FUNDAÇÃO DE ENSINO “EURIPEDES SOARES DA ROCHA”
CENTRO UNIVERSITÁRIO EURÍPEDES DE MARÍLIA – UNIVEM
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

AUGUSTO LUENGO PEREIRA NUNES

UM FRAMEWORK CONCEITUAL PARA DESENVOLVIMENTO DE APLICAÇÕES DE INTERFACES TANGÍVEIS COLABORATIVAS

MARÍLIA
2010

AUGUSTO LUENGO PEREIRA NUNES

UM FRAMEWORK CONCEITUAL PARA DESENVOLVIMENTO DE APLICAÇÕES DE INTERFACES TANGÍVEIS COLABORATIVAS

Trabalho de Curso apresentado ao Curso de Ciências da Computação da Fundação de Ensino "Eurípedes Soares da Rocha", mantenedora do Centro Universitário Eurípedes de Marília – UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Ciências da Computação.

Orientador
Profº Ms. Leonardo Castro Botega

MARÍLIA
2010

Nunes, Augusto Luengo Pereira
Um framework conceitual para desenvolvimento de aplicações de Interfaces Tangíveis colaborativas / Augusto Luengo Pereira Nunes; orientador: Leonardo Castro Botega. Marília, SP: [s.n.], 2010.
72 f.

Trabalho de Curso (Graduação em Ciências da Computação) – Curso de Ciências da Computação, Fundação de Ensino "Eurípedes Soares da Rocha", mantenedora do Centro Universitário Eurípedes de Marília – UNIVEM, Marília, 2010.

1. Interfaces Tangíveis 2. Sistemas Colaborativos 3. Interação

CDD: 005.428

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Augusto Luengo Pereira Nunes

UM FRAMEWORK CONCEITUAL PARA DESENVOLVIMENTO DE APLICAÇÕES DE INTERFACES TANGÍVEIS COLABORATIVAS

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 10.0 (DEZ)

Orientador: Leonardo Castro Botega

1°. Examinador: Ildeberto de Gênova Bugatti

2°. Examinador: Fábio Dacêncio Pereira

Marília, 03 de dezembro de 2010.

À Deus pelo sopro de vida que há em mim e por permitir que neste projeto eu dedicasse meus esforços;

À meus pais Rosely Luengo e Manuel B. Alves Pereira Nunes, por guiarem meus passos durante minha vida;

À minha namorada Claudia Abe Gargel, pelo amor que me levou a concluir este trabalho.

Aos amigos pelo incentivo;

AGRADECIMENTOS

Agradeço aos professores que lecionaram durante minha graduação em Bacharelado em Ciência da Computação, através dos quais, acumulei conhecimento e incentivos que resultaram na produção deste trabalho.

Ao amigo Davi Yoshinori Cangussu Nakano, pelos incentivos para que pudesse transpor barreias no decorrer do desenvolvimento deste projeto.

Aos colegas de laboratório os quais troquei conhecimentos fundamentais sobre o campo de Técnicas de Interação.

De forma especial ao Prof. Ms. Leonardo Castro Botega, pelo auxílio e orientação prestados durante toda a produção deste trabalho.

“Pense como uma pessoa de ação e aja como uma pessoa que pensa.”

Henri Louis Bergson

NUNES, Augusto Luengo Pereira. **Um framework para desenvolvimento de Interfaces Tangíveis colaborativas**. 2010. 72 f. Trabalho de Curso (Bacharelado em Ciências da Computação) – Centro Universitário Eurípedes de Marília, Fundação de Ensino "Eurípedes Soares Rocha", Marília, 2010.

RESUMO

Sistemas computacionais são empregados em diversas tarefas e áreas do conhecimento humano. Para interagir com tais sistemas, técnicas de interação proporcionam diferentes maneiras de se manipular e obter informação digital. As Interfaces Tangíveis (TUI) representam a classe de sistemas que reconhecem interações executadas sobre artefatos físicos, e as aplica no contexto de seu *software*, característica que permite o surgimento de sistemas de tempo real em áreas ainda não exploradas plenamente por sistemas de Interface Gráfica (GUI), como no campo de gerenciamento de emergências, que dado o aspecto crítico e colaborativo de seu ambiente, representa uma das áreas nas quais TUI agrega maior contribuição. Para a construção de tais sistemas, é ideal que se tenha um modelo conceitual que garanta a correta aplicação do paradigma TUI, agregando aos mesmos, características colaborativas. Considerando tais premissas, este projeto propõe o desenvolvimento de uma aplicação na área de gerenciamento de emergências, chamada *Route Manager* (Gerenciador de Rotas), com base num novo modelo conceitual que agrega as técnicas de interação avançadas do paradigma TUI aplicadas à sistemas colaborativos, chamado TIC (*Tangible User Interface Collaborative*).

