Revista Engenho, vol.7 – Abril de 2013

DESENVOLVIMENTO DE PROTÓTIPO DE SOFTWARE PARA RECONHECIMENTO DE POSIÇÕES E GESTOS UTILIZANDO MICROSOFT KINECT

Helder Fausto Narcizo
Hot One Comércio de Confecções Ltda.
helder.narcizo@gmail.com.

Carlos Eduardo Câmara
Centro Universitário Padre Anchieta
dinhocamara@gmail.com.

RESUMO

Este artigo apresenta numa visão geral o funcionamento e as características de *hardware* e *software* do dispositivo *Microsoft Kinect*, a implementação de um sistema que através de aquisição de dados retornados deste dispositivo, reconhece posições e gestos a partir de modelos que devem ser pré-definidos por meio de um cadastro no sistema. São demonstradas também, algumas aplicações desse tipo de tecnologia e o nível de interatividade que ela pode trazer num campo diferente da sua proposta original, a saúde. O estudo teve como base uma pesquisa exploratória tendo como alvo o dispositivo e sua documentação oficial e uma pesquisa experimental feito ao longo do desenvolvimento do protótipo de software.

Palavras Chaves: *Microsoft Kinect*, Projeto Natal, *SDK Kinect*, interatividade, desenvolvimento de *software*.

ABSTRACT

This article presents an overview of the operation and characteristics of hardware and software of the Microsoft Kinect device, implementation of a system through acquisition of data returned this device recognizes gestures and positions based on models that must be pre-defined by a registration system. Are also shown, some applications of such technology and

the level of interactivity it can bring in a different field from your original proposal, health. The study was based on an exploratory targeting the device and its official documentation and experimental research done throughout the development of the software prototype.

Keywords: Microsoft Kinect, Project Natal, SDK Kinect, interactivity, software development.

1. INTRODUÇÃO

Este artigo tem por objetivo apresentar ferramentas e modelos de desenvolvimento de um *software* utilizando o *SDK (Software Development Kit)* da *Microsoft* para o *Kinect*, realizando testes para avaliar as suas limitações. Para isso, foi realizado um levantamento de material bibliográfico sobre a mecânica de funcionamento das leituras e obtenção de dados retornados pelo equipamento. Alguns estudos complementares foram realizados, como o estudo do ambiente de programação, linguagem de programação e teoremas matemáticos, para tornar possível a compreensão dos conceitos necessários à implementação de rotinas capazes de reconhecer gestos e posições através deste dispositivo (Figura 1).

A *Microsoft,* em 2009, deu início a um projeto chamado "*Project Natal*", cujo objetivo foi desenvolver um sensor capturador de movimentos de última geração, capaz de permitir aos jogadores/usuários interagir com jogos eletrônicos e outras aplicações, sem a necessidade de utilizar um controle ou intermediador, servindo como acessório ao console de videogame Xbox360, sensor denominado hoje como *Microsoft Kinect* (QUALLS, 2010).

Figura 1: *Microsoft Kinect* (Microsoft MSDN, 2012)

Esta disponibilidade tecnológica proporciona o suporte necessário para o desenvolvimento de aplicações voltadas à área de realidade virtual aumentada, onde objetos reais interagem com objetos virtuais, portadora de várias características apresentadas como inovadoras até então.

Pretendeu-se desenvolver um protótipo de *software* para o dispositivo *Microsoft Kinect*,que auxiliasse no estudo e na avaliação de um exercício motor a fim de reconhecer a atividade humana na forma de movimentos identificando posições e gestos das articulações, segundo os passos:

- Cadastro das posições;
- Cadastro dos gestos;
- Cadastro de exercícios;
- Reconhecimento do esqueleto;
- Reconhecimento das posições;
- Reconhecimento dos gestos;
- Reconhecimento dos exercícios.

Diversas atividades de interação com o usuário podem ser realizadas com o uso de tecnologias de reconhecimento de movimentos, possibilitando inúmeras utilidades, como por exemplo, na área da saúde.

Segundo UNICID (2012), a clínica de fisioterapia da própria universidade, já utiliza no tratamento de doenças como Acidente Vascular Cerebral (AVC), paralisias, e lesões musculoesqueléticas esse tipo de tecnologia, com resultados cada vez mais satisfatórios na melhora dos tratamentos de reabilitação dos pacientes utilizando jogos.

Em 2011, GÓMEZ publicou um artigo fundamentado na utilização de um sistema chamado eBaViR baseado no dispositivo do console Wii, denominado *Wii Balance Board*, com o objetivo de auxiliar pacientes com problemas de distúrbios motores, que apresentaram melhoras significativas no equilíbrio através de exercícios de motivação e adaptação.

AMARATI (2012) em um de seus projetos em Jundiaí, o projeto *Wii* Reabilitação, utiliza o console *Wii*, que através de jogos e desafios gerados, estimula a atividade cerebral, fortalece a musculatura, facilitando o indivíduo a recuperar seus movimentos e aumentar sua capacidade de concentração e equilíbrio.

Foi observado que o uso destas tecnologias e a utilização de jogos, não retornam informações sobre o desempenho ou as respostas dos movimentos realizados pelos usuários.

Talvez o comportamento durante a realização dos movimentos de interação seja correto para os jogos, mas não para avaliação da capacidade motora dos pacientes. Percebeu-se então a necessidade do desenvolvimento de um *software* mais específico que retorne informações técnicas com a finalidade de auxiliar profissionais que as avaliam. E o dispositivo *Kinect* facilita a interação, uma vez que ele não necessita de controle ou um intermediador, pensou-se numa forma de permitir esta avaliação automática entre o previsto e o realizado pelo usuário.

2. O DISPOSITIVO MICROSOFT KINECT

O dispositivo *Microsoft Kinect* dispõe de uma arquitetura bem projetada, sendo capaz de capturar e processar dados, a fim de realizar o reconhecimento de objetos e pessoas.

2.1. Arquitetura física

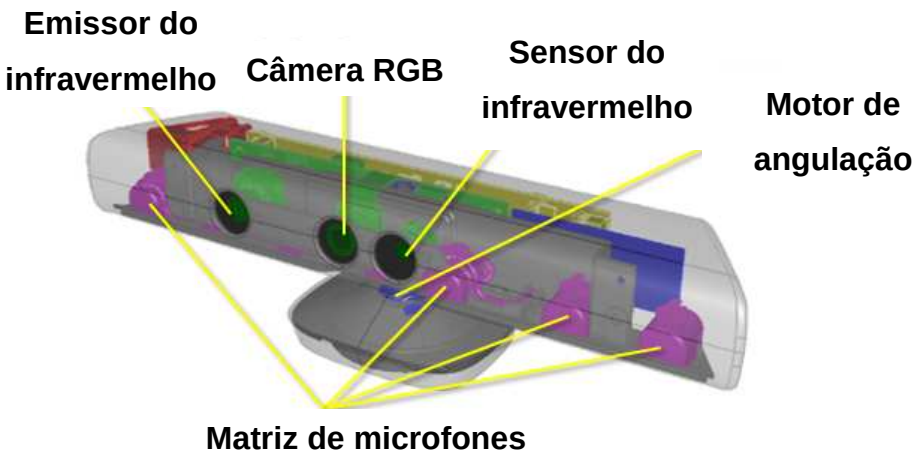

Figura 2: *Microsoft Kinect* por dentro (Microsoft MSDN, 2012)

Segundo a *Microsoft* (Microsoft MSDN, 2012), o dispositivo *Microsoft Kinect* contém, como na Figura 2:

- Uma câmera RGB que captura imagens coloridas em uma resolução de no máximo 1280x960.
- Um emissor de infravermelhos (IR) e um sensor de profundidade IR. O emissor emite feixes de luz infravermelha e o sensor de profundidade lê estes feixes de luz infravermelha que são refletidos de volta para o sensor. Os feixes refletidos são convertidos em informação de profundidade que realiza a medição da distância entre o objeto e o sensor, possibilitando a captura de uma imagem de profundidade dos objetos.

