UNIVERSIDADE DE CAXIAS DO SUL

Centro de Computação e Tecnologia da Informação

Curso de Bacharelado em Tecnologias Digitais

Andriel Giovanella

SISTEMA DE LOCALIZAÇÃO BASEADO EM INTERAÇÃO GESTUAL

CAXIAS DO SUL

19/11/2012

Andriel Giovanella

SISTEMA DE LOCALIZAÇÃO BASEADO EM INTERAÇÃO GESTUAL

Trabalho de Conclusão de Curso para obtenção do Grau de Bacharel em Tecnologias Digitais da Universidade de Caxias do Sul.

Carlos Eduardo Nery

Orientador

CAXIAS DO SUL

19/11/2012

RESUMO

Nesse trabalho são abordadas características da interação humano-computador, visando criar um sistema de interação não-convencional. Para tal é apresentado um estudo em reconhecimento gestual e de padrões. O sistema a ser desenvolvido será criado no Adobe Flash, que também é estudado no trabalho a seguir.

Palavras-chave: Interação Humano-Computador. Reconhecimento de Padrões. Reconhecimento Gestual.

ABSTRACT

This paper discusses the characteristics of human-computer interaction, to create a system of interaction unconventional. For such, is presented a study in gestural recognition and standards. The system to be developed will be created in Adobe Flash, which is also studied in the work below.

Keywords: Human-Computer Interaction. Pattern Recognition. Gesture Recognition.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A interação humano-computador trata da troca de informações entre o usuário e o computador. No princípio, o controle de máquinas era visto como algo complexo, onde a operação poderia ser feita, na maioria dos casos, apenas por especialistas. Avanços nessa área fizeram com que as pessoas passassem a se comunicar com as máquinas de maneira cada vez mais simples e intuitiva.

Atualmente, são encontrados os mais diversos dispositivos e formas de se operar uma máquina. Teclados, mouses, telas *touchscreen* e joysticks são elementos comuns no ramo da tecnologia.

A evolução da interação também é observada na área do entretenimento, onde se tem jogos empregando avançadas tecnologias visando maior imersão e realismo. O modo como o usuário joga tem se alterado radicalmente, como pode ser visto em consoles como o Nintendo Wii e dispositivos como o Kinect, que fazem com que o jogador atue no mundo real praticamente da mesma forma que seu personagem atua no mundo virtual.

Os estudos que resultaram nas alterações do modo como as pessoas interagem com o computador, surgiram anos antes das tecnologias atuais. Grande parte desses estudos é o de reconhecimento de padrões, empregados no reconhecimento dos gestos e movimentos que tornam essa interação possível.

O presente trabalho visa o desenvolvimento de um aplicativo que identifique movimentos realizados diante de uma câmera conectada ao computador. Esses movimentos irão interagir com o computador e, dessa forma, controlar opções e funções de um aplicativo de busca e localização a ser desenvolvido. Para a aplicação do modelo de interação proposto, será criado um sistema de busca e localização com base nas instalações da Cidade Universitária da UCS. Assim, os pontos de referência serão as principais referências dentro da universidade, como blocos, biblioteca, ginásios e demais prédios de maior importância.

No trabalho são tratados aspectos da interação humano-computador, com um histórico da mesma, mostrando os passos realizados desde a linha de comando até o Kinect; o reconhecimento de padrões, abordando o reconhecimento gestual adotado por tecnologias como o Kinect; e o Adobe Flash, que pode suportar a interação por gestos, se utilizando de sua linguagem de programação, o ActionScript.

2 INTERAÇÃO HUMANO-COMPUTADOR

No século XVIII, com a criação da máquina a vapor, iniciou-se a relação entre homem e máquina. Relação que foi se aproximando com o passar do tempo, e se tornando cada vez mais intuitiva. Desde as primeiras máquinas a vapor até a criação do computador, passaram-se quase cinco décadas, onde o modo de operação dos mais diferentes inventos foi otimizado. Como o computador demonstrava seu potencial para diversas tarefas, a forma de trabalhar com o mesmo teve grande destaque.

Quando interage com o computador, o usuário passa informação para o mesmo que responde à pessoa com outros dados. O processo de transferência de informações, seja do usuário para o computador ou o oposto, é denominado interação humano-computador (DIX et al., 1998).

A interação com o computador inicialmente era tratada como ”uma sequência de estímulos e respostas, como na interação de corpos físicos” (BARBOSA; DA SILVA, 2010, p. 20). Com o passar do tempo, o conceito foi repensado, tratando a interação humano-computador como “a comunicação com máquinas, em vez de a operação de máquinas” (CARD; MORAN; NEWELL, 1983 apud BARBOSA; DA SILVA, 2010, p. 20). Para resultar em uma interação satisfatória com o computador, o usuário deve interagir como um sistema, de forma exata e padronizada. Dessa forma a interação pode ser considerada uma analogia à transmissão de dados (BARBOSA; DA SILVA, 2010).

O estudo da interação humano-computador tomou forma no início da década de 1980, e na década de 1990 já existiam grandes aprofundamentos e bases teóricas na área, tornando necessários designers de interação para a criação de novos produtos, além dos engenheiros de software (BENYON, 2011).

2.1 INTERFACES E SEUS DISPOSITIVOS DE ENTRADA

As tarefas de um computador são realizadas através de uma linguagem binária, ou seja, 0 ou 1, diferente dos humanos que raciocinam com palavras, ideias, e imagens. A interface torna possível a interação entre ambos (JOHNSON, 1997). Por ser o único meio em que o usuário tem contato com o sistema, a interface por vezes é confundida com o sistema em si (BARBOSA; DA SILVA, 2010).

Com a evolução do *hardware* nos computadores, a interface ficou cada vez mais próxima do usuário, que a toca e a sente (BENYON, 2011), como nas telas *touchscreen* e dispositivos com *feedback* de força.

Para que o usuário possa agir na interface de um determinado aplicativo, o computador dispõe de vários dispositivos nos quais a pessoa pode inserir informações e outros que permitem que ele visualize as respostas do sistema. Dispositivos de saída, como impressoras, monitores e alto-falantes, são os que respondem aos comandos do usuário. Já os dispositivos de entrada são os que permitem a inserção de informação, como o teclado e o mouse, por exemplo (BARBOSA; DA SILVA, 2010).

2.1.1 Interface de linha de comando

Os primeiros computadores a serem comercializados somente possuíam interface de linha de comando, onde, para cada ação a ser desenvolvida, o usuário deveria conhecer os comandos desejados para assim poder operar a máquina. Isso tornava as máquinas muito complexas, limitando o número de usuários capazes de as operarem. Esses computadores contavam apenas o teclado como dispositivo de entrada de dados.

2.1.1.1 Teclado

A história por trás do teclado tem início muito antes da criação dos computadores. Na década de 1870, o modelo de máquina de escrever desenvolvido por Christopher Latham Shole se tornou um sucesso, pois, devido ao

posicionamento das letras, fez com que os usuários digitassem com índice de erros menor (SHNEIDERMAN, 1992). O modelo foi chamado de QWERTY, devido à posição das primeiras seis teclas do teclado (DIX et al., 1998). O modelo de teclado feito por Shole se tornou padrão durante todo o século posterior.

O desenvolvimento de teclados eletrônicos eliminou os problemas mecânicos de seus antecessores, o que fez com que os inventores do século XX propusessem layouts alternativos, que diminuíssem a distância entre os dedos (MONTGOMERY, 1982 apud SHNEIDERMAN, 1992). O layout Dvorak, desenvolvido na década de 1920, supostamente reduziu a distância entre os dedos, aumentando a capacidade dos datilógrafos de 150 palavras por minuto para 200 palavras por minuto, enquanto diminuía os erros. Mesmo assim, a aceitação do Dvorak foi baixa (SHNEIDERMAN, 1992).

Um terceiro modelo de teclado foi desenvolvido pensando nos usuários que não estavam acostumados a digitar. O modelo ABCDE tornava mais fácil a busca pelas letras, visto que as mesmas estavam em ordem alfabética (SHNEIDERMAN, 1992). Estudos apontaram que o teclado em ordem alfabética não torna mais ágil a digitação para datilógrafos treinados, somente resultando em melhor resposta em usuários ocasionais ou novatos (DIX et al., 1998).

O primeiro teclado da IBM adotou o modelo QWERTY e inseriu novas teclas como Enter, Shift, entre outras teclas de funções. Com o tempo, além das teclas, foram inseridos LEDs para indicar o status das teclas de CAPSLOCK, NUMLOCK e SCROLL LOCK. Porém alguns dispositivos adotavam modelos reduzidos de teclado, eliminando algumas teclas inseridas no modelo convencional (SHNEIDERMAN, 1992).

2.1.2 Interface Gráfica

A interface de linha de comando aos poucos foi sendo substituída pela interface gráfica do usuário, ou GUI, onde as informações se apresentavam em forma de imagens e ícones. Os estudos que desenvolveram tal interface tiveram

início na década de 1960, liderados por Douglas Engelbart, os quais buscavam uma forma mais simplificada para a interação humano-computador. No outono de 1968, no San Francisco Civic Auditorium, Engelbart apresentou uma série de metáforas visuais que poderiam proporcionar uma manipulação direta sobre as tarefas realizadas no computador, e assim substituir a complicada sintaxe utilizada pela linha de comando das máquinas da época (JOHNSON, 1997). Ao adotar uma interface que substituía as linhas de comando por ícones e símbolos, a interação entre humanos e computadores tornou-se mais simples. As tarefas que os computadores realizavam poderiam ser comparadas com as que eram feitas em máquinas de escrever e calculadoras; os arquivos salvos, com documentos, podendo ser organizados em pastas separadas; assim surgiu a metáfora da área de trabalho, ou desktop (JOHNSON, 1997).

Os estudos iniciados por Engelbart tiveram continuidade com a Xerox, que aprimorou o que havia sido desenvolvido e aplicou em seu Xerox Alto, computador que, apesar de ter sido desenvolvido para fabricação em massa, nunca chegou a ser comercializado. A ideia da concepção do Alto veio da necessidade de criar um computador funcional para escritórios, mas que esse não ocupasse muito espaço como os concorrentes da época e que fosse possível operá-lo sentado como se estivesse trabalhando em uma mesa (WADLOW, 1981).

Outro resultado dos estudos de Douglas Engelbart foi a invenção do mouse, primeiro dispositivo utilizado para escolher as opções disponíveis na interface gráfica desenvolvida. O primeiro computador pessoal a possuir mouse e interface gráfica foi o Lisa, da Apple, lançado em 1983. O fundamento principal para sua criação era tornar mais simples e intuitiva a operação do computador. Baseando-se nas teorias de Engelbart, as funções deixaram de ser chamadas através de códigos e passaram a ser indicadas por elementos gráficos, chamados de ícones. Boa parte desses ícones foi baseada em objetos encontrados em mesas de trabalho de escritório, daí surgindo o nome de Desktop para os computadores de mesa, mesma denominação utilizada para descrever a tela inicial, também conhecida como Área de Trabalho. Para indicar os itens na tela, o dispositivo agora utilizado seria o mouse. Boa parte

dos artefatos usados no desenvolvimento do Lisa já haviam sido criados anteriormente e a maioria vinha dos conceitos abordados por Engelbart anos antes. Pode-se dizer que os engenheiros que criaram o Lisa apenas juntaram ferramentas e as utilizaram em uma interface mais amigável e dinâmica (HORMBY, 2005).

Apesar de todos os avanços, o Lisa não se tornou muito popular pelo seu preço elevado, cerca de dez mil dólares (HORMBY, 2005). Apesar do seu fracasso, o seu sucessor, o Macintosh, obteve grande sucesso, sendo considerado um dos principais responsáveis pela popularização da interface gráfica da forma como ainda utilizamos até hoje.

A popularização dos computadores foi muito influenciada pela interface gráfica que simplificou a operação dessas máquinas. O raciocínio do usuário antes usado principalmente para memorizar comandos, pôde se voltar para outras tarefas. Erros, antes proporcionados principalmente por comandos digitados de forma incorreta, foram anulados. Dessa forma desempenho, aprendizagem e satisfação do usuário se aprimoraram. Boa parte desses avanços veio graças à aplicação de dispositivos em que o usuário pudesse apontar o que deseja.

2.1.2.1 Dispositivos de Apontamento

Dispositivos de apontamento são aplicáveis em seis tipos de interação:

- Selecionar: para escolher determinados itens;
- Posicionar: para desenhar, deslocar blocos;
- Orientar: indicar direção, mover;
- Traçar caminho: para desenhar linhas curvas, traçar rotas em mapas;
- Quantificar: especificar um valor numérico para escolher o número de uma página, volume, velocidade em simulações;
- Texto: indicar onde alterar o texto, mover, editar, alterar fonte.

Todas essas funções eram originalmente realizadas somente através do teclado e podem continuar sendo feitas dessa forma. Entretanto, o uso de um dispositivo de apontamento torna a realização de tais tarefas mais rápida e com menor índice de erros.

Os dispositivos de apontamento podem ser agrupados em dispositivos que oferecem controle direto, como telas *touchscreen* e *lightpen*, e os que oferecem controle indireto, tais como o mouse, *trackball* e *joysticks*.

