes no quadro “Criar segunda Voz Musical como:”. Nota-se que neste caso foi solicitada a criação de uma 4ª (intervalo de quarta).

FIGURA 35 – TELA DE ABERTURA



Na opção de menu Arquivo -> Abrir (figura 36) é aberto o arquivo no qual se deseja adicionar uma segunda voz musical (figura 37). Este arquivo pode ser gerado em qualquer software musical, que exporte arquivos no formato MIDI. O arquivo deverá ser gerado no formato 1, contendo três blocos **MTrk**. O primeiro bloco contém os eventos relacionados ao sinal de compasso, a velocidade da música, a armadura da clave e outros. O segundo bloco deverá conter somente a melodia, ou seja, a primeira voz musical, armazenada no canal 1. Por fim, no terceiro bloco, são armazenados no canal 2 a harmonia musical (contendo as três notas musicais, que formam o acorde).

Na figura 38, pode-se observar que foi utilizado o editor musical **Encore 4.21** para a criação de uma partitura musical. Nota-se que a partitura em questão é a do arquivo aberto na figura 37 com o nome de “Parabéns pra Você modo lídio” e que a linha melódica encontra-se no canal 1, enquanto que a harmonia encontra-se no canal 2. Cada melodia pertence à tonalidade de **Dó** maior e será interpretada pelo protótipo como sendo cada trecho de notas (canal 1 do arquivo MIDI) existente entre um acorde e outro (canal 2 do arquivo MIDI). Portanto, existem duas melodias na figura. Todas as notas serão convertidas de seu formato

padrão no arquivo MIDI (Hexadecimal) para o formato decimal, pois é através desse formato que se pode trabalhar com as parametrizações definidas no protótipo.

FIGURA 36 – OPÇÃO PARA ABRIR UM ARQUIVO MIDI

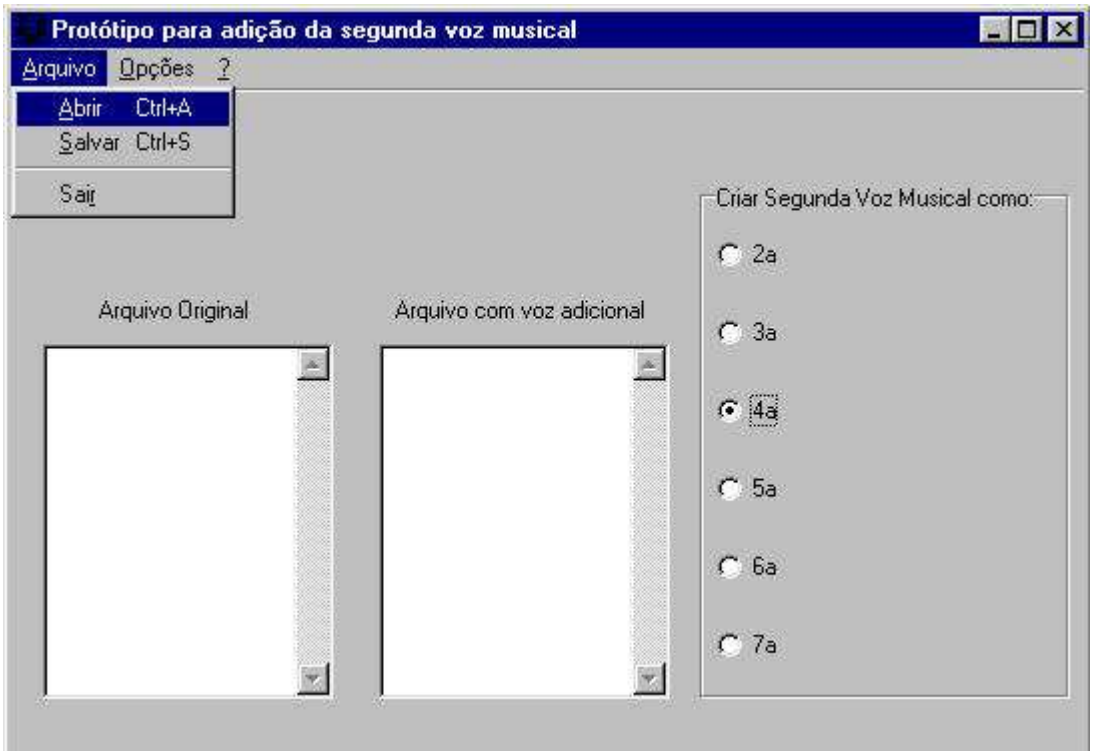


FIGURA 37 – ABRINDO UM ARQUIVO MIDI

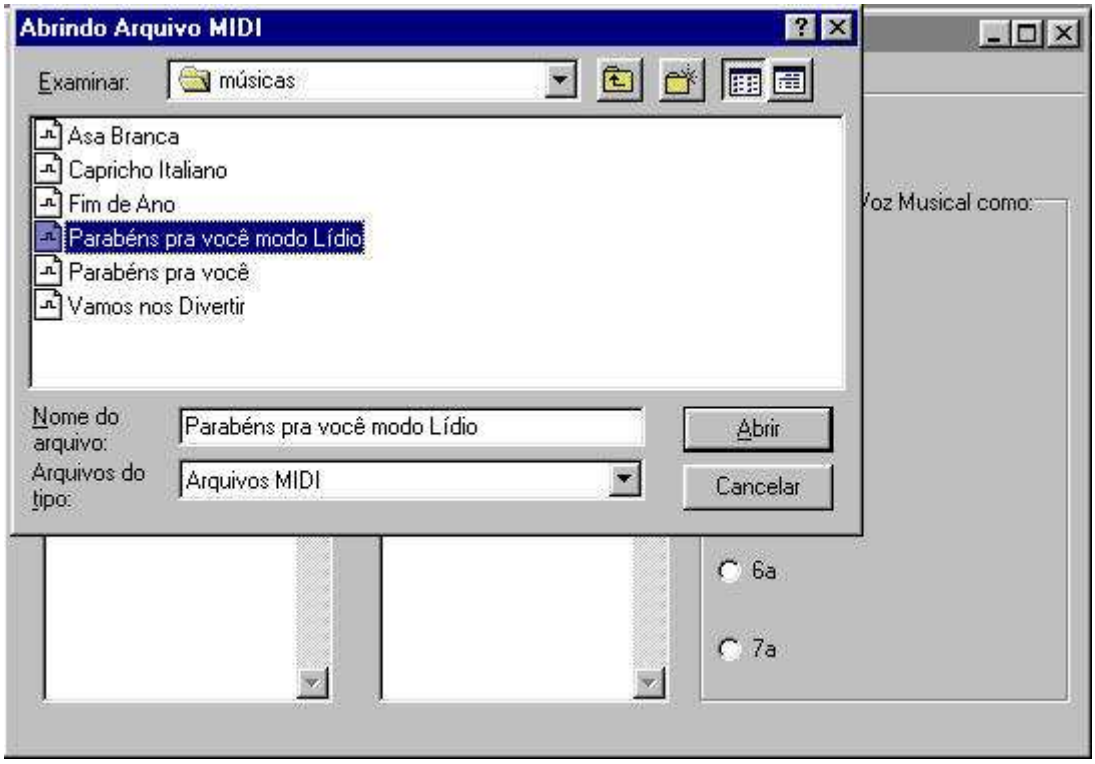
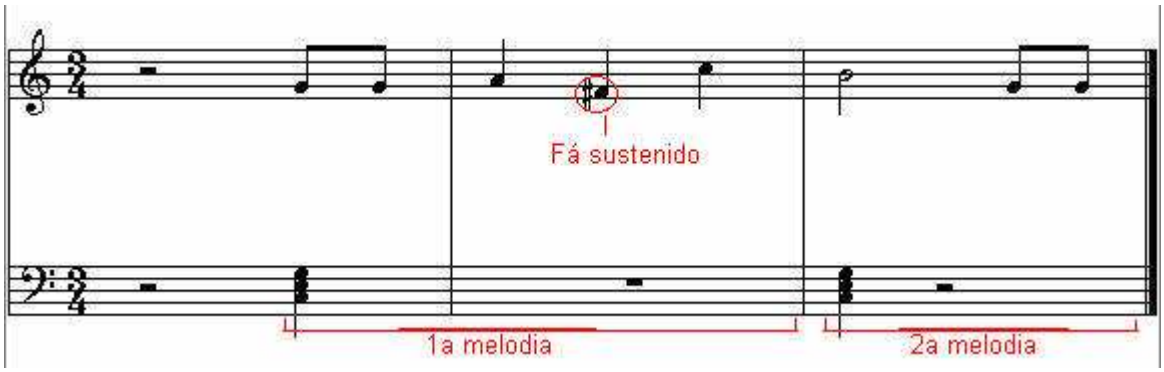