- Uma matriz de entrada de áudio, contendo quatro microfones para captura do som.
- Um motor de inclinação que suporta um alcance adicional de +\- 27 graus, o que aumenta consideravelmente o espaço para interação à frente do sensor, como representado na Figura 3.

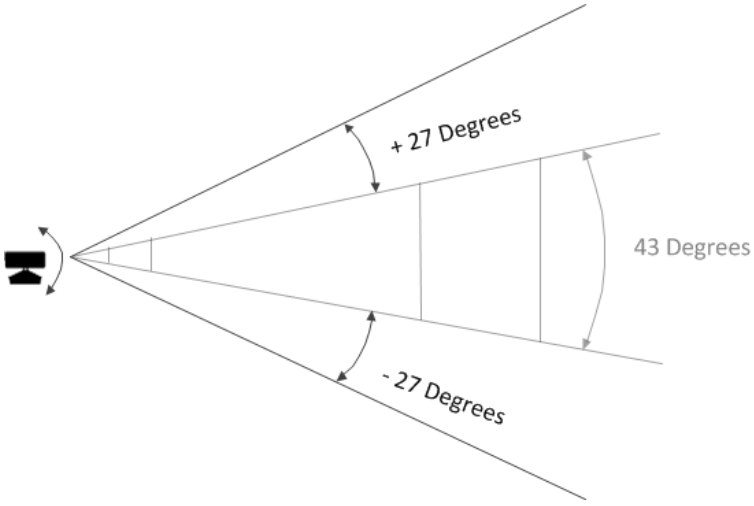

Figura 3: Inclinação do sensor (Microsoft MSDN, 2012)

2.2. Diferenças entre Kinect for Xbox 360 e Kinect for Windows

Segundo a *Microsoft*, o sensor *Kinect* para *Windows* foi totalmente testado e com suporte no *Windows* e com adicional do recurso "*Near Mode*" e melhoramentos ao rastreamento esquelético, API e conexão *USB*. O sensor para *Windows* foi concebido especificamente para ser utilizado com os computadores, e inclui um cabo *USB* encurtado. O *Kinect* para *Xbox 360* foi construído e testado apenas com o *Xbox 360*, e não com qualquer outra plataforma, e com isso não está licenciado para uso comercial, suporte, ou garantia, quando utilizado em qualquer outra plataforma.

A *Microsoft* tem uma grande equipe de engenheiros que se dedica a melhoria contínua do *hardware* e do *software* associado ao *Kinect* para *Windows*, e está empenhada em fornecer acesso contínuo ao grande investimento no rastreamento humano e reconhecimento de voz.

2.3. Funcionalidades do Microsoft Software Development Kit (SDK):

De acordo com a *Microsoft* (Microsoft MSDN, 2012), as funcionalidades suportadas pelo *SDK* são:

- Rastreamento esquelético.
- Determinação da distância entre um objeto e o sensor usando dados de profundidade.
- Captação do áudio e a localização de sua fonte.
- Motor de reconhecimento de voz permitindo o uso de uma gramática programada.

2.4. Requisitos mínimos de sistema

Segundo a *Microsoft* (Microsoft MSDN, 2012), os requisitos necessários pelo sistema exigem uma configuração mínima de *hardware* e *software* para ser executado, pois o processamento de cada aplicação consome uma quantidade de recursos. Não diferente, o *SDK* fornecido pela Microsoft também exige recursos mínimos de *hardware* e *software* para ser executado:

Sistemas operacionais

- *Windows 7 / Windows 8 (x86/x64)*

Hardware

- Processador *Dual-core* de 2,66 GHz ou mais rápido
- Barramento *USB* 2,0 dedicado ao *Kinect*
- 2 GB de *RAM*
- Placa de vídeo compatível com *DirectX 9.0c*
- Sensor *Microsoft Kinect*

Software

- *Microsoft Visual Studio 2010 Express* ou outra edição do *Visual Studio* 2010 (apenas para o desenvolvimento da aplicação)
- *.NET Framework 4*

2.5. Arquitetura lógica

Para entender melhor o funcionamento do dispositivo *Kinect*, a Figura 4 contém uma ilustração da arquitetura lógica e um pouco das características de seus componentes.

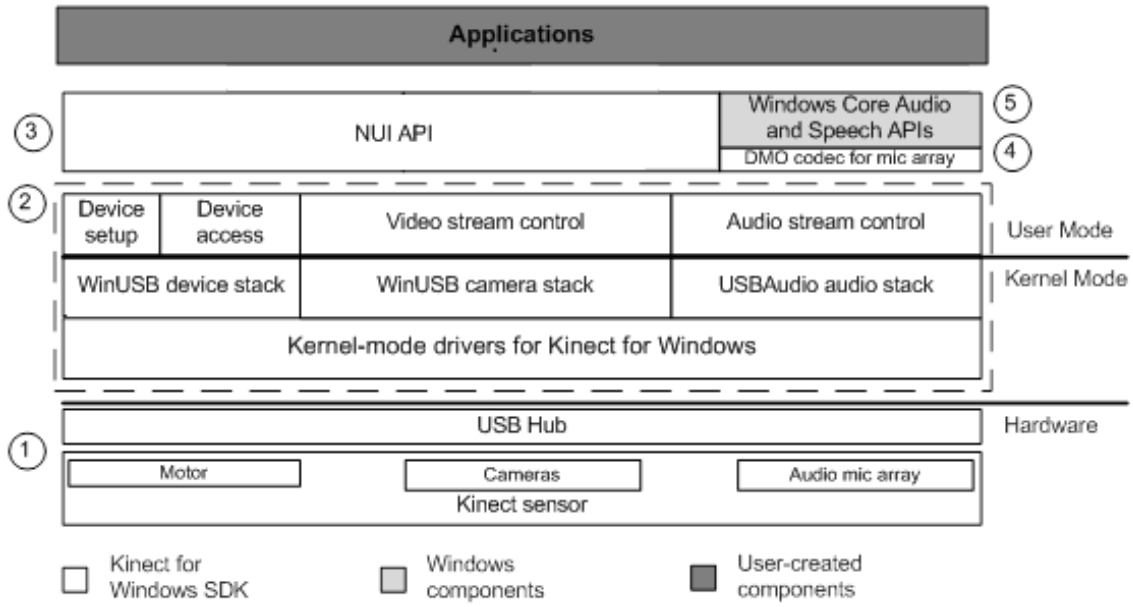

Figura 4: Arquitetura entre *Hardware* e *Software* (Microsoft MSDN, 2012)

De acordo com a *Microsoft* (Microsoft MSDN, 2012), estes componentes incluem:

1. *Hardware Kinect* - os componentes de *hardware*, incluindo o *Kinect* e o conector *USB* no qual o sensor está ligado ao computador.
2. *Drivers* do *Kinect* - são instalados no processo de instalação do *SDK*.
   - O microfone do *Kinect* permanece como um dispositivo de áudio em modo *kernel* que pode ser acessado por meio de *APIs* de áudio do *Windows;*
   - Controles de *streamings* de áudio e vídeo (profundidade, vídeo de cor e do esqueleto);
   - Funções de enumeração do dispositivo que permitem que um aplicativo use mais de um *Kinect*.
3. Componentes de Áudio e Vídeo - Rastreamento do esqueleto, áudio, imagens de vídeo e profundidade.
4. *DirectX Media Object (DMO)* - Faz a formação de feixe e da localização da fonte de áudio.
5. *APIs* do *Windows* - *SDK* do *Windows* e do *Microsoft Speech*.

3. FUNDAMENTOS TEÓRICOS

3.1. Base matemática

O reconhecimento de posições e gestos exige que sejam realizados cálculos trigonométricos com os dados de determinados pontos do corpo humano, capturados através

do dispositivo. Esses cálculos são necessários para obter resultados que comprovem se o usuário encontrava-se na posição desejada em um determinado momento, ou se o mesmo realizou um gesto correto ou próximo do esperado (ambos estabelecidos pelo sistema por meio de comparações com um cadastro realizado anteriormente).

