SAMUEL DAMAZIO DA COSTA

VITRINE INTERATIVA UTILIZANDO O MICROSOFT KINECT

UNIVERSIDADE DO VALE DO SAPUCAÍ

POUSO ALEGRE - MG

2014

SAMUEL DAMAZIO DA COSTA

VITRINE INTERATIVA UTILIZANDO O MICROSOFT KINECT

Projeto de Pesquisa e implementação apresentado à disciplina de TCC do Curso de Sistemas de Informação como requisito parcial para obtenção de créditos.

Orientador: Artur Ribas Barbosa

UNIVERSIDADE DO VALE DO SAPUCAÍ

POUSO ALEGRE - MG

2014

AGRADECIMENTOS

Agradeço ao orientador Artur Barbosa pelos conselhos. aos meus Amigos Donizette Domingues, Leda Grasiele e Rinaldo Bueno que me forneceram todo o apoio técnico e psicológico durante esse projeto, e à minha família que sempre me apoiou durante toda essa jornada.

The ideal system so buries the technology that the user is not even aware of its presence. The goal is to let people get on with their activities, with the technology enhancing their productivity, their power, and their enjoyment, ever the more so because it is invisible, out of sight, out of mind. People should learn the task, not the technology. They should be able to take the tool to the task, not as today, where we must take the task to the tool. And these tools should follow three axioms of design: simplicity, versatility, and pleasurability.

(Norman, 1998)

COSTA, S. D. **VITRINE INTERATIVA USANDO O KINECT** - Curso de SISTEMAS DE INFORMAÇÃO, Universidade do Vale do Sapucaí, Pouso Alegre, 2014.

RESUMO

Vitrines vêm sendo utilizadas para atrair e despertar o desejo de consumidores por milhares de anos, porém isso ocorre passivamente, onde a interação é uma forma de engajar o cliente. Por outro lado, interfaces naturais permitem que essa interação ocorra de forma natural e transparente. Em novembro de 2010, a Microsoft lançou o Kinect, um sensor que possui uma câmera 3D capaz de detectar movimentos de forma mais precisa do que as câmeras convencionais. Sendo assim, esse projeto faz uso da linguagem de programação C Sharp e do Kinect SDK para construir uma aplicação WPF que permita aos usuários interagirem com uma vitrine através de controles de movimento captados pelo Kinect, visualizando os produtos disponíveis. Os testes demonstraram que o Kinect consegue reconhecer movimentos de forma precisa desde que não esteja exposto a luz solar, e que a aplicação obteve uma boa recepção, sendo usada sem problemas por pessoas de todas idades. Esta pesquisa fornece amplas possibilidades de extensão, visto que o conhecimento exposto sobre interfaces naturais e o Kinect, pode ser usado nas mais diversas aplicações, sejam elas relacionadas a entretenimento, produtividade ou até mesmo robótica e medicina, que possam vir a ser exploradas futuramente por outros acadêmicos.

Palavras-chave: Interfaces Naturais de Usuário, Kinect, Kinect SDK, C Sharp, Câmeras 3D, Captação de movimentos, Vitrine Interativa.

COSTA, S. D. **VITRINE INTERATIVA USANDO O KINECT** - Curso de SISTEMAS DE INFORMAÇÃO, Universidade do Vale do Sapucaí, Pouso Alegre, 2014.

ABSTRACT

Showcases have been used to attract and arouse the interest of consumers for thousands of years, but it happens passively; interaction is necessary to engage the client. On the other hand, natural user interfaces allow this interaction to happen in a natural and transparent way. In November 2010, Microsoft launched Kinect, a sensor which is able to recognize movements more precisely than standard cameras. This project uses the language C Sharp and the Kinect SDK to build a WPF application that permits users to interact with a showcase using only gestures, to navigate through the available products. Tests have shown that Kinect can recognize gestures very accurately, as long as it is not under direct sunlight. In addition, the application had a very high acceptance level, being used by people of all ages and backgrounds without difficulties. This research offers great potential for further works, since knowledge about Kinect and natural user interfaces that now has been exposed can be used in projects that belong to a wide range of areas, from entertainment and productivity to robotics and medicine. These areas will likely be explored in the future by other academics.

Key words: *Natural User Interfaces, Kinect, Kinect SDK, C Sharp, Depth Cams, Movement recognition, Interactive Showcase.*

ÍNDICE DE ABREVIATURAS

GUI	*Graphical User Interfaces*
TUI	*Text User Interfaces*
NUI	*Natural User Interfaces*
ENIAC	*Electronic Numerical Integrator Analyser and Computer*
3D	*3 Dimensional*
MS	*Microsoft*
SDK	*Software Development Kit*
RGB	*Red Green Blue*
LISA	*Local Integrated Software Architecture*
WPF	*Windows Presentation Foundation*

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

INTRODUÇÃO

Vitrines vêm sendo utilizadas como forma de publicidade para atrair e despertar o desejo de consumidores por milhares de anos, porém isso ocorre passivamente. A forma como os produtos são expostos pouco mudou nesse intervalo. Para alcançar êxito no diálogo com o consumidor deve-se, de acordo com Sant'Anna (2006, p. 2) "basear-se no conhecimento da natureza humana. Para atrair a atenção é imprescindível saber como captá-la; para interessar é necessário conhecer cada uma das reações do ser humano, seus instintos e sentimentos", então é necessário conhecer o consumidor para definir a melhor forma de comunicação com o público.

Por outro lado, embora o controle de interfaces por movimentos e por voz já seja realidade há um longo tempo, devido às limitações dos dispositivos de captação utilizados, seu uso é bastante limitado e seu desempenho é prejudicado por fatores externos. Buscando suprir essa lacuna, a Microsoft lançou em 2010 o Kinect. Segundo Silveira (2011), desde os anos 80, computadores são largamente utilizados através de teclado e *mouse*, e vemos filmes de ficção científica nos quais os computadores são usados por meio de interfaces naturais, seja entendendo qualquer comando dado por voz, ou gestos em uma tela holográfica. O Microsoft Kinect é um passo para tornar interfaces naturais uma realidade no dia a dia do uso computacional.

Em 2011, Silveira desenvolveu o projeto "Técnica de Navegação em Documentos Utilizando Microsoft Kinect" onde mostra o uso da técnica, em uma aplicação de apresentação de slides, onde o método não convencional de interação resulta em maior facilidade na navegação.

Outro exemplo é o Kinect SDK *Dynamic Time Warping* (DTW) *Gesture Recognition*, criado por Celebi, Aydin, Temiz e Arici em 2010 na Turquia, que é um projeto que faz uso do SDK oficial da Microsoft, lançado em julho de 2011, onde diferentemente da maior parte das aplicações nas quais são usados gestos pré-definidos, o Kinect é utilizado para gravar gestos feitos por um usuário, armazená-los, e após isso interpretar as ações do usuário com base nos gestos armazenados.

Este trabalho tem como objetivo utilizar a linguagem de programação C Sharp para desenvolver um aplicativo WPF que permita aos usuários interagirem de forma

natural com uma vitrine de roupas, usando controles corporais e de voz que serão captados com a utilização do Microsoft Kinect, além de demonstrar e estudar o funcionamento dos *depth maps* e demais dados captados pelo sensor, tornando a experiência entre o cliente e a vitrine mais natural e interativa.

Ao tornar uma vitrine interativa, permite-se que ela atinja um novo paradigma dentro de sua função, adicionando a ela opções que normalmente só seriam possíveis em uma loja virtual, facilitando o processo de venda e despertando o interesse e a curiosidade do consumidor.

Do ponto de vista acadêmico, este trabalho é relevante por fazer uso de *depth cams,* que por terem se tornado acessíveis apenas recentemente com o lançamento do Kinect, tiveram seus recursos pouco explorados e possuem muito de suas possíveis utilizações práticas ainda desconhecidas, além do uso da linguagem C Sharp que não foi abordada durante o curso.

Futuramente poderá servir como referência no uso do Microsoft Kinect e demais dispositivos de captura de 3D, visto que os dados captados por esses dispositivos e o conhecimento adquirido durante a criação desse projeto podem ser usados com foco totalmente diferente do que foi abordado aqui, nas mais diversas áreas, como saúde, segurança, entretenimento, arquitetura e transporte.

2 QUADRO TEÓRICO

Neste capítulo serão discutidas as técnicas, tecnologias e metodologias que serão empregadas no desenvolvimento deste trabalho.

2.1 INTERFACES DE USUÁRIO

De grandes chaves manuais, passando por interfaces textuais e gráficas, até aos controles mentais, desde os primeiros computadores, foi necessário algum meio de controlá-los e interagir com os sistemas nele presentes. Conhecidos como interfaces de usuário, esses meios passaram por uma evolução tão relevante quanto o aumento na capacidade de processamento e armazenamento dos computadores. Como pode ser observado na Tabela 1, a interação homem-máquina tem sido uma área de pesquisa e desenvolvimento que muito se expandiu nos últimos 40 anos, sempre buscando dar ao usuário mais controle sobre o hardware e software utilizados.

"Novas tecnologias proveem poder às pessoas que as dominam. Sistemas computacionais e interfaces acessíveis são novas tecnologias em rápida disseminação. Explorar o poder do computador é tarefa para designers que entendem da tecnologia e são sensíveis às capacidades e necessidades humanas" (Rocha e Baranaukas, 2003 p. 10).