FIGURA 38 – PARABÉNS PRA VOCÊ MODO LÍDIO



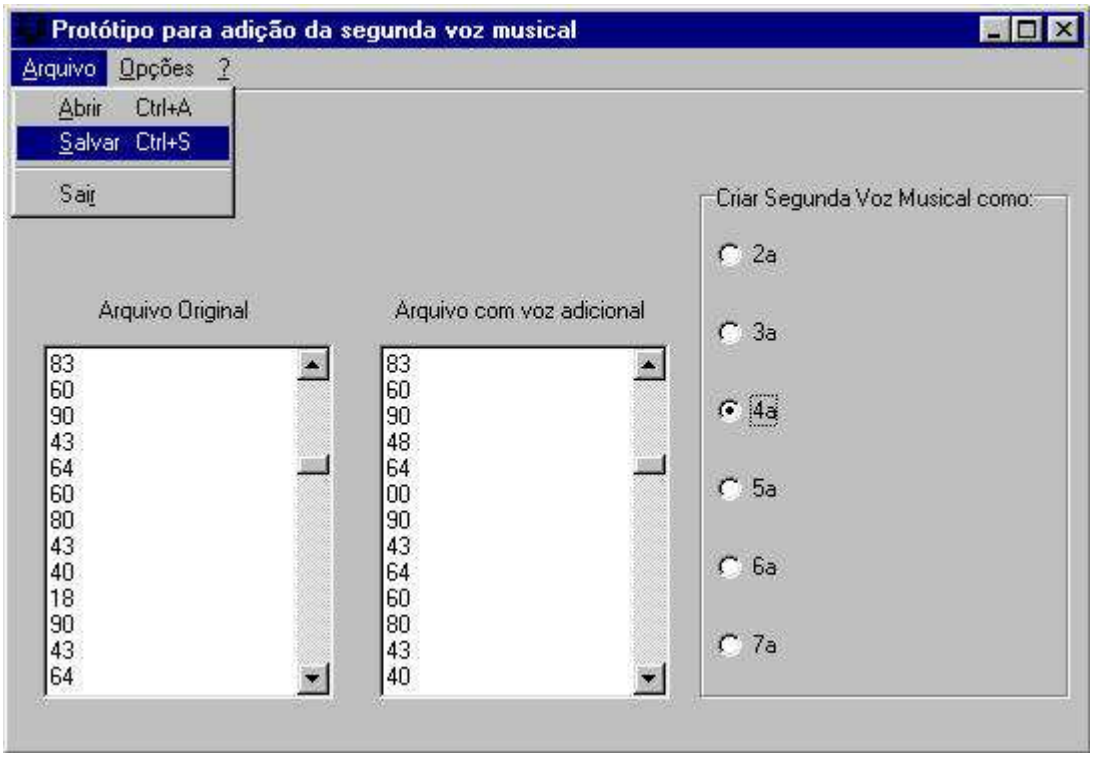
O protótipo cria todas as notas que compõem a escala da melodia em questão, tomando como base a tônica do acorde (nota que dá nome ao acorde) e o modo Jônio (modo default), e as salva em um vetor. Depois, o protótipo faz uma busca analisando todas as notas em cada melodia, comparando com certas notas existentes no vetor mais um determinado valor de intervalo, isto é, procurando por notas que podem descaracterizar o modo default (Jônio) para um outro, definindo assim o modo correto referente ao campo harmônico em cada trecho, em cada melodia. Se encontrar um modo diferente do default, este então é “sobrescrito”. Na

figura 38, note que existe um **Fá** sustenido na primeira melodia. Esta nota faz com que o protótipo interprete a melodia como pertencente à escala de **Dó** maior no modo Lídio, e não Jônio como no default, interpretação que não acontece com a segunda melodia justamente porque ela não tem nenhuma nota que a descaracterize como sendo pertencente ao modo default.

Por fim, este é o recurso usado para a criação da segunda voz musical através do método da parametrização, o protótipo utiliza a escala de intervalos referente ao modo encontrado e calcula com perfeição a segunda voz musical solicitada pelo usuário, neste caso, uma 4ª (quarta), como foi especificado na figura 35. Na figura 39 temos uma descrição detalhada de como foi adicionada a segunda voz musical no protótipo, bem como o evento e o tempo de nota. Agora o arquivo está pronto para ser salvo na opção de menu Arquivo -> Salvar (figura 40).

Figura 39 – Descrição detalhada da adição da segunda voz musical

Figura 40 – Salvar arquivo Midi com segunda voz musical



Na opção de menu Opções -> Executar Original, pode ser ouvido o arquivo MIDI antes de conter a segunda voz (figura 41), e na figura 42 tem-se o arquivo MIDI original executando. Na opção de menu Opções -> Executar Atual, pode ser ouvido o arquivo MIDI após ter sido feito a complementação da segunda voz musical através do método da parametrização (figura 43), e na figura 44 tem-se o arquivo MIDI executando, após a adição da segunda voz musical.