3.2. Base técnica de desenvolvimento

Para alcançar o objetivo da proposta, o *software* desenvolvido necessitou do uso de diversos tipos de classes, métodos, eventos e propriedades, cada qual com sua importância e finalidade, disponibilizadas no *SDK* fornecido pela *Microsoft*. Para isso, foram aplicados conhecimentos teóricos e práticos adquiridos em MICROSOFT MSDN (2012), CHANNEL9 (2012) e I-PROGRAMMER (2012), onde os mesmos disponibilizam conceitos e exemplos. Pode ser citado o projeto KinectDTW (2011), onde uma ideia parcial foi aproveitada para o atual desenvolvimento. Tal projeto, de maneira simplificada, reconhece gestos previamente cadastrados comparando posteriormente com movimentos do usuário, porém não de modo detalhado.

Fornecido pela *Microsoft*, o gerenciador de execução e recursos do *Kinect* são responsáveis pela comunicação, recepção e processamento de dados enviados por tal, que por sua vez são mapeados pelo *SDK*, resultando em informação na forma de fluxo de dados, como cor (*ColorImageFrame*), profundidade (*DepthImageFrame*), áudio (*AudioSource*) (que não entra no escopo desse artigo) e rastreamento esquelético dos usuários (*SkeletonFrame*). O acesso a essas informações em ambiente de desenvolvimento torna a criação da aplicação muito mais simples, dando aos programadores apenas a responsabilidade de focar na lógica objeto do desenvolvimento.

A seguir serão mostrados alguns passos para capturar qualquer tipo de fluxo de dados transmitido pelo *Kinect*, esses exemplos foram desenvolvidos utilizando a linguagem C# com amparo da ferramenta Visual Studio.

- Declarar o *sensor Kinect*.

```
private KinectSensor sensor; // Declaração de uma instancia do sensor Kinect
```

- Capturar um sensor disponível, no caso fixou-se o primeiro (0).

```
if (KinectSensor.KinectSensors[0].Status == KinectStatus.Connected){
    sensor = KinectSensor.KinectSensors[0];
}
```

- Habilitar os fluxos a serem manipulados.

```
if (sensor != null)
{
  sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
  sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);
  sensor.SkeletonStream.Enable();
}
```

- Iniciar o dispositivo.

```
this.sensor.Start();
```

Fluxo de cor

O fluxo das imagens de cor está disponível em dois formatos de dados, *RGB* e *YUV*, onde no padrão *RGB* se utiliza 32 *bits* de dados e opera nas resoluções de 1280x960 em 30 FPS (Frames Por Segundo), 640x480 em 30 FPS ou 640x480 em 15 FPS. Já no padrão *YUV*, se utiliza 16 *bits* de dados operando na resolução de 640X480 em 15 FPS.

Abaixo segue o código para registrar o evento, disparado toda vez que uma imagem está pronta e sua implementação.

- Como habilitar o evento:

```
if (sensor != null)
{
    sensor.ColorFrameReady += sensor_ColorFrameReady;
}
```

- Como implementar:

```
private void sensor_ColorFrameReady(object sender, ColorImageFrameReadyEventArgs e)
{
   using (ColorImageFrame colorFrame = e.OpenColorImageFrame())
   {
     if (colorFrame != null)
     {
       byte[] colorPixels;
       colorPixels = new byte[sensor.ColorStream.FramePixelDataLength];
       colorFrame.CopyPixelDataTo(colorPixels);
       ...
     }
   }
 }
```

O método *OpenColorImageFrame*, disponível no *SDK*, acessa os dados da imagem retornados pelo sensor, que quando disponível, armazena-os na variável "*colorPixels*" na forma de *bytes*.

Fluxo de profundidade

O fluxo das imagens de profundidade está disponível em três resoluções: 640x480 (o padrão), 320x240 e 80x60, cada *pixel* contém a distância em milímetros entre o dispositivo e

o objeto, representados em 2 *bytes* que são divididos em dois tipos de informação: 13 *bits* representam os dados de profundidade e 3 *bits* representam os dados do usuário.

O código necessário para manipular esse fluxo é muito parecido com o do fluxo de cor, porém com algumas particularidades é apresentado a seguir:

- Como habilitar o evento:

```
if (sensor != null)
{
    sensor.DepthFrameReady += sensorDepthFrameReady;
}
```

- Como implementar:

```
private void sensorDepthFrameReady(object sender, DepthImageFrameReadyEventArgs e)
{
    using (DepthImageFrame imageFrame = e.OpenDepthImageFrame())
    {
        if (imageFrame != null)
        {
          short[] depthPixels;
          depthPixels = new short[sensor.DepthStream.FramePixelDataLength];
          imageFrame.CopyPixelDataTo(depthPixels);
          ...
        }
    }
}
```

O método *OpenDepthImageFrame*, disponível no *SDK*, acessa os dados da imagem que são retornados pelo sensor disponível na variável "*depthPixels*", onde são armazenadas as informações da imagem de profundidade, e utiliza o tipo de dados "*short*".

Rastreamento esquelético

O sensor com o *SDK* é capaz de reconhecer até seis usuários. Destes seis, até dois podem ser mapeados em detalhes, conhecendo as posições de suas articulações e acompanhando seus movimentos, conforme representado na Figura 5.

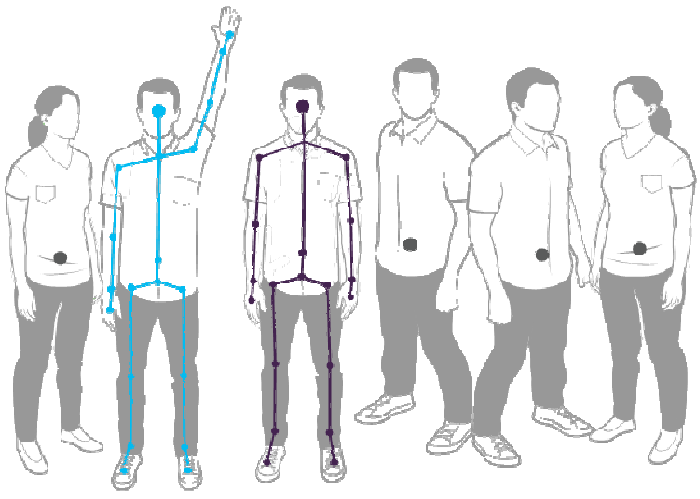

Figura 5: *Kinect* reconhece até 6 pessoas (Microsoft SDK, 2012)

Os demais usuários estabelecem um estado mapeado como “posição somente” contendo informações de sua posição, porém não mapeia suas articulações. Já um esqueleto mapeado em detalhes fornece a informação completa sobre a sua posição e também a posição das 20 articulações que o sensor é capaz de identificar. Na Figura 6, uma ilustração dos pontos e seus nomes de acesso.

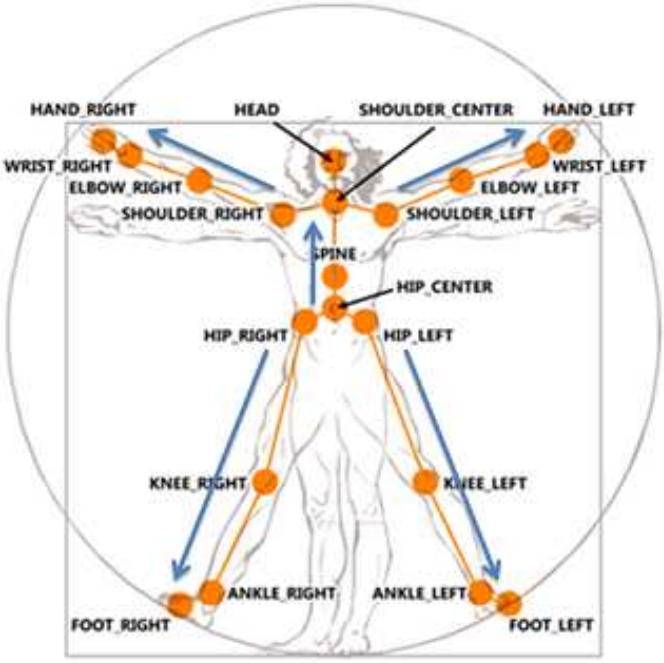

Figura 6: Mapa das articulações identificadas pelo *SDK* (Microsoft SDK, 2012)

O rastreamento esquelético trabalha em dois modos, o modo “padrão” que reconhece 20 articulações e o modo “sentado” que reconhece 10 articulações como demonstrado na figura 7.