GERAÇÃO	TECNOLOGIA DE HARDWARE	MODO DE OPERAÇÃO	LINGUAGENS DE PROGRAMAÇÃO	TECNOLOGIA TERMINAL	TIPO DE USUÁRIOS	IMAGEM COMERCIAL	PARADIGMA DE INTERFACE DE USUÁRIO
-1945 pré-histórica	Mecânia e eletromecânica	Usado somente para cálculoes	Movimento de cabos e chaves	Leitura de luzes que piscam e cartões perfurados	Os próprios inventores	Nenhuma (computadores não saíram dos laboratórios)	Nenhum
1945-1955 pioneira	Válvulas, máquinas enormes e com alta ocorrência de falha	Um usuário a cada tempo usa a máquina (por um tempo bastante limitado)	Linguagem de máquina 001100111101	TTY. Usados apenas nos centros de computação	Especialistas e pioneiros	Computador como uma máquina para cálculos	Programação, batch
1955-1965 histórica	Transistores, mais confiáveis. Computadores começam a ser usados fora de laboratórios	Batch (computador central não acessado diretamente)	Assembler ADD A,B	Terminais de linha glass TTY	Tecnocratas, profissionais de computação	Computador como um processador de Informação	Linguagens de Comando
1965-1980 tradicional	Circuito integrado. relação custo-benefício justifica a compra de computadores para muitas necessidades	Time-sharing	Linguagens de alto nível (Fortran, Pascal, C)	Terminais full screen, caracteres alfa-numéricos. Acesso remoto bastante comum	Grupos especializados sem conhecimento computacional (caixas automáticos, p.ex.)	Mecanização das atividades repetitivas e não criativas	Menus hierárquicos e preenchimento de formulários
1980-1995 moderna	VLSI. Pessoas podem comprar seu computador.	Computador pessoal para um único usuário	Linguagens orientadas a problemas/objetos (planilhas de cálculo)	Displays gráficos. estações de trabalho, portáveis	Profissionais de todo tipo e curiosos	Computador como uma ferramenta	WIMP (Window,Icons,Menus, e Point devices)
1995- futura	Integração de alta-escala. Pessoas podem comprar diversos computadores	Usuários conectados em rede e sistemas embutidos	Não imperativas, provavelmente gráficas	Dynabook, E/S multimídia, portabilidade simples, modem celular	Todas as pessoas	Computador como um aparelho eletrônico	Interfaces não baseadas em comando.

Tabela 1- Evolução das Interfaces de usuário – (Fonte: Rocha, Baranaukas, 2003 p. 10)

Até a década de 60, os primeiros *mainframes* funcionavam apenas processando dados por lotes. O mais famoso exemplo é o ENIAC – Abreviatura para *Electronic Numerical Integrator Analyzer and Computer*, ou em português Computador Analisador e Integrador Eletrônico de Números - Popularmente conhecido por ser o primeiro computador. Segundo Morimoto (2011), o ENIAC era programado através de 6.000 chaves manuais e toda a entrada de dados era feita através de cartões de cartolina perfurados, onde cada um deles podiam armazenar apenas algumas simples operações. Uma equipe preparava os cartões, incluindo as operações a serem realizadas, formando uma pilha, outra ia trocando os cartões no leitor do ENIAC, e uma terceira "traduzia" os resultados, também impressos em cartões. A figura 1 mostra o ENIAC sendo operado.

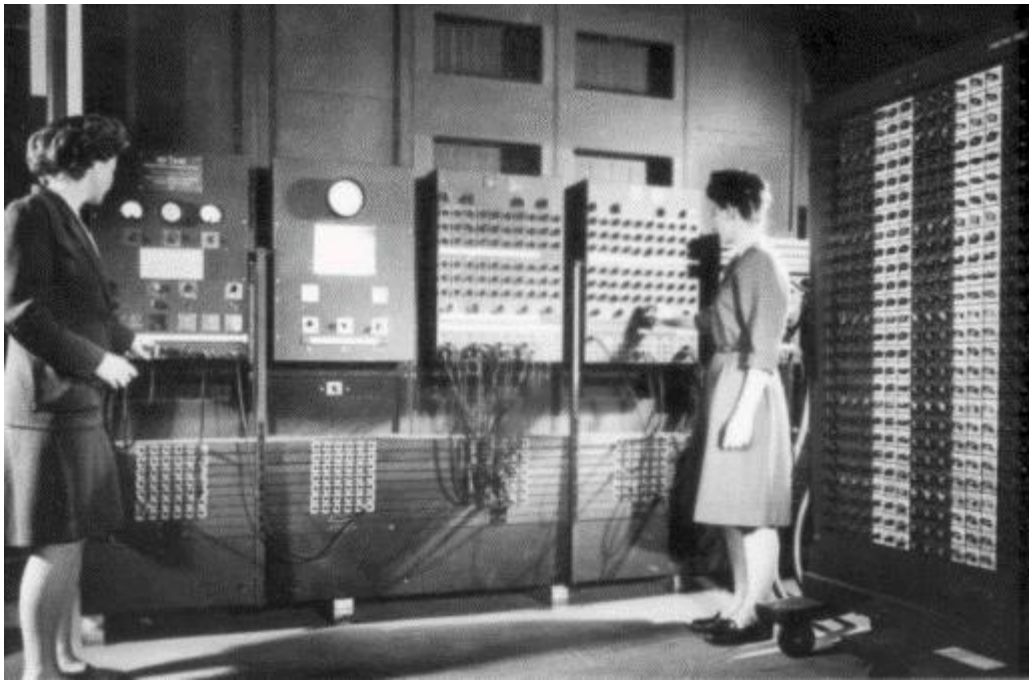

Figura 1 - Duas mulheres operando o ENIAC – (Fonte: U.S. Army)

De acordo com Garbin (2010 p. 41), no início dos anos 70 começaram a surgir os primeiros computadores operados por linhas de comando, interface que usa um teclado para enviar comandos de texto específicos para um computador. Devido ao fato de esses computadores trabalharem em sistema de tempo compartilhado, diferentemente do processamento em lotes, era possível uma real interatividade entre o usuário e o sistema, mais de um usuário poderia utilizar a máquina ao mesmo tempo e diversos processos poderiam ser executados simultaneamente. A figura 2 mostra a interface de texto do sistema operacional FreeDOS.

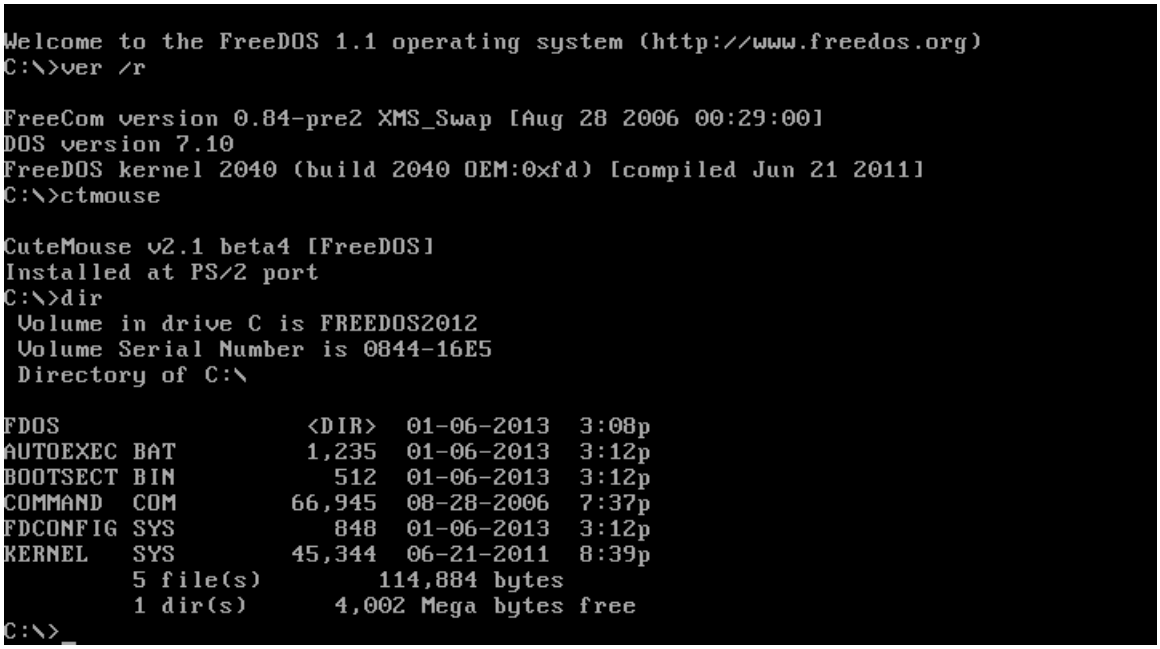

Figura 2 – Captura de tela de uma interface de texto – (Fonte: Krzysztof Burghardt)

Apesar de permitirem uma interação bastante ágil, mesmo se comparada às interfaces gráficas, o principal problema das interfaces de texto é a curva de aprendizagem. Para que o usuário possa interagir satisfatoriamente com o sistema, é necessário primeiro memorizar os comandos disponíveis e aprender a usá-los de maneira precisa, ainda que a tolerância a erros seja maior nos sistemas mais recentes. Outra barreira para a utilização do mesmo é o idioma, pois os comandos apenas fariam sentido para um usuário que conheça a língua em que o sistema está programado para utilizar.

Após o advento da popularização dos computadores, notou-se a necessidade de tornar as interfaces mais amigáveis, visto que usuários com menor conhecimento técnico deveriam ser capazes de utilizá-los em suas casas. Com isso, empresas como Apple e Microsoft começaram a comercializar dispositivos que faziam uso de uma interface gráfica parecida como a que conhecemos hoje, onde em adição ao teclado, o usuário utilizava um mouse para interagir com ícones e outras representações gráficas na tela.