FIGURA 41 – OPÇÃO PARA EXECUTAR ARQUIVO MIDI ORIGINAL

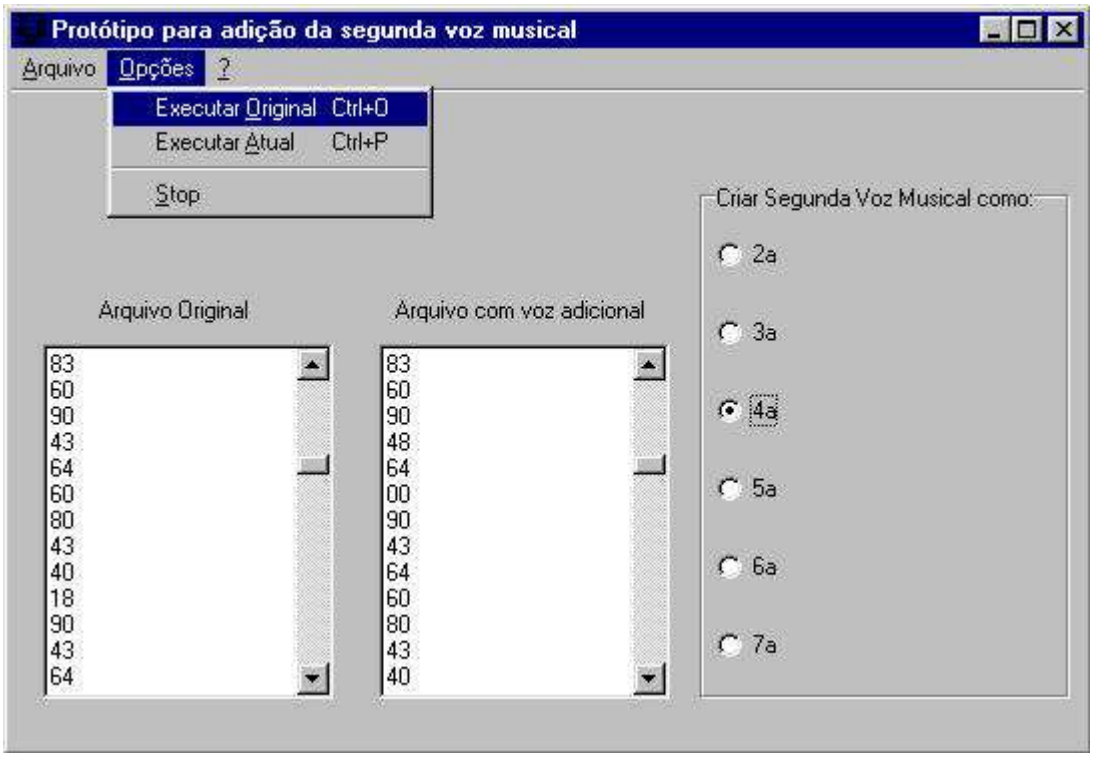


FIGURA 42 – TOCANDO ARQUIVO MIDI ORIGINAL

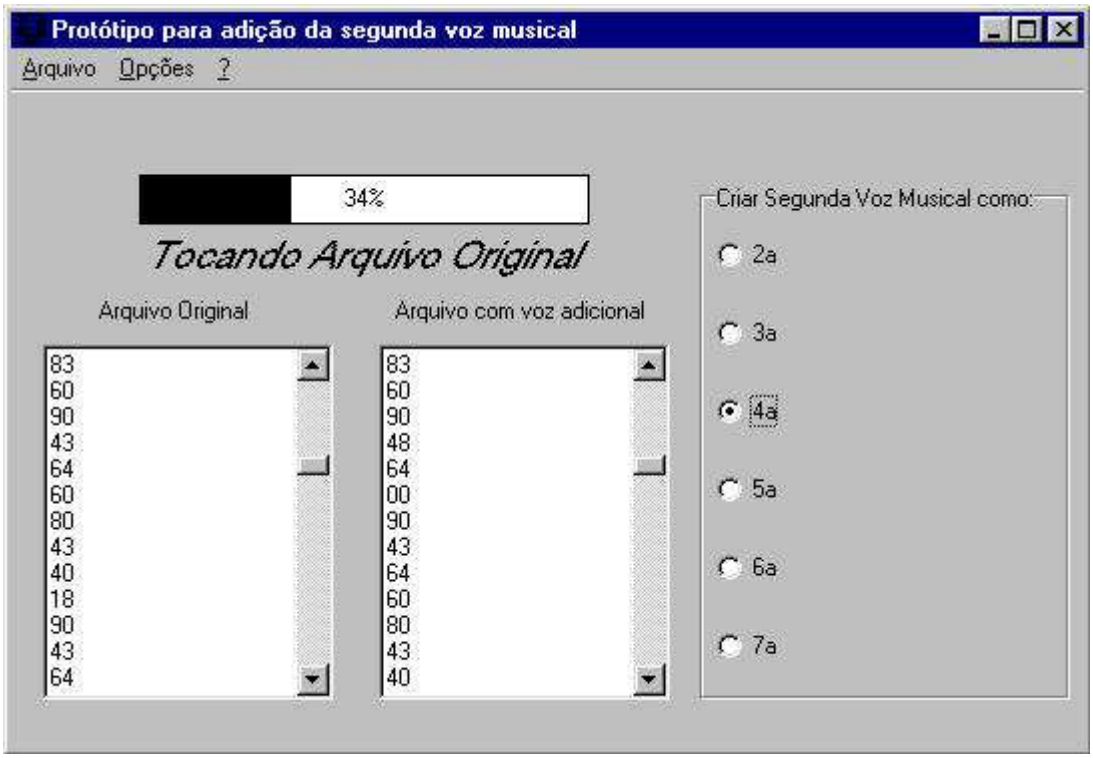


FIGURA 43 – OPÇÃO PARA EXECUTAR ARQUIVO MIDI ATUAL

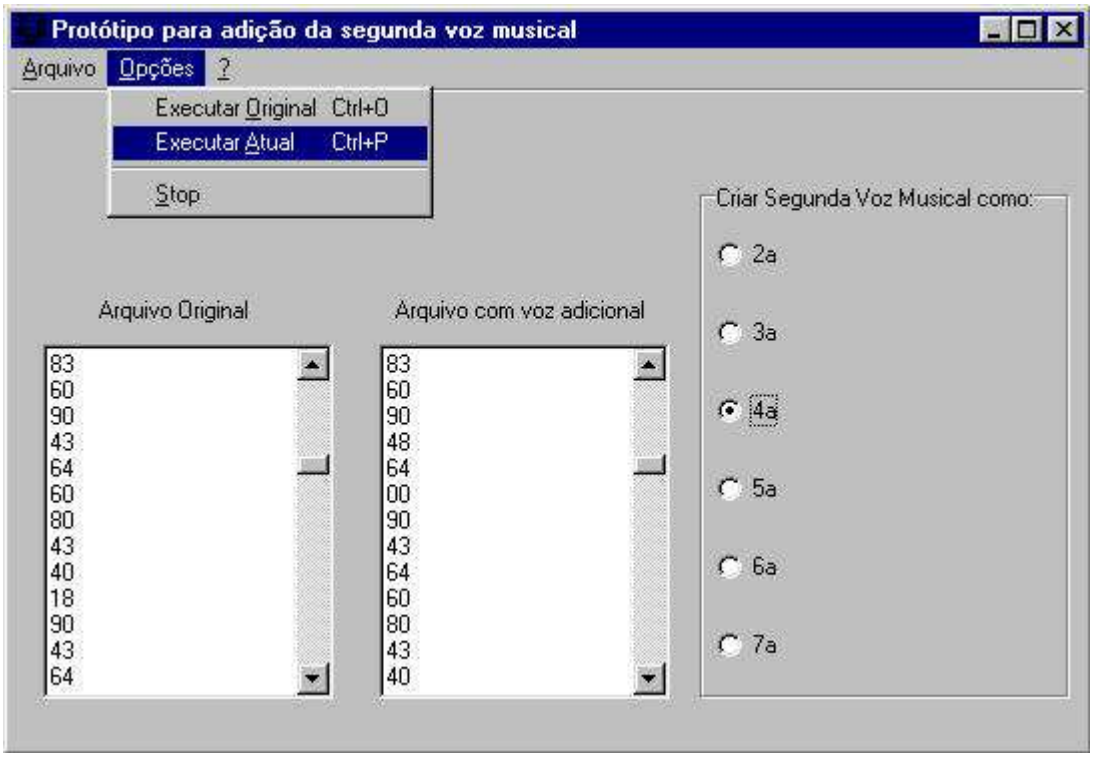
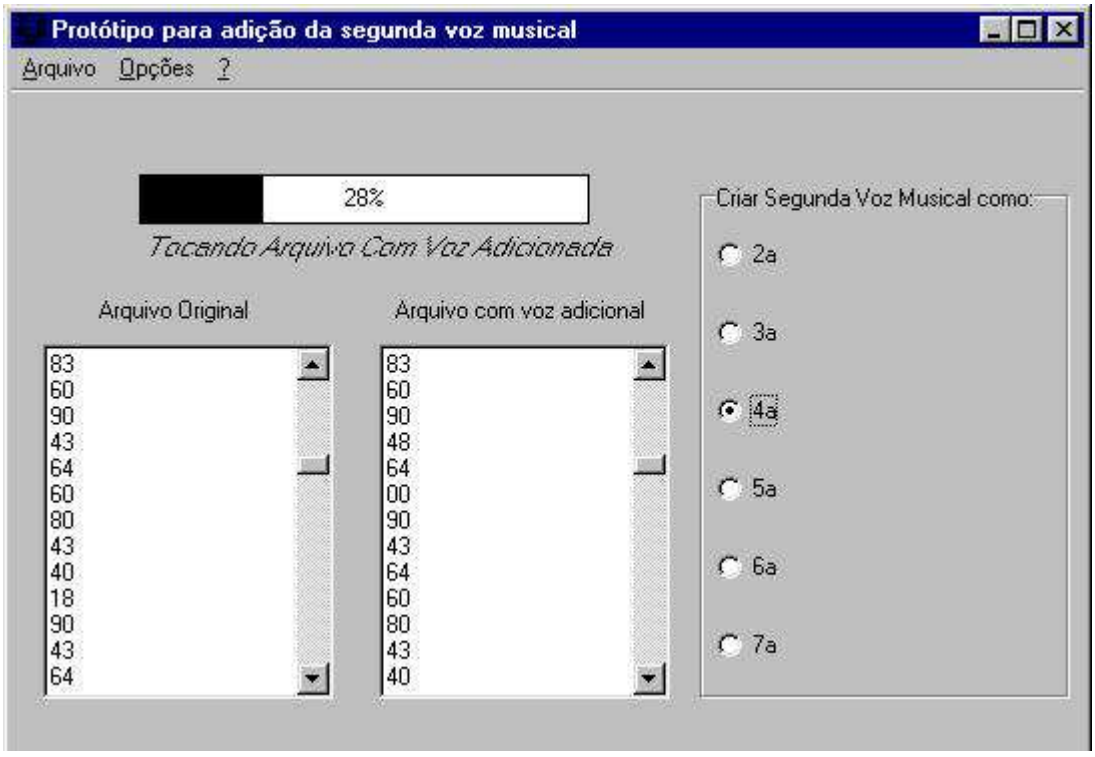
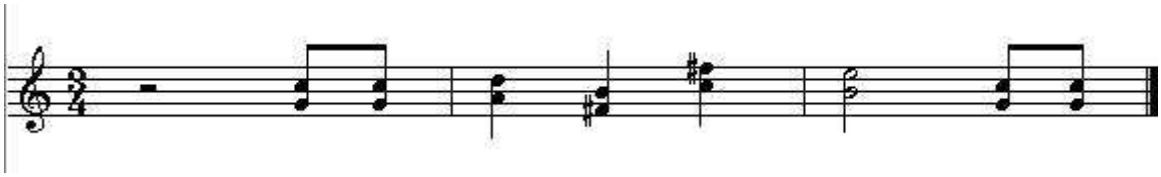


FIGURA 44 – TOCANDO ARQUIVO MIDI COM SEGUNDA VOZ MUSICAL



Após o protótipo ter concluído a criação das segundas notas, o arquivo salvo foi aberto no editor musical Encore 4.21 e pôde-se observar a segunda voz musical adicionada acima da primeira voz musical, neste caso, como uma 4ª (intervalo de quarta) conforme figura 45.