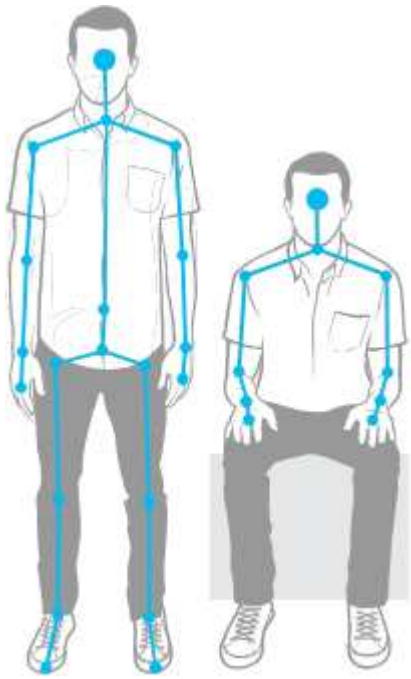

Figura 7: Modos de Rastreamento (Microsoft MSDN, 2012).

Cada articulação tem informações de coordenadas referentes à posição que está à frente do dispositivo, os eixos X e Y são números entre -1 e 1 e o eixo Z é à distância em metros em que a articulação está do dispositivo.

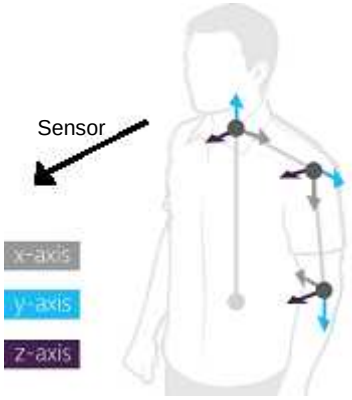

Figura 8: Representação de eixos de uma articulação (Microsoft MSDN, 2012).

O código necessário para manipular os dados do esqueleto é um pouco mais complexo devido trabalhar com diversas classes e dados. Por exemplo, as coordenadas das posições dos corpos e suas articulações.

Segue o código para registrar o evento e sua forma de implementação, para manipular esses tipos dados:

- Declaração global para armazenar todos os esqueletos capturados:

```
Skeleton [] skeletonData;
```

- Como habilitar o evento:

```
if(sensor != null)
{
    skeletonData = new Skeleton[sensor.SkeletonStream.FrameSkeletonArrayLength];
    sensor.SkeletonFrameReady +=
        new EventHandler<SkeletonFrameReadyEventArgs>(kinect_SkeletonFrameReady);
}
```

- Como implementar:

```
private void kinect_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
{
    using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
    {
        if (skeletonFrame != null && skeletonData != null)
        {
            skeletonFrame.CopySkeletonDataTo(skeletonData); // Capturando dos do esqueleto
            foreach (Skeleton skeleton in skeletonData){ // varrendo todos os esqueletos
                if (skeleton.TrackingState == SkeletonTrackingState.Tracked){
                    //Esqueleto reconhecido e mapeado Completamente
                    //Acessando Coordenadas Esqueleto
                    System.Console.WriteLine("Esqueleto X: {0} Y:{1} Z:{2}",
                        skeleton.Position.X, skeleton.Position.Y, skeleton.Position.Z);
                    //Acessando Coordenadas Articulacao
                    System.Console.WriteLine("Cabeça X: {0} Y:{1} Z:{2}",
                        skeleton.Joints[JointType.Head].Position.X,
                        skeleton.Joints[JointType.Head].Position.Y,
                        skeleton.Joints[JointType.Head].Position.Z);
                    ...
                } else if (skeleton.TrackingState == SkeletonTrackingState.PositionOnly){
                    System.Console.WriteLine("Esqueleto X: {0} Y:{1} Z:{2}",
                        skeleton.Position.X, skeleton.Position.Y, skeleton.Position.Z);
                    //Esqueleto reconhecido porem não mapeado Completamente
                    ...
                }
            }
        }
    }
}
```

O método *OpenSkeletonFrame* disponível no SDK, acessa os dados de todos esqueletos reconhecidos que são retornados pelo sensor disponível na variável "*skeletonData*", onde são armazenadas as informações do esqueleto tanto quanto de suas articulações.

4. DESENVOLVIMENTO

4.1. Ferramentas utilizadas

Para construção do aplicativo proposto no tema deste artigo, foi necessária a utilização de ferramentas que possibilitassem a extração, manipulação, armazenamento e apresentação dos dados fornecidos pelo dispositivo *Microsoft Kinect.*

Rotinas foram implementadas para a extração desses dados, para isto, foram usadas classes, métodos, eventos e propriedades presentes no *SDK (Kinect for Windows 1.5)* fornecido pela *Microsoft.*

Assim como a extração, a manipulação e apresentação desses dados ocorreram com o amparo da *IDE* do *Visual Studio* versão 2010, que proveu boas funcionalidades auxiliando a codificação em linguagem de programação C# (lê-se C *Sharp*) e *Framework .Net 4.0*.

Para utilização posterior, foi preciso realizar o armazenamento dos dados, aplicando os recursos da ferramenta *SQL Server* versão 2012.

4.2. Implementação

Depois de definidos os objetivos, tecnologia, ferramentas e requisitos, deu-se inicio à etapa de implementação, ou seja, etapa prática de desenvolvimento da proposta, sendo composta por “estratégia de armazenamento” (Banco de Dados) e “arquitetura de *software*” (*Software*).

Banco de Dados

As tabelas da base foram projetadas e criadas levando em consideração a integridade de seus dados juntamente com sua normalização, resultando em maior segurança e desempenho das rotinas que a consomem, tornando possível sua restauração.

A seguir na figura 9 é representado o diagrama de entidade relacionamento, que evidencia a estratégia utilizada de como os dados são divididos e armazenados:

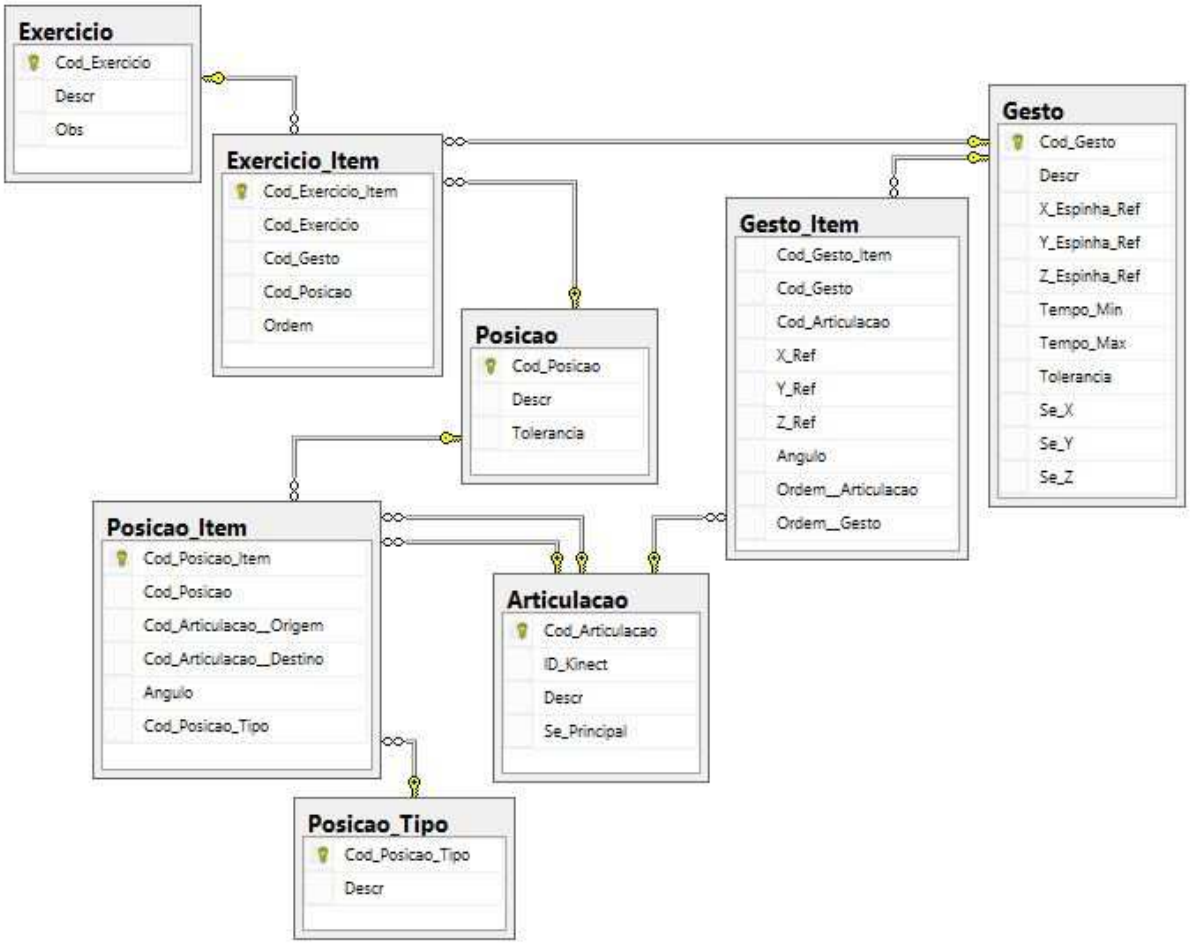

Figura 9: Diagrama de entidade relacionamento desenvolvido

Funções e finalidades de cada tabela presente no diagrama:

- **Articulacao** – representa dados das articulações que o *SDK* é capaz de reconhecer, tornando-os compatíveis para a aplicação com os retornados pelo sensor.
- **Posicao** – Guarda a identificação da posição.
- **Posicao_Item** – Armazena todos os itens da posição, ou seja, todas as articulações que uma posição possui.
- **Posicao_Tipo** – Possui informação que determina se o registro da tabela Posicao_Item é uma relação de intersecção ou angulação entre dois pontos(articulação).
- **Gesto** – Armazena a identificação do gesto e pontos de referências.
- **Gesto_Item** - Contém todos os itens do gesto, ou seja, todas as articulações e seus pontos de referencia que um gesto possui.
- **Exercicio** – Registra a identificação de um exercício.

- **Exercicio_Item** – Registra dados de todas as etapas para realização de um exercício, em forma de sequencia de posições, gestos ou os dois tipos intercalados.

Software

Com objetivo de obter maior organização e desempenho da aplicação, foram utilizados conceitos importantes e pertinentes de orientação a objetos, arquitetura de *software*, reutilização de componentes e padrões de nomenclaturas.

Na Figura 10, é demonstrado o mapa de navegação do sistema desenvolvido.

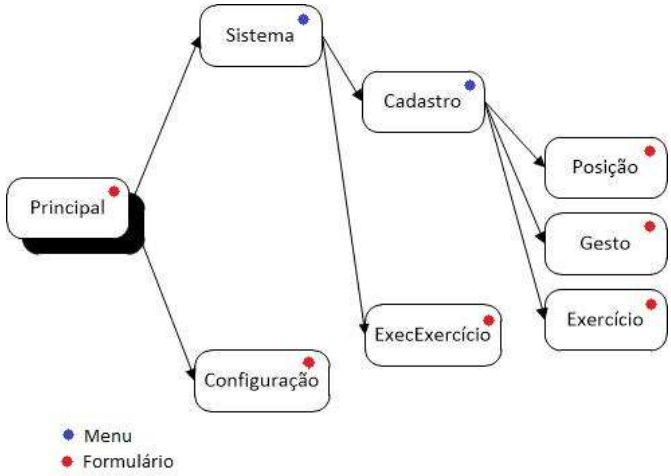

Figura 10: Mapa do sistema

Formulário Principal

Responsável por disponibilizar o acesso a todas as outras telas do sistema em forma de menu e controlar a exibição do *status* do sensor *Kinect* para o usuário (Figura 11).

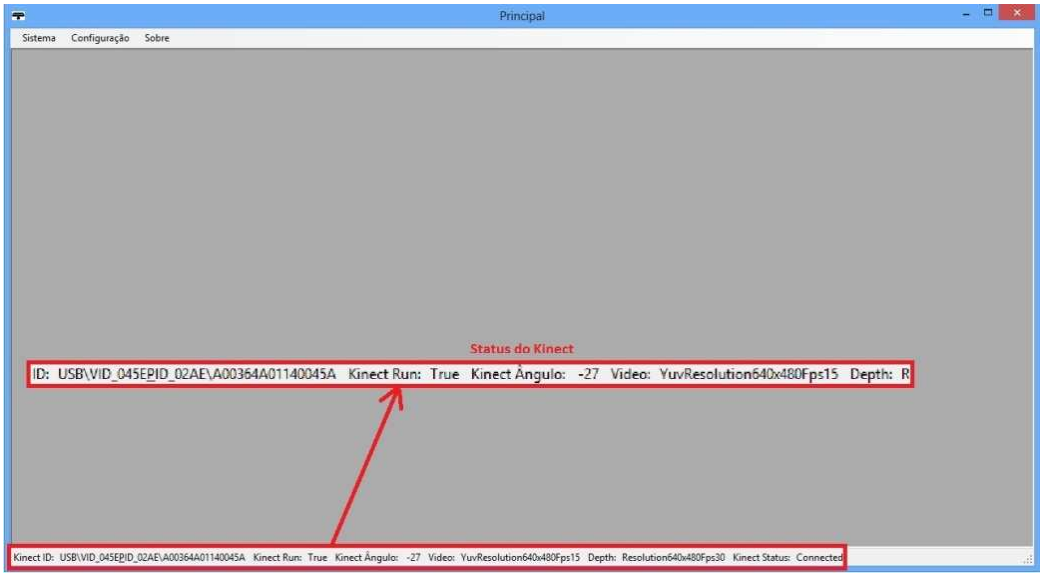

Figura 11: Formulário Principal

Formulário Posição

Possui funções de cadastro de posições a serem reconhecidas posteriormente (Figura 12).

