FURB – Universidade Regional de Blumenau
DSC – Departamento de Sistemas e Computação
Grupo de Pesquisa em Computação Gráfica, Processamento de Imagens e Entretenimento Digital
Disciplina: Sistemas Multimídia - prof. Dalton Solano dos Reis

Projeto - Sistema Multimídia – 2014/2

Etapa 01 (N2)

Equipe (nomes): ANDRÉ RAMACIOTTI / FERNANDO JUNKES / JÚLIO BATISTA

Título: LEAPMÚSICA

Problema/Dúvida: Validar a uso do Leap Motion em diferentes formas de utilizar o computador e verificar as oportunidades de interação e aprendizado musicais disponíveis com o uso deste dispositivo.

Objetivos: Desenvolver uma aplicação que possibilite o ensino de piano. A aplicação irá permitir carregar um arquivo de partitura musical que será utilizado para informar quais teclas devem ser pressionadas para tocar a música desejada. Ao pressionar as teclas virtuais serão emitidos os sons das devidas teclas pressionadas.

Relevância: Este trabalho visa permitir a introdução musical com um equipamento portátil e relativamente barato se comparado a um piano eletrônico.

Do ponto de vista computacional, este trabalho é relevante por trabalhar com um dispositivo recente, o Leap Motion e o editor Unity 3D.

Hardware escolhido: Leap Motion.

Etapa 02 (N3)

A seguir serão descritos conceitos teóricos relacionados ao projeto de pesquisa proposto.

2.1 CENÁRIO DO TRABALHO PROPOSTO

O trabalho proposto objetiva facilitar a introdução à educação musical. Para isso, será modelado um teclado virtual que será tocado através do uso do dispositivo Leap Motion. Através dessa ferramenta, o aluno terá noções básicas do que compõe uma música, como notas, ritmo, dinâmica, harmonia e melodia (PARANÁ, 2014).

De maneira sucinta, podem-se definir esses conceitos como na lista abaixo. Esses itens também explicam como os conceitos se relacionam com como um teclado é tocado.

- Nota: determina se um som é mais mais grave ou mais agudo. Formalmente, diz-se que uma nota mais grave é mais baixa que uma nota mais aguda (mais alta). Isso pode causar alguma confusão, pois leigos estão habituados a usar essas palavras para descrever a dinâmica. Em um teclado, cada tecla representa uma nota diferente.
- Ritmo: é determinado pela duração dos sons, que pode ser longa ou curta. No teclado, o ritmo é delimitado pela frequência com que as teclas são pressionadas.
- Dinâmica: dita a intensidade de um som, podendo ser mais fraca ou mais forte. Geralmente, quando um leigo descreve um som como "baixo" ou "alto" é a isso que ele se refere. No caso do teclado, a dinâmica depende da velocidade com que as teclas são pressionadas, se são pressionadas com mais ou menos força.
- Harmonia: é a combinação de sons sendo reproduzidos simultaneamente. Quando essa combinação tem um resultado agradável, diz-se que o som é harmonioso. Em um teclado, a harmonia é importante principalmente quando se pressionam duas ou mais teclas ao mesmo tempo.
- Melodia: é uma sequência de sons Normalmente, é a parte mais destacada de uma música. No caso do teclado, uma melodia é construída pressionando-se teclas em determinada sequência e em determinado ritmo.

2.2 TRABALHOS CORRELATOS

2.2.1 GarageBand

O GarageBand é um aplicativo desenvolvido pela Apple que traz inúmeras ferramentas que podem ajudar praticantes de música em todas as esferas (DUARTE, 2014, p. 33). Segundo Duarte (2014, p.33) "O aplicativo pode ser muito útil para músicos iniciantes ou músicos experientes, pois o software possui uma grande coleção de instrumentos virtuais que o usuário pode escolher e começar a tocar de maneira rápida e intuitiva.".

O aplicativo vem com uma coleção de lições de piano e violão e é possível ter feedback instantâneo enquanto toca para ajudar a aprimorar suas habilidades (APPLE, 2014, tradução nossa). As características *Magic* e *Lessons* reforçaram a ideia da Apple utilizar o GarageBand como uma ferramenta de aprendizado (MUSICRADAR, 2011, tradução nossa).

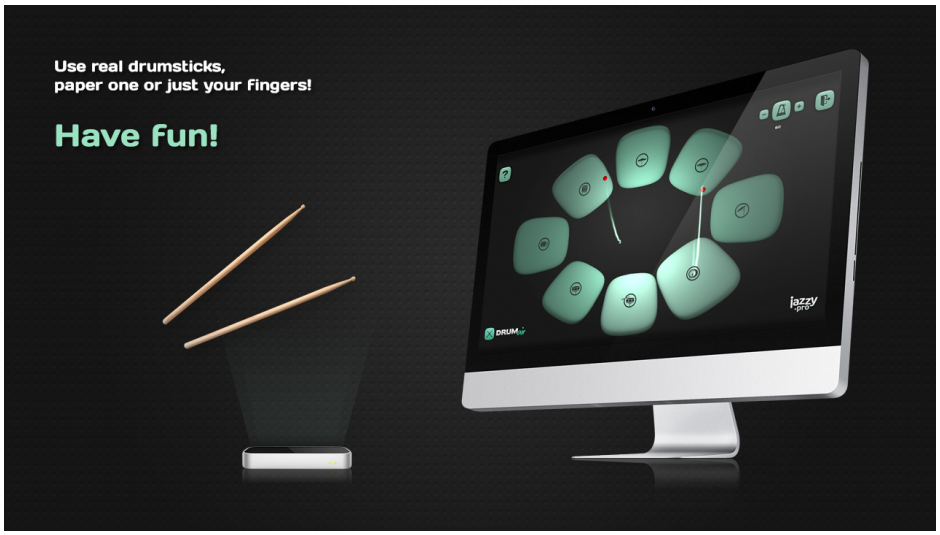
Segundo Duarte (2014, p. 33) o GarageBand dispõe de instrumentos como "guitarras [...], violão, baixos, baterias [...], pianos, órgãos, sintetizadores, naipe completo de cordas friccionadas, tocador e editor de sampler, gravador de áudio com efeitos para a voz e simuladores de efeitos de guitarra.". Além dos instrumentos, também estão disponíveis sintetizadores híbridos, digitais e analógicos para sons eletrônicos (MUSICRADAR, 2011, tradução nossa).

É possível gravar instrumentos através de software, entradas USB, dispositivos MIDI e teclados em tela (BEATON; HARRISON; TATAR, 2010, p. 1420, tradução nossa). Segundo Duarte (2014, p. 33) "As ferramentas nele contidas possibilitam desde a realização de gravações com qualidade profissional, com uso de samplers, instrumentos virtuais, simuladores de efeitos para guitarra, gravação multi-pista e toda sorte de recursos modernos encontrados em softwares de gravação profissional [...]".

2.2.2 DRUMair e Bongos!

O DRUMair (Figura 1) é um aplicativo desenvolvido pela Jazzy Innovations que traz um conjunto de bateria para tocar utilizando o Leap Motion. O aplicativo permite utilizar tanto as mãos quando baquetas para tocar os objetos virtuais que emitem o som da bateria.