Segundo Morimoto (2011), vários dos primeiros sistemas operacionais com interface gráfica enfrentaram dificuldades comerciais devido ao alto custo necessário para satisfazer os requisitos de sistema, como por exemplo o LISA, desenvolvido pela Apple, que foram produzidas cerca de cem mil unidades em dois anos, porém a maior parte delas foram vendidas com grandes descontos, muitas vezes, abaixo do preço de custo. Como a Apple investiu aproximadamente US$ 150 milhões no desenvolvimento do LISA, a conta

acabou ficando no vermelho. Juntamente com as dificuldades comerciais, esses sistemas sofreram duras críticas de profissionais da indústria, seja por serem considerados desnecessários ou porque o hardware necessário para rodá-los tinha um custo elevado para a época.

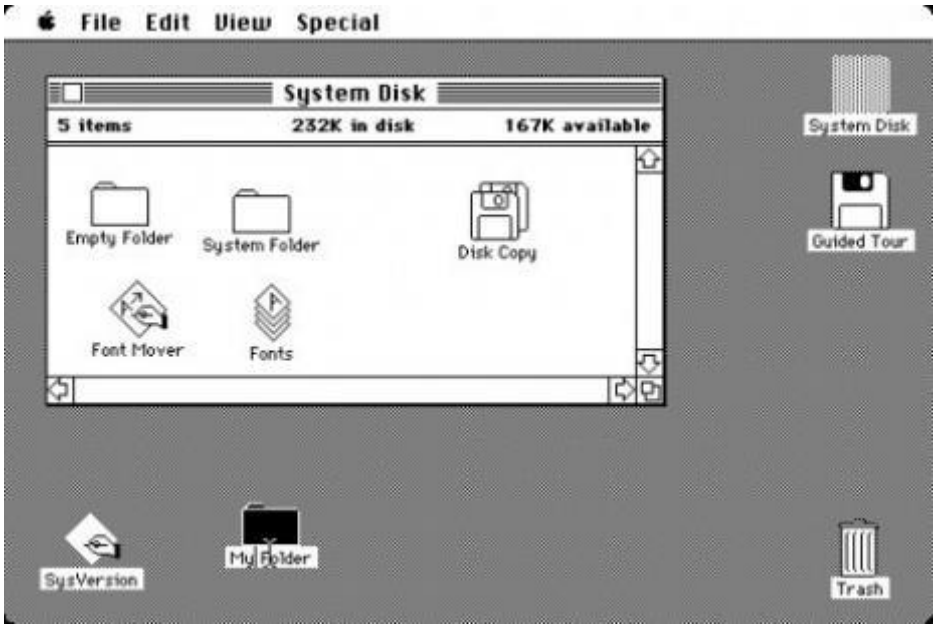

Figura 3 - Versão 1.0 do Mac OS – (Fonte: Morimoto)

Contrário às previsões da época, as interfaces gráficas não só conquistaram seu espaço como também se tornaram predominantes. Para um usuário comum, dificilmente será necessário usar comandos de texto, visto que atualmente até mesmo as máquinas mais simples como roteadores fornecem uma interface gráfica simplificada. A figura 3 demonstra a interface gráfica da primeira versão do Mac OS, modelo seguido na maioria dos sistemas atuais.

Desde os anos 70, as interfaces gráficas evoluíram muito, comparadas com os sistemas disponíveis atualmente, as primeiras versões são praticamente inutilizáveis. Os sistemas operacionais atuais permitem uma ampla gama de possibilidades ao usuário, possuem um design mais amigável, maior velocidade, praticidade e tolerância a erros, possibilitando assim uma experiência mais agradável.

De acordo com Rocha e Baranaukas (2011), individualmente, essas interfaces de usuário têm mudado a vida de muitas pessoas: médicos estão podendo fazer diagnósticos mais precisos; crianças estão expandindo os horizontes em ambientes de aprendizagem;

artistas gráficos podem explorar mais possibilidades criativas; e pilotos têm mais segurança em seus voos.

Porém, mesmo com a perceptível evolução, essas interfaces continuaram a ser uma barreira, um passo a mais entre o usuário e o objetivo que ele busca cumprir.

Segundo Norman (1998), "O sistema ideal enterra a tecnologia de tal forma que o usuário nem mesmo percebe a sua presença, as pessoas deveriam ter de aprender apenas a realizar a tarefa, e não como manipular a ferramenta". Buscando atingir esse paradigma, surgiram então as NUI - *natural user interfaces* ou Interfaces Naturais de Usuário – que são definidas por Buxton (2010) como "uma interface de usuário desenvolvida para reusar habilidades existentes na interação direta com o conteúdo".

Ou seja, uma interface que o usuário pode utilizar satisfatoriamente e de forma intuitiva, sem a necessidade de passar por um longo processo de aprendizagem, visto que ele usará ações similares às já usadas no dia a dia. Isso pode ser observado no exemplo da Figura 4, onde a usuária faz o personagem dentro do jogo acertar a bola com a raquete fazendo o mesmo movimento que faria para acertar a bola na vida real. Em adição ao controle de movimentos, também estão incluídos dentro das interfaces naturais, os comandos por voz, a escrita manual e as telas de toque.

Figura 4 – Usuária controlando um jogo por meio de movimentos (Fonte: Microsoft)

Garbin (2011) diz que “Interfaces que se baseiam em reconhecimento possuem algumas diferenças com relação às interfaces de texto e gráficas no que se refere à precisão do input do usuário, já que podem ocorrer erros de interpretação durante o reconhecimento, tornando-o incerto. Por isso, devem fornecer um feedback ao usuário de modo que ele mesmo possa se monitorar, checar a causa do erro de interpretação e corrigi-lo. Ao mesmo tempo, ao se programar um software de uma aplicação deste tipo, deve-se utilizar profundos conhecimentos sobre o contexto da aplicação para que haja menos erros de interpretação no reconhecimento dos dados de entrada do usuário.”

Desta forma, torna-se necessária uma maior tolerância a erros, tanto por parte do software quanto por parte do dispositivo de entrada. Câmeras e microfones vêm sendo usados como formas de *input* há vários anos, porém esses dispositivos carecem da precisão requerida, pois sofrem interferência de fatores externos como iluminação e ruído. Buscando resolver esse problema, foram então criadas as *Depth Cams.*

2.2 DEPTH CAMS

Basicamente, as câmeras comuns que utilizamos no dia a dia para tirar fotografias e gravar vídeos, capturam a luz de uma cena e geram uma visualização da mesma em duas dimensões, armazenando a cada pixel, sua posição no eixo X Y (horizontal e vertical) e a informação referente a cor capturada naquela posição.

Essas imagens vêm sendo usadas na interação com computadores por um longo tempo, porém elas são afetadas drasticamente pela distância, pela qualidade da câmera e principalmente pelas condições de iluminação, tornando as mesmas inconstantes e inconsistentes, dificultando a análise e identificação por um computador que não tem a mesma capacidade de abstração de um cérebro humano. Mesmo com inúmeros algoritmos criados com o objetivo de estabilizar, corrigir e padronizar essas imagens, não foi possível com a atual tecnologia, contornar de forma eficiente essas limitações.

Figura 5- Representação gráfica de um depht map

Foi então que surgiram as *Depth Cams* (câmeras de profundidade), que funcionam de forma parecida com as câmeras comuns, porém ao invés de capturar a luz e armazenar informações referentes à cor de cada pixel da cena, as *depth cams* lançam ao ambiente uma luz infravermelha que não é visível pelo olho humano. Ao ser bloqueada por algum

objeto presente no local, a luz retorna a seu ponto de origem, então o *software* que pode estar presente tanto em um computador interno na câmera quanto em um equipamento externo, realiza um cálculo tomando como base o tempo gasto para que o infravermelho retornasse ao projetor, definindo a distância entre que aquele ponto do eixo X Y e a câmera. Os raios infravermelhos não são afetados pela luz visível, funcionando mesmo em locais sem nenhuma iluminação, sendo assim, fornecem uma informação precisa, consistente e que pode ser facilmente entendida por um *software*. Dessa forma, a imagem final não é mais gerada apenas em duas dimensões (horizontal e vertical), mas sim em três, adicionando o eixo Z (profundidade). Uma representação gráfica de um *depht map* pode ser visto na figura 5.

2.3 KINECT

O Kinect (ou como era conhecido antes de seu lançamento "Project Natal") encabeçado pela Microsoft e lançado em novembro de 2010, é um dispositivo que inicialmente visava permitir aos jogadores interagir com o console XBOX 360 sem a necessidade de ter em mãos um controle.

Apesar de já haverem outras *Depth Cams* disponíveis na época de seu lançamento, o que tornou o Kinect um marco foi sua disponibilidade. A tecnologia que antes só era acessível por grandes empresas e alguns profissionais capazes de investir uma grande quantia em dinheiro, agora podia ser adquirida pela grande massa de desenvolvedores e usuários comuns, permitindo que uma enorme variedade de pesquisas e aplicativos fosse desenvolvida na área.

Como inicialmente era uma tecnologia fechada que não permitia por padrão o seu uso em dispositivos que não fossem os consoles Xbox, iniciou-se uma corrida para o desbloqueio do acessório. Segundo Biggs (2010), empresas como a *Adafruit* ofereceram prêmios em dinheiro ao primeiro que disponibilizasse *drivers* funcionais que permitissem a utilização do Kinect em computadores, cerca de uma semana após o lançamento, já haviam diversas formas de acessar o Kinect, tanto em sistemas operacionais Linux quanto em Windows. Consequentemente apareceram centenas de aplicativos que usavam a câmera da Microsoft, contribuindo ainda mais para sua popularização. Em

fevereiro de 2011, com todas as condições favoráveis e em conjunto com o marketing pesado criado pela Microsoft, o Kinect alcançou a marca de 10 milhões de unidades vendidas em 4 meses, quebrando recordes e se tornando o acessório eletrônico mais vendido da história.