Palavras-chave: Interfaces Tangíveis, Sistemas Colaborativos, Técnicas de Interação.

NUNES, Augusto Luengo Pereira. **Proposta de aplicação tangível para gerenciamento de emergências**. 2010. 72 f. Trabalho de Curso (Bacharelado em Ciências da Computação) – Centro Universitário Eurípedes de Marília, Fundação de Ensino "Eurípedes Soares Rocha", Marília, 2010.

ABSTRACT

Computer systems are employed in various tasks and areas of human knowledge. To interact with these systems, interaction techniques provide different ways to manipulate and retrieve digital information. The Tangible User Interfaces (TUI) represents the class of systems that recognizes interactions performed on physical artifacts, and applies them in the context of its software, a feature that allows the emergence of real-time systems in areas not yet fully exploited by Graphical User Interaction (GUI) systems, as in the field of emergency management, that given the critical and collaborative aspect of its environment, represents one of the areas where TUI adds greatest contribution. For the construction of such systems, it is ideal to have a conceptual model that ensures correct application of the TUI paradigm, adding thereto, collaborative features. Considering these assumptions, this project proposes to develop an application in the field of emergency management, called Route Manager, based on a new conceptual model that adds the techniques of advanced interaction of the TUI paradigm applied to collaborative systems called TIC (Tangible User Interface Collaborative).

Keywords: Tangible Interfaces, Collaborative Systems, Interaction Techniques.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

INTRODUÇÃO

O desenvolvimento de sistemas computacionais é uma atividade complexa e que procura atender às necessidades de vários campos da atuação humana. Para orientar tal processo, os *frameworks* conceituais representam a organização dos requisitos para que se possa atender com plenitude a demanda por um sistema computacional. Porém, para que se possa interagir com um sistema computacional, é necessário considerar particularidades técnicas do próprio sistema, e em função da maneira adotada para receber estímulos de um usuário e responder ao mesmo de forma que possa compreender a informação digital, o que impede que um modelo genérico de construção de *software* possa ser empregado para todos os tipos de implementações de interfaces de comunicação com tais sistemas (YUAN *et* al, 2007).

Para cada tipo de interface, encontram-se requisitos específicos que necessitam ser considerados no desenvolvimento de um sistema.

Interfaces Tangíveis (TUI) podem ser definidas como aquelas que compreendem interações realizadas em artefatos físicos, como estímulos para interferir no contexto e representações de informação digital. Para esta classe de interface, modelos de arquitetura de *software* usualmente empregados em sistemas de Interface Gráfica, não atendem os requisitos (FISHKIN, 2004) (ISHII e ULMER, 1997).

A capacidade das TUI em reconhecer interações aplicadas sobre objetos e aplicá-las no contexto de um sistema computacional, abre possibilidade de tornar interações com computadores, mais próximas das interações com processos do mundo real, e para alguns campos de atuação de sistemas, tal característica pode significar melhor desempenho ou maior satisfação de requisitos sistêmicos. No campo de gerenciamento de emergências, por exemplo, informações em tempo real da situação num local de uma ocorrência, com a possibilidade de interagir de forma mais natural com o ambiente retratado num mapa, representa atendimento com melhor desempenho, o que pode reduzir danos e prejuízos. Esta característica é comum em ambientes de processos críticos, onde ainda além da rápida compreensão por parte do usuário da situação atual para que possa realizar a tomada de decisão estratégica, também é desejável que vários usuários possam cooperar na execução de uma mesma tarefa, ampliando assim o atendimento à demanda (RADICCHI *et* al, 2010).

Como objetivo geral, este trabalho propõe um *framework* conceitual para orientar o desenvolvimento de software com TUI aplicada, onde o domínio do sistema seja composto de ambientes no qual a colaboração entre usuários seja requisito fundamental. Tal modelo deve

observar os conceitos do paradigma TUI, nos quais objetos físicos são interpretados pelo sistema como pontos de entrada de interação; além de também conter definições com relação às atividades colaborativas que um conjunto de usuários pode realizar sob um possível sistema construído nos moldes do *framework*, de forma que o contexto das atividades de cada usuário seja mantido com consistência.