Figura 1 - Aplicativo DRUMair



Fonte: Innovations (2013)

O aplicativo Bongos! (Figura 2) permite ao usuário tocar um conjunto de dois bongos através do Leap Motion. Para tocar os instrumentos, o usuário utiliza as próprias mãos. O aplicativo trás 3 pares de bongos diferentes com áudios gravados com alta fidelidade a 44.1khz (ROTWEIN, 2013). Um diferencial do aplicativo está na possibilidade de fazer o som sair mais alto de acordo com a força utilizada para bater no objeto virtual.

Figura 2 - Aplicativo Bongos!

Fonte: Rotwein (2013)

2.2.3 LeapGesture

Nowiki et al. (2014) apresenta o desenvolvimento de uma biblioteca para reconhecimento de gestos estáticos e dinâmicos para o dispositivo Leap Motion. Foram utilizadas máquinas de vetores de suporte para o reconhecimento de gestos estáticos e modelos ocultos de Markov para reconhecimento de gestos dinâmicos.

Segundo Nowiki el al. (2014, p. 19, tradução nossa) "As máquinas de vetores de suporte foram escolhidas porque existe uma fundamentação matemática sólida apoiando a simples ideia de maximização da margem entre classes". Como implementação de SVM foi utilizada a biblioteca libSVM, entretanto, devido a problemas de desempenho durante o treinamento para classificação dos gestos, também foram feitos testes com a biblioteca libLinear. A troca da biblioteca melhorou o desempenho do treinamento, que passou de aproximadamente 12 horas, para 5 segundos com 5000 exemplos. Entretanto, a troca de biblioteca implicou em uma perda de aproximadamente 5% à 17% na precisão do reconhecimento.

Os modelos ocultos de Markov foram implementados de forma que cada estado sempre tem uma transição para ele mesmo e para o próximo estado. Segundo Nowiki el al. (2014, p. 35, tradução nossa) "as transições para si mesmo foram utilizadas para modelar diferentes velocidades dos gestos e permite conseguir um sistema mais robusto".

A biblioteca atingiu 99% de precisão no reconhecimento 5 gestos estáticos e 85% de precisão para o reconhecimento de 10 gestos estáticos. Na tarefa de reconhecimento de gestos dinâmicos, a precisão foi de 85% para um conjunto com seis gestos dinâmicos.

No Quadro 1 pode ser observado as principais características entre o trabalho proposto e os correlatos.

Quadro 1 – proposto versos correlatos

Características	Proposto	GarageBand	DRUMair e Bongos!	LeapGesture
Utiliza o LeapMotion	sim	não	sim	sim
Auxilia no ensino para tocar instrumentos musicais	sim	sim	não	não
Comunicação com instrumentos reais	não	sim	não	não

2.3 CONCEITOS RELACIONADOS A SISTEMAS MULTIMÍDIA

2.3.1 Áudio Digital

A digitalização do som é a forma de trazer o som para o ambiente computacional, representando-o numericamente e assim, podendo alterá-lo e reproduzi-lo (ALVARENGA, 2013, p. 14). Segundo Alvarenga (2013, p. 14) "O aúdio digitalizado gera um grande volume

de dados, porém contém apenas informação sonora.".

Segundo Menezes Júnior (2007, p. 11) "MIDI (Musical Instrument Digital Interface) é um protocolo padrão que apresenta um conjunto de mensagens capazes de levar toda a informação necessária a um equipamento musical eletrônico digital para torná-lo capaz de gerar ou reproduzir músicas.". Este protocolo consiste em pacotes multi-byte contendo variáveis que descrevem eventos musicais como nome da nota, velocidade de ativação pela tecla do controlador e o canal de atribuição e é comumente utilizado para manipular informações musicais (ALVARENGA, 2013, p. 14; COLLYER; BOATHRIGHT-HOROWITZ; HOOPER, 1997, p. 347, tradução nossa).

Segundo Roads (1996, p. 999, tradução nossa) "Embora que a especificação MIDI original tenha padronizado a linguagem de controle musical, ela não descreve um formato de arquivo para esses dados.". Com isto, a comunidade MIDI adotou uma extensão a especificação MIDI chamada de Standard MIDI Files (SMF) onde o objetivo principal era a troca de sequências de dados criados em diferentes programas (ROADS, 1996, p. 999, tradução nossa).

O processo de geração do som utilizando o protocolo MIDI era a síntese subtrativa e posteriormente passou a ser substituída pela síntese por wavetable que gera o som a partir de amostras digitais de sons natuais (MENEZES JÚNIOR, 2007, p. 31). Devido ao interesse de algumas empresas começarem a explorar os recursos da síntese por wavetable surgiu a tecnologia SoundFont (MENEZES JÚNIOR, 2007, p. 32).

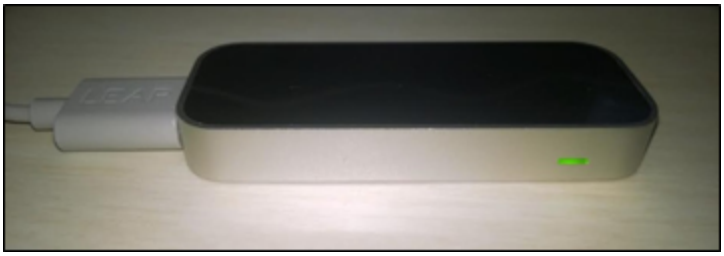
Segundo Menezes Júnior (2007, p. 3) "[...] podemos entender que SoundFonts são bancos de timbres que podem ser gravados diretamente de instrumentos acústico, editados e armazenados na memória das placas de som que suportam esta tecnologia.". Este formato pode ser comparado as fontes de texto, pois ele armazena os timbres utilizados para audição do conteúdo musical de um arquivo MIDI (MENEZES JÚNIOR, 2007, p. 32). Segundo Menezes Júnior (2007, p. 33) um arquivo SoundFont armazena "[...] amostras de áudio digitalizadas e as instruções para síntese, tais como características de filtros, loop, parâmetros do envelope de volume tais como tempos de ataque, sustain, release e outros.".

2.3.2 Leap Motion

O Leap Motion (Figura 3) é um dispositivo desenvolvido pela empresa Leap Motion, Inc. e promete precisão milimétrica para detecção das mãos (WEICHERT et al., 2013, tradução nossa). O *hardware* do dispositivo é um tanto simples, consistindo apenas de duas

