"TTAIRBUTTON": Biblioteca para Criação de Botões Virtuais com Execução de Ações

Jesimon Barreto Santos[1], Almerindo Nascimento Rehem Neto[2]

[1]Instituto Federal de Sergipe (IFS)

Lagarto-SE-Brasil

[2]Universidade Federal da Bahia (UFBA)

Salvador-BA-Brasil

jesimonbarreto@gmail.com, almerindo.rehem@gmail.com

Abstract. *This paper present a library focused on using virtual buttons that run Java virtual shares or shares in real equipment with the Kinect sensor. Ease and speed for both developers and end users to have goals and priorities to be achieved in the development of this tool. Allow beginners to develop an area with new and unfamiliar technology goals were also achieved.*

Resumo. *Este artigo apresenta uma biblioteca voltada para uso de botões virtuais em Java que executam ações virtuais ou ações em equipamentos reais com o sensor Kinect. Facilidade e velocidade tanto para os desenvolvedores quanto para usuários finais foram objetivos e prioridades a serem alcançados no desenvolvimento dessa ferramenta. Possibilitar iniciantes a desenvolverem para uma área com tecnologia nova e pouco conhecida também foram objetivos alcançados.*

1. Introdução

No inicio não se achava necessária pesquisas em modos de melhorar os equipamentos tecnológicos, até porque existiam apenas usos avançados para cálculos em laboratório [Harms 2004]. Em 1980 a compra de computadores cresceu, sendo usados por pessoas que não tinham conhecimento na área de computação, a partir daí iniciou-se uma grande preocupação com IHC ou interação humano computador [Gasparini 2013]. O IHC é a área que estuda formas de melhorar a relação do homem com computadores por meio de software, de modo que não sejam só funcionais, mas que também fossem práticos e com interfaces gráficas atrativas, esse fator hoje é considerado de extrema importância e sem ele pode ocorrer o abandono do software pelo usuário [Cordeiro 2008].

O IHC é uma área que traz muita preocupação e é explicada pela dependência das outras áreas para facilidade, disponibilidade e agilidade no uso de computadores [De Carvalho 2003], e devido à dificuldade enfrentada pelos usuários finais, o uso do software muitas vezes não acontece com o alcance esperado. Poucas interfaces estão conseguindo alcançar o objetivo de serem de fácil manuseio e agrade aos olhos dos usuários, algo que

traga facilidade de uso, grande comodidade juntamente com boa aparência, irá agradar os usuários. A aceitação da invenção do controle remoto é um grande exemplo de comodidade, controle de equipamentos a certa distancia sem o usuário precisar se mover.

O costume do homem ao acesso e controle de itens de grande porte usando tecnologia com facilidade e sem esforço quase sempre foi direcionado a controles remotos, mais necessariamente controles por botões. Botões podem ter sua ação deduzida facilmente pelo desenho, palavra ou até mesmo letra exposta em si, realizando a ação relacionada a ele com um simples clique. O uso de botões tem grandes qualidades e para inovar associamos com sensor que pode detectar cliques no ar, o *Kinect* [Pheatt 2012].

Após analise de pesquisa e software que pudessem propor uma solução encontrou-se um, desenvolvido pela *Microsoft* [Microsoft Corporation 2011], criado na linguagem *C#,* foi observada uma grande limitação que acaba se tornando um dos nossos objetivos. Iniciantes em programação e maioria dos cursos de programação estudam linguagens atuais e se agradam por nível mais simplificado de programação.

Com isso, tentamos alcançar o objetivo de propor um ambiente atrativo de fácil manuseio para usuários finais, um desenvolvimento rápido, simplificado e em uma linguagem atual para iniciantes. Além disso, definir um padrão de programação nessa área de pesquisa que ainda é muito nova e assim tentarmos interessar pesquisadores a se voltarem pra essa área que tem muito a crescer. Para cumprir os objetivos foi criado foco em códigos menores, desenvolvimento mais rápido, garantia da interação do maior número de usuários finais, de forma a exigir o esforço mínimo no uso, assim, desenvolvendo uma biblioteca para a linguagem *Java* de controle de botões com sensor *Kinect* e ações com suporte para controle de software (programas, aplicativos, jogos) e hardware (Lâmpadas, televisões, portões) alcançamos esse objetivo.