O objetivo específico do *framework* apresentado neste trabalho é fornecer subsídio para a construção de um sistema de gerenciamento de rotas para o campo de gerenciamento de emergências, o qual é destinado às plataformas que contenham superfícies sensíveis ao toque e capazes de reconhecer objetos físicos. O sistema desenvolvido neste trabalho tem o objetivo de validar a aplicabilidade do modelo conceitual utilizado. A utilização do *framework* no campo de gerenciamento de emergências não impede que o mesmo seja utilizado em outros ambientes com características colaborativas.

O primeiro capítulo do presente trabalho inicia-se com um levantamento bibliográfico e um estudo da evolução das interfaces computacionais, bem como dos paradigmas e técnicas de interação empregadas na comunicação com computadores ao longo dos anos, para enfim evidenciar a contribuição das interfaces tangíveis neste contexto. Há ainda a definição do paradigma TUI e suas classificações, além do levantamento dos tipos de sistemas decorrentes de sua aplicação.

Finalizando o primeiro capítulo, há a relação de tecnologias utilizadas para o desenvolvimento de Interfaces Tangíveis, e no segundo capítulo estão detalhadas as tecnologias que apóiam a construção de TUI e que foram utilizadas na construção da aplicação produzida por este trabalho sob as diretrizes do *framework*.

O terceiro capítulo demonstra inicialmente a metodologia para construção do *framework* proposto, fazendo uma revisão de trabalhos correlatos nos quais o modelo é embasado, em seguida há a apresentação da aplicação desenvolvida e de suas funcionalidades, demonstrando a integração entre as tecnologias utilizadas além das medidas técnicas utilizadas para programar a aplicação. Ao final, há o relato dos resultados de testes com o sistema desenvolvido.

No quarto capítulo há o relato das conclusões deste trabalho, bem como dos trabalhos futuros que poderão ser realizados à partir conhecimento levantado pelo presente projeto, levando em consideração o novo *framework* conceitual e o exemplo de aplicação construída sob o modelo, destinados à contribuir para o avanço das Interfaces Tangíveis em sistemas colaborativos.

CAPÍTULO 1 - INTERFACES COMPUTACIONAIS

A reunião de todos os aspectos que afetam o uso de um sistema compõe uma definição geral de interface (SMITH, 1982). Em sistemas baseados em computador as interfaces são direcionadas para o tratamento da informação digital e manipulação pelo usuário.

A necessidade de se criar métodos de tradução das linguagens naturais humanas para linguagens que possam ser entendidas por dispositivos computacionais, e vice-versa, impulsionou o surgimento de um vasto campo de estudos. As pesquisas desenvolvidas pela área de Interface Humano-Computador (IHC) tornaram a realização das atividades junto a um dispositivo computacional, mais intuitivas e afastadas de detalhes técnicos ou operacionais. Em resumo a IHC defende que as atividades de um usuário com um computador, devem ser enxergadas pelo usuário em alto nível, ou seja, sua preocupação deve estar apenas no campo do seu problema e não no processo ou capacidade computacional.

A palavra Interação é definida como “Influência recíproca de dois ou mais elementos” (PRIBERAM, 2010). As interações envolvendo um usuário e um computador, compõem a classe estudada pela IHC.

Ao longo da recente história da IHC foram sugeridas várias maneiras de se interagir com dispositivos computacionais, geralmente expressados através de um paradigma atrelado a novos dispositivos de interface e inovações nas soluções de *software*. Um dos primeiros meios a serem adotados era constituído de sequências de chaves eletro-mecânica como entrada de dados, e uma sequência de lâmpadas enfileiradas como saída de dados. Mais tarde surgiu o paradigma de linha de comando, que consiste em textos enviados ao computador através de um teclado, que são interpretados e então executados. Este tipo de interface se consolidou rapidamente em razão de sua eficiência e precisão, o usuário envia ao computador um comando, que é executado em seguida. Ainda hoje são encontrados sistemas ou programas que utilizam este conceito, como por exemplo, terminais Linux ou Unix (CAMPOS, 2006). Porém as interfaces baseadas em linha de comando exigem certo grau de conhecimento do usuário, é necessário que o mesmo saiba sintaxes e atributos dos comandos, o que dificulta o seu uso. Foram realizadas pesquisas para aperfeiçoar o uso deste tipo de interface, chegando a alguns consensos, como por exemplo, o de que o nome do comando deve ser o mais fácil de lembrar em função de seu propósito (MYERS, 1988). A Figura 1 demonstra o fluxo da interação através da interface baseada em linha de comando.