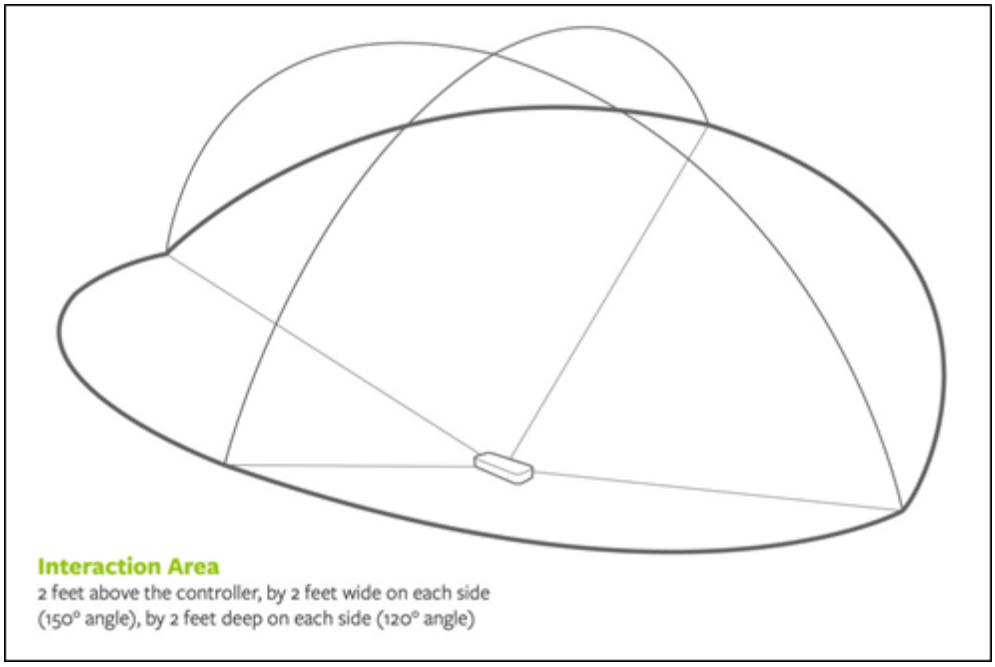
câmeras estéreo e três LEDs infravermelhos (COLGAN, 2014, tradução nossa).

Figura 3 - Leap Motion.



O *hardware* do dispositivo oferece uma área de interação de aproximadamente 2,5 metros cúbicos (Figura 4), consistindo em 60,960 centímetros acima do controle, 60,960 centímetros ou 150º para cada lado e 60,960 centímetros ou 120º de profundidade para cada lado (COLGAN, 2014, tradução nossa).

Figura 4 - Área de interação do Leap Motion.



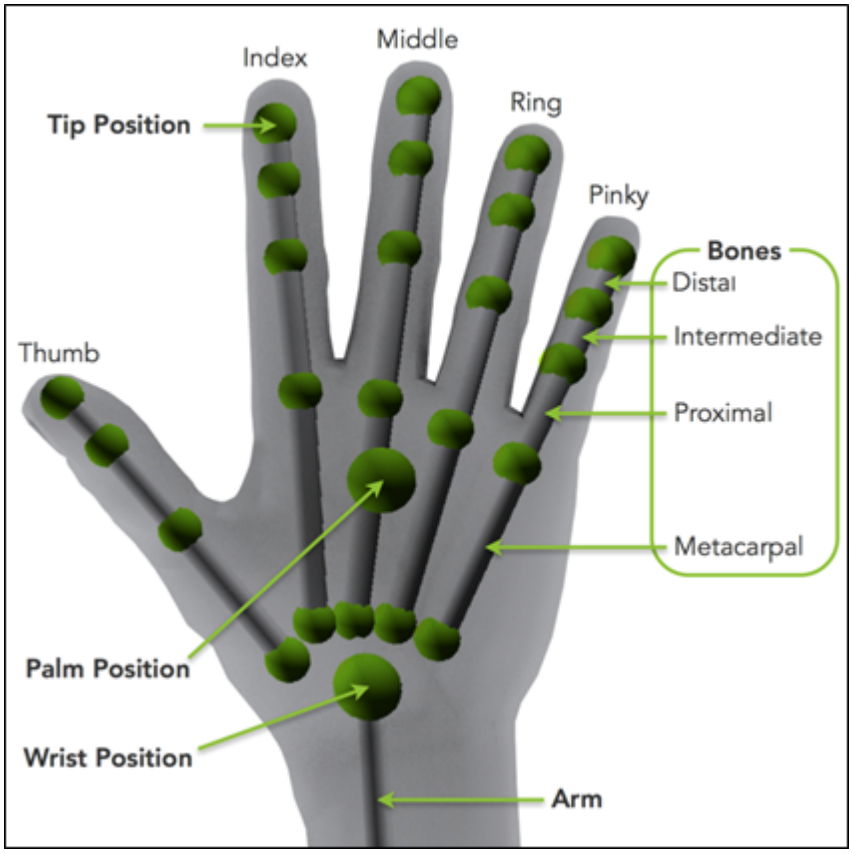
Fonte: Colgan (2014).

O dispositivo lê os dados dos sensores em sua memória local e faz os ajustes necessários para então enviar os dados para o computador, onde são feitos os processamentos matemáticos (COLGAN, 2014, tradução nossa). Segundo Colgan (2014, tradução nossa) "Diferentemente do que se pensa, o Leap Motion não gera um mapa de profundidade – em vez disso ele aplica algoritmos avançados aos dados puros do sensor". Após o processamento dos dados do sensor, uma camada com algoritmos de rastreamento extrai informações como dedos, ferramentas e infere a posição de objetos oclusos (COLGAN, 2014, tradução nossa).

Também são aplicadas técnicas de filtragem para garantir coerência temporal dos dados (COLGAN, 2014, tradução nossa).

Os dados processados são enviados no formato de frames através de um protocolo de transporte utilizando o protocolo TCP para aplicações nativas, ou *WebSocket* para aplicações web. De acordo com Davis (2014, tradução nossa) "No nível mais básico, a API do Leap Motion retorna os dados de rastreamento na forma de frames". Em cada frame é possível acessar as entidades rastreadas como mãos (Figura 5), dedos e ferramentas, objetos representando gestos reconhecidos, descrição do movimento das mãos na cena e também os dados puros do sensor (DAVIS, 2014, tradução nossa).

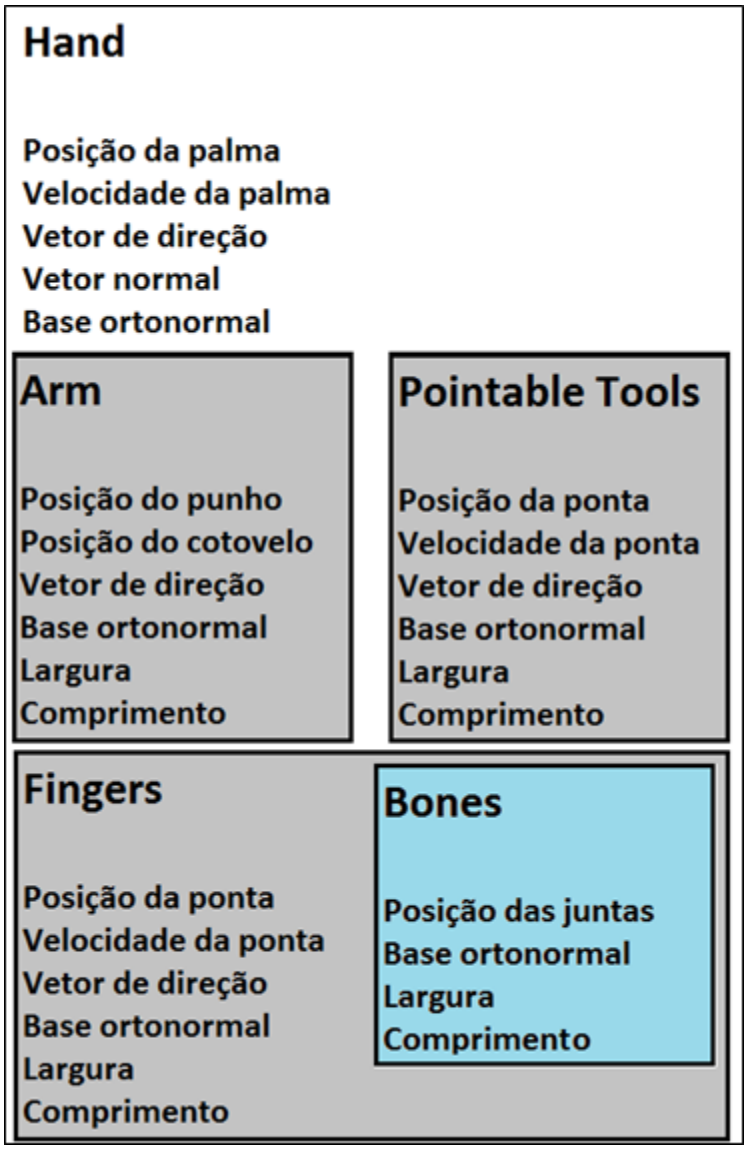
Figura 5 - Informações da mão retornadas pela API do Leap Motion.