2.1.2.1.1 Dispositivos de Apontamento Direto

Os dispositivos de apontamento direto são aqueles que o usuário pode indicar o que deseja diretamente sobre a tela do computador. Os principais dispositivos desse tipo são:

Lightpen: foi o primeiro dispositivo que possibilitou realizar as seis funções de interação de apontamento diretamente na tela. Existem *lightpens* de diversos tamanhos, pesos e formatos, sendo que a maioria possui um botão para indicar quando o cursor deve ser pressionado (SHNEIDERMAN, 1992). É conectada por um cabo e, quando acionada, um facho de luz é lançado na tela que o detecta. Dessa forma, a *lightpen* possui uma maior precisão em relação às telas *touchscreen* (DIX et al., 1998). Existem três desvantagens no uso da *lightpen*: a mão do usuário oculta parte da tela, a mão se afasta muito do teclado para utilizá-la e o usuário deve pegar a caneta e se aproximar da tela para poder usá-la (SHNEIDERMAN, 1992).

Touchscreen: diferente da *lightpen*, as telas *touchscreen* não precisam de dispositivos para alterar itens na tela, sendo que o usuário pode indicar diretamente com o dedo o que deseja alterar. As telas sensíveis ao toque começaram a ser desenvolvidas na segunda metade da década de 1960, sendo que diversas técnicas foram criadas. O primeiro computador com esse tipo de tela, o PLATO IV, que começou a ser utilizado em 1972, adotava uma forma pioneira do infravermelho para captar a posição do objeto em contato com a tela e sua finalidade era a educação. Todavia, o primeiro computador com tela *touchscreen* a ser comercializado foi o HP-150, lançado em 1983. (BUXTON, 2007). A tela *touchscreen* continuou com os

problemas de esconder parte da tela com a mão do usuário e afastar a mão do teclado, já enfrentados pela *lightpen*, além de ter apontamento por vezes impreciso e sujar o monitor. As telas *touchscreen* têm sido utilizadas frequentemente em espaços públicos devido à ausência de elementos móveis e o seu custo relativamente baixo (SHNEIDERMAN, 1992). Existem telas *touchscreen* que distinguem somente um toque e as *multitouch*, que reconhecem toques simultâneos (BENYON, 2011). Além disso, vários dispositivos como celulares e *notebooks* se utilizam dessa tecnologia, eliminando a necessidade do teclado.

Stylus: a criação dos *notebooks* estimulou o interesse por dispositivos em formato de caneta, semelhantes à *lightpen*, para que o usuário pudesse escrever e desenhar à mão diretamente em seu computador. Dessa forma, foi desenvolvida a caneta *stylus*, basicamente possuindo as mesmas funções da *lightpen*. O formato do *notebook* permite que o usuário descanse a mão ao utilizar a *stylus*, de uma maneira semelhante a que se utiliza uma caneta em um caderno, por exemplo. Assim, pôde superar um dos problemas enfrentados pelas primeiras *touchscreens*, a fadiga causada ao usuário ao fazê-lo escrever em uma tela na vertical. Apesar dos algoritmos que possibilitaram o reconhecimento da escrita à mão, a escrita pelo teclado ainda é preferida em diversas situações.

2.1.2.1.2 Dispositivos de Apontamento Indireto

Esses dispositivos não possuem os problemas de fadiga manual e obscurecimento da tela pela mão, causados pelas primeiras telas *touchscreens* e *lightpens*. Outras adversidades permanecem, como o afastamento da mão do teclado e o ato de ter que pegar outros dispositivos. Por outro lado, os dispositivos de apontamento indireto necessitam de maior coordenação para indicar o ponto que deseja com o cursor na tela. Alguns dos principais dispositivos são:

Mouse: se tornou atrativo pela forma em que é utilizado, com o braço e a mão descansando sobre a mesa enquanto utiliza o mesmo, além de proporcionar um apontamento preciso (SHNEIDERMAN, 1992). Desenvolvido por volta de 1964 por Douglas Engelbart, originalmente possuía duas rodas para mover seu cursor nas

coordenadas x e y (DIX et al., 1998). Foram desenvolvidos vários modelos de mouses, com vários botões, tamanhos e pesos, fazendo com que o usuário pudesse escolher o que melhor se adaptasse às suas tarefas (SHNEIDERMAN, 1992). Dentre os dispositivos de apontamento indireto é o de maior sucesso e o mais utilizado em computadores (DIX,1998).

Trackball: a *trackball* fica imóvel na mesa, como o teclado, ou seja, diferente do mouse, não precisa de um espaço maior para sua manipulação (DIX,1998). Possui uma bola em sua superfície, que ao ser manipulada faz com que o cursor da tela se mova (SHNEIDERMAN, 1992). Usuários com deficiência motora ou problemas de lesão por esforço repetitivo – LER – costumam adotar a *trackball* ao invés do mouse, pelo menor índice de movimento necessário para operá-la. A maior dificuldade quanto à usabilidade da *trackball* é para desenhar ou fazer movimentos longos (DIX,1998).

Mesa digitalizadora: também conhecida como *graphicstablet* ou tablete gráfico, a mesa digitalizadora é uma superfície plana sensitiva ao toque, usada deitada sobre a mesa, fazendo com que o usuário não trabalhe diretamente no monitor. Geralmente utilizada para desenhar, é atrativa por não causar tanta fadiga manual nem ocultar a tela, diferente das telas *touchscreen* e *multitouch*. Podem ser operadas com o dedo, lápis, caneta ou qualquer outro instrumento (SHNEIDERMAN, 1992).

Touchpad: foi usado extensivamente nos computadores portáteis da Apple e são encontrados na maioria dos *notebooks*. Substitui o mouse e pode ser operado movendo um dedo sobre a superfície, de forma semelhante a *trackball*. O maior problema encontrado no uso do *touchpad* é quando o usuário pressiona com uma força um pouco maior fazendo com que o sistema considere o impulso como um clique (DIX,1998).

Eyegaze: consiste em um aparelho montado sobre a cabeça do usuário que dispara um laser de baixo poder sobre o olho, que reflete sobre a retina, identificando a direção do olhar e movendo assim o cursor. Aplicações militares para

indicação de alvos e usuários impossibilitados de usar suas mãos são os maiores beneficiados pelo uso desse dispositivo. É um sistema preciso, porém devido ao seu alto custo, não é muito disseminado (DIX,1998).

2.2 VIDEOGAMES

Em paralelo à evolução dos dispositivos de interação humano-computador, os videogames também influenciaram no modo como as pessoas se comunicam com as máquinas. Ao observar o histórico dos consoles domésticos de videogame, pode-se notar que os dispositivos de interação evoluíram juntamente com a popularização dos próprios consoles.

O Odyssey, lançado em 1972, e o Pong, de 1974, são considerados consoles da primeira geração, possuindo controladores limitados, que só permitiam alterar a posição de um item na tela. A segunda geração de consoles comporta o Odyssey² e o Atari 2600, onde ambos possuem um controlador para a posição e um único botão para ação. Da terceira a sexta geração de consoles, os controladores não sofreram muitas alterações que não fossem a adição de novos botões de ação. Dessas gerações podemos citar como principais consoles: o NES, de 1983, e o Master System, de 1985, da terceira geração de consoles; Mega Drive, de 1988, SNES, de 1990, da quarta geração; Playstation, de 1994, Nintendo 64, de 1996, da quinta geração; Dreamcast, de 1998, Playstation 2, de 2000, Xbox, de 2001, da sexta geração.

A sétima geração de consoles foi a que teve a maior revolução em seus controladores. O PlayStation 3, de 2006, manteve o padrão dos controladores das gerações anteriores. Diferente de seu concorrente, o Nintendo Wii, do mesmo ano, já possuía um avanço significativo em relação ao modo de interação entre o usuário e o jogo. Seu controlador se assemelha a um controle remoto de uma televisão convencional, porém o mesmo possui acelerômetros e giroscópios que possibilitam identificar ações como inclinação, rotação e aceleração em qualquer direção (BRAIN, 2005). Dessa forma, ao invés do jogador usar um botão para indicar a posição, ele move o controle para o lado em que deseja ir, por exemplo. O controle

também pode assumir função de uma arma em um jogo de tiro, apontando-o para mirar e atirar de uma forma mais natural. Em outros casos, ele pode simular uma vara de pesca, baquetas de bateria, até mesmo luvas de boxe, fazendo com que a luta simulada pelo Nintendo Wii seja interpretada fora da tela da televisão, tornando-a mais interativa e realista.

2.3 KINECT E SEMELHANTES

Um dos maiores avanços em matéria de interação homem-máquina veio com o lançamento do Kinect. Visando competir a altura com o Nintendo Wii, a Microsoft, em parceria com a PrimeSense, lançou o Projeto Natal. O nome faz referência à cidade brasileira, pois Alex Kipman, um dos líderes do projeto, é brasileiro e achou interessante homenagear seu país, além de que a palavra “natal” em latim significa nascer, representando bem o conceito por trás do projeto. A liderança do mesmo era dividida entre o próprio Alex Kipman com Kudo Tsunoda, Andrew Blake e Jamie Shotton.

O projeto buscava uma forma de eliminar os controladores convencionais através de rastreamento de movimento do jogador diante do console. O sistema para o console Xbox 360 utiliza-se de um sensor de profundidade, uma espécie de câmera baseada em sinais infravermelhos; uma câmera de vídeo, para reconhecer o rosto do jogador e capturar imagens; e um microfone para reconhecimento de voz, que ajuda a diferenciar jogadores presentes no mesmo ambiente.

A Microsoft indica que o usuário deve ficar de 1,8m a 2,4m de distância do sensor para poder ser reconhecido e interagir com o mesmo. Com as informações capturadas pela câmera infravermelha, o sistema divide o corpo humano em 48 pontos de articulação, assim gera uma espécie de gráfico em 3D que indica a posição de cada parte do corpo do jogador. O Kinect consegue processar 30 frames por segundo, tornando o movimento mais suave. Para aplicar todos esses dados obtidos, a Microsoft também foi conhecer estúdios em Hollywood que trabalhavam com *motion-capture*, bastante utilizado para aplicar elementos de computação gráfica a atores reais (SHOTTON et al., 2011).

Além de se preocupar com a jogabilidade, os desenvolvedores do Kinect já visavam utilizá-lo para outras funções, como por exemplo, compras em lojas virtuais, onde a pessoa poderia simular como ficaria vestido com determinado produto. Dessa forma, além de proporcionar um novo método de interação entre homem e máquina, o Kinect também levaria a realidade aumentada ao cotidiano das pessoas. Outro segmento que também se utiliza da ferramenta é o de esportes, visto que já existem alguns jogos onde a pessoa deve se exercitar diante do console e assim atinge determinada pontuação.

No começo de 2012, a Microsoft anunciou o lançamento do Kinect para computadores. O dispositivo para Xbox 360 na época custava US$ 119, já o para Windows sai por US$ 250. A diferença é que nos produtos voltados para o videogame a Microsoft lucra com o licenciamento da tecnologia do Kinect, diferente dos aplicativos voltados para computador, cujo kit de desenvolvimento (SDK) é gratuito. Além disso, o sensor para Windows trabalha com distância mínima de 50 cm, proporcionando uma maior proximidade do usuário com a tela (VELOSO, 2012). A atualização do SDK do Kinect para Windows possibilita o reconhecimento de expressões faciais, possibilita aplicações em linguagens como C++ e C# e possui o Kinect Studio, que permite gravar e reproduzir vídeos de usuários utilizando seus aplicativos. Lançado em fevereiro, o sensor para Windows só entrou no mercado brasileiro em junho de 2012 (CAMPI, 2012).

A empresa PrimeSense foi a responsável pela tecnologia por trás do sistema de reconhecimento de gestos do Kinect. As tecnologias anteriores a da empresa utilizavam luz infravermelha que calculava as distâncias de um determinado objeto calculando o tempo e comprimentos de onda de luz que retornavam para câmeras especialmente desenvolvidas para isso. Já o método da PrimeSense codifica a informação em padrões de luz e a câmera procura por deformações nesses padrões. Assim que a câmera recebe o retorno da luz infravermelha, o sistema constrói uma forma básica do ambiente e as pessoas nele. Assim que o dispositivo reconhece a pessoa, ele passa a calcular a forma dos membros e seus pontos de movimento (SCHRAMM, 2010).

A parte da interface e software do Kinect foram desenvolvidos pela Microsoft, já as demais tecnologias que foram feitas pela PrimeSense podem ser aplicados a outras funções que não voltadas para jogos. A empresa vem desenvolvendo outras tecnologias como televisões interativas e até mesmo um “Kinect” para computadores. O Xtion desenvolvido em parceria com a Asus tem as mesmas características que o sensor do Xbox, sendo que além de poder ser utilizado em jogos, pode ser utilizado para navegar na internet e outras diversas funções. Juntamente com a Hillcrest Labs, a PrimeSense adaptou para o computador o navegador de internet projetado para aparelhos de TV, Kylo Browser, tornando possível a navegação de diversos sites utilizando o Xtion. Segundo a ASUS, diversos jogos são compatíveis com o Xtion, como Street Fighter, Need for Speed: Hot Pursuit e Angry Birds (ASUS, 2012).