Esse artigo toma o conceito de biblioteca como sendo coleção de subprogramas utilizados para o desenvolvimento de software. Requer noções básicas de programação orientada a objetos em Java.

2. A Biblioteca "TTAirButton"

A biblioteca TTAirButton é uma inovação no que diz respeito a controles remotos, possibilitando controle de vários sistemas sofisticados em um software e que pode ser programado facilmente por estudantes que tem conhecimentos básicos de Orientação a objetos na linguagem *Java*. Traz um padrão de programação para uso de botões virtuais que são acionados pela interação do usuário através do sensor *Kinect*, quando acionados podem executar qualquer ação, ficando a critério do desenvolvedor. A TTAirButton já tem em seu pacote classes que facilitam o uso de arduíno para controle de hardware.

Em relação ao usuário final, a interação acontece de forma divertida e fácil de modo a parecer um jogo, os botões só aparecem na tela do computador e a partir da imagem e da localização da mão dada pelo sensor, o usuário consegue selecionar ou clicar no botão.

2.1. Ambiente para Desenvolvimento da Biblioteca

O ambiente de desenvolvimento escolhido foi o *Netbeans* 7.0, a linguagem *Java*, *Microsoft Kinect,* com sistema operacional *Microsoft Windows* 7. Para o uso do sensor foi necessário a instalação dos drivers da *OpenNI* [OpenNI] e do *NITE* [NITE], esse passo é necessário para todo o serviço de conexão e resgate dos valores do sensor.

Para usar os valores das juntas do usuário dadas pelo sensor mesmo usando as bibliotecas da *OpenNI* e do *NITE*, é difícil a programação e é grande a quantidade de linhas de código para uso do *Kinect,* então esse problema foi resolvido com o uso do framework que trata os valores dos sensores como camadas [Renhem Neto 2013], facilita e padroniza a programação.

O *Kinect* é usado por obter sensores que possibilitam a detecção das articulações humanas, assim como mãos, em três dimensões ou 3D, além do sensor de detecção possui uma câmera colorida chamada de sensor *RGB*, usado para que o usuário possa se enxergar na tela e saber com qual botão está interagindo.

2.2. Alguns Casos para uso da Biblioteca ou softwares produzidos pela mesma

Uso de alguns recursos do computador é cansativo e não oferece nenhuma interação atrativa ao usuário, como por exemplo, leitura de um arquivo de texto. Usando a biblioteca com conceitos básicos de orientação a objetos em linguagem de programação Java, com pouco tempo e com pouco esforço teria um software para leitura de arquivos de texto, podemos abrir (botão1), passar (botão2), ou voltar (botão3) a página, arquivo de texto com clique no ar. A interação de usuários que não são acostumados com tamanha quantidade de teclas e comandos existentes no mouse e teclado acontecerá de forma fácil, com poucas ações e botões.

Outro caso é o controle de eletrodomésticos que já são controlados por botões em controles remotos, como televisões, lâmpadas, entre outros equipamentos automatizados, com a vantagem de interação natural e o controle de todos esses equipamentos em um único software.

3. Arquitetura da Biblioteca

A arquitetura da biblioteca funciona baseada em duas camadas, camada visual e camada de ação. As camadas estão ligadas por um canal baseado em troca de mensagens a todo o momento, esse controlador indica o modo que se encontra o botão, sendo possível assim manter as camadas em constante sincronia. A camada visual aguarda interação do usuário, que é detectado com os valores do *Kinect*. Essa arquitetura é demonstrada mais claramente na figura 1 e no decorrer desse artigo com exemplo de uso.