Fonte: Davis (2014)

Toda entidade rastreada está na hierarquia do objeto *Hand* (Figura 6). A API também consegue distinguir entre mão direita e mão esquerda e informar a força dos movimentos agarrar e pinça (DAVIS, 2014, tradução nossa).

Figura 6 - Hierarquia de objetos rastreados pela API do Leap Motion.



A hierarquia apresentada na Figura 6 exibe como a API do Leap Motion retorna os dados do sensor. Inicialmente existe um objeto *Hand* com informações sobre a palma da mão. Este objeto ainda é composto pelo objeto *Arm* que contém informações sobre o punho do usuário e uma coleção de ferramentas apontadoras (*Pointable Tools*), que podem ser canetas, por exemplo. Por fim, o objeto *Hand* também contém uma coleção de dedos (*Fingers*) que traz informações sobre cada dedo de cada mão, onde cada dedo contém um conjunto de informações sobre os ossos (*Bones*) que formam o dedo.

2.4 AMBIENTES UTILIZADOS NO DESENVOLVIMENTO

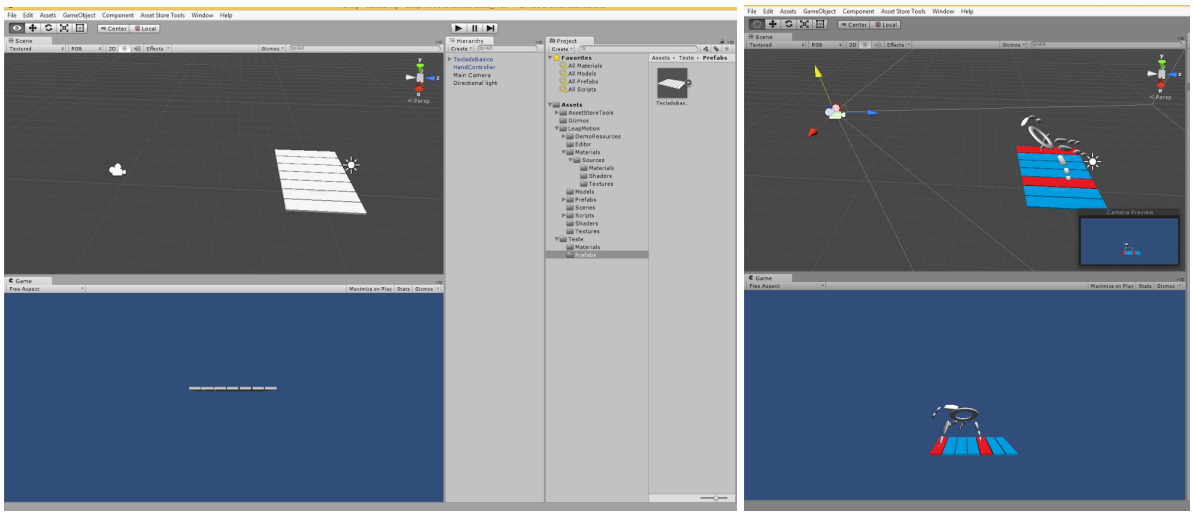
Para o desenvolvimento do projeto utilizaremos o Unity3d para criação do cenário e o Visual Studio para ambiente de desenvolvimento. Dentre as bibliotecas que utilizamos se encontra a SDK do LeapMotion e uma biblioteca chamada Sanford Multimidia.

2.4.1 Unity3d

A principal ferramenta do projeto. O Unity3d (Figura 7) é uma engine para desenvolvimento de jogos que vem amplamente sendo utilizada para desenvolvimento de outras aplicações multimídias e/ou mobile. Como exemplo o LeapMotion e o Vuforia.

Não há nenhuma particularidade na utilização do Unity3d para o LeapMotion, basta ter a SDK instalada e importar os *assets* do LeapMotion disponibilizados no site do fabricante.

Figura 7 - O ambiente Unity 3d

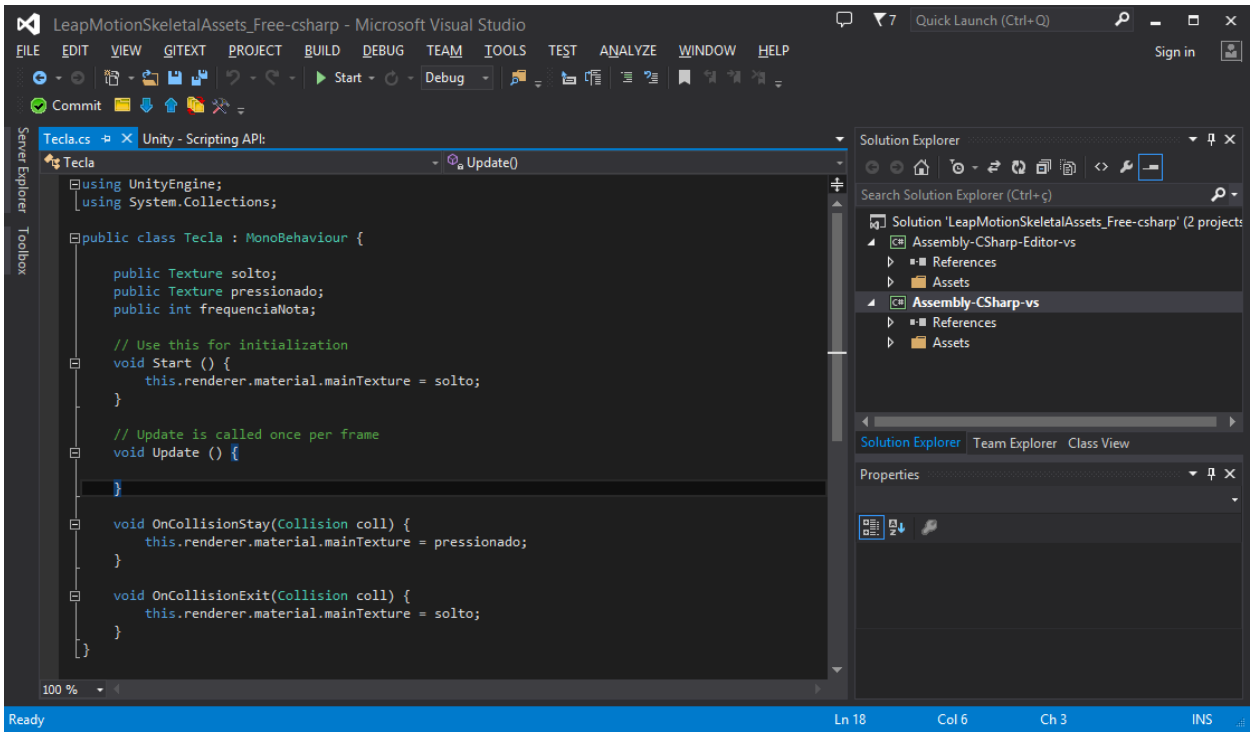


Adicionando a cena o componente do LeapMotion para gerar a mão, e criando um teclado modelo para nossos primeiros testes. Utilizamos em conjunto as DLLs de Sanford para emitir os sons das teclas.