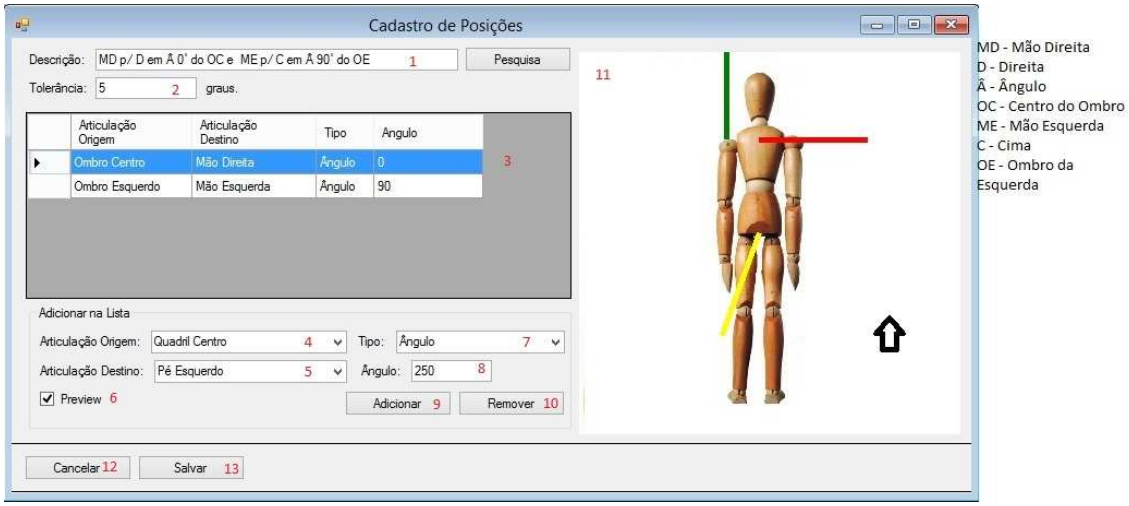

Figura 12: Formulário cadastro de uma posição

1. Descrição da posição;
2. Tolerância em graus quando selecionado o tipo “ângulo” e pontos para tipo “em cima”;
3. Lista das posições combinadas entre duas articulações que compõe a posição final;
4. Lista de articulações de origem ou ponto de referência para o ponto de destino;
5. Lista de articulações de destino;
6. Habilita a pré-visualização (item 11);
7. Recebe o valor “em cima” quando uma articulação está sobreposta à outra ou “ângulo” quando uma articulação destino precisa estar sobre um ângulo da articulação origem;
8. Quando selecionado “ângulo” no item 7, insere o ângulo;
9. Adiciona item a lista (item 3);
10. Remover item da lista (item 3);
11. Ilustração da posição que irá ser cadastrada com todos os itens que foram adicionados na lista, onde os riscos em vermelho são os itens selecionados da lista, verdes são os validados e amarelos são os da pré-visualização (item 6);
12. Cancela o cadastro da posição;
13. Salvar posição.

Formulário Gesto

Possui funções de cadastro de gestos a serem reconhecidas posteriormente (Figura 13).

Figura 13: Formulário de cadastro do gesto

1. Descrição do gesto;
2. Lista das articulações que conterá o gesto que irão ser mapeados;
3. Lista de articulações reconhecidas;
4. Adicionar articulação selecionada no (item 3) a lista (item 2);
5. Remover articulação do (item 2);
6. Quantidade de quadros que conterá o modelo padrão;
7. Tempo em segundos que o processo de captura irá utilizar para capturar todos os quadros (item 6);
8. Tolerância em graus;
9. Tempo mínimo para reconhecimento de um gesto;
10. Tempo máximo para reconhecimento de um gesto;
11. Dimensões que serão consideradas para que um gesto seja reconhecido;
12. Tempo restante para terminar o processo de captura dos quadros;
13. Quantidade de quadros capturados;
14. Ponto de referência para o modelo padrão;
15. Vídeo que demonstra as articulações a serem consideradas a para o reconhecimento do gesto que está em destaque com pontos de cor vermelha;
16. Cancela cadastro do gesto;
17. Inicia processo de captura;

18. Salva captura realizada.

Formulário Exercício

Possui funções de cadastro de exercícios a serem reconhecidas posteriormente (Figura 14).

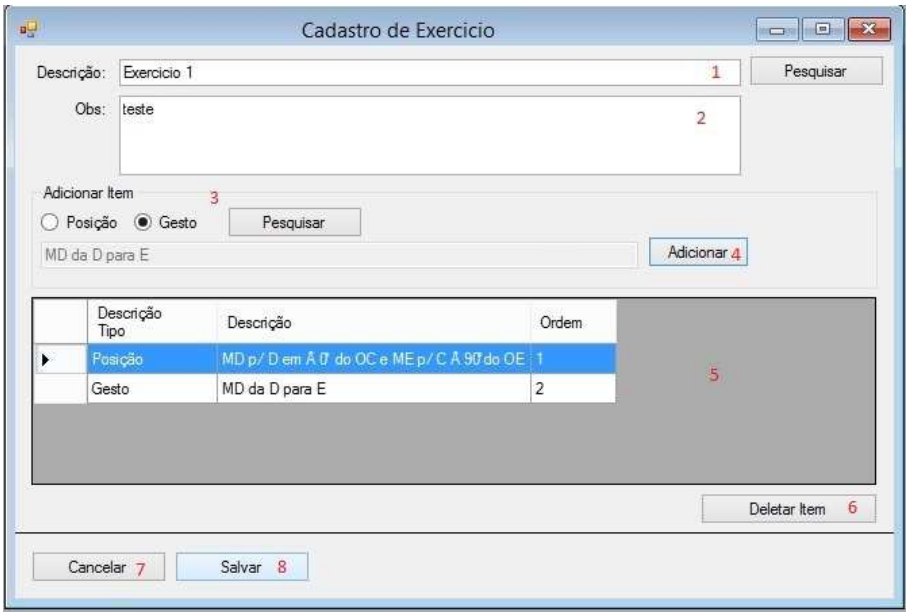

Figura 14: Formulário cadastro do exercício

1. Descrição do exercício;
2. Observação;
3. Seleção de uma etapa do exercício que poderá ser do tipo “posição” ou “gesto”;
4. Adicionar a lista de etapas do exercício (item 5);
5. Lista das etapas do exercício;
6. Apagar uma etapa da lista (item 5);
7. Cancelar o cadastro do exercício;
8. Salvar o exercício.

Formulário Execução de Exercício

Esse tipo de formulário possui funções para o reconhecimento das posições e gestos realizados pelo usuário comparando-os com as posições e gestos previamente cadastros pelo modelo padrão (Figura 15). É importante ressaltar que enquanto o usuário não finaliza uma etapa do exercício não passará para a próxima etapa.

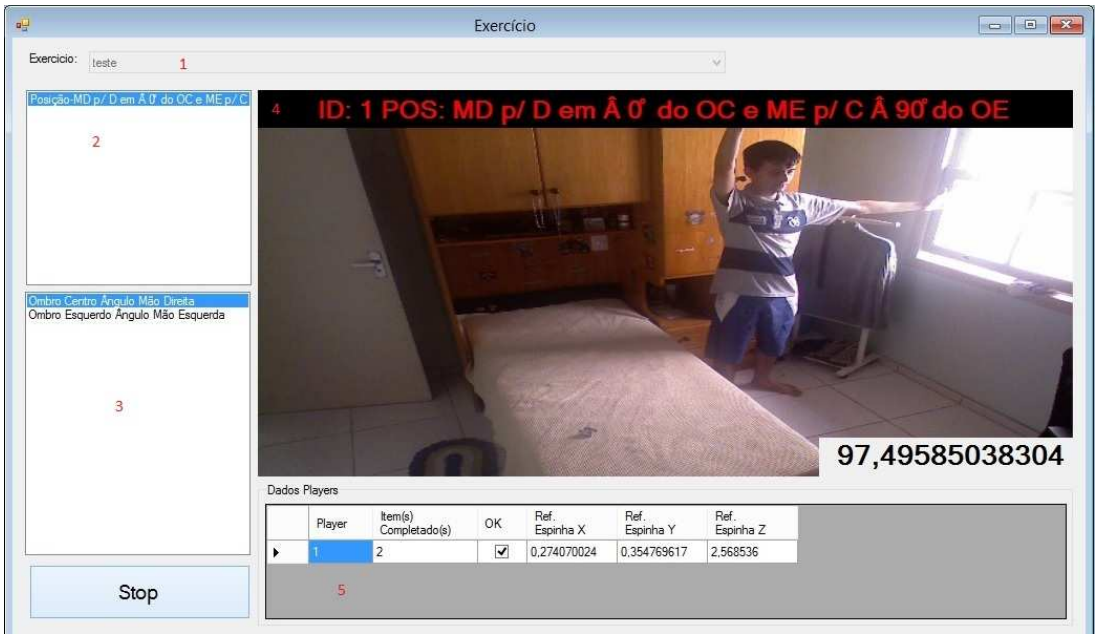

Figura 15: Formulário execução do exercício

1. Lista com os exercícios previamente cadastrados;
2. Lista com as etapas do exercício selecionado (item 1);
3. Lista com articulações envolvidas de uma etapa selecionada (item 2);
4. Tarja que exibe a identificação de uma posição ou gesto reconhecido a partir do modelo padrão;
5. Lista dos usuários identificados no ambiente.

Formulário Configuração

Responsável por configurar o dispositivo para uma boa execução durante o uso da aplicação (Figura 16).