FIGURA 45 – ARQUIVO MIDI COM SEGUNDA VOZ MUSICAL



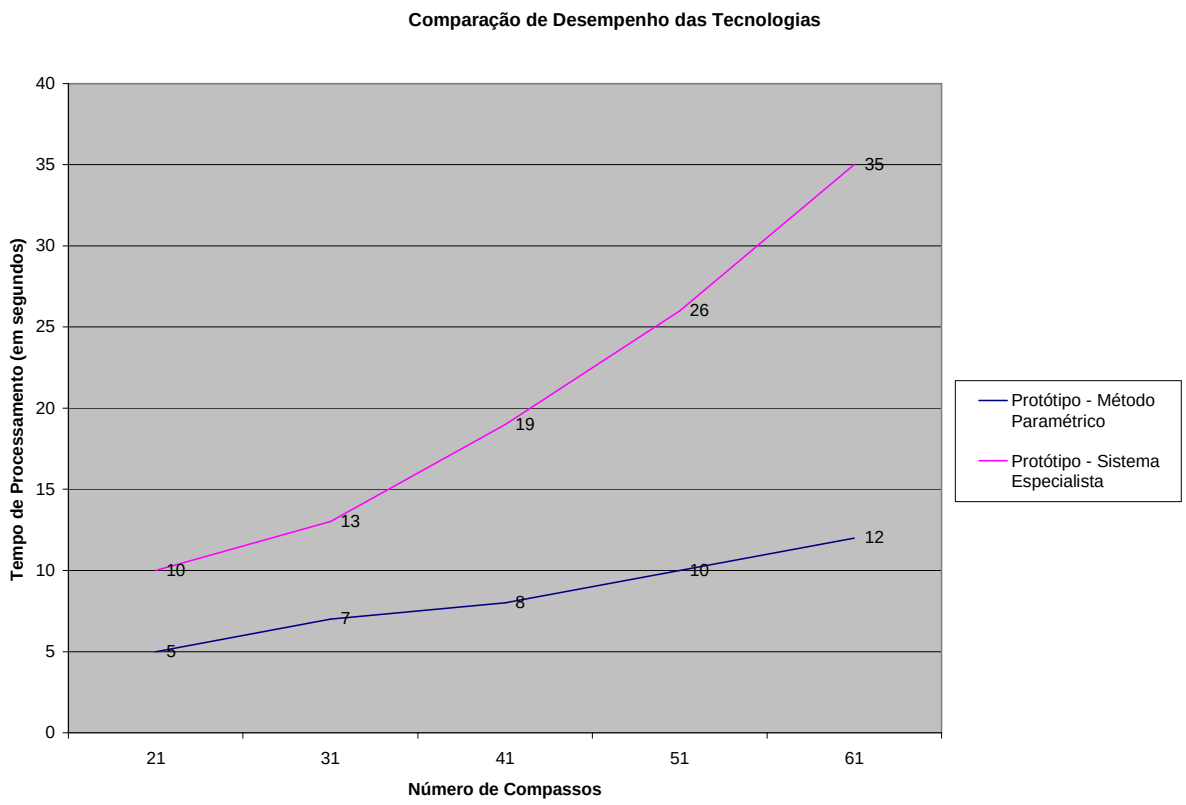
5 RESULTADOS E DISCUSSÃO

O protótipo atual apresenta uma diversidade maior para a criação de uma segunda voz musical, pois utiliza o conceito de intervalos. Enquanto que no protótipo de Maske (2000) poderia apenas ser criado um tipo de voz musical, nesta nova implementação existem 6 tipos diferentes, variando desde o intervalo[3] de 2ª. (segunda) até o intervalo de 7ª. (sétima). Outra melhora está relacionada aos tipos de acordes utilizados nas músicas. No protótipo atual pode-se utilizar acordes maiores e menores com acidentes, o que não era permitido anteriormente, além de permitir que qualquer acorde possa ser escrito em qualquer parte da partitura musical, isto é, dentro da pauta ou nas linhas suplementares. O protótipo de Maske (2000), somente permitia que os acordes fossem escritos dentro da pauta musical.

Com o abandono do sistema especialista no protótipo de Maske (2000) e sua substituição pelo método de parametrização, notou-se uma redução de tempo em seu desempenho, calculando e criando a segunda voz musical de uma maneira muito mais rápida e eficiente. Para comparar o desempenho e demonstrar a análise feita entre ambas as tecnologias, foi criado um gráfico demonstrativo, através da ferramenta Microsoft Excel, do cálculo de uma segunda voz musical para a música “Asa Branca”. Nota-se que a música foi analisada cinco vezes , mas em cada uma com o seu número de compassos alterados, modificando a duração ou tamanho da música, conforme eixo “X”. No eixo “Y” está representado a duração em segundos que cada tecnologia levou para a criação da segunda voz musical. Deve-se ressaltar que o protótipo atual (tecnologia de parametrização matemática), leva o mesmo tempo para a criação desta segunda voz independente da opção de intervalo selecionada (2ª, 3ª, 4ª, etc...).

[3] A definição do conceito de intervalos musicais encontra-se no item 2.2

FIGURA 46 – COMPARAÇÃO DE DESEMPENHO DAS TECNOLOGIAS



6 CONCLUSÕES

A proposta do presente trabalho foi de desenvolver uma aplicação semelhante àquela desenvolvida inicialmente por Maske (2000), porém, eliminando o uso de um sistema especialista. Em seu lugar foi implementado um método de parametrização matemática para a criação de uma segunda voz musical, objetivando auxiliar o músico na inclusão desta na escrita de partituras.

A dificuldade inicial foi a de encontrar uma forma paramétrica para identificar os modos das escalas, tornando assim possível o cálculo da segunda voz musical através do conceito de intervalos, já que este é baseado nas escalas e seus respectivos modos musicais. Após um estudo aprofundado em teoria musical e uma sucedida modelagem dos modos musicais em vetores, concluiu-se também que a música, segundo sua teoria, possui muitas características de uma ciência exata, tais como uma forte ligação com a lógica e a aritmética.

O método de parametrização desenvolvido neste trabalho demonstrou que os objetivos inicialmente propostos foram alcançados. O Sistema Especialista desenvolvido por Maske (2000) agora não utiliza mais as regras de sua base de conhecimento, tornando-se assim, não mais um Sistema Especialista e sim um protótipo baseado em parametrizações matemáticas para a formulação de uma segunda voz musical, tornando-se muito mais rápido, eficiente e fiel à teoria musical.

O protótipo implementado no presente trabalho possui as seguintes limitações:

a) foram utilizados somente acordes maiores e menores e seus respectivos modos na sua forma natural (Iônico, Dórico, Frígio, Lídio, Mixolídio, Eólio e Lócrio);
b) limitou-se a entrada de arquivo MIDI ao uso do formato 1, pois este formato separa os canais em blocos diferentes.

6.1 EXTENSÕES

Para as tecnologias apresentadas neste estudo, muitos caminhos poderão se abrir, mostrando um vasto número de aplicações possíveis. No caso específico deste trabalho, poderia ser feito um estudo mais aprofundado nos outros formatos de arquivo MIDI, já que o usado neste trabalho não salva as armaduras de sustenidos e bemóis no começo do arquivo, ao contrário dos outros, onde então já pode ser informado qual o tipo de modo usado na melodia

(Iônico, Dórico, etc...). Desta forma, o protótipo não precisaria ficar analisando nota por nota em uma determinada melodia para saber a que modo ela pertence, pois já estaria especificado no começo do arquivo.

Poderiam ser implementados, utilizando do método da parametrização, outros tipos de modos como os pertencentes às escalas menor harmônica ou melódica e seus respectivos acordes, abrangendo ainda mais a teoria musical.

REFERÊNCIAS BIBLIOGRÁFICAS

CANTU, Marco. **Dominando o Delphi 5** – bíblia. São Paulo: Makron Books. 2000. 860 p.

CHEDIAK, Almir. **Harmonia e Improvisação.** Rio de Janeiro: Lumiar Editora. 1986. 355 p.

FÁVERO, Alexandre José. **Sistemas especialistas**, [S.l.], [2001?]. Disponível em: <http://www.din.uem.br/ia/especialistas/>. Acesso em 05 set. 2001.

GONTIJO, Christian Haagensen. **Sítio Web sobre Midi**, [S.l.], [1998?]. Disponível em: <http://www.geocities.com/SiliconValley/Campus/7086/>. Acesso em: 28 ago. 2001.

LIA. **Sistemas inteligentes aplicados**, Fortaleza, abril [1997?]. Disponível em: <http://www.lia.ufc.br/~bezerra/exsinta/index.html>. Acesso em: 01 set. 2001.