Vendo a expansão dos serviços inteligentes em seus aparelhos de TV, como acesso a internet, a Samsung lançou a UE55ES8000, a primeira TV a tentar substituir o controle remoto convencional por um meio alternativo de interação. O aparelho pode ser controlado por voz, gestos ou via controle remoto *touchpad*. O controle por voz é amplo, possuindo diversos comandos e não demonstrando dificuldade em diferenciar vozes masculinas, femininas e infantis. O controle por gestos é feito através de uma câmera localizada na parte superior do aparelho, onde o usuário utiliza a mão aberta como um cursor do mouse e para selecionar um item, basta fechar a mão. O controle em questão apresenta algumas dificuldades em ambientes escuros, por vezes não localizando a mão da pessoa. Outro problema é quando o sensor confunde a cabeça com a mão, fazendo com que o usuário distancie os dois membros para evitar dificuldades. Mesmo assim, a navegação na internet via controle por gestos se mostra mais eficaz que por controle remoto convencional. Já o controle com *touchpad* se mostra útil para navegação e possui microfone embutido, permitindo que o usuário possa controlar por voz sem ter que falar alto para que o receptor de áudio da TV o escute (ARCHER, 2012).

Os dispositivos voltados inicialmente para o Kinect e aplicados nas demais tecnologias citadas, demonstram que as tecnologias têm migrado para formas de

interação alternativas, substituindo os dispositivos de entrada convencionais. O reconhecimento de padrões tem tornado possível a interação por voz e gestos, dispensando a utilização dos dispositivos que conectem fisicamente o usuário com o sistema. Tal área vem gerando um grande campo de estudo para o desenvolvimento de novos produtos interativos, que podem simplificar cada vez mais o modo de operação de computadores e das mais diversas máquinas.

3 RECONHECIMENTO DE PADRÕES

Reconhecimento de padrões é uma atividade que humanos fazem inconscientemente, utilizando-se de vários órgãos e sentidos, como visão e audição. O cérebro recebe a informação e a identifica conforme padrões estabelecidos anteriormente (PAL; MITRA, 2004).

Reconhecimento de padrões visa classificar objetos em categorias ou classes, ou seja, trata do problema de desenvolvimento de algoritmos capazes de fazer a análise e classificação dos mais diversos itens, buscando uma maior precisão e velocidade para tal ação (PAL; MITRA, 2004). Esses objetos podem ser imagens, sinais ou qualquer outro que possa ser classificado e aqui são tratados pelo termo padrões (THEODORIDIS; KOUTROUMBAS, 2003).

Antes dos anos 1960 era objeto de estudo da área de estatística. Porém, com o advento da computação ocorre um aumento na demanda de aplicações que utilizam reconhecimento de padrões. Assim, este passa a ser um dos alicerces da automatização da produção industrial, além de ser parte integral da maioria dos sistemas de inteligência artificial (THEODORIDIS; KOUTROUMBAS, 2003).

Segundo Theodoridis e Koutroumbas (2003), algumas das principais aplicações da área de reconhecimento de padrões são:

Visão da máquina: O sistema captura imagens por intermédio de uma câmera e as analisa para descrever o que é a imagem. Uma aplicação típica de máquina é na manufatura da produção, onde uma inspeção visual automatizada é utilizada para averiguação de qualidade do produto.

Reconhecimento de caracteres: Reconhecimento óptico de caracteres já é uma aplicação comum, onde o texto escaneado é reconhecido pela máquina que transforma a imagem capturada em caracteres editáveis. Há também o reconhecimento de escrita manual, utilizado por bancos para validação de cheques, por exemplo. Existem aplicações voltadas aos

correios, para separação de correspondências conforme o CEP do destinatário. Tais funções podem ser consideradas tediosas para serem feitas por pessoas.

Diagnóstico auxiliado por computador: Vários dados podem ser analisados por computadores, como raios-X, tomografias, eletrocardiogramas, eletro encefalogramas, dentre outros. Apesar de não substituir o trabalho do médico, os diagnósticos auxiliados por computador dão uma "segunda opinião", podendo detectar problemas que possam passar despercebidos por alguma pessoa.

Reconhecimento de voz: a fala é o mais natural dos meios de comunicação dos seres humanos. Construir máquinas inteligentes que reconheçam comandos de voz meche tanto com engenheiros e cientistas como com escritores de ficção científica. Aplicações em potencial são inúmeras, podendo ser utilizadas para controle de máquinas das mais diversas formas. Existem softwares, como o Dragon Dictation, para iPhone, que já vem utilizando o reconhecimento de voz para captar informações via microfone e transformar em texto, tornando a escrita o dobro mais rápida que se fosse feita por um datilógrafo, além de tornar mais simples a comunicação com pessoas surdas.

Reconhecimento gestual: Vem sendo estudada há pouco tempo. É fortemente ligada com outras disciplinas como linguística, computação gráfica e visão. Este será detalhado na seção 3.1.

Um sistema de reconhecimento de padrões pode ser dividido em três fases: aquisição de dados, seleção de características e classificação (PAL; MITRA, 2004).

A aquisição de dados pode ser uma coleção de medições ou consulta de um banco de dados (HIMBERG et al., 2001). Dependendo de como os dados estão dispostos, o computador os absorve por meio de algum determinado sensor, como câmera ou microfone.

As medidas usadas para classificar um objeto são conhecidas como características. Quanto maior o número de possíveis características, melhor o resultado gerado na classificação do padrão avaliado (THEODORIDIS; KOUTROUMBAS, 2003). A extração de características capta de todos os dados a informação mais relevante de acordo com o propósito de classificação, a fim de minimizar a variação dentro da classe e aumentar a diferença entre outras classes. Além disso, nesse processo, a proporção dos dados é minimizada para diminuir os processamentos realizados pelo computador (LAMPINEN; LAAKSONEN; OJA, 1998). A seleção de características a serem utilizadas na classificação de um item começa com a remoção de *outlier*. Um *outlier* é um ponto de referência muito distante dos demais valores encontrados. A eliminação de tal atributo faz com que a classificação posterior seja mais precisa. Após, é feita a normalização dos dados, onde, para os mesmos, são atribuídos valores menores, gerando um custo de processamento menor (THEODORIDIS; KOUTROUMBAS, 2003).

Os passos anteriores são realizados visando o sucesso na fase de classificação. É nessa parte do processo de reconhecimento de padrões que os dados de entrada geram resultado (LAMPINEN; LAAKSONEN; OJA, 1998). Existem diversos métodos de classificação e funções que podem ser aplicados para a classificação. Na aprendizagem são gerados os classificadores que diferenciarão os elementos a serem inseridos no sistema (DUDA; HART; STORK, 2000). Com base no método de aprendizagem, o reconhecimento de padrões pode ser categorizado como supervisionado ou não-supervisionado.

A aprendizagem é considerada supervisionada quando compreende exemplos de vetores de entrada juntamente com seus vetores alvo correspondentes (BISHOP, 2006). Conjunto de treinamento é assim denominado o grupo de padrões anteriormente classificados que são adotados na aprendizagem supervisionada (THEODORIDIS; KOUTROUMBAS, 2003). Alguns dos principais métodos de reconhecimento de padrões que adotam aprendizagem supervisionada são classificadores baseados na teoria de decisão Bayesiana, utilizado em reconhecimento de padrões em imagens de sensoriamento remoto, para gerar

estudos geológicos e previsão do tempo, dentre outras aplicações (GONZALEZ; WOODS, 2010); classificadores lineares, devido a sua simplicidade computacional não são considerados ideais para todo tipo de reconhecimento de padrões devido as suas limitações, porém sua velocidade pode compensar a perda de desempenho (FUKUNAGA, 1990); e Matching, utilizado para classificar o padrão desconhecido com base na distância entre ele e os vetores de características das classes geradas pelo próprio sistema (GONZALEZ; WOODS, 2010).

Quando não existem dados que possam ajudar na aprendizagem de um sistema de reconhecimento de padrões, ela é considerada não supervisionada. A aprendizagem não supervisionada, também conhecida como *clustering*, é assim nomeada quando os padrões são agrupados pelo algoritmo, com base nas semelhanças observadas pelo próprio sistema (DUDA; HART; STORK, 2000). Nessa forma de classificação não existem classes predefinidas ou essas classes são desconhecidas (LAMPINEN; LAAKSONEN; OJA, 1998). *Clustering* costuma ser utilizado para fornecer classes protótipo para uso em classificadores supervisionados (WEBB; COPSEY, 2011).

Os métodos de reconhecimento de padrões são utilizados em diversas áreas, conforme explicado anteriormente. Uma das aplicações é o reconhecimento gestual que pode ser utilizado como base para um modo de interação diferente do convencional. Por se tratar do foco de estudo, o reconhecimento gestual será melhor detalhado no capítulo a seguir.

3.1 RECONHECIMENTO GESTUAL

A área de reconhecimento de gestos tem crescido gradualmente e dispositivos que adotam essas técnicas, como o Kinect, se tornaram muito populares. Apesar de estudos estarem datados desde 1973 (JOHANSSON), nos últimos anos a área deu um salto devido à ampla gama de aplicabilidade em projetos de interface natural ao usuário (BOBICK; WILSON, 1997).

Técnicas de visão computacional e reconhecimento de padrões podem envolver extração de características, detecção de objetos e classificação, características essenciais ao reconhecimento gestual (MITRA; ACHARYA, 2007). Essas características devem abordar aspectos importantes do gesto, observando componentes do movimento de forma confiável, além de saber quais os aspectos que são vagos e assim calcular sua alta variabilidade (BOBICK; WILSON, 1997). Por exemplo, em ambientes naturais, a localização da mão pode ser ambígua, devido ao movimento de fundo e presença de outras pessoas. Além disso, os gestos geralmente aparecem dentro de um fluxo contínuo de movimento. Detectar automaticamente onde um gesto começa e termina é um problema desafiador (ALON et al., 2009).

Para reconhecer gestos manuais em vídeo, um sistema de visão computacional deve estar capacitado a interpretar tanto gesto espacial quanto temporal. Gesto espacial se refere a onde o gesto ocorre, ou seja, onde a mão gesticulando está localizada. Já o gesto temporal é quando o gesto começa e termina (ALON et al., 2009). Para caracterizar alterações temporais em um evento de vídeo, devem ser analisadas as condições do evento atual e do passado, fazendo um comparativo entre eles e assim determinar o movimento realizado (MITRA; ACHARYA, 2007).

Sistemas de reconhecimento gestual podem interpretar a mão detectando a pele, utilizando histogramas de cores para calcular a probabilidade de o elemento captado ser a mão do indivíduo. Dessa forma, entretanto, a mão pode ser confundida com o rosto, entre outros erros (ALON et al., 2009).

A popularidade dos modelos ocultos de Markov (HMMs) levou à sua utilização no reconhecimento de ação e gesto (BOBICK; WILSON, 1997). O HMM é rico em estruturas matemáticas e modela eficientemente informações espaço-temporais de uma forma natural. O modelo é chamado de "oculto" porque tudo o que pode ser visto é apenas uma sequência de observações (MITRA; ACHARYA, 2007). Em geral, é criado um rastreador que adota uma abordagem baseada em HMM para

reconhecer a forma da mão através de seu contorno, e assim pode detectar variações em sua forma (RAMAMOORTHY et al., 2003).

Há duas definições importantes em reconhecimento gestual: postura e gestos. A postura se refere a localização estática da mão. O gesto é a sequência de posturas de mão conectados por movimentos contínuos em determinado tempo. O movimento pode ser considerado global, quando a mão é movimentada e local quando apenas o dedo se move (CHEN; GEORGANAS; PETRIU, 2007).

Para realizar tarefas de reconhecimento gestual estão disponíveis diversas técnicas e ferramentas. Utilizando métodos de reconhecimento de padrões, dispositivos como o Kinect foram desenvolvidos e outros dispositivos também podem ser criados. Um software que pode adotar os princípios de reconhecimento de padrões para assimilar gestos e movimentos capturados por uma câmera é o Adobe Flash.

4 FLASH

A disseminação da internet modificou as interações entre as pessoas. Não é de hoje que é possível conversar com qualquer pessoa pela web em tempo real. Sites de compras se tornaram grandes empresas e roubam muitos clientes das lojas físicas. A evolução da internet passa pelos desenvolvedores e softwares que os auxiliam na criação de sites interativos, chamativos e criativos, que fazem com que o usuário se sinta cada vez mais deslumbrado com o que encontra na rede de computadores. Uma das ferramentas que mais revolucionou a maneira como os sites são desenvolvidos é o Flash (REINHARDT; LENTZ, 2001).

A história por trás do Flash começa em 1993, quando Jonathan Gay e Charlie Jackson fundam a empresa FutureWave Software para trabalhar com o mercado de softwares que utilizassem caneta em telas *touchscreen*. Supondo que desenhar na tela pudesse ser melhor que desenhar no papel, Gay e Jackson começaram a trabalhar no SmartSketch, programa para essa finalidade. Porém a empresa Go, que criaria o sistema operacional onde o SmartSketch seria aplicado, foi adquirida pela AT&T e cancelou o lançamento do sistema em questão, eliminando o mercado do programa da FutureWave (GAY, 2006).

A empresa então decidiu voltar seu produto para os sistemas Windows e Macintosh, que já possuíam os aplicativos Illustrator e FreeHand para desenho vetorial (GAY, 2006).