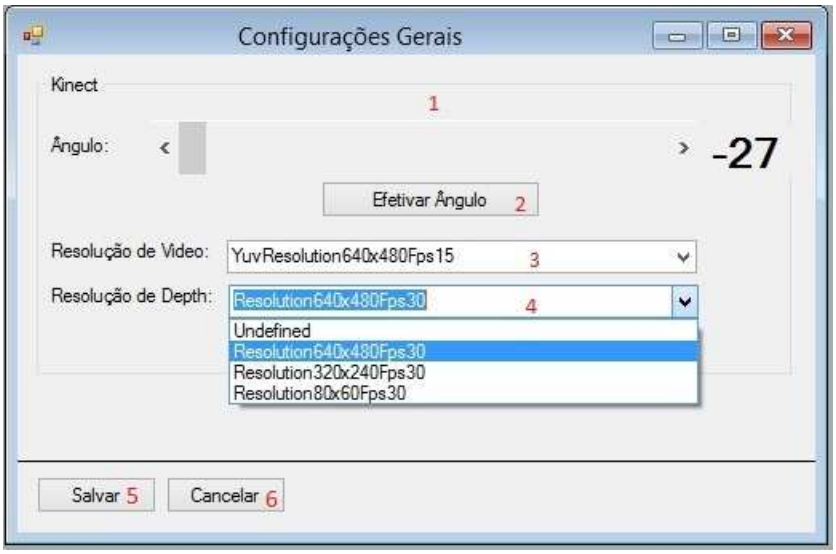

Figura 16: Formulário de Configuração

1. Configura o motor de inclinação no ângulo em que melhor aproveita o espaço para interação na frente do sensor;
2. Efetiva o ângulo escolhido em (item 1) passando o comando para o dispositivo;
3. Lista com as configurações de vídeo suportadas pelo dispositivo;
4. Lista com as configurações da resolução de resposta para o mapa das profundidades dos objetos;
5. Salva configurações selecionadas;
6. Cancela edição das configurações.

Geral

Para permitir a visualização das classes que compõem o sistema com seus respectivos métodos, propriedades e atributos é demonstrado na figura 17 o diagrama de classes.

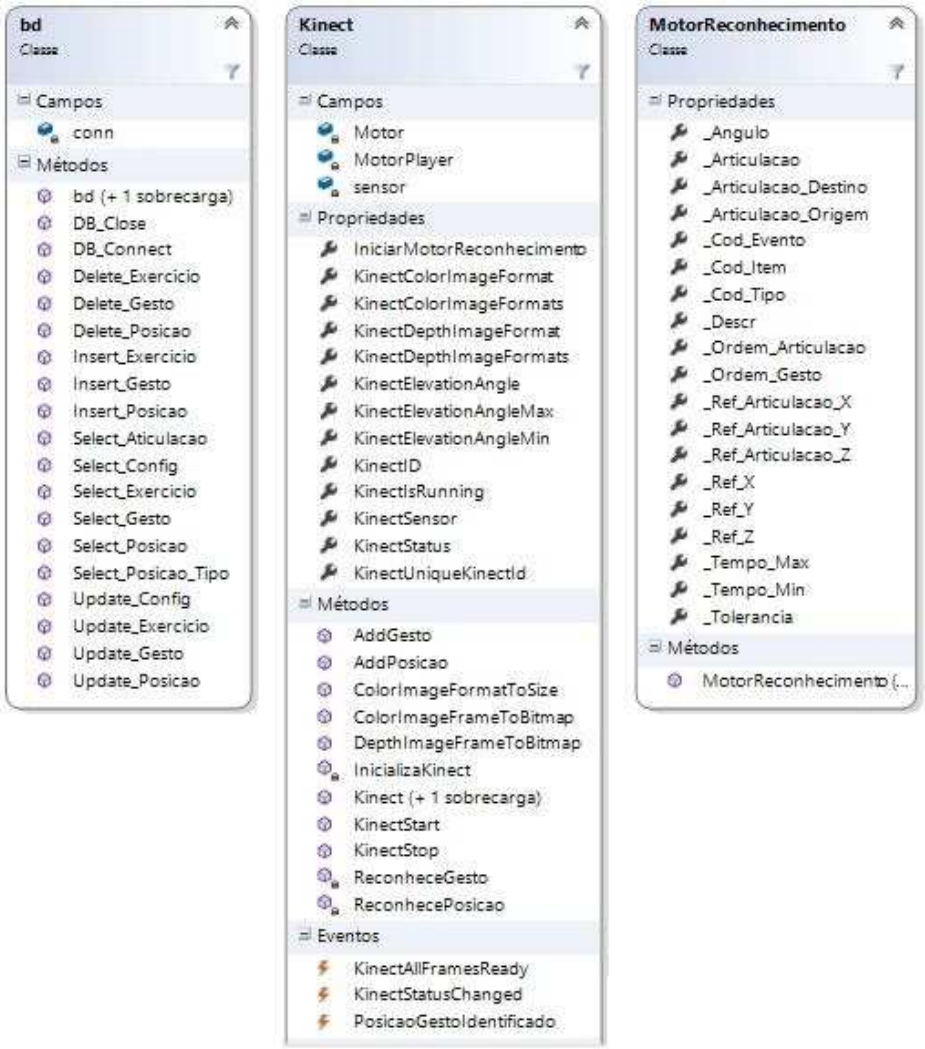

Figura 17: Diagrama de Classes

4.3. Teste

Testes foram desenvolvidos e aplicados singularmente durante o desenvolvimento do *software*.. Baseando-se nos resultados dos mesmos, correções eram realizadas, quando necessária, colaborando com a qualidade final do sistema.

Primeiramente, é realizado um cadastro de posições ou gestos descrevendo como deve postar-se diante do dispositivo referindo à descrição, tolerância, angulação e os demais itens do cadastro para as posições ou gestos desejados.

O segundo passo é a criação dos exercícios, onde são vinculados às posições e gestos cadastrados anteriormente, criando uma série de etapas a serem executadas pelo usuário.

Após as definições dos exercícios, são listados todos os itens da série que o usuário deverá executar. O sistema ficará aguardando o usuário realizar a etapa até o mesmo

conseguir realizar a posição ou gesto corretamente com a tolerância de erros estipulada anteriormente mediante cadastro.

A figura 18 apresenta um diagrama de Caso de Uso que mostra as funcionalidades do sistema do ponto de vista do usuário.