MASKE, Roberto Carlos. **Protótipo de um sistema para auxiliar na composição musical utilizando-se de regras de harmonia.** 2000. 124 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

MOORS, Michael. **The unofficial rose web site**, [S.l.], [1999?]. Disponível em: <http://www.rationalrose.com/>. Acesso em: 16 set. 2001.

OLIVEIRA, Reinaldo G. **Teoria musical**, [S.l.], [1999?]. Disponível em: <http://www.geocities.com/gaitifera/teoria.html>. Acesso em: 25 ago. 2001.

PLATE, Eduardo. **Viol@o.hpg**, São Paulo, maio [2000?]. Dísponível em: <http://www.violao.hpg.ig.com.br/index.html>. Acesso em: 07 nov. 2001.

RATTON, Miguel. **Criação de música e sons por computador.** Rio de Janeiro: Campus. 1995. 202 p.

RIBEIRO, Horácio da Cunha e Souza. **Introdução aos sistemas especialistas.** Rio de Janeiro: Livros Técnicos e Científicos Editora. 1987. 203 p.

RIBEIRO, Wagner. **Elementos de Teoria da Música.** São Paulo: Coleção F.T.D. Editora. 1965. 122 p.

ANEXO A

Principais procedimentos e funções implementados no Protótipo (Delphi 5.0).

```
Procedimento Varredura;
//analisa cada nota no arquivo midi e seu respectivo acorde para
identificar o modo
Var
     Aux_Fim,Aux_Tempo_0:Boolean;
     Aux_i,Aux_Tempo_Compasso,Aux_Qde_Compasso,Aux_Indice:integer;
     Aux_Tempo1,Aux_Qde_Tempo1:variant;

 Begin
     Aux_i := i;
     Aux_Tempo_Compasso := Tempo_Compasso;
     Aux_Qde_Compasso := Qde_Compasso;
     Aux_Tempo_0 := Tempo_0;
     Aux_Tempo1 := Tempo1;
     Aux_Qde_Tempo1 := Qde_Tempo1;
     Aux_Fim := Fim;
     Aux_Indice:= Indice;
     While i <= Comeco_bloco3-12 do
    begin
         if      (((form2.memo1.lines[i]      =      '90')      and
((form2.memo1.lines[i+2] = '64')or(form2.memo1.lines[i+2] = '50')))
or
            ((form2.memo1.lines[i]        =        '80')        and
((form2.memo1.lines[i+2] = '40')or(form2.memo1.lines[i+2] = '50'))))
then
         begin
              if form2.memo1.lines[i] <> '80' then
              begin
                   inc (i);
                   nota:= form2.HexToInt(form2.memo1.lines[i]);
                   if ((Indice <> (Canal2.Count-2)) and (Indice <
(Canal2.Count-2))) then
                   begin
                        While          ((Tempo_Compasso          >=
(PMusica(Canal2[Indice+1])^.Tempo)) and
                                (Qde_Compasso                      =
(PMusica(Canal2[Indice+1])^.Compasso))) do
                        begin
                             inc(Indice);
                        end;
                   end
                   else
```

```
                  If not fim then
                     If             ((Tempo_Compasso             >=
(PMusica(Canal2[Indice+1])^.Tempo)) and
                      (Qde_Compasso                              =
(PMusica(Canal2[Indice+1])^.Compasso))) then
                     begin
                          Inc(Indice);
                          fim:= true;
                     end;
                   Acorde:= (PMusica(Canal2[indice])^.Acorde);
                   if ( Indice <> Aux_Indice ) then
                     begin

form2.Insercao(aux_i,Aux_Tempo_Compasso,Aux_Qde_Compasso,Aux_Indice,
Aux_Tempo1,Aux_Qde_Tempo1,Aux_Tempo_0,Aux_Fim);
                        Aux_i := i - 1;
                        Aux_Tempo_Compasso := Tempo_Compasso;
                        Aux_Qde_Compasso := Qde_Compasso;
                        Aux_Tempo_0 := Tempo_0;
                        Aux_Tempo1 := Tempo1;
                        Aux_Qde_Tempo1 := Qde_Tempo1;
                        Aux_Fim := Fim;
                        Aux_Indice:= Indice;
                        Modo:= 'Jônio';
                        end;

                    form2.Tonica_escala (Acorde);
                    procura_modo (nota);
                    i:= i + 2;
               end
               else
               begin
                 i:= i + 3;
               end;
          end
          else
          begin
               if (form2.memo1.lines[i] <> '00') then
               begin
                    If (form2.memo1.lines[i] <> 'C0') then
                    begin
                         If (form2.memo1.lines[i] < '80') then
                         begin
                              var1:=
form2.HexToInt(form2.memo1.lines[i]);
                              var2:= 0;
                              inc (i);
                         end
                         else
                         if   (form2.memo1.lines[i]   >   '7F')   and
(form2.memo1.lines[i+1] < '80')then
                         begin
```

```
                              var1:=
form2.HexToInt(form2.memo1.lines[i]);
                              var2:=
form2.HexToInt(form2.memo1.lines[i+1]);
                              var3:= 0;
                              i:= i + 2;
                         end
                         else
                         if   (form2.memo1.lines[i]   >   '7F')   and
(form2.memo1.lines[i+1]   >   '7F')and   (form2.memo1.lines[i+2]   <
'80')then
                         begin
                              var1:=
form2.HexToInt(form2.memo1.lines[i]);
                              var2:=
form2.HexToInt(form2.memo1.lines[i+1]);
                              var3:=
form2.HexToInt(form2.memo1.lines[i+2]);
                              var4:= 0;
                              i:= i + 3;
                         end
                         else
                         begin
                              var1:=
form2.HexToInt(form2.memo1.lines[i]);
                              var2:=
form2.HexToInt(form2.memo1.lines[i+1]);
                              var3:=
form2.HexToInt(form2.memo1.lines[i+2]);
                              var4:=
form2.HexToInt(form2.memo1.lines[i+3]);
                              i:= i + 4;
                         end;
                         //divide o valor em unidade de tempo
                         Tempo_0:= False;
                         Tempo1:=
(form2.ReadVarLen(var1,var2,var3,var4)/Unidade_Tempo);
                         Qde_Tempo1:= Qde_Tempo1 + Tempo1;
                         While (Qde_Tempo1 >= numerador) do
                         begin