Em 1995, a FutureWave decidiu mudar o foco do SmartSketch para animações. A decisão surgiu devido a Web, que possuía grande potencial e demonstrava que as pessoas poderiam gostar dos elementos gráficos que seu programa poderia gerar (GAY, 2006).

Na época, os navegadores somente reproduziam animações através do Java, o que as tornavam muito lentas. Porém, pouco tempo depois, o Netscape foi

lançado com seu plug-in API[1] que tornou possível a criação de uma ferramenta que melhorou as condições das animações na internet (GAY, 2006).

Essa ferramenta é antecessora do Flash Player, que permite a execução de arquivos SWF[2] gerados pelo Flash em navegadores web. O Flash Player evoluiu em paralelo ao Flash, se adequando as inovações geradas pelo software de criação.

Com essas melhorias, os criadores do SmartSketch decidiram mudar o nome de seu software para FutureSplash Animator, focando em sua capacidade de gerar animações, que pôde ser lançado em maio de 1996. Meses depois, em agosto, a Microsoft trabalhava no MSN[3] e, com a ideia de tornar a internet semelhante à televisão, adotou a tecnologia do FutureSplash. Outro cliente importante dessa mesma época foi a Disney Online, que também trabalhava com o Macromedia Shockwave. Devido a essa ligação, a Macromedia se interessou no software de animação e adquiriu a FutureWave em dezembro de 1996, transformando o FutureSplash Animator no Macromedia Flash 1.0 (GAY, 2006).

O Flash 2, de 1997, começou a tornar o software como referência para ilustradores e animadores. Essa versão inseriu o conceito de biblioteca no programa, que possibilitou uma melhor manipulação dos objetos criados no programa, implantou funções de botão e efeitos de som e texto (MALOPINSKY, 2010).

A terceira versão do programa foi lançada em 1998, onde foram introduzidas opções de transparência e os conceitos de *movieclip*, função utilizada para separar um clipe de vídeo do resto do arquivo (MALOPINSKY, 2010).

---

[1] API, sigla para “application programming interface” ou interface de programação do aplicativo, é uma ferramenta que adiciona funcionalidades a um software.

[2] SWF ou Shockwave Flash é o formato de arquivo geralmente exportado pelo Flash e amplamente usado na web.

[3] MSN, derivado de The Microsoft Network, é uma rede de serviços on-line da Microsoft, que inclui e-mail, programa de comunicação instantânea, entre outros.

Lançado em 1999, o Flash 4 teve várias inovações, com melhor adequação de suas funções à área de trabalho do programa, camadas de controle e suporte a arquivos MP3. O grande diferencial dessa versão foram as *actions*, que viriam a se tornar a linguagem de script do programa e que aumentaram a gama de opções que o software oferecia (MALOPINSKY, 2010).

A linguagem de programação do Flash, o ActionScript, foi inserida na versão 5 do programa, lançada em 2000. Baseado na sintaxe e semântica do ECMAScript, o script do Flash recebeu adequações com o lançamento do Flash Player nas versões 6 e 7, mas não teve alterações radicais (BRIMELOW, 2008). O software em si teve poucas alterações, a maioria, visuais e de caráter organizacional, mas, com a inserção do script, aumentaram as criações feitas com o programa, principalmente com lançamento de jogos criados nele (MALOPINSKY, 2010).

Em 2002 a Macromedia lançou o Flash MX, incluso em um pacote com outros programas da empresa. Nessa versão, o programa passou a suportar vídeos, fator importante para a criação de sites de *streaming* de vídeos, como o YouTube (MALOPINSKY, 2010).

Nessa mesma época foi lançado o Flash Player 6, que possuía a classe *Camera* que passou a dar acesso a webcam do usuário, tornando possível a publicação de vídeos ao vivo, detecção de movimentos, entre outras opções (REINHARDT, 2007).

O Flash MX 2004, de 2003, trouxe como diferencial a inserção do ActionScript 2, que também foi usada no Flex 1.0. O Flex é um programa voltado para o desenvolvimento de aplicações ricas para a Internet, também conhecidas como RIAs (Rich Internet Applications). São consideradas aplicações desse tipo aquelas que possuem características de interação diferenciadas, tornando mais simples as ações que o usuário deve realizar para interagir com o sistema (WARD, 2007). O ActionScript 2 possuía poucas diferenças em relação à versão anterior, pois a base da linguagem era a mesma (BRIMELOW, 2008). A maior diferença é

relativa à introdução de palavras-chave familiares a usuários acostumados a trabalhar com Java e C++ (ADOBE, 2009).

No Flash 8, lançado em 2005, foram inseridas novas ferramentas de desenho que tornavam o programa mais parecido com o Illustrator, software para ilustração. O codec de vídeo, software para decodificar sinais, utilizado nas versões anteriores foi substituído e, coincidentemente, três meses depois, o YouTube foi ao ar (MALOPINSKY, 2010). Nesse mesmo ano, a Macromedia foi adquirida pela Adobe por 3,4 bilhões de dólares (ADOBE, 2005).

Durante o planejamento do Flash Player 9, seus desenvolvedores da Adobe acharam que era necessário criar uma nova *engine* de programação para superar as versões anteriores do ActionScript. Assim foi criado o ActionScript 3.0, lançado no Flex 2.0 e reconhecido a partir da nona versão do player (BRIMELOW, 2008).

O primeiro Flash lançado pela Adobe foi o CS3, de 2007. Vendido junto ao pacote Adobe Creative Suite, teve sua interface adequada aos produtos da empresa e possuía o ActionScript 3.0 e sua versão anterior (BRIMELOW, 2008).

Com o pacote CS4, lançado em 2008, veio a nova versão do Flash que ganhou melhoras na parte de animação, além de suporte básico a elementos 3D (BRIMELOW, 2008).

Em 2010 o Flash CS5 entrou no mercado com inovações na parte de animação, novos formatos de arquivo e maior integração com os demais produtos do pacote da Adobe (BRIMELOW, 2008).

O pacote Adobe CS6 trouxe o novo Flash em 2012. Este possui um simulador que permite ao desenvolvedor emular inclinação e rotação de um dispositivo móvel, além de funções de toque e zoom realizados na tela. Também foram inclusos recursos que possibilitam a exportação de animações em JavaScript para HTML5 (WILLIAMS, 2012).

4.1 APLICAÇÕES COM FLASH

A primeira função a qual o Flash se propôs foi a criação de animações voltadas para a Internet. Com o tempo, suas funções cresceram significativamente, ampliando seu número de usuários, além de popularizar o Flash Player que passou a ser incluído a navegadores e sistemas operacionais (REINHARDT, LENTZ, 2001). A gama de produção do Flash cresceu relativamente com a inserção do ActionScript, que permitiu a criação de aplicações mais complexas. Assim o Flash passou a atuar com três componentes, o software de desenvolvimento, o plug-in que permite a execução dos projetos criados e a linguagem de script. Sua popularização também se deve a ausência de concorrentes na época de sua criação, pois itens semelhantes aos criados no Flash eram somente os aplicativos em Java (MACHADO, 2012). Aos poucos, as animações criadas no software passaram a incluir novas funções e de coadjuvante de um site, acabou por se tornar o próprio site.

O arquivo SWF, principal formato exportado pelo Flash, é o que pode ser lido pelo Flash Player. Esse formato pode conter vídeo, som, animações, dados, entre outras informações, criando assim um arquivo multimídia (DEHAAN, 2007). Com a união de tantos elementos, um grande mercado surgiu: desenvolvimento de jogos em Flash. O diferencial dos produtos dessa área é que o usuário não necessita de um computador potente para poder jogar e geralmente são disponibilizados gratuitamente na internet. Atualmente jogos desenvolvidos no Flash são facilmente encontrados em redes sociais e dispositivos móveis (RIVELLO, 2009).

Além de trabalhar com a maior parte dos sistemas operacionais, o Flash também é compatível com alguns consoles. O Nintendo Wii e o Playstation 3 podem ter o Flash Player instalado, tornando possível a criação de jogos e outros projetos (FRANÇA, 2010). O Xbox 360, através do Kinect, também dá suporte a itens criados no Flash. A princípio, essa integração somente era possível com a utilização de *hacks*, assim chamado um conjunto de modificações em um sistema, porém, em

2011, a empresa Blitz criou uma ferramenta que permitiu a integração entre o console e arquivos criados no Flash (COLDEWEY, 2011).

Outra área em que o Flash pode ser utilizado é na de dispositivos *multitouch*. Desde a versão 10.1 do Flash Player e é capaz de compreender eventos de toque realizados em uma tela. A manipulação de eventos *multitouch* é disponível para ActionScript 3, com a utilização da classe *Multitouch*. Produtos voltados para dispositivos com essa tecnologia devem se adequar ao fato de não poderem assumir funções a eventos como pairar, ou seja, passar o mouse sob algum elemento. Outro fator, é que nesses dispositivos é possível fazer vários toques na superfície, como se fosse possível utilizar diversos mouses ao mesmo tempo e assim fazer gestos como dar zoom afastando ou juntando os dedos na tela (CANTRELL, 2009).

O desenvolvimento de produtos para dispositivos móveis também é um dos pontos fortes do Flash, principalmente porque esse mercado é um dos que mais crescem atualmente (FRANÇA, 2010). Existem vários sistemas operacionais para esses aparelhos e a criação para algumas plataformas pode ser mais complexa do que para outras. O iPhone da Apple, por exemplo, não dá suporte ao Flash, fato que se tornou um problema para a Adobe que viu outras soluções, como o HTML5, substituírem aplicativos criados por seu produto (MACHADO, 2012). Apesar das últimas versões do Flash possuir funções voltadas para dispositivos móveis, como APIs para orientação da tela, dentre outros (DURA, 2010), a Adobe anunciou em 2011 que não dá mais suporte ao Flash Player para dispositivos como Android e BlackBerry (MACHADO, 2012). Entretanto a versão CS6 possui avanços significativos em HTML5, o que facilita o desenvolvimento tanto para a web como para as plataformas de smartphones disponíveis no mercado (WILLIAMS, 2012).

Diferente da realidade virtual, que troca a realidade física pela gerada pelo computador (RIVELLO, 2009), a realidade aumentada permite que o mundo real e o virtual se misturem, mudando a maneira como o usuário interage com o computador (HAUTSCH, 2009). O Flash pode criar projetos com realidade aumentada graças ao

FLARToolkit, conjunto de funções para ActionScript 3 baseada na ARToolKit, biblioteca originária do C. O FLARToolkit lê um marcador, detectado através de uma câmera, e o substitui na tela por um elemento predefinido (NICKULL, 2009).

No ano de 2002, a Macromedia lançou um documento que descreve uma série de parâmetros sobre como tornar mais ricas as aplicações para internet utilizando o Flash MX. Surgia assim o conceito de RIA, Rich Internet Application. As diretrizes fazem questão de expor a necessidade de tornar os arquivos mais ágeis e principalmente a dinâmica na interação com o sistema a ser criado no Flash. Um dos aspectos mais importantes diz a respeito da comunicação com o servidor do site, pois a maioria das ações é feita no próprio computador do usuário, evitando realizar funções no servidor, fato que tornava as aplicações lentas (ALLAIRE, 2002). RIAs unem multimídia, maior interatividade e funcionalidade de aplicações desktop a aplicações para internet. Não resolvem todos os problemas da web, porém solucionam os relativos a processamento, dados, configuração e *feedback* (PRECIADO et al., 2005).

O desenvolvimento de RIA é historicamente ligado ao Flash, porém se assemelha com aplicações anteriormente feitas em Java, NET e C++ (GULDMAN, 2006). Atualmente existem várias ferramentas utilizadas para criar produtos desse gênero. Além do Flash, essas aplicações também podem ser desenvolvidas em Flex e AJAX, dentre outros (FRANÇA, 2005).

As propriedades de RIA ampliaram as possibilidades na web. Suas características tornaram sites mais dinâmicos, eliminando a lenda que dizia que o Flash servia para tornar a internet mais lenta e tirar o foco com suas animações. As bibliotecas do ActionScript que tornaram possível a criação de produtos com realidade aumentada também passaram a ser utilizadas frequentemente, tornando comum a utilização das *tags* de RA em publicidade e propaganda, principalmente. A união de características desses dois mercados do Flash podem gerar produtos ainda mais interessantes.

5 INTERAÇÃO GESTUAL

Com base no funcionamento do Kinect, surgiu a ideia de criar um aplicativo que não utilize periféricos convencionais para controle e inserção de informações. No Kinect, o método para controle das ações é baseado na captura de movimentos feitos por uma câmera infravermelha. No console, as informações rastreadas são a posição completa do corpo do jogador, sua distância do aparelho e coordenadas de membros, gerando uma espécie de mapa corporal completo.

O Kinect realiza esse rastreamento completo, em 3D, para que a pessoa faça no mundo real o que estaria fazendo no jogo ou na aplicação no qual o aparelho é adotado. Caso o rastreamento fosse parcial, desconsiderando a profundidade da movimentação, ou seja, capturando apenas movimentos na horizontal e na vertical, a câmera infravermelha poderia ser substituída por uma câmera RGB convencional, encontrada até mesmo em vários dispositivos como *notebooks* e celulares. Ao utilizar a câmera convencional, qualquer dispositivo com esse artefato poderia empregar interação semelhante a do videogame citado.