De acordo com as especificações divulgadas pela Microsoft, o Kinect é composto por duas câmeras, sendo a primeira uma câmera VGA comum, responsável por capturar a imagem com cores e a segunda, a câmera de profundidade, responsável por capturar as imagens em 3D. Ambas possuem a resolução de 640x480 pixels e alcance entre 1 e 4 metros, além do projetor infravermelho que lança os raios no ambiente para que seja calculada a distância dos objetos. Também podemos encontrar um eixo motorizado controlada via *software*, capaz de mover o sensor verticalmente em uma angulação de 27° e em seu interior um grupo de 4 microfones capazes de capturar o som e definir de que posição ele se originou. A figura 6 mostra o Kinect sem a capa, onde é possível observar o posicionamento das câmeras e microfones.

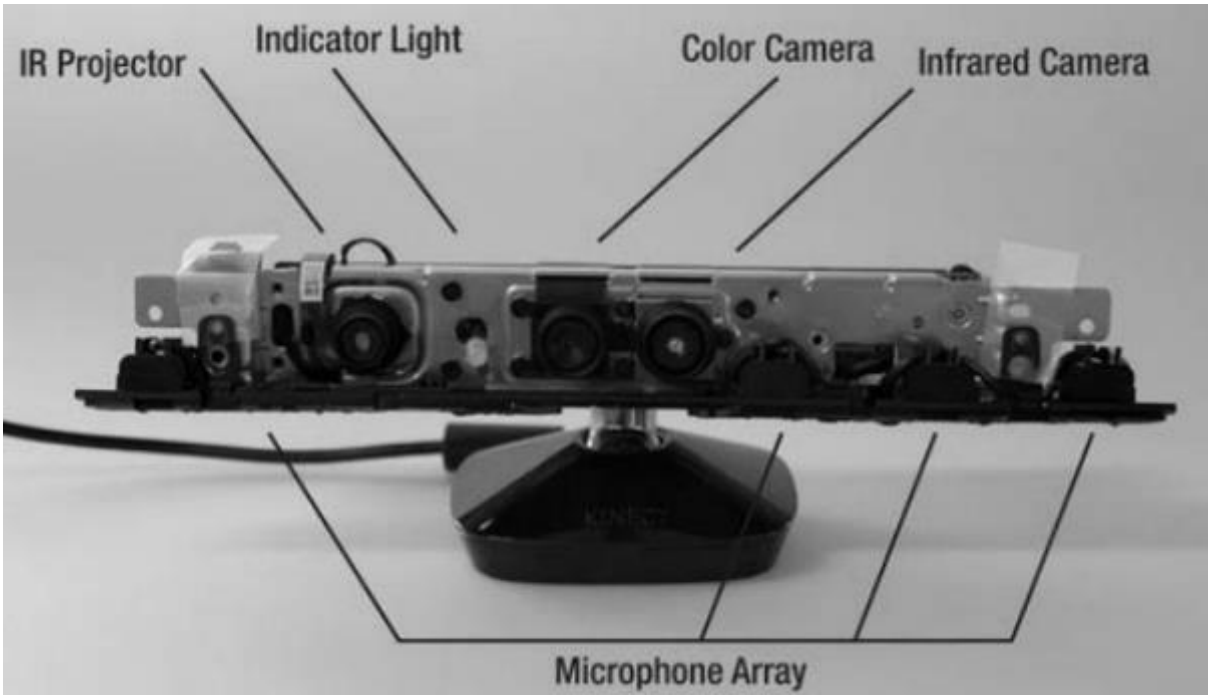

Figura 6 - Kinect sem a capa – (Fonte: Ashley, Webb)

O Kinect foi escolhido para este projeto por seu fácil acesso e por suas especificações atenderem aos requisitos de captação necessários, além da maior disponibilidade de conteúdo e documentação em relação aos seus concorrentes.

2.4 FERRAMENTAS MICROSOFT

Aqui serão analisadas as ferramentas distribuídas pela Microsoft, que foram escolhidas para o desenvolvimento do projeto.

2.4.1 KINECT SDK

Um SDK (sigla para *Software Development Kit*, ou em português, Kit de Desenvolvimento de Software), é um conjunto de ferramentas e documentos que auxiliam os desenvolvedores a criar aplicações seguindo um padrão de desenvolvimento de uma plataforma em questão.

O Kinect SDK é um conjunto de bibliotecas que permite programar aplicações em diversas plataformas de desenvolvimento da Microsoft usando o Kinect como dispositivo de captação de dados e manipular esses dados de forma fácil e padronizada.

O Kinect SDK foi o principal fator na decisão de usar um ambiente Windows, ao invés das ferramentas livres disponíveis, pois fornece melhor suporte no reconhecimento de esqueletos e possui além do reconhecimento de voz que não é encontrada nos demais, facilitando o processo de desenvolvimento de aplicações que fazem uso do sensor.

De acordo com o blog oficial do Kinect, a última versão do SDK compatível com a primeira versão do sensor é a versão 1.8, lançada em setembro de 2013, que trouxe melhorias nas ferramentas de remoção de background, captação de profundidade e cor simultaneamente e interação com múltiplos sensores.

O SDK é compatível com os sistemas operacionais Windows 7, Windows 8 e Windows 8.1, funciona utilizando o .NET Framework 4.x, é documentada de forma eficiente e é compatível com as linguagens C Sharp e C++.

2.4.2 C SHARP

C#, também escrito como C Sharp, é uma linguagem de programação orientada a objetos, fortemente tipada que combina os melhores elementos de múltiplas linguagens amplamente conhecidas como C++, Java, Visual Basic ou Delphi. Foi desenvolvida pela Microsoft em 2000, sendo parte da iniciativa .NET e encabeçada por Anders Heljsberg, diretor de *design* da Microsoft na época.

Teve como objetivo desde sua elaboração, ser uma linguagem simples, moderna e de uso geral, buscando prover suporte aos princípios da engenharia de *software*. Seu uso é amplo, apesar de não ser tão econômica em recursos como o C. Ela pode ser usada tanto em sistemas embarcados quanto em sistemas distribuídos ou aplicações para PC.

Segundo Pozzebon (2010), C# é, de certa forma, a linguagem de programação que mais diretamente reflete a plataforma .NET sobre qual todos os programas .NET executam. O C# está de tal forma ligado a essa plataforma que não existe o conceito de código não gerenciado em C#. Suas estruturas de dados primitivas são objetos que correspondem a tipos em .NET.

3 QUADRO METODOLÓGICO

Neste capítulo serão abordadas as metodologias e processos que foram usados na execução desse projeto, assim como os envolvidos e o meio em que ele será implantado e testado.

3.1 TIPO DE PESQUISA

> Pesquisa é a construção de conhecimento original de acordo com certas exigências científicas. Para que seu estudo seja considerado científico você deve obedecer aos critérios de coerência, consistência, originalidade e objetivação. É desejável que uma pesquisa científica preencha os seguintes requisitos: a) a existência de uma pergunta que se deseja responder; b) a elaboração de um conjunto de passos que permitam chegar à resposta; c) a indicação do grau de confiabilidade na resposta obtida.
>
> (GOLDEMBERG, 1999, p.106)

Pesquisa científica é mais especificamente definida por (GALLIANO, 1977, p. 6) como "a realização concreta de uma investigação planejada e desenvolvida de acordo com as normas consagradas pela metodologia científica". Uma pesquisa pode ser tipificada de diversas formas, sendo levados em consideração seus procedimentos técnicos, seu objetivo e sua abordagem. Diversos fatores devem ser levados em consideração ao escolher qual o tipo de pesquisa ser usada no projeto. Pode-se citar como exemplo principalmente a natureza do problema, o meio em que ele se encontra, e a capacidade e disponibilidade do pesquisador para realizar cada uma delas, visto que podem requerer diferentes quantidades de recursos e esforços. As categorias em que uma pesquisa é colocada não são excludentes, ou seja, uma mesma pesquisa pode estar em mais de uma categoria ao mesmo tempo, desde que atenda os pré-requisitos de cada uma delas.

Segundo Marconi e Lakatos (2009, p. 6), "Pesquisa aplicada caracteriza-se por seu interesse prático, isto é, que os resultados sejam aplicados ou utilizados, imediatamente, na solução de problemas que ocorrem na realidade".

Observando que este projeto está dirigido à resolução de um problema prático e específico, e que para isso será desenvolvida uma aplicação *desktop* e a sua implantação para testes em um estabelecimento comercial, ele se enquadra como uma pesquisa aplicada.

3.2 CONTEXTO

Inicialmente o aplicativo funcionará exclusivamente com o sistema operacional Windows e terá como público alvo estabelecimentos de varejo dos mais diversos setores. Para fins de pesquisa e obtenção de dados, o sistema será testado em uma loja varejista de roupas, calçados, brinquedos e acessórios, localizada na cidade de Bom Repouso - MG, tendo seu número de clientes atendidos mensalmente estimado em 600 (seiscentos). Cerca de 60% da clientela é formada por mulheres de 20 a 40 anos, das classes B, C e D. O município do interior de Minas Gerais tem uma população de 10.000 habitantes e sua economia é baseada principalmente no cultivo da batata inglesa e do morango. Cerca de 50% da população vive na zona rural e a loja está localizada próxima à extremidade de uma das principais ruas da cidade.

A implantação em uma cidade de maior população tornaria o sistema acessível a um maior número de usuários, porém o baixo custo, a disponibilidade e a facilidade fizeram com que o estabelecimento de Bom Repouso fosse escolhido.

O sistema será testado em dois ambientes diferentes, sendo um, a vitrine localizada no interior da loja, estando disponível em horário comercial, e o outro, a vitrine exterior, que funcionará 24 horas por dia durante o período de testes que se iniciará no segundo semestre.