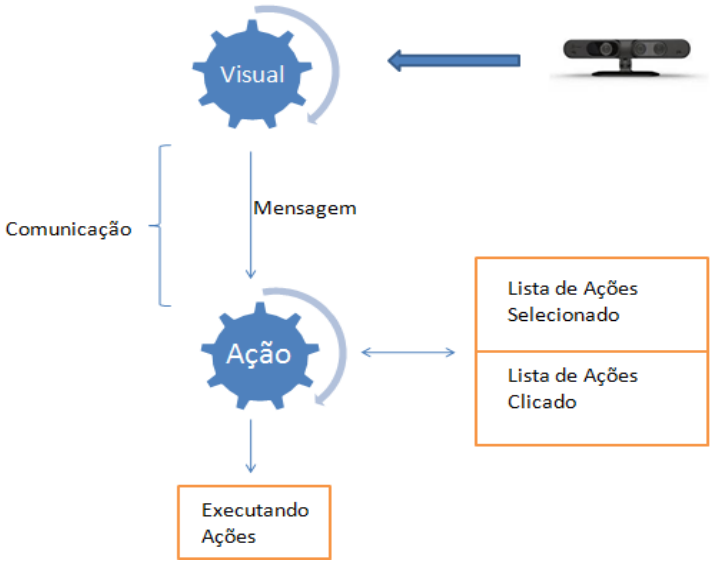

Figura 1. Diagrama da arquitetura das camadas e comunicação do TTAirButton

3.1. Camada Visual

A camada visual apesar de se chamar assim, também atua com vários cálculos antes de enviar a mensagem para a próxima camada.

Após a conexão com o *Kinect,* essa camada busca além da imagem da câmera *RGB*, os valores de cada dimensão das mãos do usuário. Essa camada ainda tem a função de sincronismo na exibição do botão, que usa imagens diferentes para cada modo. A imagem e os valores salvos são separados e tratados da sua devida forma.

A imagem é sempre posta somente para exibição, e servi de referencia para o que o usuário saiba onde se encontram as mãos e com qual botão ele deseja interagir.

Antes dos botões serem adicionados na camada para uso, precisam de três imagens e indicar as informações do botão. As imagens devem ser selecionadas, renomeadas 'Nx, Sx, Cx' onde 'x' é o nome dado a identificação do botão, compactadas para o formato "zip" e copiadas para o diretório dessa biblioteca (C://TTAirButton) criado após a primeira execução.

As informações do botão são elas: Ponto inicial, ponto final e identificação. Ponto inicial é reconhecido pelos valores da posição '0' nessa biblioteca, e são os valores na horizontal e vertical onde se iniciam as dimensões dos botões (X_o, Y_o). Ponto final tem por representação os valores da posição '1' nessa biblioteca, e indicam o tamanho que terão as dimensões do botão (X_t, Y_t). A identificação é uma representação em forma de variável *String* que é usada para organização das imagens que representam os modos do botão, e na mensagem que será enviada para a camada de ação, essa mensagem estará a indicar qual botão sofreu alteração e em qual modo se encontra, veja mais detalhes na seção 3.2. Nos botões foram usados apenas (X, Y), mas para detecção do clique é necessário o uso da distancia ou (Z).

O *Kinect* foi usado para possibilitar o uso dos valores da localização em três dimensões (X, Y, Z) das juntas do usuário, neste caso só foram usados os valores das mãos e do pescoço (um referencial do corpo para detecção do clique). Portanto, com o valor das mãos na horizontal e na vertical (X, Y), é verificado se uma das mãos encontra-se dentro de algum botão, se confirmado neste instante o botão passará para o modo "selecionado". A diferença básica entre selecionado e clicado é a distancia entre determinado ponto de referencia, esse ponto de referencia é o pescoço e irá servir do que seria intensidade de força em um botão físico, neste caso será a distância, quanto mais longe do pescoço, mais perto do *Kinect* maior a intensidade. O sensor sendo usado como único referencial tornaria a biblioteca muito propensa a erro, ou melhor, era obrigatório o usuário sempre estar na mesma posição todas às vezes de uso, sabemos que isso é quase impossível.