                              Qde_Tempo1:= Qde_Tempo1 - numerador;
                              Inc (Qde_Compasso);
                         end;
                         Tempo_Compasso:= (Round(Qde_Tempo1+1));
                    end
                    else
                    begin
                //           for L:= comeco_bloco2 to comeco_bloco2+2
do
                //             memo2.lines.add(memo1.lines[L]);
                         i:= i + 2;
                    end;
               end
```

```
              else
              If (Tempo_0) and (form2.memo1.lines[i+1] <> 'C0') then
              begin
                 Tempo_0:= False;
                 inc (i);
              end
              else inc (i);
         end;
    end;

form2.Insercao(aux_i,Aux_Tempo_Compasso,Aux_Qde_Compasso,Aux_Indice,
Aux_Tempo1,Aux_Qde_Tempo1,Aux_Tempo_0,Aux_Fim);
 end;

Procedimento Insercao
(i,Tempo_Compasso,Qde_Compasso,Indice:integer;Tempo1,Qde_Tempo1:vari
ant;Tempo_0,Fim:boolean);
//Analisa cada nota no arquivo midi e calcula a sua respectiva
segunda voz musical, criando um novo arquivo midi

Var
  Aux_Indice:integer;
  Begin
  Aux_Indice:= Indice;
    While i <= Comeco_bloco3-12 do
    begin
         if      (((form2.memo1.lines[i]      =      '90')      and
((form2.memo1.lines[i+2] = '64')or(form2.memo1.lines[i+2] = '50')))
or
            ((form2.memo1.lines[i]        =        '80')        and
((form2.memo1.lines[i+2] = '40')or(form2.memo1.lines[i+2] = '50'))))
then
         begin
              if form2.memo1.lines[i] <> '80' then
              begin
                   inc (i);
                   nota:= form2.HexToInt(form2.Memo1.lines[i]);
                   if ((Indice <> (Canal2.Count-2)) and (Indice <
(Canal2.Count-2))) then
                   begin
                        While          ((Tempo_Compasso          >=
(PMusica(Canal2[Indice+1])^.Tempo)) and
                             (Qde_Compasso                        =
(PMusica(Canal2[Indice+1])^.Compasso))) do
                        begin
                             inc(Indice);
                        end;
                   end
                   else
                   If not fim then
                      If             ((Tempo_Compasso             >=
(PMusica(Canal2[Indice+1])^.Tempo)) and
```

```
                        (Qde_Compasso                                    =
(PMusica(Canal2[Indice+1])^.Compasso))) then
                    begin
                         Inc(Indice);
                         fim:= true;
                    end;
                Acorde:= (PMusica(Canal2[indice])^.Acorde);
                if ( Indice <> Aux_Indice ) then
                  begin
                  break;
                  end;

                form2.Memo2.lines.Add('90');
                Retorno_aux:= RetornaSolucao(Acorde,nota);
                Nota_Retorno:=
copy(form2.IntToHex(Retorno_aux),3,2);
                form2.Memo2.Lines.Add(Nota_Retorno);
                form2.Memo2.Lines.Add('64');
                form2.Memo2.Lines.Add('00');
                form2.Memo2.lines.Add('90');
                form2.Memo2.lines.Add(form2.Memo1.lines[i]);
                form2.Memo2.lines.Add('64');
                i:= i + 2;
            end
            else
            begin
                form2.Memo2.lines.Add('80');
                form2.Memo2.Lines.Add(form2.Memo1.lines[i+1]);
                form2.Memo2.Lines.Add('40');
                form2.Memo2.lines.Add('00');
                form2.Memo2.lines.Add('80');
                Retorno_aux:= RetornaSolucao(Acorde,nota);
                Nota_Retorno:=
copy(form2.IntToHex(Retorno_aux),3,2);
                form2.Memo2.Lines.Add(Nota_Retorno);
                form2.Memo2.lines.Add('40');
                i:= i + 3;
            end;
        end
        else
        begin
            if (form2.memo1.lines[i] <> '00') then
            begin
                If (form2.memo1.lines[i] <> 'C0') then
                begin
                    If (form2.memo1.lines[i] < '80') then
                    begin

form2.memo2.lines.add(form2.memo1.lines[i]);
                        var1:=
form2.HexToInt(form2.memo1.lines[i]);
                        var2:= 0;
                        inc (i);
```

```
                        end
                        else
                        if   (form2.memo1.lines[i]   >   '7F')   and
(form2.memo1.lines[i+1] < '80')then
                        begin

form2.memo2.lines.add(form2.memo1.lines[i]);

form2.memo2.lines.add(form2.memo1.lines[i+1]);
                             var1:=
form2.HexToInt(form2.memo1.lines[i]);
                             var2:=
form2.HexToInt(form2.memo1.lines[i+1]);
                             var3:= 0;
                             i:= i + 2;
                        end
                        else
                        if   (form2.memo1.lines[i]   >   '7F')   and
(form2.memo1.lines[i+1]   >   '7F')and   (form2.memo1.lines[i+2]   <
'80')then
                        begin

form2.memo2.lines.add(form2.memo1.lines[i]);

form2.memo2.lines.add(form2.memo1.lines[i+1]);

form2.memo2.lines.add(form2.memo1.lines[i+2]);
                             var1:=
form2.HexToInt(form2.memo1.lines[i]);
                             var2:=
form2.HexToInt(form2.memo1.lines[i+1]);
                             var3:=
form2.HexToInt(form2.memo1.lines[i+2]);
                             var4:= 0;
                             i:= i + 3;
                        end
                        else
                        begin

form2.memo2.lines.add(form2.memo1.lines[i]);

form2.memo2.lines.add(form2.memo1.lines[i+1]);

form2.memo2.lines.add(form2.memo1.lines[i+2]);

form2.memo2.lines.add(form2.memo1.lines[i+3]);
                             var1:=
form2.HexToInt(form2.memo1.lines[i]);
                             var2:=
form2.HexToInt(form2.memo1.lines[i+1]);
                             var3:=
form2.HexToInt(form2.memo1.lines[i+2]);
```

```
                              var4:=
form2.HexToInt(form2.memo1.lines[i+3]);
                              i:= i + 4;
                         end;
                         //divide o valor em unidade de tempo