3.4 METODOLOGIA

Para a execução do projeto de forma eficiente, foi utilizada a metodologia ICONIX, já exposta no quadro teórico. Essa metodologia usa marcos e passos como forma de medir e planejar o desenvolvimento, buscando sempre manter uma documentação clara e coesa. Baseado nas informações coletadas no início dessa pesquisa, iniciou-se a modelagem do sistema. Os passos para o desenvolvimento de ambos os módulos estão descritos nas próximas seções.

3.5 LEVANTAMENTO DE REQUISITOS

Antes de iniciar o desenvolvimento da aplicação, foram realizadas algumas entrevistas com os gerentes do estabelecimento do local, buscando obter dados como: horário com maior movimento; produtos de maior e menor exposição; público alvo; gastos e processos necessários com o atual sistema; possibilidade de maior automatização do processo; como alimentar a base de dados, entre outras.

Então será iniciada uma análise do cotidiano do local para complementar e confirmar as informações obtidas nas entrevistas.

Seguindo o ICONIX, o levantamento de requisitos inicial requer a análise das principais funções que devem estar no sistema. Sugere-se que seja elaborada uma lista seguindo o padrão "O sistema deve...", para ser então apresentada ao cliente, buscando validar a relação desses requisitos às suas necessidades.

Deste modo, foram levantados os principais requisitos funcionais:

- O sistema deve exibir as roupas disponíveis.
- O sistema deve opcionalmente mostrar informações sobre as roupas.
- O sistema deve ser controlado por movimentos.
- O sistema deve permitir a inserção de novos produtos de forma fácil.
- O sistema deve ser capaz de ser usado por pessoas não familiarizadas à tecnologia.

Após levantados e analisados os requisitos, seguiu-se então para o desenho e desenvolvimento.

3.6 DESENHO E MODELAGEM

Em posse dos requisitos validados pelo cliente, criou-se então o diagrama de casos de uso, para definir de que forma os atores irão interagir com o sistema.

Como pode ser observado na figura 7, a maior parte da interação será entre o programa e os clientes do estabelecimento. Os funcionários do estabelecimento interagirão apenas eventualmente para adicionar ou retirar produtos da lista a ser exibida para os consumidores.

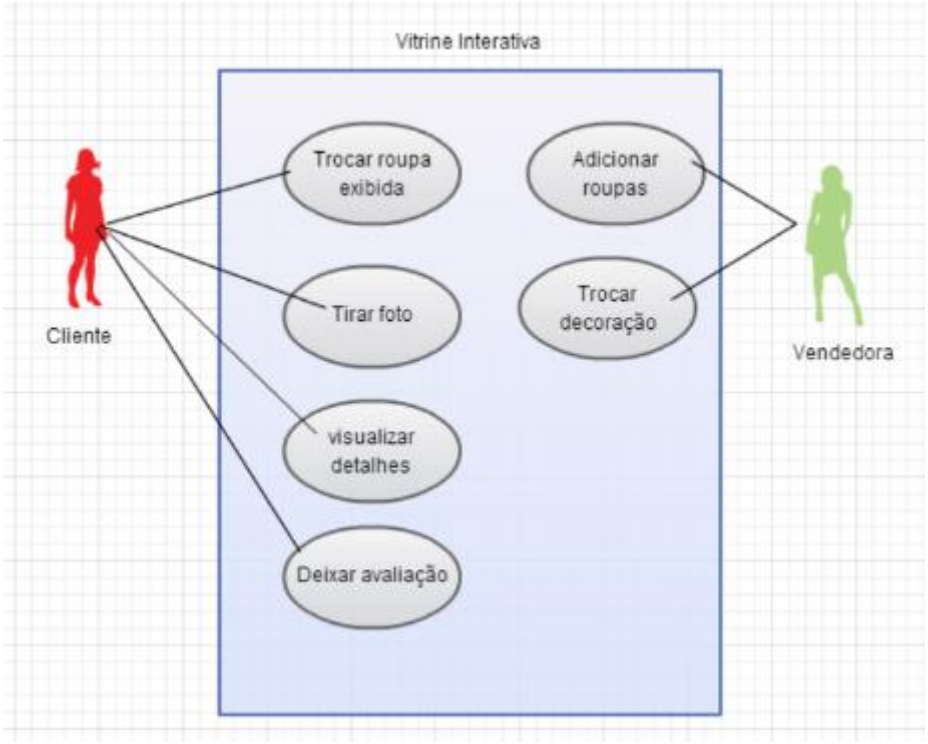

Figura 7 - Diagrama de casos de uso

3.7 PREPARAÇÃO DO AMBIENTE

Para a realização deste trabalho, foi configurado o ambiente de desenvolvimento que consiste na instalação das ferramentas e *SDKs* necessários para a desenvolvimento da aplicação e a instalação e configuração do banco de dados, além do ambiente físico que envolve o posicionamento da câmera e do projetor de forma que possibilite a interação. Para iniciar o desenvolvimento de uma aplicação que faz uso do Kinect e do Kinect SDK, é necessário instalar os seguintes componentes em um sistema operacional Windows 7 ou superior:

- Microsoft Visual Studio
- Microsoft .NET Framework 4
- Kinect for Windows drivers
- Kinect for Windows SDK (x86 or x64)
- DirectX 9.0c
- Speech Platform API
- Kinect for Windows Runtime Language Pack
- Microsoft Speech Platform SDK

Em adição aos componentes requeridos pelo Kinect SDK, também foram instalados o *Kinect toolkit explorer* e a biblioteca *Coding 4 Fun* que auxiliam no desenvolvimento.

Quanto aos requisitos de Hardware, as especificações mínimas abaixo precisam ser atendidas:

- Sensor Kinect
- Processador dual-core, 2.66-GHz
- Placa de vídeo com suporte ao Microsoft DirectX 9.0
- 2 GB de Memória Ram
- Fonte de energia USB para o Kinect

3.8 DESENVOLVIMENTO

Como exposto no quadro teórico, no desenvolvimento da aplicação foi utilizada a linguagem de programação *C Sharp* e o Microsoft Visual Studio como IDE[1]. Como o Kinect SDK tem suporte à aplicações WPF[2], esse foi o modelo escolhido porque facilita o desenvolvimento da interface e sua interação com o código em si.

Inicialmente notou-se a necessidade de desenvolver uma interface de *debugging* para melhor compreender o funcionamento do Kinect e seu SDK e também analisar mais facilmente se os dados estavam sendo captados e interpretados corretamente. Para construir essa interface decidiu-se criar duas *streams* para exibir os dados da câmera RGB e da câmera 3D, além de um *canvas* que é usado para desenhar as linhas das juntas e articulações.

Esses elementos são inseridos na interface WPF através de um código XAML[3] que pode ser observado na figura 8. Como as *streams* são geradas com a criação sequencial de *bitmaps*, usa-se controles do tipo imagem. Assim como no XML, propriedades como tamanho, margens e nome, usado para referenciar o elemento no código, podem ser adicionadas para personalizar uma *tag* no XAML.

```
<Image Name="colorStream" Height="240" Width="320" />
<Image Name="dephtStream" Height="240" Width="320" />
<Canvas Name="canvasSkeleton" Height="240" Width="320" />
```

Figura 8 - Código XAML referente à definição de controles WPF para Streams e Canvas

Para que a aplicação consiga se comunicar corretamente com o Kinect e fazer uso do SDK é necessário adicionar primeiramente "Microsoft.Kinect" nas referências do projeto, e então à lista de *namespaces* no início do código, como mostrado na figura 9.

---

[1] *Integrated Development Environment (ou IDE)* é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de *software* com o objetivo de agilizar este processo.

[2] *Windows Presentation Foundation (*ou *WPF*) é um sistema gráfico para renderização de interfaces em aplicações baseadas em Windows.

[3] Linguagem de marcação derivada do XML.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.IO;
using Microsoft.Kinect;
```

Figura 9 - Lista de *namespaces* inseridos no início do código

Após essa adição é possível utilizar as classes e métodos do Kinect. Cria-se então um objeto do tipo KinectSensor para referenciar o sensor utilizado. É possível utilizar vários sensores caso seja necessário.

O Kinect envia três fluxos de dados para a aplicação, os da câmera RGB, os da câmera 3D e os dados dos esqueletos detectados, cada um desses fluxos precisa ser habilitado separadamente antes de ativar o sensor com o método *Start*. Os métodos responsáveis por habilitar esses fluxos podem receber parâmetros referente à qualidade e resolução da imagem, número de *frames* por segundo e modo de captura.

Sempre que o sensor adquire todos os dados referentes a um fluxo, ele dispara um evento avisando que o *frame* referente a este fluxo está pronto. Esses eventos são ColorFrameReady, DephtFrameReady e SkeletonFrameReady, adicionalmente há o evento AllFramesReady que é disparado somente quando os dados referentes aos três fluxos estão disponíveis, útil nos casos em que ter *frames* descartados não é um problema e se faz necessário sincronizá-los.

São criados então *event handlers* para cada fluxo, onde os dados recebidos podem ser usados. A figura 10 mostra o trecho de código referente à habilitação dos fluxos, inicialização do sensor e criação dos *event handlers*.