Se as variações de movimentos em profundidade forem desconsideradas, muitas formas de ação possíveis no Kinect ficam impossibilitadas. Por outro lado, a aplicação se torna mais simples e com menor índice de cálculos a serem realizados, não necessitando de toda a estrutura que o Xbox 360 possui. Devido a esses aspectos, o método proposto pretende observar apenas movimentos bidimensionais.

Outros itens que foram levados em conta na criação do Kinect podem ser desconsiderados. O dispositivo do Xbox 360 consegue distinguir jogadores com base na captura de imagens feitas pela câmera em RGB, diferenciando-os visualmente. E possui um microfone, que realiza o reconhecimento das vozes das pessoas que o utilizam naquele momento. Essa caracterização de cada usuário não é levada em consideração na interatividade proposta, pois esta visa ser usada por apenas uma pessoa por vez, sem necessidade de reconhecê-la numa segunda oportunidade.

O reconhecimento de padrões se dará através da análise das imagens captadas pela câmera. O computador analisará a posição da mão do usuário e a usará como base para controle do cursor que se movimentará junto com o membro. A ação pode ser comparada ao reflexo diante de um espelho, substituindo a mão pelo cursor na tela.

O sistema foi desenvolvido no Adobe Flash com a utilização da linguagem ActionScript, adotando funções de webcam junto ao Flash Player. A câmera capturará a imagem e os movimentos feitos diante dela serão os responsáveis pela movimentação do cursor.

O aplicativo criado para testar a interação proposta é um sistema de busca, para ajudar o usuário encontrar uma determinada localização em uma área específica. No caso, foi adotada a situação em que o usuário se encontre na Universidade de Caxias do Sul e deseja encontrar alguma localização na área da instituição, como biblioteca, blocos, protocolo e entre outros.

A primeira tela que se apresenta no programa é um menu com o nome dos locais que poderão ser encontrados. O controle desse menu se dará através da interação gestual, onde o usuário utilizará sua mão aberta diante da câmera e terá o controle do cursor, com o qual a pessoa escolherá a localização desejada no menu. Depois de escolhido o destino, um mapa da universidade abrirá na tela, indicando onde o usuário se encontra e o local escolhido anteriormente, mostrando o caminho a ser percorrido.

O mapa da universidade foi modelado em 3D no programa 3DMax, com base no mapa da cidade universitário encontrado no site da UCS. No sistema desenvolvido, não serão modelados todos os prédios indicados no mapa do site. O sistema sempre tomará como ponto de partida o bloco 71, porém ao ser implementado por completo, além de possuir indicação para todos os locais possíveis dentro da universidade, o ponto inicial deve variar de acordo com o local em que o sistema estiver instalado. Para isso devem ser criadas diferentes versões das localizações, de acordo com a posição do próprio aplicativo.

A interface gráfica do sistema adota um modelo semelhante ao encontrado no Google Maps, com o mapa à direita e o caminho descrito textualmente na esquerda. Diferente do Google Maps, não é possível deslocar, dar zoom ou indicar uma localização diretamente no mapa. Somente a indicação de alguma localização é possível, porém utilizando o menu do aplicativo, usando interação gestual.

A interação e o modo como a pessoa se aproveitará dela são o foco do sistema, visto que o problema tratado é a criação de um modelo de interação novo, com base nas inovações surgidas com o lançamento do Kinect e ferramentas semelhantes. Para tanto serão aplicados designs que levem em conta princípios de usabilidade para interfaces humano-computador, procurando uma melhor forma de interação entre o usuário e o sistema.

6 DESENVOLVIMENTO DO APLICATIVO

Com o objetivo de criar um sistema de localização utilizando interação gestual capturada por uma câmera digital convencional, o trabalho de desenvolvimento do mesmo foi divido em duas etapas:

1 - Mapa: Modelagem e renderização do mapa.

2 - Interface: Desenvolver a interface e os controles do aplicativo.

6.1 MAPA

Para modelar o mapa em 3D, foi optado pela utilização do software 3D Max 2011, fabricado pela Autodesk. Existem outros softwares semelhantes, como o Maya, da mesma produtora, e, inclusive softwares livres, como o Blender. Junto com o 3D Max, foi utilizado o plug-in VRay 2.01, fabricado pelo Chaos Group, principalmente para complementar a posterior renderização das imagens.

O sistema de localização proposto visa encontrar determinados pontos em uma área específica, nesse caso, nas instalações da Cidade Universitária da UCS. Para facilitar o reconhecimento visual do usuário, foi optado por modelar um mapa com os blocos e demais referências. O site da UCS possuía um mapa com todos os prédios da Cidade Universitária (Figura 1), que foi utilizado como base para a modelagem do que viria a ser usado no aplicativo.

FIGURA 1 – Mapa da Cidade Universitária da UCS

Fonte: UCS. Disponível em: <http://www.ucs.br/ucs/ucsnaregiao/campusuniversitario/localizacao/mapa_cidade_universitaria> Acesso em: 28 de out. 2012.

Alguns dos pontos de referência encontrados no mapa foram descartados por serem pouco conhecidos ou de menor importância, assim, decidiu-se que os prédios a serem modelados seriam os seguintes:

Bloco A – Reitoria;

Bloco B – CETEC;

Bloco C – CETEC;

Bloco D – Centro de Ciências Exatas e Tecnologia;

Bloco E – Centro de Filosofia e Educação;

Bloco F – Centro de Ciências da Administração;

Bloco G – Centro de Ciências Exatas e Tecnologia;

Bloco H – Centro de Ciências Humanas;

Bloco I – Laboratório de Informática;

Bloco J – Centro de Ciências Econômicas, Ciências Contábeis e Comércio Internacional;

Bloco K – Salas de Aula;

Bloco L – Programa de Línguas Estrangeiras;

Bloco M – UCS Teatro;

Bloco S – Centro de Ciências e Saúde;

Bloco V – Centro de Ciências Exatas e Tecnologia;

Bloco 43;

Bloco 46 – Salas de Aula / Pós Graduação;

Bloco 58 – Centro de Ciências Jurídicas;

Bloco 70 – Fisioterapia e Instituto de Medicina do Esporte;

Bloco 71 – Centro de computação;

CETEL – Centro de Teledifusão Educativa, Centro de Ciências da Comunicação;

Quiosque de Informações;

Terminal Rodoviário;

Gráfica / Editora;

Ambulatório Central;

Casa do Professor Visitante;

Biblioteca Central;

Casa do Professor II;

Museu de Ciências Naturais;

Galeria Universitária;

Centro de Convivência;

Bar do Bloco 58;

Restaurante Universitário;

Jardim Zoológico;

Serpentário;

Ginásio de Esportes I;

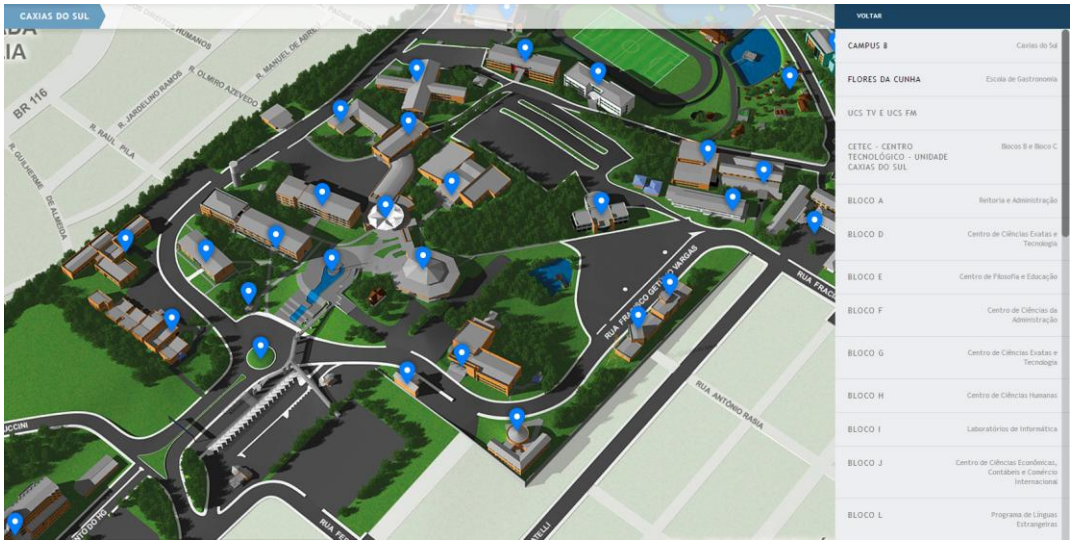
Ginásio de Esportes II – Piscina;

Ginásio de Esportes III – Poliesportivo;

Quadras Cobertas.

No desenvolver da modelagem, a UCS trocou o modelo do mapa da Cidade Universitária, optando por um modelo mais interativo e com algumas mudanças em relação ao mapa anterior. Na figura 2, o mapa interativo atualmente disponível no site da UCS.

FIGURA 2 – Mapa interativo da Cidade Universitária da UCS



Fonte: UCS. Disponível em <http://www.ucs.br/site/tourvirtual/unidades/detalhe/1/caxias-do-sul> Acesso em: 28 de out. 2012.

Apesar do mapa em si não ter sofrido alterações, alguns dos pontos de referência foram alterados:

Vila Poliesportiva – uniu em um ponto único o bloco 43 e os ginásios de esportes I, II e III;

CETEC – uniu os blocos B e C;

UCS Rádio e UCS FM – adicionado;

Central de atendimento – adicionado;

Quadras Cobertas – excluído;

Serpentário – excluído;

Bloco K – excluído;

Gráfica / Editora – excluído;

Casa do Professor II – excluído.

As informações anteriores refletem na renderização do mapa, pois o agrupamento de prédios deve ficar visível na imagem a ser gerada, e na interface do aplicativo, devido aos botões com o nome de cada um dos pontos de referências.

6.1.1 Modelagem

A modelagem, com o software 3D Max 2011, foi iniciada com a criação do terreno com o recurso Line, usado para traçar uma linha reta que estendida, criando assim a base do mapa. O terreno do mapa não segue o relevo da Cidade Universitária, principalmente devido à necessidade de uma pesquisa maior para que o relevo fosse desenhado de forma fiel.

Os prédios foram modelados com a ferramenta Box, que gera um cubo. Para deixar os prédios no formato adequado, foi utilizada a edição chamada Edit Poly, que permite uma série de alterações nas formas. Com tal ferramenta é possível selecionar as arestas de qualquer objeto modelado e posicioná-las da forma desejada.

Após a modelagem do terreno e dos prédios, foram inseridos a vegetação e os carros por meio de Merge. Merge permite inserir elementos de outros arquivos para que sejam copiados e inseridos na cena com todos os recursos de edição possíveis. A vegetação e os carros foram pegos de um banco de modelos 3D adquiridos de diversos sites da internet.

Com todos os objetos modelados, foi iniciado o processo de texturização com VRay Materials. VRay Materials são materiais fornecidos com o VRay que possibilita uma iluminação mais correta, com melhor distribuição de luz, além de reflexão e refração mais precisos. Foram criados 12 materiais diferentes para essa cena.

Com os materiais criados, foi iniciado o mapeamento com UVW Map. O mapeamento é a aplicação de materiais nos modelos para atribuir realismo as superfícies na renderização. A aplicação dos materiais é diferente de acordo com o elemento em que o material será aplicado, ou seja, a textura é a mesma para um

prédio grande ou pequeno, por exemplo. Para que os tijolos tenham o mesmo tamanho, ao aplicar o material, o UVW Map permite a edição da textura na aplicação da mesma, podendo redimensionar ou girar conforme o desejado. Assim os prédios, principalmente os com textura de tijolos, puderam ficar semelhantes com o mapeamento adequado.

Após os materiais aplicados, foi feito o processo de iluminação, com a utilização de luzes Standard, Target Direct e VRay Light. A diferença entre as luzes é basicamente no modo como ela se dispersa na cena. A Target Direct foi utilizada em todos os prédios, para que se destacassem na cena. As demais ficaram responsáveis pela iluminação do cenário, do terreno e de seus elementos secundários.

Por fim, foi realizada a programação do render com recursos do VRay. O render é o resultado final da modelagem, que pode ser um vídeo ou uma imagem estática. O VRay permite diversas configurações para uma melhor imagem e uma finalização mais rápida, principal atrativo do plugin.