2.4.2 Visual Studio

Por padrão o editor de código do Unity3d é o MonoDevelop, porém optamos pela utilização do Visual Studio por ser uma ferramenta mais completa. Até a versão 4.5 do Unity3d havia um problema de compatibilidade com o Visual Studio 2013 (Figura 8), porém na versão 4.6 isso já não ocorre, permitindo configurar no próprio Unity3d para utiliza-lo como editor padrão.

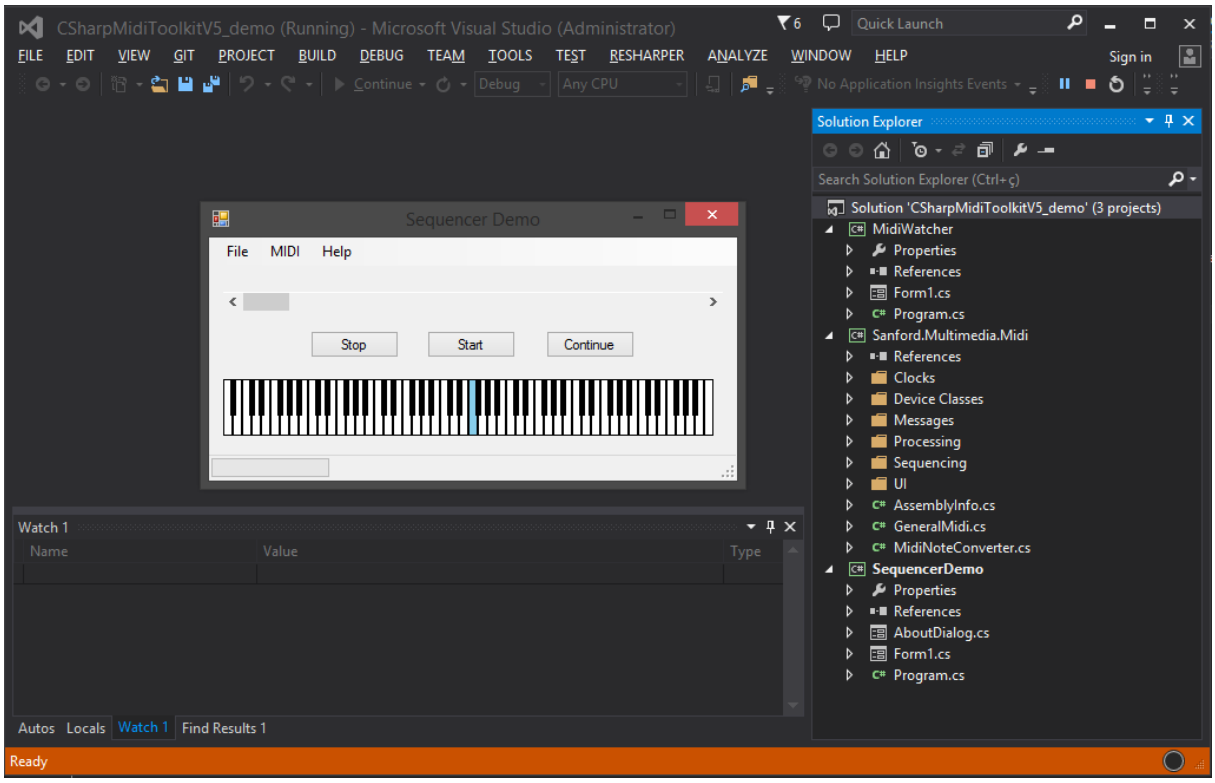
Figura 8 - O ambiente Visual Studio



2.4.3 Biblioteca de Sanford

Essa biblioteca foi desenvolvida por Leslie Sanford, que consiste em gerar o audio como se fosse um piano (Figura 9). O Conjunto de DLLs implementa toda a parte de controle de temporização e sintetização dos efeitos de áudio. Além disso a biblioteca consegue importar aquivos MIDI e executar o áudio.

Figura 9 - Demonstração da biblioteca de Sanford



2.5 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

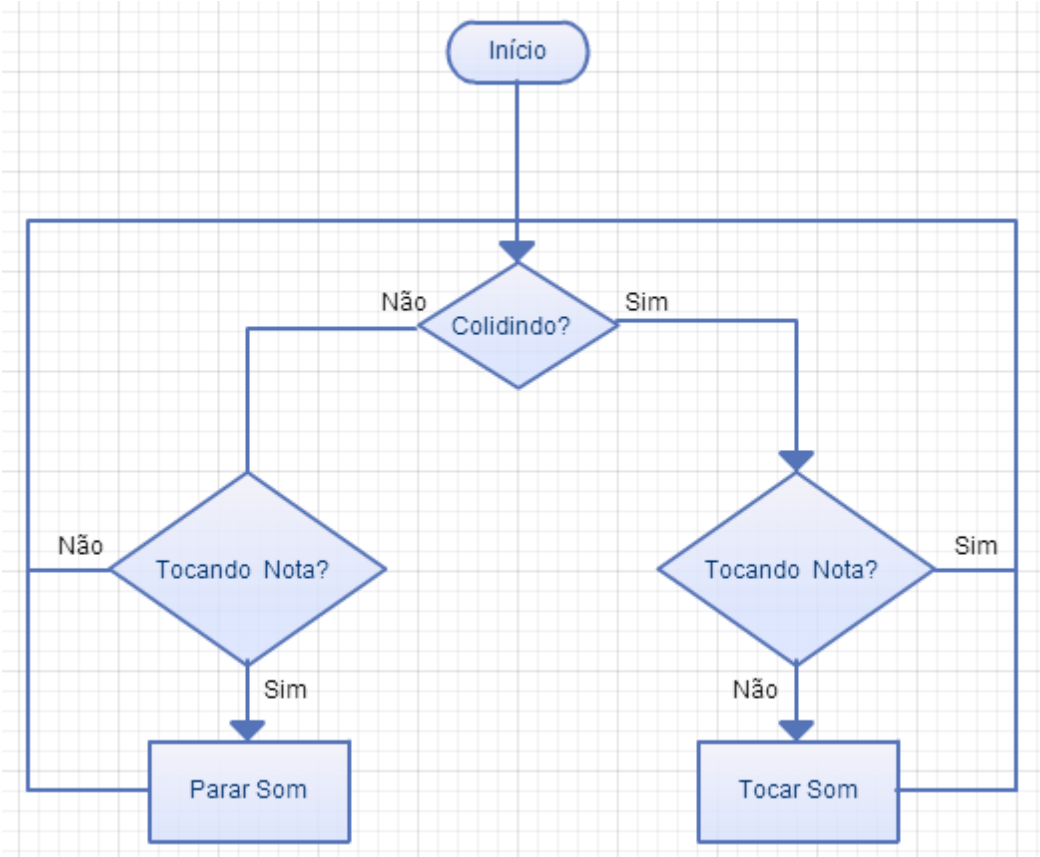
O sistema descrito nesta proposta deverá:

- exibir o modelo 3D de um piano (Requisito Funcional - RF);
- exibir o modelo 3D das mãos do usuário (RF);
- executar uma nota quando uma das teclas virtuais for pressionada (RF);
- permitir a execução de mais de uma nota ao mesmo tempo (RF);
- alterar a intensidade do som dependendo da velocidade do movimento (RF);
- ser desenvolvido para o sistema operacional Windows (Requisito Não-Funcional - RNF);
- ser desenvolvido sobre o ambiente Unity3D (RNF);
- utilizar o dispositivo Leap Motion (RNF);

2.6 ESPECIFICAÇÃO

O projeto consiste em reconhecer o toque nas teclas virtuais e reproduzir seus sons. Para tal será utilizado as colisões entre a mão virtual gerada pelo Toolkit do LeapMotion e as teclas do teclado virtua. Temos o seguinte fluxo para os scripts das teclas (Figura 10).

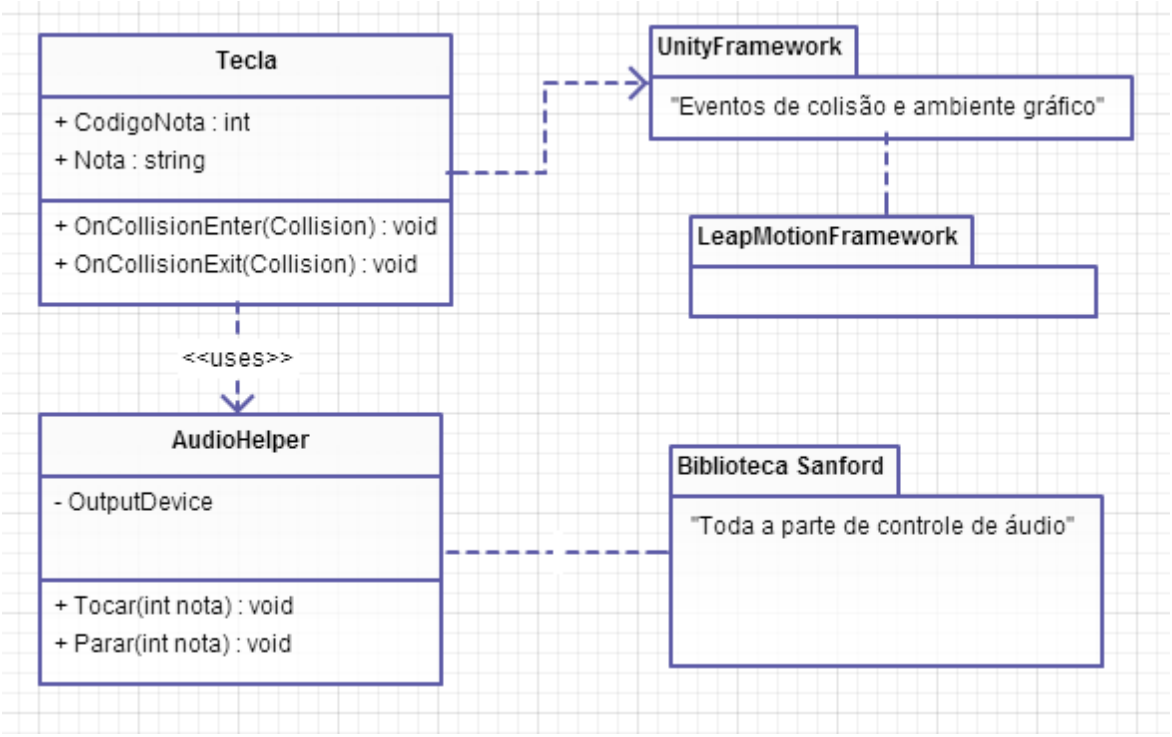
Figura 10 - Fluxo dos scripts das teclas



A maior dificuldade se encontra nas imprecisões do LeapMotion e da sensibilidade do reconhecimento de colisão. Em nossos testes vemos que ficou muito fácil de tocar as teclas erradas, pois o menor contato já é determinado como colisão. Para resolver essa situação optamos por criar a caixa de colisão menor do que as teclas, assim colisões acidentais, ou apenas na borda da tecla não irá resultar num falso positivo de tecla pressionada.

A programação base do projeto consiste apenas em reproduzir o audio quando a tecla pressionada e parar quando a tecla for solta. Como todo o processo de reprodução é igual, e deve ser inclusive global, acabamos com o seguinte diagrama de classes (Figura 11).

Figura 11 - Diagrama de classes



A maior parte das operações complexas são realizadas pelas bibliotecas ou pela própria engine, cabendo a nós apenas a lógica, ou tratar casos específicos como a sensibilidade da colisão.

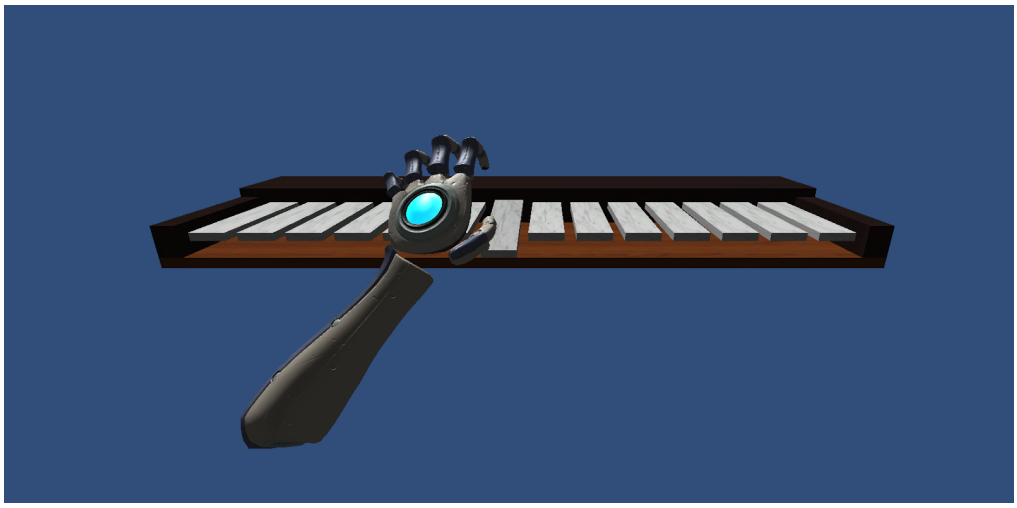
Etapa 03 (N4)

A seguir serão descritos conceitos relacionados com a implementação do projeto de pesquisa proposto.

3.1 IMPLEMENTAÇÃO

Para o desenvolvimento da aplicação (Figura 12) foi utilizado o SDK para desenvolvimento utilizando o linguagem C# dentro do Unity 3D fornecido pela Leap Motion, Inc. Este SDK contém vários objetos prontos para serem incluídos em um projeto do Unity 3D. Dentre os objetos fornecidos estão: modelos de mão, modelos de ferramentas e componentes de física para utilizar com tratamento de colisão. Além dos modelos o SDK traz uma API que permite acesso aos dados diretos do controlador no formato descrito no item 2.3.2.