O uso da compactação das imagens de um mesmo botão para o formato "zip", garante a organização de todo o projeto. A biblioteca tem seu próprio diretório (C:/TTAirButton), criado no inicio da primeira execução, o diretório é criado mesmo sem nenhum botão adicionado. Esses são pequenos pontos que, em grandes projetos fazem diferença. Na figura 2 encontra-se um exemplo do resultado de uso de um projeto com apenas um botão e as linhas de códigos necessárias para chegar a esse resultado estão descritas e exibidas na figura 3.

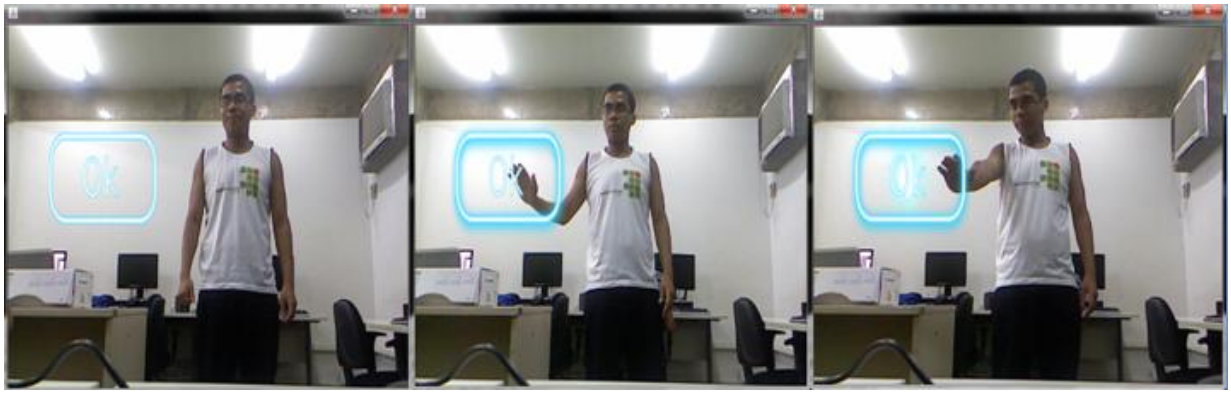

Figura 2. Resultado visual do uso da biblioteca TTAirButton

```
18          //Criando um botao
19          TTAirButton ok = new TTAirButton();
20          /*Usa-se o método setPoint(int,int,0 ou 1) é a localização do botão
21          em relação ao painel. Como último parâmetro quando "0" indica posição
22          inicial, e quando "1" indica o tamanho*/
23          ok.setPoint(50, 100, 0);
24          ok.setPoint(250, 300, 1);
25          /*O método setIdentification(String) é usado para indicar o nome do
26          botão, mesmo nome usado  no arquivo .zip*/
27          ok.setIdentification("ok");
28          /*Criando um Painel gerenciador dos botões*/
29          MyScreenPanel msp = new MyScreenPanel();
30          /*Adicionando botão ao painel*/
31          msp.AddButton(ok);
32          /*Uso da API para uso do kinect. Adicionando a imagem colorida
33          da câmera do kinect.
34          */
35          ALayerShape rgb = null;
36          try {
37              rgb = new LayerRGB();
38          } catch (Exception erro) {
39              System.out.println("Erro ao Adicionar o imagem da camera do kinect");
40          }
41          /*Adicionando imagem colorida ao Painel*/
42          msp.addLayerShape(rgb);
43          JFrame frame = new JFrame();
44          frame.setSize(640,480);
45          /*Adicionando Painel ao Frame*/
46          frame.add(msp);
47          frame.setVisible(true);
```

Figura 3. Código necessário para usar a camada Visual da TTAirButton

Toda a arquitetura de programação que foi desenvolvida essa biblioteca, exibição das imagens e uso de painel seguem o modelo do Touching The Air Framework [Rehem Neto 2013].

3.2. Comunicação

A comunicação que também pode ser chamada de camada de comunicação é apenas o canal entre as camadas e a mensagem enviada da camada visual para a camada de ação.