```
KinectSensor miKinect;

        public MainWindow()
        {
            InitializeComponent();
        }


        private void Window_Loaded(object sender, RoutedEventArgs e)
        {

            if (KinectSensor.KinectSensors.Count == 0)
            {
                MessageBox.Show("Nenhum Kinect Detectado", "Nenhum Kinect :(");
                Application.Current.Shutdown();
            }

            try
            {
                miKinect = KinectSensor.KinectSensors.FirstOrDefault();

miKinect.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
                miKinect.DepthStream.Enable();
                miKinect.SkeletonStream.Enable();
                miKinect.Start();
            }
            catch
            {
                MessageBox.Show("Ocorreu um erro ao receber dados do Kinect",
"Kinect fail =(");
                Application.Current.Shutdown();
            }

            miKinect.ColorFrameReady += new
EventHandler<ColorImageFrameReadyEventArgs>(miKinect_ColorFrameReady);
            miKinect.DepthFrameReady += new
EventHandler<DepthImageFrameReadyEventArgs>(miKinect_DepthFrameReady);
            miKinect.SkeletonFrameReady += new
EventHandler<SkeletonFrameReadyEventArgs>(miKinect_SkeletonFrameReady);
        }
```

Figura 10 – Código para inicialização do Kinect

Para poder exibir os dados da câmera RGB, primeiramente se criou um vetor do tipo byte[ ] para receber os dados enviados, o tamanho necessário para esse vetor varia de acordo com a resolução da imagem, então se deve usar o atributo PixelDataLenght para defini-lo.

Em seguida cria-se uma variável do tipo WriteableBitmap, que gasta menos recursos do que um Bitmap comum, e a instancia com os bytes copiados anteriormente

para o vetor. Por último, atribui-se essa imagem criada ao atributo *Source* do *controller* colorStream que foi definido anteriormente no código XAML. Esse processo se repete a cada quadro enviado pelo Kinect. O código referente aos dados câmera RGB pode ser visto na figura 11.

```
WriteableBitmap bitmapEficiente = null;
byte[] datosColor = null;


void miKinect_ColorFrameReady(object sender, ColorImageFrameReadyEventArgs e)
        {
            using (ColorImageFrame framesImagen = e.OpenColorImageFrame())
            {
                if (framesImagen == null)
                    return;

                int stride = framesImagen.Width *
framesImagen.BytesPerPixel;//basically, 4 bytes per pixel
                byte[] datosColor = new byte[framesImagen.PixelDataLength];


                framesImagen.CopyPixelDataTo(datosColor);


                 if (bitmapEficiente == null)
                    {
                        bitmapEficiente = new WriteableBitmap(framesImagen.Width,
framesImagen.Height, 96, 96, PixelFormats.Bgr32, null);

                    }

                    bitmapEficiente.WritePixels(
                        new Int32Rect(0, 0, framesImagen.Width,
framesImagen.Height),
                    datosColor,
                    stride, 0

                        );

                    colorStream.Source = bitmapEficiente;


            }
```

Figura 11- Código referente à exibição dos dados da câmera RGB

O procedimento para a câmera de profundidade é semelhante, mas como esse fluxo passa informações referentes à distância de cada *pixel* ao invés da cor, para exibir esses dados de forma gráfica é necessário atribuir cores para cada um desses valores,

normalmente são utilizados diferentes tons de cinza. Então se cria um outro objeto do tipo *WriteableBitmap* usando as cores referentes à cada distância e o atribui ao atributo *Source* do *controller* depthStream, também definido no código XAML.

O sistema de detecção de esqueletos é onde o Kinect SDK se destaca, devido a um intensivo processo de *machine learning*, é possível reconhecer esqueletos nas mais diversas posições e com alta precisão, mesmo em momentos em que partes do corpo estão sobrepostas ou fora do campo de visão do sensor.

Assim como nos outros fluxos, o sensor envia os dados referentes aos esqueletos reconhecidos sempre que termina a captura de um *frame*. O Kinect envia para a aplicação um vetor do tipo Skeleton com todos os esqueletos detectados naquele frame. O código que armazena os dados dos esqueletos reconhecidos é mostrado na figura 12.

```
void miKinect_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
        {

            Skeleton[] esqueletos = null;
            using (SkeletonFrame framesEsqueletos = e.OpenSkeletonFrame())
            {
                if (framesEsqueletos != null) {
                    esqueletos = new
Skeleton[framesEsqueletos.SkeletonArrayLength];
                    framesEsqueletos.CopySkeletonDataTo(esqueletos);

                }

            }
            if (esqueletos == null) return;
            foreach (Skeleton esqueleto in esqueletos)
            {
                if (esqueleto.TrackingState == SkeletonTrackingState.Tracked)
{
              …
```

Figura 12 - Trecho do código referente à detecção de esqueletos

O sensor é capaz de detectar um máximo de seis esqueletos, porém só é capaz de de reconhecer as juntas de dois esqueletos simultaneamente, essa limitação pode ser melhor visualizada na figura 13. Por padrão são rastreadas as juntas dos dois esqueletos que se encontram mais próximos do sensor, mas é possível selecionar qualquer um deles.

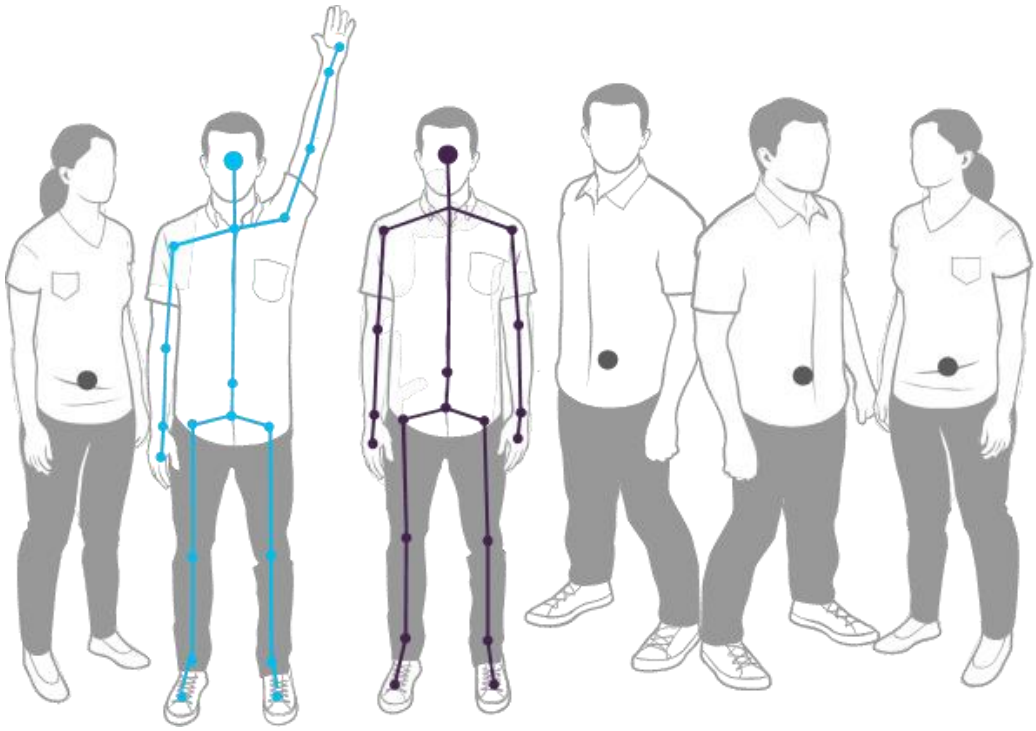

Figura 13 - Limite de detecção de esqueletos do Kinect – Fonte: Microsoft

Como pode ser observado na figura 13, cada esqueleto rastreado possui um vetor com as dezesseis juntas referentes aos pés, tornozelos, joelhos, quadris, mãos, pulsos, cotovelos, ombros, espinha e cabeça. Cada uma dessas juntas armazena uma posição nos eixos XYZ e uma angulação.

Em posse desses dados, é possível realizar toda uma gama de processos para verificar por exemplo a distância que o usuário se encontra da câmera, a distância entre duas juntas distintas ou se o esqueleto está em uma certa posição.

Apesar de ser possível utilizar coordenadas absolutas, na maior parte das vezes isso não é o ideal, visto que desta forma pessoas de estaturas diferentes não teriam a mesma experiência. Então para verificar se o usuário está com o braço esticado no eixo X por exemplo, compara-se a distância entre a junta referente à mão e a junta referente à cabeça, como mostrado no início do código na figura 14.

```
Joint jointCabeza = esqueleto.Joints[JointType.Head];
Joint handJoint = esqueleto.Joints[JointType.HandRight];

if (handJoint.Position.X - jointCabeza.Position.X < 0.5)
    {
        strechedArm();

    }


Line linhaBracoDir = new Line();
                    linhaBracoDir.Stroke = new SolidColorBrush(Colors.Red);
                    linhaBracoDir.StrokeThickness = 5;

                    ColorImagePoint pointMano =

miKinect.CoordinateMapper.MapSkeletonPointToColorPoint(handJoint.Position,
ColorImageFormat.RgbResolution640x480Fps30);
                    linhaBracoDir.X1 = pointMano.X;
                    linhaBracoDir.Y1 = pointMano.Y;