Figura 12 - Leap Música



Como a interação do usuário mantendo o teclado fixo na cena ficaria prejudicada, optou-se por fazer o teclado seguir as mãos do usuário. Para isso, quando as duas mãos estão presentes na cena, elas precisam estar praticamente na mesma altura. O teclado será posicionado entre as mãos do usuário há uma altura que permita pressionar as teclas apenas movimentando os dedos (Quadro 2).

Quadro 2 - Método que calcula a posição entre as mãos

```
Vector3 CalculateVectorBetweenHands()
{
        var leftHandPosition =
currentFrame.Hands.Leftmost.PalmPosition.ToUnityScaled();
        var rightHandPosition =
```

```
currentFrame.Hands.Rightmost.PalmPosition.ToUnityScaled();
        var vectorBetweenHands = Vector3.Lerp(leftHandPosition, rightHandPosition,
0.5f);
        vectorBetweenHands.x = 0;
        vectorBetweenHands.y -= HandKeyboardYDistance;
        vectorBetweenHands.z += HandKeyboardZDistance;

        return vectorBetweenHands;
}
```

O cálculo da posição do teclado é feito através de interpolação linear pelo método Vector3.Lerp() disponibilizado na API do Unity 3D. Este método recebe o vetor de origem, o vetor de destino e o tempo da interpolação. Após o cálculo da posição entre as mãos, são aplicadas as distâncias de margem *HandKeyboardYDistance*, *HandKeyboardZDistance*, nos eixos "y" e "z" respectivamente.

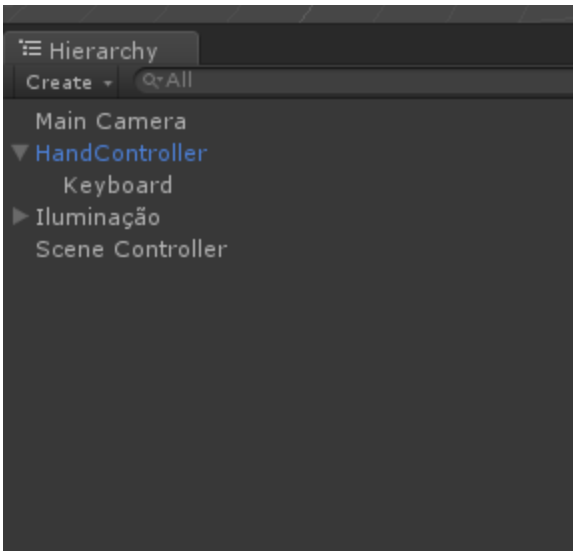
Quando há apenas uma mão na cena, para melhorar a interação, ao invés de esconder o teclado, ele é mantido visível. Porém, o teclado apenas acompanha a mão nos eixos "y" e "z", ficando fixo na posição "x" permitindo que o usuário passe por todas as teclas (Quadro 3).

Quadro 3 - Método que calcula a posição do teclado para apenas uma mão.

```
Vector3 CalculateVectorForHand()
{
        var handPosition = currentFrame.Hands[0].PalmPosition.ToUnityScaled();
        handPosition.x = gameObject.transform.localPosition.x;
        handPosition.y -= HandKeyboardYDistance;
        handPosition.z += HandKeyboardZDistance;
        return handPosition;
}
```

Para que a posição do teclado alterasse de acordo com a movimentação das mãos é preciso utilizar a técnica de Coordenadas de Dispositivos Normalizados (NDC) que a API do Leap Motion implementa no método *ToUnityScaled()*. Além de aplicar a técnica NDC também é preciso manter o objeto que deseja movimentar como filho do *HandController* na hierarquia de objetos do Unity 3D (Figura 13).

Figura 13 - HIerarquia de objetos no projeto do Unity 3D.



Como o ângulo de visão do Leap Motion diminui conforme a mão está mais próxima do sensor na altura, foi implementado um método para aumentar e diminuir o comprimento do teclado de acordo com a posição da mão do usuário (Quadro 4). Neste método também foi utilizada a técnica de interpolação linear, que interpola um ponto entre os valores 0.02 (KeyboardMinZ) e 0.05 (KeyboardMaxZ).

Quadro 4 - Calculo do comprimento do teclado de acordo com a mão do usuário

```
void UpdateScale()
{
        float iterationPoint = currentFrame.Hands[0].PalmPosition.ToUnityScaled().y / HandMaxY;

        float zScale = KeyboardMinZ + (KeyboardMaxZ - KeyboardMinZ) * iterationPoint;

        Vector3 scale = new Vector3 (gameObject.transform.localScale.x, gameObject.transform.localScale.y, zScale);

        gameObject.transform.localScale = scale;
}
```

Para tocar as notas ao pressionar as teclas foi utilizado o protocolo MIDI através da biblioteca Sanford. A biblioteca fornece uma API simples para enviar mensagens no protocolo MIDI (Quadro 5).

Quadro 5 - Enviando uma mensagem para tocar uma nota na biblioteca Sanford

```
OutputDevice device = new OutputDevice(0);
```

```
var message = new ChannelMessage(ChannelCommand.NoteOn, 0, note, 127);
device.Send(message);
```

Inicialmente é necessário um dispositivo que irá receber a mensagem, neste caso a variável *device*. Posteriormente é necessário montar uma mensagem através da classe *ChannelMessage*. O construtor recebe o tipo da nota *ChannelCommand.NoteOn* para ativar a nota e *ChannelCommand.NoteOff* para desativar a nota. Também é necessário informar a nota desejada através da variável *note* e tmabém o volume para tocar a nota. Como é um *device* para todo o teclado foi criado uma propriedade estática para armazenamento deste, compartilhada entre todas as teclas.

Como foram implementadas apenas 14 teclas do piano, na Tabela x1 são apresentados os valores para tocar as notas de cada tecla.

Tabela 1 - Valores necessários para tocar determinadas notas musicais

Nota	Valor da variável *note*
Dó	48
Ré	50
Mi	52
Fá	53
Sol	55
Lá	57
Si	59
Dó	60
Ré	62
Mi	64
Fá	65
Sol	57
Lá	59
Si	61

O teclado foi feito de maneira simples, um cubo como base em textura de madeira

clara, as laterais e parte traseira com uma textura de mogno e as teclas com uma textura de mármore. As teclas foram feitas com uma força de torque inversa a força aplicada pelos dedos para que quando sejam soltas voltem para a posição inicial, dando uma fluidez ao teclado.

3.2 RESULTADOS E DISCUSSÃO

Os resultados obtidos foram positivos, embora sejam necessárias extensões ao trabalho para que se chegue mais próximo a um piano físico. Para efeitos comparativos, segue a tabela abaixo (Tabela 2).

Tabela 2 - Comparação entre Leap Música e um piano físico

	LeapMúsica	Piano Físico
Preço (R$)	400 (LeapMotion)	2.000
Comprimento (cm)	35	132
Peso (kg)	2	12
Número de teclas	14	88
Resposta tátil	Não	Sim

Para as comparações, considerou-se que a pessoa já possui um computador e deseja adquirir um instrumento musical. Uma das possibilidades seria a compra de um LeapMotion e o uso do software desenvolvido neste trabalho, e a outra, a compra de um piano digital. Como base da comparação, foi escolhido o modelo P-35 da marca Yamaha, por ser o piano de 88 teclas mais barato do mercado.