O canal entre as camadas é o padrão de envio pela camada visual e o recebimento pela camada de ação. Esse padrão, chamado de protocolo de comunicação entre camadas, é formado por duas regras, a camada visual sempre envia a mensagem, mesmo que não tenha acontecido nenhuma interação o que garante a sincronização em tempo real da camada de ação em relação à camada visual.

A mensagem enviada é composta por duas informações, identificação do botão e o modo em que se encontra o botão. A identificação usada é o valor *String* cadastrado em linhas de código na camada visual. O modo é detectado pela própria camada, se não houve interação do usuário, se o usuário interagiu parcialmente ou se o usuário interagiu. Exposto na tabela 1 o valor enviado de acordo com o modo.

Tabela 1. Valores da variável do modo do botão para cada interação que acontece na camada visual

MODO	VALOR

Sem interação	“simInteração”
Interação Pacial	“selecionado”
Interação ou Clique	“clicado”

3.3. Camada de Ação

A camada de ação assim como a camada visual está sempre ativa e pode ser dividida em mais de uma função, é responsável por armazenar as ações e por executá-las.

O armazenamento das ações é feito em listas representadas pela identificação do botão. Quando criamos um botão na camada visual, são criadas duas listas para ele que tem como referencia o valor de sua identificação, uma das listas é usado quando o botão entrar no modo selecionado e outra quando for clicado.

Quando essa camada recebe a mensagem, verifica primeiro o valor do modo, se for “semInteração” é passado pra próxima mensagem, mas se tiver com outro valor a camada busca a identificação do botão, salva a lista de ações que tem por representação a identificação do botão e executa toda a lista de ações. Nessa versão não tem opção de escolha, caso o usuário mantenha por muito tempo o mesmo modo não será executada a mesma lista de ações várias vezes.

Tem um modelo de ação para que seja adicionada a lista de ações, esse modelo pode ser de dois tipos, ações virtuais ou ações reais. Esses tipos foram criados para facilitar o uso da biblioteca, para ação virtual quer dizer que a ação está relacionada com software. A ação real tem métodos que são necessários para conexão com hardware, mais necessariamente, com o arduíno. As ações precisam de uma informação antes de ser adicionada a camada, é a qual lista será adicionada a ação, para o modo selecionado ou clicado.

Na figura 4 temos um exemplo de uma classe do tipo ação virtual, essa ação exibe a mensagem “Botão Ok- Selecionado”

```
public class AcaoSelecaoBotaoOK extends VirtualAction{


    @Override
    public void run() {
        System.out.println("Botão Ok- Selecionado ");
    }

}
```

Figura 4. Exemplo de código necessário para criar uma ação virtual

Para garantir a associação da ação com o botão, o método para adicionar requer o botão a que pertencerá à ação. A figura 5 mostra um exemplo do código para criar ação e adicioná-la a camada e tornar pronto para uso. O exemplo de ação segue o exemplo inicial da camada visual.

```
/*Criando acao*/
AcaoSelecaoBotaoOK acao=new AcaoSelecaoBotaoOK();
/*É usado para indicar em qual lista do modo será adicionado,
Clicked ou Selected*/
acao.setStateAction("Selected");
/*Adicionando o ação e relacionando com o botão*/
msp.addActionButton(ok, acao);
```

Figura 5. Exemplo de código para Associar uma ação a um botão

O código acima deve ser colocado depois da linha que adiciona o botão ao painel, para que o painel reconheça o botão quando for feita a associação.

4. Usuários Finais

O Uso de botões para usuários finais torna-o mais interativo pelo uso das imagens, um dos problemas que podem acontecer no uso de imagens é colocar imagens de difícil compreensão [Saúde 2012]. Problema que é evitado por usar imagens não fixadas (possível alterar as imagens mudando o arquivo no diretório), o que torna o software livre para usar imagens de qualquer natureza e associá-la a ação que ache melhor.