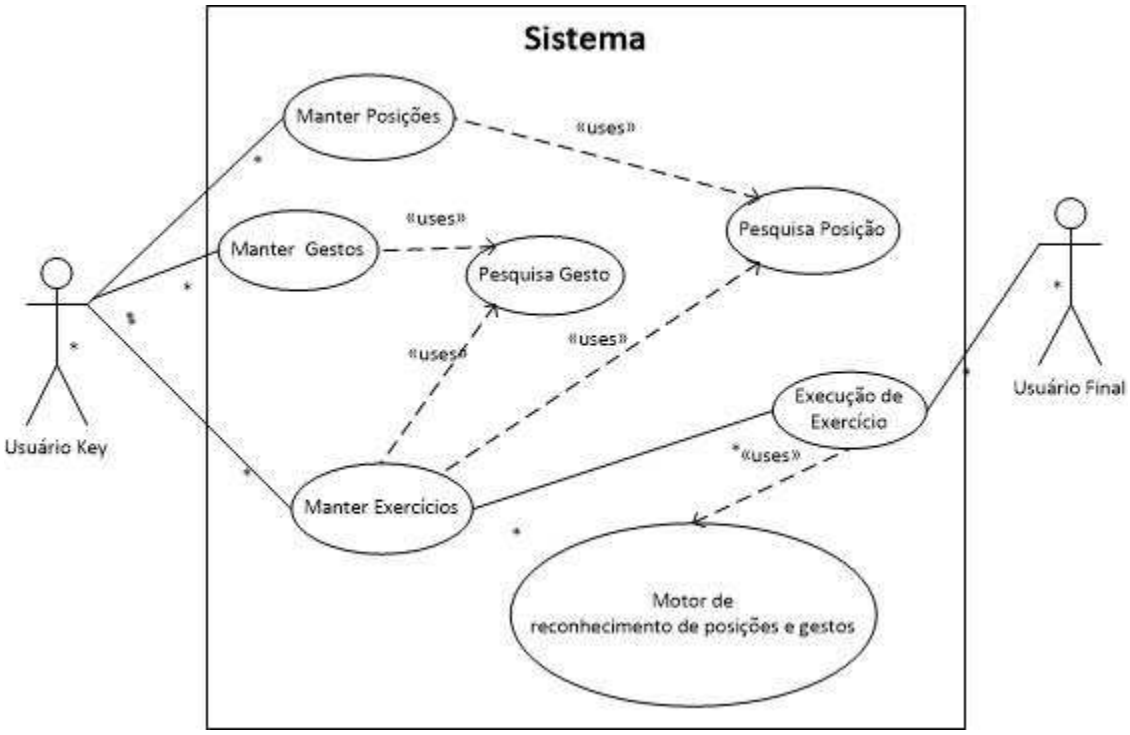

Figura 18: Diagrama de Caso de Uso

Os testes foram aplicados com quatro pessoas de faixas etárias e tamanhos diferentes, e o *software* conseguiu reconhecer permitindo a mesma sequência de exercício para todos.

O exercício realizado foi uma série contendo uma posição, um gesto e para finalizar outra posição, onde detalharei a seguir.

A primeira etapa, a posição consistia em:

a) A “mão direita” deveria estar sobre um ângulo de 0º do ponto de referência “ombro direito”;

b) A “mão esquerda” deveria estar sobre um ângulo de 90º do ponto de referência “ombro esquerdo”.

A segunda etapa, o gesto consistia em:

a) A “mão direita” passaria da direita para esquerda em no mínimo 1 segundo e no máximo 3 segundos.

A terceira etapa e ultima posição consistia em:

a) O “pé esquerdo” deveria estar sobre um ângulo de 250º do ponto de referência “quadril”.

5. CONCLUSÃO

O sistema de informação nos permite uma interação do homem com a máquina. É por meio do dispositivo *Microsoft Kinect* que a manipulação por meios específicos supre a necessidade dos processos de diversas áreas, que busca mapear, agilizar, e garantir a eficácia de diversos segmentos.

Atualmente, o *Kinect* é visto como entretenimento através de jogos, mas esse dispositivo é capaz de fornecer mais utilidades como é o caso do Hospital Evangélico de Londrina (TABORDA, 2012), que auxilia os profissionais de saúde no centro cirúrgico, através de comandos realizados pelo próprio profissional permitindo visualizar regiões com mais agilidade e precisão.

A proposta do *software* desenvolvido ao longo de todo o trabalho é a comunicação de diversas áreas com a área da tecnologia da informação. O estudo inicialmente foi focado em artigos da área, buscando informações sobre uso de consoles de games prontos e como eram utilizados em treinamentos. Assim, a ideia apresentada é uma ferramenta que permite montar ou elaborar uma serie de movimentos através de um cadastro e, como resposta avaliar se os momentos realizados pelo usuário são equivalentes e, principalmente, se são realizados dentro de uma margem de erro suportável. Dessa forma, surge a necessidade de buscar as limitações do *SDK* e estudar métodos e maneiras de programar estas avaliações, talvez de forma automática. Foi então que percebeu-se a existência desta possibilidade e trabalhou-se em um programa, desenvolvido neste *SDK* que pode trazer respostas interessantes e com certa precisão. Por isso, pensou-se em aplicações principalmente para profissionais da área de saúde. Para isso, foram necessárias pesquisas e conhecimentos técnicos a fim de manipular os dados coletados desenvolvendo um *software* capaz de apresentar informações corretas e precisas de forma a permitir uma avaliação mais segura dos resultados.

A aplicação desenvolvida é capaz de executar todas as operações correspondentes à proposta inicial - reconhecimento de posições e gestos utilizando *Microsoft Kinect*, porém não de forma extremamente precisa. Esse aprofundamento requer uma maior demanda de tempo em estudos e um conhecimento mais profundo na manipulação de certos dados, essencial para futuros trabalhos que pretendam evoluir e aperfeiçoar o projeto para darem continuidade à proposta.

O uso deste dispositivo facilita o reconhecimento de objetos em um ambiente permitindo que através de manipulação dos dados retornados reconheça variações no ambiente. Atualmente, existe uma deficiência desse tipo de *software* por tratar-se de uma tecnologia nova e o seu principal uso ser em jogos.

A utilização dessa forma de sistema facilitaria a precisão de profissionais da área da saúde permitindo o uso de uma ferramenta que complementa a aplicação com a automação do reconhecimento da qualidade e maior precisão dos movimentos e exercícios possibilitando a execução dos mesmos sem a presença de um profissional o tempo todo, mas sim a sua supervisão nas tarefas desenvolvidas previamente cadastradas pelo mesmo. O sistema permite que não seja possível a continuidade da sequencia sem que o movimento cadastrado seja realizado de forma correta. Para o profissional, o maior ganho com o uso do disposto é a divisão com o tempo nas sessões, podendo supervisionar mais de um paciente em um mesmo momento. Já para o paciente é a interatividade deixando que seus exercícios sejam repetitivos e monótonos, e claro, se o paciente possuir o mesmo dispositivo em casa, poderia realizar mais séries ao longo da semana reduzindo o tempo de reabilitação gerando maiores expectativas e resultados positivos em seu tratamento.

Portanto, foi compensador explorar essa tecnologia com pesquisas, métodos e técnicas a fim de aperfeiçoar processos em ambientes criativos e inovadores, possibilitando soluções de problemas e agregando valor ao mercado.

6. REFERÊNCIAS BIBLIOGRÁFICAS

AMARATI, Associação de Educação Terapêutica para Portadores de Lesões Neurológicos. Disponível em http://www.amarati.org.br/site_novo/, acesso em 04/11/2012.

CHANNEL9. Kinect for Windows Quickstart Series. Disponível em <http://channel9.msdn.com/Series/KinectQuickstart>, acesso em 02/02/2012.

GÓMEZ GIL, Journal of Neuroengineering and Rehabilitation - Effectiveness of a Wii balance board-based system (eBaViR) for balance rehabilitation: a pilot randomized clinical trial in patients with acquired brain injury (2011). Disponível em <http://www.jneuroengrehab.com/content/8/1/30>, acesso em 24/03/2012.

I-PROGRAMMER. Practical Windows Kinect In C#. Disponível em <http://www.i-programmer.info/ebooks/practical-windows-kinect-in-c/3738-introduction-to-kinect.html>, acesso em 04/11/2012.

KinectDTW, Project Kinect SDK Dynamic Time Warping (DTW) Gesture Recognition. Disponível em http://kinectdtw.codeplex.com/, acesso em 04/11/2012.

MICROSOFT MSDN, Biblioteca MSDN Kinect for Windows SDK. Disponível em <http://msdn.microsoft.com/en-us/library/hh855347>, acesso em 10/09/2012.

MICROSOFT, Kinect For Windows. Disponível em <http://www.microsoft.com/en-us/kinectforwindows/develop/resources.aspx>, acesso em 01/02/2012.

QUALLS Eric, About.com Guide, Microsoft project natal fact sheet. Disponível em http://xbox.about.com/od/news/a/natalfactsheet.htm, acesso em 10/09/2012.

TABORDA Cauã, da revista info, Hospital brasileiro usa Kinect para fazer cirurgias. Disponível em http://exame.abril.com.br/tecnologia/noticias/hospital-brasileiro-usa-kinect-para-fazer-cirurgias, acesso em 04/11/2012.

UNICID Universidade Cidade de São Paulo – Xbox Kinect é utilizado para reabilitação na fisioterapia. Disponível em <http://www.unicid.br/cgi/cgilua.exe/sys/start.htm?infoid=3183&query=simple&search_by_field=tax&sid=130&text=kinect>, acesso em 12/03/2012.