Através do quadro, fica visível que o LeapMúsica apresenta vantagens nos quesitos preço e portabilidade, sendo significativamente mais barato, leve e menor que o piano. Por outro lado, o piano possui um número maior de teclas e resposta tátil ao toque nas teclas.

3.3 CONCLUSÕES

Neste trabalho, estudou-se a possibilidade de utilizar o Leap Motion como forma de interagir com um piano digital. Os resultados iniciais mostraram-se positivos, embora extensões sejam necessárias até que se chegue à simulação de um piano completo.

Entre os pontos positivos da solução desenvolvida estão a facilidade de uso e o baixo custo em comparação a um piano real. Há também a vantagem de um notebook com Leap Motion poder ser transportado muito mais facilmente que um piano ou mesmo um piano digital.

Por outro lado, a solução apresentada não possui o mesmo número de teclas que um piano e não apresenta uma resposta tátil como um instrumento de verdade. Embora o primeiro problema possa ser solucionado com as extensões sugeridas abaixo, o segundo exigirá pesquisa mais aprofundada com outros tipos de dispositivos.

3.4 EXTENSÕES

Algumas sugestões de extensões deste trabalho são listadas a seguir:

a. gravar os movimentos durante a execução de uma música para que depois se possa exibir que teclas devem ser pressionadas;
b. exportar o áudio de uma música para que outras pessoas possam escutar;
c. aumentar o tamanho do teclado, permitindo mais teclas;
d. permitir a escolha de sons de outros instrumentos, como cravo, órgão e sintetizadores;
e. fornecer resposta tátil para o usuário através de dispositivos como luvas (data gloves).

REFERÊNCIAS BIBLIOGRÁFICAS

ALVARENGA, Gustavo Garcia. **Ferramenta para criação de composições musicais para Android**. 2013. 59 f, il. Trabalho de Conclusão de Curso (Graduação em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2013. Disponível em: <http://www.bc.furb.br/docs/MO/2013/353880_1_1.pdf>. Acesso em: 20 out. 2014.

APPLE, Inc. **GarageBand for Mac**: Incredible music. In the key of easy. [S.l.], [2014]. Disponível em <https://www.apple.com/mac/garageband/>. Acesso em 20 out. 2014.

BEATON, Bobby; HARRISON, Steve; TATAR, Deborah. Digital drumming: a study of co-located, highly coordinated, dyadic collaboration. In: SIGCHI Conference on Human Factors in Computing Systems, 10., 2010, Nova Iorque. **Proceedings**... Proceedings of SIGCHI Conference on Human Factors in Computing Systems, 2010, p. 1417-1426. Disponível em: <http://dmrussell.net/CHI2010/docs/p1417.pdf>. Acesso em: 20 out. 2014.

COLGAN, Alex. **How Does the Leap Motion Controller Work?**. [San Francisco], 2014. Disponível em: <http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/>. Acesso em: 20 out. 2014.

COLLYER, Charles E.; BOATRIGHT-HOROWITZ, Seth S.; HOOPER, Sari. A motor timing experiment implemented using a musical instrument digital interface (MIDI) approach. **Behavior Research Methods, Instruments, & Computers**, [S.l.], v. 29, n. 3, p. 346-352. 1997.

DAVIS, Alan. **Getting Started with the Leap Motion SDK**. [San Francisco], 2014. Disponível em: <http://blog.leapmotion.com/getting-started-leap-motion-sdk/>. Acesso em: 20 out. 2014.

DUARTE, Alex M. **Aplicativos musicais para tablets e smartphones**: Novos recursos para a educação musical. 2014. 58 f, il. Trabalho de Conclusão de Curso (Licenciatura em música) - Instituto de Artes, Universidade de Brasília, Brasília, 2014. Disponível em: <http://bdm.unb.br/bitstream/10483/7951/1/2014_AlexMarquesDuarte.pdf>. Acesso em: 20 out. 2014.

INNOVATIONS, Jazzy. **DRUMair**: Release your inner rockstar. [S.l.], [2013]. Disponível em <https://apps.leapmotion.com/apps/drumair--2/windows>. Acesso em 20 out. 2014.

MENEZES JÚNIOR, Carlos R. F. de. **Editor MIDI para violão com articulação humanizada nota a nota e qualidade acústica em linguagem funcional pura**. 2007. 133 f, il. Dissertação (Mestrado em Engenharia Elétrica) - Curso de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Uberlândia, Uberlândia, 2007. Disponível em: <http://repositorio.ufu.br/handle/123456789/338>. Acesso em: 20 out. 2014.

NOWIKI, Michal et al. **Gesture recognition library for Leap Motion controller**. 2014. 65f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Instituto de ciência da computação, Universidade de Tecnologia de Poznan, Poznan.

PARANÁ. Stella Maris Oliveira Ludwig. Secretaria Estadual da Educação. Compreendendo a Música. In: PARANÁ. Cristiana Gonzaga. Secretaria Estadual da Educação (Ed.). **Dia-a-dia Educação:** Portal Educacional do Estado do Paraná. Curitiba: Secretaria da Educação do Paraná, 2014. Disponível em: <http://www.arte.seed.pr.gov.br/modules/conteudo/conteudo.php?conteudo=136>. Acesso em: 22 out. 2014.

ROADS, Curtis. **The computer music tutorial**. Cambridge : The MIT Press, c1996. xx, 1234 p, il.

ROTWEIN, Cody. **Bongos!**: Feel as you're playing real bongos!. [S.l.], [2013]. Disponível em <https://apps.leapmotion.com/apps/bongos/windows>. Acesso em 20 out. 2014.

WEICHERT, Frank et al. **Analysis of the Accuracy and Robustness of the Leap Motion Controller**. Sensors, Basel, v. 13, n. 5, 2013. Disponível em: <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3690061/>. Acceso em: 20 out. 2014.

FICHA DE AVALIAÇÃO

ASPECTOS AVALIADOS	Peso	Nota		
Avaliação N2	Etapa 1	O **problema** está claramente formulado?		
		O **objetivo** está claramente definido e é passível de ser alcançado?		
		Apresenta um grau de **relevância** em computação que justifique o desenvolvimento?		
	Etapa 2	O **cenário** descrito demonstra aonde o trabalho está inserido?		
		São apresentados 3 **trabalhos correlatos**?		
		As principais **características entre o trabalho** proposto e os correlatos foram descritas?		
		Conceitos relacionados a Sistemas Multimídia são suficientes e têm relação com o tema proposto?		
		O **ambiente** a ser utilizado foi detalhado?		
		Os **requisitos** a serem desenvolvido foram claramente descritos?		
		É apresentado o **diagrama/fluxograma** e este é coerente com o trabalho proposto?		
Avaliação N4	Etapa 3	São descritos os principais trechos de códigos utilizados na **implementação**?		
		São apresentados os **resultados** dos testes/avaliações fazendo comentários sobre os mesmos?		
		As **conclusões/extensões** estão descritas?		
	Geral	As **referências bibliográficas** são apresentadas e obedecem às normas da ABNT?		
		As informações retiradas de outros autores estão devidamente **citadas** no texto?		
		A **exposição do assunto** é ordenada (organização, apresentação gráfica, idéias e linguagem)?		
C