                    ColorImagePoint pointElbow =

miKinect.CoordinateMapper.MapSkeletonPointToColorPoint(elbowJoint.Position,
ColorImageFormat.RgbResolution640x480Fps30);

                    linhaBracoDir.X2 = pointElbow.X;
                    linhaBracoDir.Y2 = pointElbow.Y;

                    canvasSkeleton.Children.Add(linhaBracoDir);
```

Figura 14 - Trecho do código que verifica a distância entre a mão e a cabeça do usuário

Para desenhar um osso no *canvas*, primeiro cria-se um objeto do tipo Line e define-se sua cor e a espessura em *pixels*, em seguida se atribui ao início e fim da linha, as coordenadas referentes à duas juntas vizinhas. Após definir todos os atributos da linha, usa-se o método Children.Add para desenhar a linha no *canvas*. Vale notar que as coordenadas passadas pelo *Skeleton Stream* são diferentes das coordenadas encontradas na imagem RGB, portanto é necessário usar o método MapSkeletonPointToColorPoint para alinhá-las. A tela de *debugging* concluída pode ser vista na figura 15.

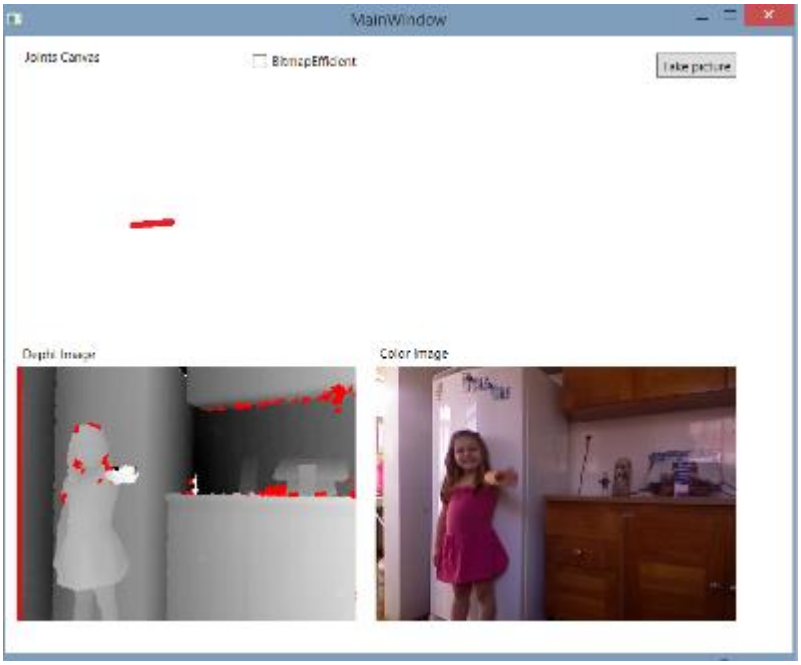

Figura 15 - Tela de debugging

Para encerrar o Kinect, basta usar o método *Stop* ao fechar a janela da aplicação, como mostrado no trecho de código encontrado figura 16

```
private void Window_Closed(object sender, EventArgs e)
        {
            miKinect.Stop();
        }
```

Figura 16 - Código para encerrar o Kinect

Com os conhecimentos adquiridos nesse primeiro estágio sobre a interação com o Kinect, iniciou-se o desenvolvimento da aplicação em si.

Primeiramente é realizada uma busca pelas fotos das roupas em um determinado diretório usando o método getFiles, em seguida as fotos encontradas são adicionadas em uma lista, então quando o botão *Next* é pressionado, o programa atribui o caminho da próxima foto ao atributo *Source* do controle WPF usado para exibir as roupas. Esse processo pode ser observado na figura 17.

```
List<Image> upperClothesAvailable = new List<Image>();

        foreach (string myFile in
        Directory.GetFiles(upperDirectory, "*.PNG", SearchOption.AllDirectories))
             {
                 Image image = new Image();
                 BitmapImage source = new BitmapImage();
                 source.BeginInit();
                 source.UriSource = new Uri(myFile, UriKind.RelativeOrAbsolute);
                 source.EndInit();
                 image.Source = source;

                 upperClothesAvailable.Add(image);
             }

        private void kinectNextUpperButton_Click(object sender, RoutedEventArgs e)
         {
             currentPicture++;
             if (currentPicture >= upperClothesAvailable.Count)
             {
                 currentPicture = 0;
             }


             suitImages.Source = upperClothesAvailable[currentPicture].Source;

         }
```

Figura 17 - Código para a exibição das roupas

Somente quando essa etapa foi terminada iniciou-se a integração com o Kinect. Para a interação com os elementos da tela, criou-se um cursor, que é um objeto do tipo InteractionHandPointer e segue o movimento da junta referente à mão do usuário, cuja a posição posteriormente será comparada à posição dos botões da interface para que se possa ativá-los.

Para facilitar esse processo foi utilizado a biblioteca KinectToolKit, que acompanha o SDK. Essa biblioteca une exemplos e ferramentas que buscam simplificar o desenvolvimento de aplicações que façam uso do Kinect. Ao referenciar a biblioteca, torna-se possível declarar certos controles WPF relacionados ao sensor no código XAML.

Nesse projeto foram utilizados os controles KinectCircleButton, que cria um botão que pode interagir com o cursor previamente criado, o KinectUserViewer, responsável por selecionar o usuário que levantar uma das mãos e ceder a ele o controle do cursor, e o KinectRegion que limita o cursor à uma certa área. O código XAML referente a esses elementos é mostrado parcialmente na figura 18.