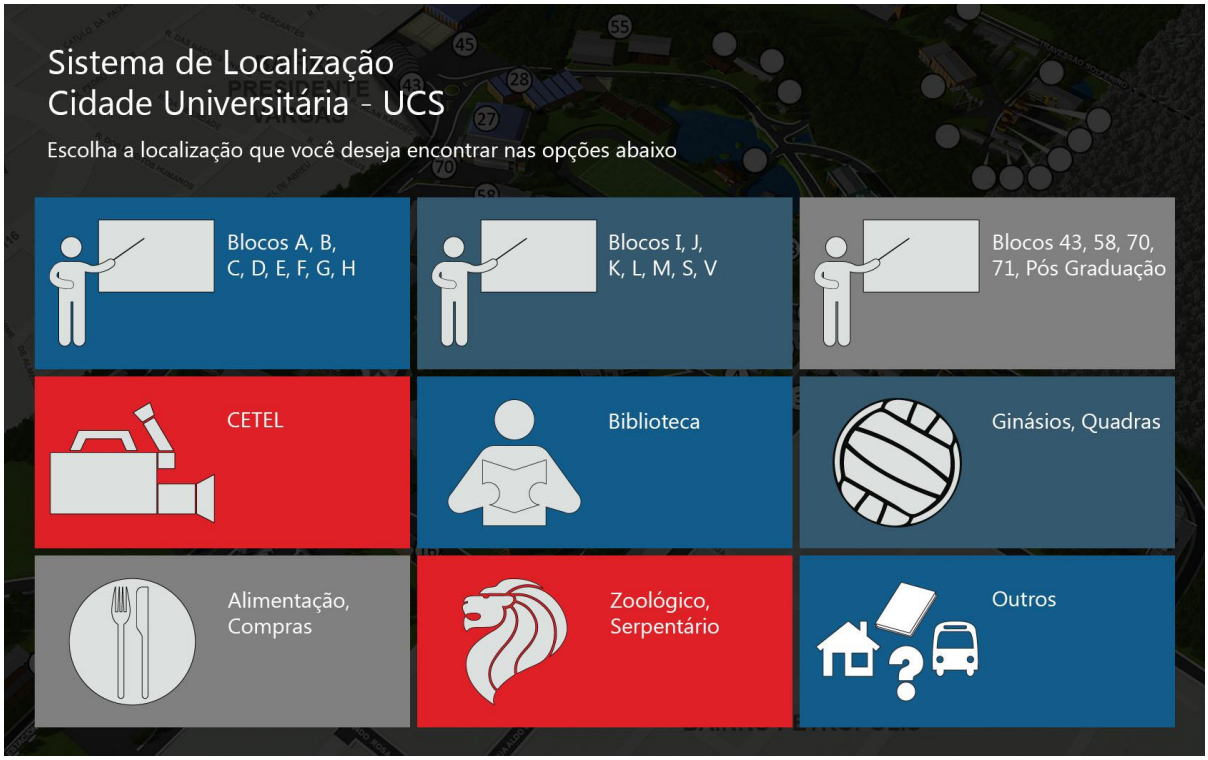
No trabalho aqui descrito, o objetivo inicial era a renderização de vídeos, traçando um caminho entre um prédio e outro. Porém, primeiramente devido a estrutura limitada, os vídeos foram descartados. Na renderização do primeiro vídeo, o computador do autor teve a fonte queimada. Para evitar maiores problemas, foi adotada uma abordagem diferente, adotando o modelo de mapa utilizado no Google Maps, ou seja, a vista aérea de um terreno em forma de imagem estática. A adaptação do mapa para a renderização de uma imagem ao invés de um vídeo foi mínima, principalmente pelo fato da iluminação ter sido aplicada de forma homogênea, pois o vídeo que seria criado seria determinado por uma câmera que andaria pelo terreno, assim não podendo haver mudanças de iluminação pelo caminho. Para tanto, as câmeras que fariam o caminho entre prédios foram eliminadas e somente uma vista aérea foi adotada e uma imagem geral foi gerada como resultado final. No total foram gerados cinco renders, um completo, de 1680 x 1050 pixels, e outros quatro, de 4100 x 2750 pixels, dividindo toda a cena. Após a

renderização, as quatro imagens foram unidas e essa imagem foi cortada de acordo com o prédio tomado como ponto de partida e o de destino, gerando, por fim, 30 imagens diferentes.

6.2 INTERFACE

O desenvolvimento da interface do aplicativo iniciou-se com a criação do layout do mesmo. O visual dos menus do aplicativo foi baseado no design Metro da Microsoft. Encontrado inicialmente no Encarta 95, uma enciclopédia digital da Microsoft, e no MSN 2.0, o design Metro se popularizou ao ser empregado no sistema operacional Windows Mobile, no Windows Phone 7 e, recentemente, no Xbox Live e Windows 8 (MASSEY, 2012). O design se caracteriza pela disposição dos menus como azulejos ou ladrilhos, em quadrados ou retângulos, e a utilização da fonte Segoe (GREENE, 2012). No menu do aplicativo, foram adicionados botões com sua descrição, que pode remeter a uma localidade ou um conjunto delas. No primeiro caso, é aberta a tela de resultado da busca, já no segundo, o usuário é remetido a um novo menu, com as localizações indicadas separadamente. Os botões são aplicados com opacidade em 50%, e o mesmo fica com a opacidade em 100% quando houver o posicionamento do cursor sobre ele. O layout da tela de abertura do aplicativo é mostrado na figura 3.

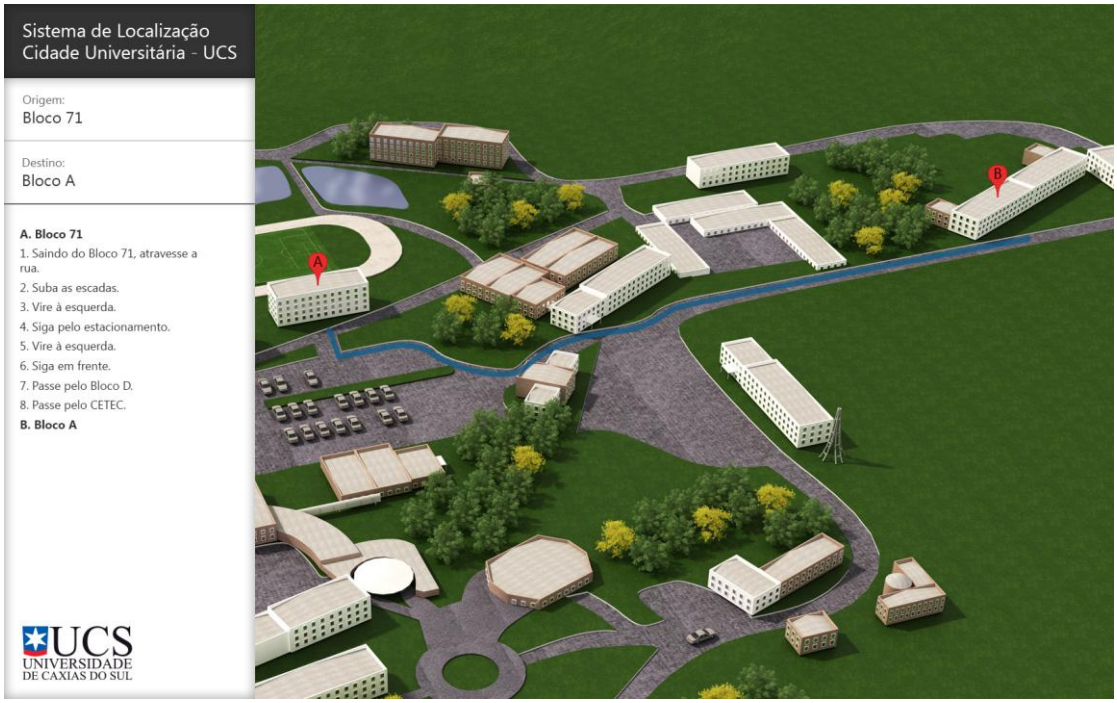
FIGURA 3 – Menu inicial do aplicativo



Fonte: Autor.

Ao escolher alguma localização, o usuário é levado para a tela de resultado da busca, onde, a princípio, haveria o vídeo e a descrição textual do caminho para chegar ao lugar desejado. Apesar da não adoção das animações, o layout da tela em questão foi pouco alterado, somente substituindo o espaço reservado para o vídeo pelo mapa estático. O layout foi baseado nas características do Google Maps: à esquerda da tela a descrição do caminho a ser realizado e, à direita, o mapa com o caminho traçado entre os dois pontos. Na figura 4, o layout da tela em questão.

FIGURA 4 – Layout mapa



Fonte: Autor.

Finalizado o layout do aplicativo, foi iniciada a construção da interface e seus controles no Flash. Os primeiros estudos sobre captura de movimento com o Flash, ainda na primeira parte do trabalho, foram desenvolvidos com ActionScript 2.0, com base em arquivos encontrados no site FFiles.com (UTELLIOGLU, 2009). Nesse primeiro estudo, o cursor do mouse mudaria de posição junto com a mão do usuário, cuja imagem era capturada por uma webcam. Como a aplicação se baseia em uma espécie de grid imaginário para calcular as posições na tela, uma imagem de uma tabela foi usada como background nesse arquivo.

Primeiramente são chamadas as bibliotecas “greensock”, para uma interpolação de imagem mais dinâmica, e a “BitmapData” para trabalhar com bitmap. A seguir foram criadas as variáveis dos objetos “vídeo” e “cam”. A primeira receberá as imagens capturadas pela webcam através da segunda variável.

```
import com.greensock.*; //importando a biblioteca de interpolação

import flash.display.BitmapData; // importando a classe BitmapData
```

```
var video:Video; //declara o objeto video

var cam:Camera = Camera.get(); //declara o objeto câmera, que capturará imagens
via webcam

video.attachVideo(cam); //atribui a imagem da câmera à variável video
```

Para dar continuidade à execução do aplicativo, é verificada a atividade da câmera:

```
cam.onActivity = function(isActive:Boolean) { //quando houver atividade, executar:

  trace("movimento");

  trace(cam.currentFps)

  trace(cam.motionLevel)

};
```

Após, com base no vídeo, são identificadas as coordenadas possíveis em x e y além de sua largura e altura.

```
var videoX:Number = video._x; //identificação das coordenadas em x do vídeo

var videoY:Number = video._y; //identificação das coordenadas em y do vídeo

var videoW:Number = video._width; //identificação da largura do vídeo

var videoH:Number = video._height; //identificação da altura do vídeo
```

Em seguida, é feita uma proporção entre o tamanho do vídeo na tela e o tamanho da imagem capturada pela webcam, dividindo o valor da primeira pelo da segunda. Também são adquiridas a largura e altura da imagem captada pela webcam.

```
var proporcao:Number = videoW/cam.width; //calcula a proporção do tamanho do vídeo
ao tamanho da imagem captada pela câmera

trace(cam.width) // largura da imagem captada pela webcam
```

```
trace(cam.height) // altura da imagem captada pela webcam
```

Para detectar onde o movimento ocorre foram criadas as variáveis “now” e “before”, onde ambas captam o mapa de bitmaps capturado pela câmera. Esses mapas de bitmaps são praticamente fotos feitas pela webcam. Mais tarde essas imagens serão comparadas para detectar onde ocorre mudança entre elas, determinando assim o movimento realizado diante da câmera.

```
var now = new BitmapData(cam.width, cam.height); //declara e atribui o mapa de bitmaps capturado pela câmera à variável “now”

var before = new BitmapData(cam.width, cam.height); //declara e atribui o mapa de bitmaps capturado pela câmera à variável “before”
```

Os quadrados (células) da tabela somente possuem identificação espacial em relação ao arquivo em que se encontram. Para atribuir uma posição perante o que é capturado pela webcam, foi criada uma variável “quadradoX”. Essa variável assume o valor de x do quadrado (que é conhecido pelo próprio Flash), menos a coordenada x do vídeo (identificada anteriormente). O resultado dessa subtração é dividido pela variável “proporcao”. Os mesmos passos são realizados para atribuir a posição y do quadrado a ser atribuída na variável “quadradoY”.

```
function hitDetect() {

        var quadradoX:Number = (quadrado1._x-videoX)/proporcao //posição x do quadrado 1 menos posição x do vídeo dividido pela proporção

        var quadradoY:Number = (quadrado1._y-videoY)/proporcao // mesmo cálculo anterior, porém com a coordenada y
```

Depois de indicadas as coordenadas x e y dos quadrados, um novo mapa de bitmap é adicionado à variável “now”, ou seja, uma nova imagem é atribuída a essa variável, substituindo a anterior. Assim, as variáveis “now” e “before”, que antes possuíam um mesmo mapa de bitmaps, passam a possuir atributos diferentes, com

“before” apresentando uma imagem capturada anteriormente a que está em “now” nesse momento.

```
        now.draw(video) //a variável “now” é atualizada com o mapa de bitmap
captado pela webcam nesse momento
```

A comparação entre as variáveis “now” e “before” é realizada pela diferença de incidência de vermelho sobre a área do quadrado da tabela. A cor vermelha foi escolhida considerando que a pessoa irá controlar o cursor com a palma da mão estendida diante da câmera. Para verificar a alteração tonal, foram criadas as variáveis “valNowQuad1” e “valBeforeQuad1”. Cada uma recebe um valor de acordo com a incidência de vermelho, descrito como “16 & 0xFF”, sobre a posição x e y de cada quadrado.

```
var valNowQuad1:Number=(now.getPixel(quadradoX, quadradoY) >> 16 & 0xFF);

var valBeforeQuad1:Number=(before.getPixel(quadradoX, quadradoY) >> 16 & 0xFF);
```

Caso haja uma mudança de cor entre “valNowQuad1” e “valBeforeQuad1”, o cursor do mouse vai se movimentar para a posição com mais vermelho, se utilizando do atributo “TweenLite” da biblioteca “greensock”. A diferença entre tons necessária para que o cursor mude de posição foi de no mínimo 30, sendo que quanto menor o valor, maior a sensibilidade.

```
        if (valNowQuad1>valBeforeQuad1+30 || valNowQuad1<valBeforeQuad1-30)
{//verificação da incidência de vermelho nas variáveis

                TweenLite.to(mc, 1, {_x:quadradoX, _y:quadradoY});//movimenta o
cursor

        }
```

Após mudar a posição do cursor, a variável “before” recebe um novo mapa de bitmaps.

```
        before.draw(video) //a variável “before” é atualizada com o mapa de
bitmap captado pela webcam nesse momento

}
```

Para que o processo se repita, foi criada a variável “intervalID”, que chama a função “hitDetect” a cada 30 milissegundos.

```
var intervalID:Number = setInterval(hitDetect, 30);
```

Devido a linguagem descontinuada (ActionScript 2.0), o alto custo de processamento, e tendo em vista a maior complexidade do aplicativo proposto em relação ao estudo, esses trabalhos preliminares foram deixados de lado em um primeiro momento.