                         Tempo_0:= False;
                         Tempo1:=
(form2.ReadVarLen(var1,var2,var3,var4)/Unidade_Tempo);
                         Qde_Tempo1:= Qde_Tempo1 + Tempo1;
                         While (Qde_Tempo1 >= numerador) do
                         begin
                              Qde_Tempo1:= Qde_Tempo1 - numerador;
                              Inc (Qde_Compasso);
                         end;
                         Tempo_Compasso:= (Round(Qde_Tempo1+1));
                    end
                    else
                    begin
                         for L:= comeco_bloco2 to comeco_bloco2+2 do

form2.memo2.lines.add(form2.memo1.lines[L]);
                         i:= i + 2;
                    end;
               end
               else
               If (Tempo_0) and (form2.memo1.lines[i+1] <> 'C0') then
               begin
                  form2.memo2.lines.add('00');
                  Tempo_0:= False;
                  inc (i);
               end
               else inc (i);
          end;
     end;
  end;

Procedimento Tonica_Escala (Acorde:string);
  Var
   Base: array [1..14] of integer;
   i:integer;
   Begin
     Base[1]:= 0; Base[2]:= 2; Base[3]:= 4; Base[4]:= 5; Base[5]:=
7;
     Base[6]:=  9;  Base[7]:=  11;  Base[8]:=  12;  Base[9]:=  14;
Base[10]:=16;
     Base[11]:=17; Base[12]:=19; Base[13]:=21; Base[14]:=23;
     notabase := form2.retorna_notabase(Acorde);
     for i := 1 to 7 do        //carrega as notas no vetor notas
       begin
         vetnotas[i]:=  Base[i]  +  notabase;  //  carrega  modo  jônio
para consulta
         if vetnotas[i] > 11 then
            vetnotas[i]:= vetnotas[i] - 12;
```

```
       end;
    end;

Função Retorna_Notabase (Acorde:string):integer;
//Retorna tônica (em decimal) para cálculo da escala
 Begin
   if (Acorde = 'Dó Maior') or (Acorde = 'Lá Menor')then
   Result:= 0;
   if (Acorde = 'Dó# Maior') or (Acorde = 'Lá# Menor')then
   Result:= 1;
   if (Acorde = 'Ré Maior') or (Acorde = 'Si Menor')then
   Result:= 2;
   if (Acorde = 'Ré# Maior') or (Acorde = 'Dó Menor')then
   Result:= 3;
   if (Acorde = 'Mi Maior') or (Acorde = 'Dó# Menor')then
   Result:= 4;
   if (Acorde = 'Fá Maior') or (Acorde = 'Ré Menor')then
   Result:= 5;
   if (Acorde = 'Fá# Maior') or (Acorde = 'Ré# Menor')then
   Result:= 6;
   if (Acorde = 'Sol Maior') or (Acorde = 'Mi Menor')then
   Result:= 7;
   if (Acorde = 'Sol# Maior') or (Acorde = 'Fá Menor')then
   Result:= 8;
   if (Acorde = 'Lá Maior') or (Acorde = 'Fá# Menor')then
   Result:= 9;
   if (Acorde = 'Lá# Maior') or (Acorde = 'Sol Menor')then
   Result:= 10;
   if (Acorde = 'Si Maior') or (Acorde = 'Sol# Menor')then
   Result:= 11;
  end;

Procedimento Procura_Modo (nota:integer);
//procura por notas que descaracterizam o modo default Jônio
 Var
  Lidio,Mixolidio:integer;
   Begin
       Lidio:= vetnotas[4] + 1;
       Mixolidio:= vetnotas[6] + 1;
       if (Lidio > 11) then
           Lidio := Lidio - 12;
       if (Mixolidio > 11) then
           Mixolidio:= Mixolidio - 12;
       if ( Lidio = nota mod 12) then
           Modo:= 'Lídio';
       if ( Mixolidio = nota mod 12) then
           Modo:= 'Mixolídio';
    end;


Função RetornaSolucao (Acorde:string; nota2:integer):word;

 Var
```

```
  nota3:integer;
  i:integer;
  vetornotas: array [1..7] of integer;

 Begin
   //Modos dos Acordes Maiores
 Jonio[1]:= 0; Jonio[2]:= 2; Jonio[3]:= 4; Jonio[4]:= 5; Jonio[5]:=
 7; Jonio[6]:= 9;                          Jonio[7]:= 11;  Jonio[8]:=
 12; Jonio[9]:= 14; Jonio[10]:= 16; Jonio[11]:= 17;
 Jonio[12]:= 19; Jonio[13]:= 21; Jonio[14]:= 23;
 Lidio[1]:= 0; Lidio[2]:= 2; Lidio[3]:= 4; Lidio[4]:= 6; Lidio[5]:=
 7; Lidio[6]:= 9;      Lidio[7]:= 11; Lidio[8]:= 12; Lidio[9]:= 14;
 Lidio[10]:= 16; Lidio[11]:= 18;
 Lidio[12]:= 19; Lidio[13]:= 21; Lidio[14]:= 23;
 Mixolidio[1]:=    0;    Mixolidio[2]:=    2;    Mixolidio[3]:=    4;
 Mixolidio[4]:=    5;    Mixolidio[5]:=    7;    Mixolidio[6]:=    9;
 Mixolidio[7]:=   10;      Mixolidio[8]:=   12;   Mixolidio[9]:=   14;
 Mixolidio[10]:=   16;   Mixolidio[11]:=   17;   Mixolidio[12]:=   19;
 Mixolidio[13]:= 21; Mixolidio[14]:=  22;

{//Modos dos Acorde Menores "No presente momento não precisa ser
utilizado, devido às aquivalências dos acordes maiores e menores"
 Dorico[1]:=  0;  Dorico[2]:=  2;  Dorico[3]:=  3;  Dorico[4]:=  5;
 Dorico[5]:=  7;  Dorico[6]:=  9;  Dorico[7]:=  10;   Dorico[8]:=  12;
 Dorico[9]:= 14; Dorico[10]:= 15; Dorico[11]:= 17; Dorico[12]:= 19;
 Dorico[13]:= 21; Dorico[14]:= 22;
 Frigio[1]:=  0;  Frigio[2]:=  1;  Frigio[3]:=  3;  Frigio[4]:=  5;
 Frigio[5]:=  7;  Frigio[6]:=  8;  Frigio[7]:=  10;  Frigio[8]:=  12;
 Frigio[9]:= 13; Frigio[10]:= 15; Frigio[11]:= 17;
 Frigio[12]:= 19; Frigio[13]:= 20; Frigio[14]:= 22;
 Eolio[1]:= 0; Eolio[2]:= 2; Eolio[3]:= 3; Eolio[4]:= 5; Eolio[5]:=
 7; Eolio[6]:= 8;
 Eolio[7]:=  10;  Eolio[8]:=  12;  Eolio[9]:=  14;  Eolio[10]:=  15;
 Eolio[11]:= 17;
 Eolio[12]:= 19; Eolio[13]:= 20;Eolio[14]:= 22;}

   carrega_modo(Modo); //carrega o modo
     for i := 1 to 7 do        //carrega as notas no vetor notas
      begin
       vetornotas[i]:= vetmodo[i] + notabase;
       if vetornotas[i] > 11 then
          vetornotas[i]:= vetornotas[i] - 12;
      end;

  for i := 1 to 7 do
   Begin
     nota3 := vetornotas[i];
     if (nota3 = nota2 mod 12) then
         break;
   end;
   Result:= retorna_voz(i,nota2); //retorna segunda voz
end;
```

```
Procedimento Carrega_Modo(Modo:string);
//carrega o modo correto para criação da segunda voz
var
 i:integer;
 Begin
   if Modo = 'Jônio' then
       Begin
         for i := 1 to 14 do
         vetmodo[i] := Jonio[i];
       end;

   if Modo = 'Lídio' then
       Begin
         for i := 1 to 14 do
         vetmodo[i] := Lidio[i];
       end;

   if Modo = 'Mixolídio' then
       Begin
         for i := 1 to 14 do
         vetmodo[i] := Mixolidio[i];
       end;

 end;

Função Retorna_Voz (i:integer;nota2:integer):word;
//Retorna a voz musical conforme o tipo selecionado pelo usuário
(variável voz)
    Begin
      voz := 1 + SegundaVoz.ItemIndex;
      Result:= nota2 + (vetmodo[i+voz] - vetmodo[i]);
    end;
```