```
<Window x:Class="WpfApplication1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
         xmlns:k="http://schemas.microsoft.com/kinect/2013"
        Title="MainWindow" Height="1200" Width="800" Loaded="Window_Loaded">
    <Grid>
        <k:KinectRegion Name="zonaCursor">
            <k:KinectUserViewer  DefaultUserColor="gray" PrimaryUserColor= "red"/>
                <k:KinectCircleButton Label="Proxima" Name="kinectNextUpperButton"
                Click="kinectNextUpperButton_Click" ></k:KinectCircleButton>
         </k:KinectRegion>
    </Grid>
</Window>
```

Figura 18 - Código XAML dos controles do KinectToolKit

Apesar de o aplicativo ainda não estar finalizado, foi possível perceber que a integração entre o Microsoft Visual Studio, o Kinect SDK e o KinectToolKit reduzem consideravelmente o trabalho necessário para desenvolver uma aplicação que use o Kinect como dispositivo de entrada.

Com isso, as principais funções do sistema foram desenvolvidas e iniciou-se a etapa de testes.

4 DISCUSSÃO DE RESULTADOS

Neste capítulo serão discutidos os resultados obtidos durante o desenvolvimento do projeto, utilizando uma explicação teórico-prática dos principais pontos.

O projeto atingiu seus objetivos satisfatoriamente, porém as limitações de tempo não permitiram que as funções extras fossem desenvolvidas.

A aplicação detecta se existe um ou mais Kinects conectados e então utilizando a detecção de juntas e esqueletos, automaticamente procura por esqueletos dentro do campo de visão da câmera e coloca no controle do sistema o usuário que se encontra mais próximo. O usuário é então informado sobre qual o melhor lugar para se posicionar, e quando levanta uma das mãos o cursor controlado pelo movimento da mão aparece na tela, permitindo que ele verifique as roupas disponíveis através de gestos. A interface da aplicação pode ser vista na figura 19.

Figura 19 - Captura da tela principal do programa sendo utilizada

Além de permitir que os comandos venham do lado de fora do vidro e eliminando assim risco de que o hardware seja roubado ou sofra vandalismo, a principal vantagem de usar o controle por gestos é facilitar a curva de aprendizagem requerida para usar a aplicação.

Isso foi comprovado pelos testes aplicados (figura 20) no estabelecimento comercial, mesmo o *software* não estando finalizado e sem as ferramentas de auxílio. Pessoas dos mais variados grupos sociais e culturais, desde crianças da geração Z[4] até idosos que não possuíam nenhuma experiência no uso de computadores conseguiram usar a ferramenta sem maiores problemas após uma breve demonstração do funcionamento da mesma, sempre expressando espanto em relação à novidade tecnológica, "Parece mágica! Como que funciona?" indagou uma das funcionárias do estabelecimento após testar a aplicação pela primeira vez.

Figura 20 - Funcionária da loja demonstrando à uma cliente como usar o programa

A detecção de movimentos pelo Kinect se mostrou satisfatória, porém ainda há pontos onde se nota a necessidade de um hardware mais avançado para a captação. O sistema consegue detectar pessoas no campo de visão rapidamente, sofrendo mínima influência de fatores externos como luz e excesso de objetos na cena, o único momento em que a detecção foi prejudicada pela iluminação foi quando o sensor foi posicionado

[4] **Geração Z** é a definição sociológica para definir geração de pessoas nascidas na década de 90 até o ano de 2010.

em ambiente externo durante o dia. Isso se deve ao fato de que a luz infravermelha emitida pelo sol sobrepõe as emitidas pelo Kinect.

A detecção de esqueletos é possível mesmo com partes do corpo fora do ângulo de visão da câmera obstruídas, seja por objetos, animais ou até mesmo outras pessoas passando momentaneamente entre o usuário e o sensor. A distância mínima em que o usuário pode ficar do sensor é de um metro, e a máxima é de 4 metros, porém a distância ideal é de cerca de 2 metros. Para esse estabelecimento em específico, esse limite não chegou a ser um problema, mas pode vir a trazer complicações para a implantação em um local com uma menor área livre. Essas limitações são bastante reduzidas na segunda versão do Kinect, mas o mesmo ainda não se encontra disponível para desenvolvedores fora do *beta test*[5].

Enquanto o sistema se destaca ao reconhecer usuários e movimentos maiores mesmo em condições adversas, a resolução da câmera de profundidade presente no Kinect dificulta o reconhecimento de movimentos mais sutis como, por exemplo, levantar dedos individualmente, que é possível somente ativando o *near mode*. A remoção de *background* é eficiente, mas gera uma imagem serrilhada[6], o que piora a aparência do produto final.

A captação de som se mostrou excelente, fazendo um ótimo trabalho no isolamento de ruídos, apesar de ser necessário fazer a calibração previamente, os quatro microfones que o sensor possui permitem definir a origem de cada som, facilitando assim esse processo. Porém diferentemente da captação de movimentos, o reconhecimento de voz é menos tolerante a erros, requerendo que o mesmo seja executado de forma mais precisa.

Como peça de marketing, a vitrine interativa atingiu resultados acima dos esperados, diversas pessoas relataram ter ido até a loja com o intuito de testá-lo e ver como funcionava após ter ouvido amigos falarem sobre o sistema, e a dona do estabelecimento relatou que foi abordada na rua por uma cliente que disse "É verdade que lá na loja agora tem um computador na parede que você controla com a mão 'pra' ver as roupas? Mas que chique! Só na Ivete Modas mesmo!". Como durante os testes a aplicação não estava finalizada e o número de peças disponíveis era bastante limitado não foi

[5] Período de testes de algum sistema antes de sua finalização.

[6] Artefato irregular que pode ser visto nas linhas de uma imagem em baixa resolução, lembrando uma serra.

possível obter dados mais precisos sobre sua influência no processo de compra.

Como registrado na figura 21, crianças se divertiram usando a aplicação enquanto os pais compravam normalmente, tornando o processo de compras mais tranquilo tanto para o cliente quanto para os funcionários do estabelecimento. Baseando-se nessa experiência foram adicionadas ferramentas focadas no entretenimento de crianças de 5 a 10 anos à lista de possíveis adições ao sistema. Vale também notar que em alguns casos, o software não foi capaz de reconhecer o esqueleto de algumas crianças de dois, três e quatro anos devido à baixa estatura.

Figura 21 - Filha de uma cliente entretida com a vitrine

Um dos conceitos iniciais do projeto é projetar a imagem em uma porta de vidro para que pudesse ser utilizado pelo lado de fora do estabelecimento, fora do horário comercial, porém durante o desenvolvimento e testes foi constatado que certos tipos de vidro bloqueiam os raios infravermelhos emitidos pelo sensor, adicionando então mais um fator a ser analisado antes da implantação do sistema em um novo local. Outro empecilho percebido foi o fato de que não é possível projetar a imagem na maior parte dos vidros pois os mesmos não retêm a luz visível aos olhos, sendo assim necessária a aplicação de uma película holográfica[7], cuja demonstração pode ser vista em uso na figura

[7] Película acrílica que devido a suas características retém a luz visível aos olhos, emitidas por um projetor

22 para tornar a projeção no vidro possível. Devido a questões orçamentárias, não foi possível adquirir essa película no tamanho requerido, então os testes necessários foram feitos utilizando uma amostra pequena da mesma, impedindo assim a avaliação em um caso de uso real nessas condições.

Figura 22 - Demonstração de uso da película holográfica (Fonte: Xlusion3D)

O posicionamento e as especificações técnicas do projetor utilizado influenciam fortemente na experiência final do usuário, projetores de 1500 lumens[8] produzem resultados aceitáveis apenas em ambientes escuros, no uso externo bons resultados foram alcançados com projetores que possuem acima de 5000 lumens.

Como alternativa ao vidro, é possível usar uma parede ou uma placa qualquer como superfície de projeção, porém quando projetando a imagem em uma superfície não transparente, o usuário por padrão acaba ficando entre o projetor e a superfície de projeção, e quanto mais próxima o usuário estiver da superfície, mais inclinado o projetor tem de estar, porém, a partir de certo grau de inclinação, a projeção passa a ficar deformada. As diferenças entre as estrutura necessárias em cada situação são demonstradas na figura 23

comum, que pode ser observada de ambos os lados.

[8] Medida utilizada para definir a potência de equipamentos de projeção.

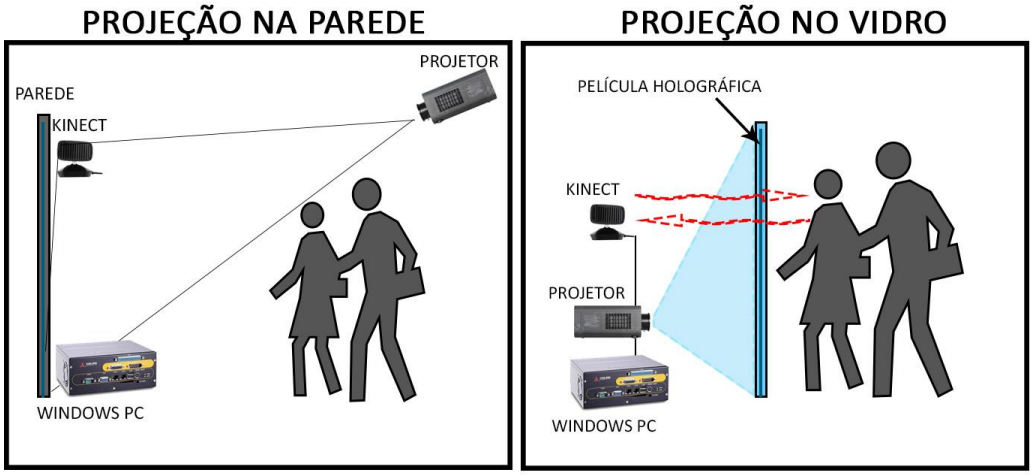

Figura 23 – Comparação entre a projeção na parede a projeção no vidro.

Na maior parte dos projetores é possível compensar essa deformação com ajustes na lente, mas o quão grande pode ser o ajuste é diferente em cada modelo.

Em resumo, apesar de ainda haver espaço para melhorias, o Kinect se provou uma ótima alternativa para a captação tanto de áudio quanto de movimentos, possuindo um robusto SDK que facilita o desenvolvimento de aplicações. O uso de interfaces naturais sem dúvidas facilita o acesso, principalmente por parte de pessoas não familiarizadas com o uso de tecnologias em geral. Tanto os clientes quanto o estabelecimento se mostraram satisfeitos com a Vitrine interativa, que além de auxiliar direta e indiretamente no processo de compra, se mostrou uma excelente ferramenta de marketing.

CONCLUSÃO

Atualmente nos encontramos em um período onde cada vez mais pessoas, que antes não possuíam contato com computadores, passam a ser incluídas digitalmente. Com isso, interfaces naturais adquirem um importante papel, tornando essa interação transparente e natural. Não há dúvidas de que apesar de terem garantido seu lugar, as interfaces naturais ainda se encontram em um estágio de maturação, e têm um longo caminho a percorrer, principalmente se tratando do reconhecimento de movimentos e voz.

O Kinect foi um grande passo em direção a tornar o reconhecimento de movimentos viável e acessível a uma gama maior de usuários e desenvolvedores, principalmente por combinar um baixo custo quando comparado às demais alternativas. Porém é notável que ainda são necessárias diversas melhorias em seu hardware, as câmeras presentes no sensor possuem uma baixa resolução, o que dificulta a detecção de movimentos mais sutis. Diversas melhorias já foram implementadas em seu sucessor, o Kinect 2, porém, devido ao fato de a câmera 3D usar raios infravermelhos, tornar sua utilização sob luz solar satisfatória continua sendo um grande desafio.

O SDK desenvolvido pela Microsoft fornece uma fácil integração entre o sensor e as ferramentas de desenvolvimento utilizadas, além de possuir um grande suporte da comunidade, o que torna esse conjunto a melhor escolha para se desenvolver uma aplicação que faça uso de todos os recursos do Kinect.

Apesar das mudanças de percurso e dificuldades encontradas, foi possível desenvolver a aplicação e seus requisitos básicos. O uso de realidade aumentada se provou mais complexo do que o esperado, principalmente devido à diferença no posicionamento das câmeras encontradas no Kinect, e não pôde ser incluído até a entrega do projeto.

Durante o período de testes o projeto se mostrou bem sucedido, tornando o processo de compra mais eficiente e agradável, seja com a exibição de produtos, no entretenimento das crianças que acompanhavam os clientes ou como ferramenta de marketing, área em que se destacou, principalmente devido à natureza inovadora do Kinect.

Esta pesquisa fornece amplas possibilidades de extensão, visto que o conhecimento exposto sobre interfaces naturais e o Kinect, pode ser usado nas mais

diversas aplicações que possam vir a ser exploradas futuramente por outros acadêmicos, sejam elas relacionadas a entretenimento, produtividade ou até mesmo robótica e medicina.

REFERÊNCIAS

AZEVEDO, Marcio. Quantas fases tem o ICONIX.
Disponível em: http://oengenheirodesoftware.blogspot.com.br/2010/11/quantas-fases-tem-o-iconix.html Acesso em: 4 jun. 2013.

BARCELOS, Helen; SCHUSTER, Kátia; CORNIANI, Fábio. Vitrinismo: Um meio de comunicar. Universidade Federal do Pampa, Campus São Borja – RS, 2010.

CELEBI, Sait; AYDIN, Ali S; TEMIZ, Talha T.; ARICI, Tarik. Gesture Recognition Using Skeleton Data with Weighted Dynamic Time Warping - Graduate School of Natural and Applied Sciences, Istanbul Sehir University, Istanbul, Turkey, 2011.

GALLIANO, A. Guilherme. O método científico teoria e prática. São Paulo: Harbra, 1986.

GOLDENBERG, Mirian. A arte de pesquisar. Rio de Janeiro: Record, 1999.
Disponível em http://metodologiadapesquisa.blogspot.com
Acesso em: 29 jun. 2013.

KEAN, Sean; HALL, Jonathan; PERRY, Phoenix. Meet The Kinect: An Introduction to Programming Natural User Interfaces.1º ed. Technology in Action, 2011.

PINTO, Anna F. M. Planejamento, estrutura e apresentação do projeto segundo as normas da ABNT. Belo Horizonte. 2010.

ROCHA, Heloisa V; BARANAUKAS, Maria C. Design e availiação de interfaces Humano Computador. Universidade Estadual de Campinas, Campinas, 2003.

SANT'ANNA, Armando. Propaganda: teoria, técnica e prática. 7° ed. São Paulo: homson Learning Edições, 2006.

SILVEIRA, Marcus A. Técnica de Navegação em Documentos Utilizando Microsoft Kinect. Universidade federal do Rio Grande do Sul, Porto Alegre, 2011.

WEBB, Jarrett; ASHLEY, James. Beginning Kinect Programming with the Microsoft Kinect SDK. 1º ed. Apress, 2012.