5. Conclusão

A TTAirButton usa ferramentas da atualidade para encontrar uma forma fácil, com velocidade e melhor interação do usuário final ajudando em outras áreas de conhecimento. Além de propor aos desenvolvedores um padrão de desenvolvimento em uma área nova de pesquisa. É uma boa forma para controle de usuários finais, uma melhor visualização e melhor qualidade de uso em *players* mp3, visualizador de imagens, controladores de equipamentos eletrônicos, teclados virtuais todos com diversão.

Uma biblioteca com foco em botões traz a variação e também novas soluções para a IHC. Possibilita com um simples passar de mão pela imagem ou clique ter todo o controle de um software ou hardware, o que torna seu computador um controle remoto universal. Entre as principais metas atingidas está compactação de códigos, com intuito de restringir ao mínimo o uso por iniciantes.

Referências

Bobeth, J. et al. (2012) "Evaluating Performance and Acceptance of Older Adults Using Freehand Gestures for TV Menu Control". EuroiTV'12. p.35-44.

Cordeiro, R.;de Oliveira, M.R.; Chanquini, T. P. (2008) "Utilização de conceitos de interface homem-máquina para adaptação da disciplina de requisitos do RUP" <http://www.centropaulasouza.sp.gov.br/pos-graduacao/workshop-de-pos-graduacao-e-pesquisa/anais/2009/trabalhos/gestao-e-desenvolvimento-de-tecnologias-da-informacao-aplicadas/trabalhos-completos/CORDEIRO,%20Renato.pdf> acesso em 29/03/2014.

Darvison, A. (2012) "Kinect Open Source Programming Secrets", http://fivedots.coe.psu.ac.th/~ad/kinect/, Setembro.

De Carvalho, J. O. F. (2003) "O papel da interação humano-computador na inclusão digital". <http://periodicos.puc-campinas.edu.br/seer/index.php/transinfo/article/viewFile/1461/1435> acesso em 29/03/2014.

Draper, G.M. et al. (2009) "A History of Computing Course with a Technical Focus". In: Proc. SIGCSE '09, ACM Press, p. 458-462.

Gasparini, I.; Kimura, M. H.; Pimenta, M. S.(2013) "Visualizando 15 anos de IHC". In IHC '13 Proceedings of the 12th Brazilian Symposium on Human Factors in Computing Systems, p.238-247.

Harms, D. E. (2004) "A Virtual Reality Simulator of the ENIAC". ITiCSE '04 Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, p.239-239.

Microsoft Corporation. (2011) "Kinect SDK for Windows – Hover Button / Hover Control". < http://blogs.msdn.com/b/tess/archive/2011/08/16/kinect-sdk-for-windows-hover-button-hover-control.aspx?Redirected=true> acesso em 24/04/2014

Moniruzzaman, B. (2011) "A Gesture Controlled User Interface for Inclusive Design and Evaluative Study of Its Usability". JSEA, Vol.4 No.9, p.513-521.

NITE. http://www.openni.org/files/nite/.

OpenNI. SDK. http://www.openni.org/.

Pheatt, C.; McMullen, J. (2012) Programming for the Xbox Kinect™ sensor. Journal of Computing Sciences in Colleges Archive,Vol. 27 Issue 5, p.140-141.

Rehem Neto, A; Aragão, L.; Santos, C.A.S.; Canuto, C. (2013) "Touch The Air: Um Framework Orientado a Eventos para Ambientes Interativos". WebMedia '13, Proceedings of the 19th Brazilian symposium on Multimedia and the web, p.11-12.

Rehem Neto, A.; Santos, C.A.S.; Andrade, M.V. R. (2011) "Interfaces para Aplicações de Interação Natural Baseadas na API OpenNI e na Plataforma Kinect". XVII Webmedia / XXVI SBBD. 1ed.Florianópolis, Brazil: SBC, v. 1, p. 35-60.

Saúde, L. M. S.; Kirner, C. (2012) "Facilitando o Entendimento do Conceito de Perspectiva através de Realidade Aumentada". <http://www.lbd.dcc.ufmg.br/colecoes/wrva/2012/0019.pdf> acesso em 29/03/2014.