Na busca por melhores soluções, foi encontrado o pacote Ostrich (ZEN, 2011), em ActionScript 3, com diversas opções para controle via webcam. O pacote permite que o cursor siga o movimento de vídeo, possibilitando a criação da interface desejada. Para a criação do aplicativo foram utilizadas três classes do Ostrich:

OstrichCamera – configura uma webcam para captura de movimento. A classe detecta se há uma câmera ligada, identifica largura e altura da imagem e espelha a câmera, fazendo com que o vídeo na tela se comporte como um reflexo em um espelho. Ainda possui outras funções, como identificação de áreas da imagem, que não foram empregadas no aplicativo.

OstrichCursor – define um cursor que segue o movimento realizado diante da câmera. Nessa classe o movimento é identificado de forma semelhante ao primeiro estudo para a interação proposta. A imagem é capturada pela câmera e, após 100 microssegundos, outro mapa de bitmaps é gerado. A seguir, as duas imagens são comparadas e as alterações entre elas são identificadas, assim reconhecendo onde o movimento ocorre. Em torno da área do movimento é colocado um retângulo invisível, cujas coordenadas são passadas para o cursor, que passa a se movimentar de acordo com as ações realizadas perante a webcam.

OstrichButton - define eventos para um botão controlado via OstrichCursor. O OstrichButton não permite clique, portanto, para realizar uma ação, é definido um tempo em que o cursor fique posicionado sobre o botão para que a tarefa desejada seja efetuada. Nos menus do aplicativo, optou-se usar dois segundos, ou seja, para que a ação se inicie, o cursor deve ficar na área do botão por esse tempo. A ação vinculada ao botão pode ser a abertura de novos menus ou a exibição do mapa com o caminho entre os pontos de referência.

No Flash, o desenvolvimento do sistema partiu de um arquivo chamado “main”, onde foram colocados o título do sistema, a imagem de fundo, um preloader[4] e um MovieClip[5] chamado “alvo” que carrega os menus ou mapas conforme o optado. Também foram criados os arquivos dos menus, onde eram chamadas as classes do Ostrich e os botões eram inseridos. Botões que levavam a outros arquivos de menu ou aos de mapa. Esse tipo de arquivo possui a descrição do deslocamento entre os pontos e o mapa. Para que o usuário volte ao menu após visualizar a imagem, primeiramente foi pensado em adicionar um botão voltar, porém a inserção do mesmo tornava a aplicação muito lenta, devido a inclusão de todas as funções da Ostrich e para somente uma operação. Assim foi adicionado uma ação que faz com que o menu inicial seja exibido sessenta segundos após a abertura do arquivo-mapa.

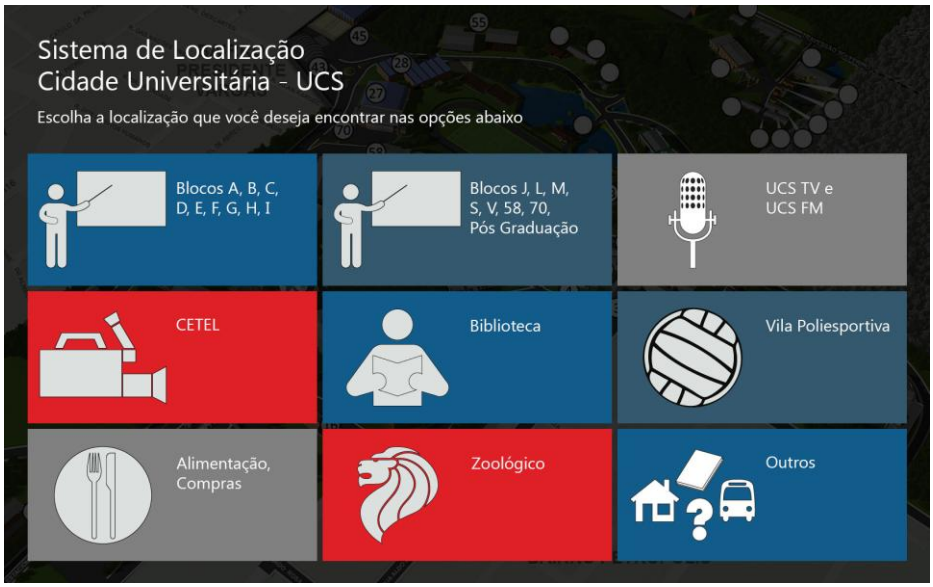
O primeiro problema encontrado foi em relação aos menus. Como no princípio foram criados cinco arquivos com menus (o inicial, dois de blocos, um de alimentação e compras e um com outros locais), cada um possuía botões com destinos específicos, porém, após carregar o primeiro arquivo, os demais acabavam carregando o mesmo script do principal. Para exemplificar, na figura 5 é mostrado a primeira tela de menu.

---

[4] Preloader é a nomenclatura normalmente utilizada para a barra de progresso do carregamento do arquivo.

[5] Movieclip é um símbolo do Adobe Flash geralmente usado para carregar arquivos externos.

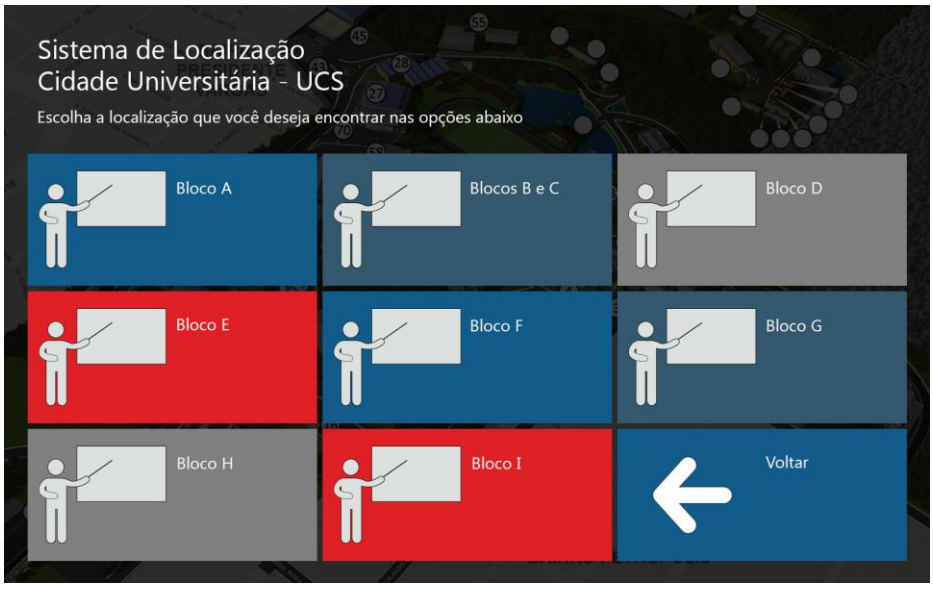
FIGURA 5 – Menu inicial do sistema



Fonte: Autor.

No menu principal, o último e os dois primeiros botões, levam a outros menus, enquanto os demais botões levam diretamente ao mapa referenciado. O primeiro botão desse arquivo abre o menu 2, visto na figura 6.

FIGURA 6 – Menu secundário



Fonte: Autor.

Nesse arquivo, todos os botões levam a mapas, porém, por exemplo, ao escolher “Bloco G” o usuário era levado ao mapa para a vila poliesportiva. O mesmo problema se repetia nos outros botões, ou seja, somente o menu principal estava funcionando e os demais chamavam os destinos dele. Esse problema surgia devido ao script do arquivo principal que não era descartado, ou seja, as funções dos botões de um arquivo secundário não eram carregadas, e os mesmos seguiam as diretrizes do primeiro arquivo.

Como solução, os menus foram adicionados a um único arquivo, com todos os botões e suas funções específicas. Como somente ficariam visíveis os botões do menu inicial, os demais ícones foram colocados fora da cena[6], lado a lado, com o mesmo agrupamento usado anteriormente. Para que os botões desejados possam ser vistos, no MovieClip “alvo” é indicado o deslocamento necessário. Por exemplo, quando o usuário escolher o primeiro botão do menu principal, ele deverá ser levado para a tela com os ícones dos blocos A ao I. Para que essa tela apareça, o “alvo” faz um deslocamento de 1680 pixels (tamanho da cena) para a direita, como uma câmera que se desloca para o lado para filmar outra área do cenário.

O posicionamento do “alvo” resultou em outro problema: o deslocamento ocorria e jogava a visualização para fora da cena, porém o cursor desaparecia, não agindo nos botões. A solução foi identificar em qual parte do menu o cursor estava e alterar sua posição x conforme o necessário. O botão é identificado como “bt1”, “bt2” e assim por diante, e é reconhecido assim que acessado. A ação inicia identificando esse botão. Se for algum que leva o “alvo” para fora da cena, ele faz com que a coordenada x do cursor receba o valor da própria coordenada mais o valor adequado. Por exemplo, ao escolher o primeiro botão, o primeiro submenu é acessado. O programa sabe que o alvo se encontra em “bt1”, cujo destino é a primeira posição fora da cena, com os blocos A ao I. A ação reconhece “bt1” e faz

[6] Cena ou stage é a área visível do arquivo Flash. Elementos podem ser adicionados fora da cena e serem funcionais, porém ficam invisíveis.

com que a coordenada x do cursor receba o seu próprio valor mais 1680, que é o tamanho da cena.

O deslocamento, tanto do “alvo” como do cursor, refletiu no deslocamento do arquivo com o mapa e no menu que aparece assim que o mapa é encerrado. Então, nos arquivos com mapa, foi adicionada uma ação que seta a posição do “alvo” de volta para zero, fazendo com que a visualização do mapa e do menu seguinte fique conforme o desejado.

Um problema de grande impacto no sistema é em relação à precisão do que a câmera captura e sua interferência na interação gestual. A primeira característica importante para aumentar a precisão da interação do sistema é a resolução da tela do computador, que deve ser de 1680 x 1050 pixels, tamanho usado para a cena no arquivo Flash. Tal propriedade é imperativa, pois resoluções diferentes podem expandir ou contrair os botões e consequentemente a área de controle dos mesmos. A segunda propriedade, que infere maior diferença, é relativa a qualidade da webcam. Quanto maior a resolução e qualidade da imagem capturada pela câmera, melhor o resultado da interação. O movimento que mais interfere na interação é o de elementos da cor vermelha, então se o usuário estiver usando uma luva vermelha, por exemplo, a tendência é que ele consiga um resultado melhor na movimentação do cursor na tela.

7 CONCLUSÃO

O desenvolvimento de um aplicativo de localização e um com interação gestual são dois trabalhos a parte. O primeiro é mais comum, visto em aplicações para GPS[7] e até mesmo o Google Maps. Porém o mesmo deve ter cuidado quanto sua precisão e sob o ponto de vista em que ele é aplicado. No trabalho desenvolvido, o sistema de localização não visa à distribuição para o público, portanto não poderia ser utilizado em dispositivos móveis. Com isso, ao invés de adotar distâncias, o mesmo deve guiar o usuário se utilizando de outros pontos de referência, tal como foi aqui aplicado.

A criação de um aplicativo com interação gestual é mais complexa, primeiramente pela considerável falta de material em relação a sistemas de localização. Porém são encontradas na internet algumas bibliotecas, principalmente para Flash, que podem auxiliar no desenvolvimento de tal software.

O maior problema nesse caso é baixa precisão principalmente devido as webcams que não possuem uma qualidade tão boa quanto a câmera usada no Kinect. A utilização de QR Code[8] junto ao aplicativo pode ser estudada como uma forma de melhorar a precisão da captura de movimento pelo mesmo.

O trabalho aqui descrito teve início antes do lançamento do Kinect para computador, o que fez com que todo o desenvolvimento fosse baseado com a captura de imagens via webcam. Um sistema de busca e localização que adote os mapas e descrições adequados, e cuja interação seja controlada por Kinect pode possuir uma precisão muito mais ampla que a gerada por uma webcam comum.

---

[7] GPS é a sigla para Global Positioning System ou sistema de posicionamento global, sistema via satélite que fornece a posição do dispositivo móvel no qual se encontra.

[8] QR Code espécie de um código de barras bidimensional reconhecido quando escaneado por câmeras, geralmente usados em publicidade.

Quanto aos mapas e descrições, os mesmos podem ser melhorados com a modelagem do terreno e a adoção de uma escala mais precisa que a aqui aplicada. A descrição do caminho pode ser trabalhada de forma diferente, mas a aqui utilizada se baseia nos próprios pontos de referência da UCS, o que torna mais simples que uma descrição que diga apenas distâncias.

A adoção de distâncias entre os pontos pode ser relevante em algum aplicativo para *smart phones*, pois parte deles possui GPS, facilitando a orientação do usuário no terreno. Um produto para tais dispositivos também tornaria mais ampla a criação do aplicativo, pois o mesmo seria usado em qualquer ambiente, diferente do criado nesse trabalho, que necessita de especificação do local aonde o computador com o sistema se encontra.

Aumentando a precisão da câmera, a aplicação poderia conter o mapa da UCS de modo permanente na tela, com os menus à esquerda em botões menores. Assim, ao escolher um destino, o caminho poderia ser traçado no mapa em tempo real, aumentando a interatividade do sistema.

Diversos pontos poderiam ser adaptados e melhorados com uma pesquisa maior juntamente com os possíveis usuários do sistema de localização proposto. Saber o que o usuário precisa torna o foco da criação mais específico. Já a interação gestual não tem como ser definida pelo usuário, cabendo ao desenvolvedor buscar melhores maneiras de deixa-la mais precisa, principal ponto deficiente no aplicativo desenvolvido.

8 REFERÊNCIAS

ADOBE. **Adobe to acquire Macromedia**. Disponível em: <http://www.adobe.com/aboutadobe/invrelations/adobeandmacromedia.html>. Acesso em: 5 jun. 2012.

_____. **Programming Adobe ActionScript 3.0**. San Jose: Adobe, 2009.

ALLAIRE, Jeremy. **Macromedia Flash MX - A next-generation rich client**. San Francisco: Macromedia, 2002.

ALON, Jonathan et al. **A Unified Framework for Gesture Recognition and Spatio temporal Gesture Segmentation**. In: *Pre-print, IEEE Transactions of Pattern Analysis and Machine Intelligence (PAMI)*, set. 2009.

ARCHER, John. **Samsung Smart TV voice and gesture control systems review**. Disponível em: <http://www.trustedreviews.com/samsung-smart-tv-voice-and-gesture-control-systems_TV_review>. Acesso em: 3 jun. 2012.

ASUS. **ASUS apresenta o Xtion, sensor de movimento para PCs**. Disponível em: <http://asus.adrenaline.uol.com.br/tecnologia/noticias/12289/asus-apresenta-o-xtion-sensor-de-movimento-para-pcs.html>. Acesso em: 2 jun. 2012.

BARBOSA, Simone D. J.; DA SILVA, Bruno S. Interação Humano-Computador. Rio de Janeiro: Elsevier, 2010.

BENYON, David. Interação Humano-Computador. São Paulo: Pearson, 2011.

BISHOP, Christopher M. **Pattern Recognition and Machine Learning**. New York: Springer, 2006.

BOBICK, Aaron F.; WILSON, Andrew D. A **State-Based Approach to the Representation and Recognition of Gesture**. In: *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 19, num. 12, p. 1325- 1337, dez. 1997.

BRAIN, Marshall. **Como funciona o Nintendo Wii**. Disponível em: <http://eletronicos.hsw.uol.com.br/nintendo-wii.htm>. Acesso em: 05 abr. 2012.

BRIMELOW, Lee. **Six reasons to use ActionScript 3.0**. Disponível em: <http://www.adobe.com/devnet/actionscript/articles/six_reasons_as3.html>. Acesso em: 5 jun. 2012.

BUXTON, Bill. **Multi-Touch Systems that I Have Known and Loved**. Disponível em: <http://www.billbuxton.com/multitouchOverview.html>. Acesso em: 29 mar. 2012.

CAMPI, Monica. **Kinect para Windows reconhece expressões faciais**. Disponível em: <http://info.abril.com.br/noticias/tecnologia-pessoal/kinect-para-windows-reconhece-expressoes-faciais-22052012-19.shl>. Acesso em 3 jun. 2012.

CANTRELL, Christian. **Multitouch and gesture support on the Flash Platform**. Disponível em: <http://www.adobe.com/devnet/flash/articles/multitouch_gestures.html>. Acesso em: 5 jun. 2012.

CHEN, Qing; GEORGANAS, Nicolas D.; PETRIU, Emil M. **Real-time Vision-based Hand Gesture Recognition Using Haar-like Features**. In: *Instrumentation and Measurement Technology Conference Proceedings, 2007*, Ottawa, p. 1-6, mai. 2007.

COLDEWEY, Devin. **Utility Lets Kinect And Flash Play Nice**. Disponível em: <http://techcrunch.com/2011/01/12/making-kinect-and-flash-play-well-together/>. Acesso em: 5 jun. 2012.

DEHAAN, Jen. **Using Flash for the first time – Part 1: Building a banner**. Disponível em: <http://www.adobe.com/designcenter-archive/flash/articles/flacs3it_firstflash_pt1.html>. Acesso em: 5 jun. 2012.

DIX, Alan J. et al. **Human-Computer Interaction**. 2 ed. London: Prentice Hall, 1998.

DUDA, Richard O.; HART, Peter E.; STORK, David G. **Pattern Classification**. 2 ed. New York: Wiley, 2000.

DURA, Daniel. **Using screen orientation APIs for smartphone application development**. Disponível em: <http://www.adobe.com/devnet/flash/articles/screen_orientation_apis.html>. Acesso em: 5 jun. 2012.

FRANÇA, Leonardo. **Desenvolvendo games para Nintendo Wii com Adobe Flash**. Disponível em: <http://www.leonardofranca.com.br/index.php/2010/09/02/desenvolvendo-games-para-nintendo-wii-com-adobe-flash/>. Acesso em: 5 jun. 2012.

_____, Leonardo. **Flash Player no PlayStation 3**. Disponível em: <http://www.leonardofranca.com.br/index.php/2010/09/28/flash-player-no-playstation-3/>. Acesso em: 5 jun. 2012.

_____, Leonardo. **Múltiplas Faces de RIA**. Disponível em: <http://www.leonardofranca.com.br/index.php/2005/12/29/multiplas-faces-de-ria/>. Acesso em: 5 jun. 2012.

_____, Leonardo. **Opções para desenvolvimento mobile**. Disponível em: <http://www.leonardofranca.com.br/index.php/2010/08/19/opcoes-para-desenvolvimento-mobile/>. Acesso em: 5 jun. 2012.

FUKUNAGA, Keinosuke. **Introduction to Statistical Pattern Recognition**. 2 ed. San Diego: Academic Press, 1990.

GAY, Jonathan. **The History of Flash**. Disponível em: <http://www.adobe.com/macromedia/events/john_gay/index.html>. Acesso em: 5 jun. 2012.

GONZALEZ, Rafael C.; WOODS, Richard C. **Processamento Digital de Imagens**. São Paulo: Pearson, 2010.

GREENE, Jay. **Why Metro now rules at Microsoft**. Disponível em: <http://news.cnet.com/8301-10805_3-57370910-75/why-metro-now-rules-at-microsoft/>. Acesso em: 28 out. 2012.

GULDMAN, Andrew - **Guidelines for Flash application development**. Disponível em: <http://www.adobe.com/devnet/flash/articles/ria_dev_guidelines.html>. Acesso em: 5 jun. 2012.

HAUTSCH, Oliver. **Como funciona a Realidade Aumentada**. Disponível em: <http://www.tecmundo.com.br/realidade-aumentada/2124-como-funciona-a-realidade-aumentada.htm>. Acesso em: 5 jun. 2012.

HIMBERG, Johan et al. **The Self-Organizing Map as a Tool in Knowledge Engineering**. In: PAL, Nikhil R. (org.). *Pattern Recognition in Soft Computing Paradigm*. Singapura: World Scientific, 2001. p. 38-65.

HORMBY, Tom. **A History of Apple's Lisa, 1979-1986**. Disponível em: <http://lowendmac.com/orchard/05/apple-lisa-history.html>. Acesso em: 29 mar. 2012.

JOHANSSON, Gunnar. **Visual perception of biological motion and a model for its analysis**. In: *Perception & Psychophysics*, Uppsala, vol. 14, num. 2, p. 201-211, 1973.

JOHNSON, Steven. **Cultura da interface**: como o computador transforma nossa maneira de criar e comunicar. Rio de Janeiro: J. Zahar, 2001.

LAMPINEN, Jouko; LAAKSONEN, Jorma; OJA, Erkki. **Pattern Recognition**. In: LEONDES, Cornelius T. (org.). *Image Processing and Pattern Recognition*. San Diego: Academic Press, 1998. p. 1-53.

MACHADO, André. **Adobe Flash: ainda vale à pena investir?** Disponível em: <http://www.sosblogueiro.com/adobe-flash-ainda-vale-a-pena-investir/>. Acesso em: 5 jun. 2012.

MALOPINSKY, Ivan. **The Evolution of Adobe Flash: From 1996 to 2010**. Disponível em: <http://www.pxleyes.com/blog/2010/07/evolution-of-flash-from-1996-to-2010/>. Acesso em: 5 jun. 2012.

MASSEY, Stephane. **Metro Ui Design Principles**. Disponível em: <http://www.stephanemassey.com/metro-design-principles/>. Acesso em: 28 out. 2012.

MITRA, Sushmita; ACHARYA, Tinku. **Gesture Recognition: A Survey**. In: *IEEE Transactions On Systems, Man, And Cybernetics - Part C: Applications And Reviews*, vol. 37, num. 3, p. 311-324, mai 2007.

NICKULL, Duane. **FLARToolKit - 3D Flash augmented Reality**. Disponível em: <http://technoracle.blogspot.com.br/2009/04/flartoolkit-3d-flash-augmented-reality.html>. Acesso em: 5 jun. 2012.

NIELSEN, Jakob. **Flash: 99% Bad**. Disponível em: <http://www.useit.com/alertbox/20001029.html>. Acesso em: 5 jun. 2012.

PAL, Sankar K.; MITRA, Pabitra. **Pattern recognition algorithms for data mining**: scalability, knowledge discovery and soft granular computing. Boca Raton: Chapman & Hall/CRC, 2004.

PRECIADO, J.C. et al. **Necessity of methodologies to model rich Internet applications**. In: *Seventh IEEE International Symposium on Web Site Evolution, 2005. (WSE 2005)*, Budapeste, p. 7-13, set. 2005.

RAMAMOORTHY, Aditya et al. **Recognition of dynamic hand gestures**. In: *Pattern Recognition 36 (2003)*, num. 9, p. 2069–2081, set. 2003.

REINHARDT, Robert. **Working with the Camera Class - Part 1: Viewing Live Output**. Disponível em: <http://www.communitymx.com/content/article.cfm?cid=2816A>. Acesso em: 5 jun. 2012.

_____, Robert; LENTZ, Jon Warren. **Flash 5, A Bíblia**. Rio de Janeiro: Campus, 2001.

RIVELLO, Samuel Asher. **An introduction to developing games on the Adobe Flash Platform**. Disponível em: <http://www.adobe.com/newsletters/inspire/october2009/articles/article1/index.html?trackingid=EXBIC>. Acesso em: 5 jun. 2012.

_____, Samuel Asher. **Augmented reality using a webcam and Flash**. Disponível em: <http://www.adobe.com/devnet/flash/articles/augmented_reality.html>. Acesso em: 5 jun. 2012.

SCHRAMM, Mike. **Kinect: The company behind the tech explains how it works**. Disponível em: <http://www.joystiq.com/2010/06/19/kinect-how-it-works-from-the-company-behind-the-tech/>. Acesso em: 1 jun. 2012.

SHNEIDERMAN, Ben. **Designing the user interface**: strategies for effective human-computer interaction. 2. ed. Massachusetts: Addison-Wesley, 1992.

SHOTTON, Jamie et al. **Real-Time Human Pose Recognition in Parts from a Single Depth Image**. Disponível em: <http://research.microsoft.com/apps/pubs/default.aspx?id=145347>. Acesso em: 30 mar. 2012.

THEODORIDIS, Sergios; KOUTROUMBAS, Konstantinos. **Pattern Recognition**. 2 ed. San Diego: Elsevier, 2003.

UTELLIOGLU – **Webcam Control** – Disponível em: <http://www.ffiles.com/flash/miscellaneous/webcam_control_2547.html>. Acesso em: 28 out. 2012.

VELOSO, Thássius. **Microsoft anuncia Kinect para Windows (mas Brasil fica de fora)**. Disponível em: <http://tecnoblog.net/87820/kinect-para-windows/>. Acesso em: 3 jun. 2012.

WADLOW, Thomas A. The Xerox Alto Computer. **Byte**, New Hampshire, n. 9, p. 58-68, set. 1981. Disponível em: <http://www.guidebookgallery.org/articles/thexeroxaltocomputer>. Acesso em: 20 de abril de 2012.

WARD, James. **What is a Rich Internet Application?** Disponível em: <http://www.jamesward.com/2007/10/17/what-is-a-rich-internet-application/>. Acesso em: 5 jun. 2012.

WEBB, Andrew R.; COPSEY, Keith D. **Statistical Pattern Recognition**. Chichester: Wiley, 2011.

WILLIAMS, Michael James. **Flash CS6: What's New?** Disponível em: <http://active.tutsplus.com/articles/news/flash-cs6-whats-new/>. Acesso em: 5 jun. 2012.

ZEN, Dan – **Ostrich – Flash / Webcam Motion Cursor** – Disponível em: <http://ostrichflash.wordpress.com/>. Acesso em: 28 out. 2012.