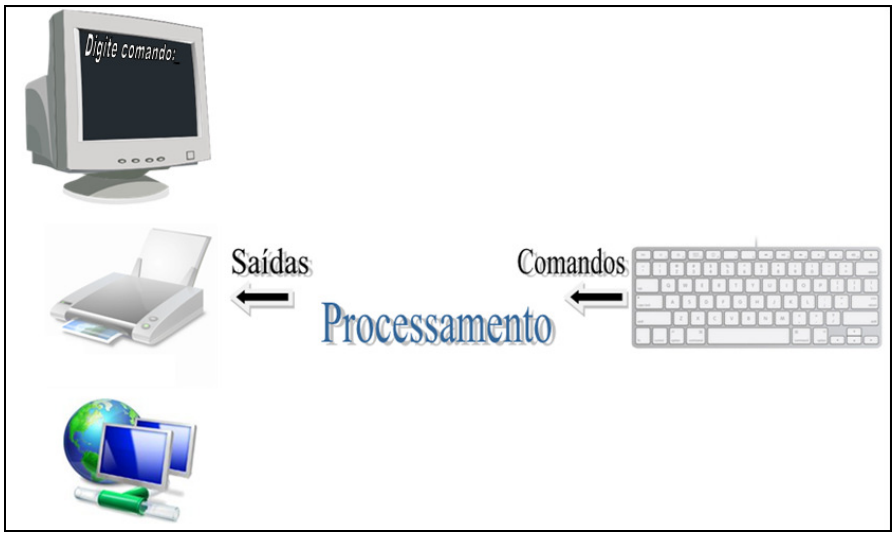

Figura 1- Fluxo da interação da interface por linha de comando

Na década de 1970 começou-se a aplicar representações gráficas para serem projetadas nos monitores, atribuindo a elas valores e funções para os programas executarem suas tarefas.

No início havia apenas caixas de diálogo mostradas aos usuários, que as controlava ainda através do teclado. Pesquisas realizadas com essas novas tendências, principalmente pelos laboratórios *Stanford Research Institute* (SRI) e *Massachusetts Institute of Technology* (MIT) levaram ao surgimento de um novo paradigma de interfaces, denominado WIMP (acrônimo das palavras: janela, ícone, menu e ponteiro), que mais tarde ficou conhecido como GUI (Interface Gráfica de Usuário), e foi usado pela Xerox (XEROX, 2010) para desenvolver um editor de textos chamado BRAVO, que dada a evolução do hardware e software da época, poderia ter mais de uma janela aberta ao mesmo tempo no sistema operacional, juntamente com um novo dispositivo de *hardware* para a interação com o computador chamado mouse, que complementava as interações providas pelo teclado. O editor BRAVO foi inicialmente acoplado ao projeto ALTO da Xerox, mas a popularidade das interfaces gráficas ganhou impulso quando em 1982 a Xerox agregou os conceitos WIMP e o editor BRAVO no seu projeto Xerox Star, e criou o conceito de metáfora de *Desktop,* que era a primeira tela mostrada para seus usuários, onde representações gráficas chamadas ícones simbolizavam arquivos e funcionalidades do sistema (CARD *et* al, 1978). Com este conjunto, foi então difundido o conceito de WYSTWYG (acrônimos da frase "o que você vê, é o que você obtém").

Em seguida a Apple apresentou seu projeto Lisa em 1983 já com as implementações das novas interfaces, e em 1984 também no Macintosh. Outra significante contribuição dos

desenvolvedores da Xerox foi a criação de um mecanismo de impressão que era fiel a imagem projetada na tela do computador, era possível imprimir imagens. O Xerox Star popularizou a metáfora de *desktop*, e através das GUI, propiciou novas experiências aos usuários, com representações virtuais sugestivas como ícones, caixas de diálogo, barras de rolagem, janelas, que davam ao usuário boa fluidez entre suas atividades que então poderiam ser controladas simultaneamente (PEW, 2003). A Figura 2 representa o fluxo da interação numa interface gráfica convencional.

Figura 2- Fluxo de interação da interface gráfica.

No início da década de noventa havia um grande crescimento das redes de computadores, com padronizações bem difundidas no mercado. Neste período foi proposto um conjunto formado pela linguagem de formatação HTML, redes e URLs (*Uniform Resource Locators*) que juntamente com bancos de dados distribuídos, formavam o embrião da *World Wide Web* (WWW), inicialmente um conceito com o nome de Mesh (BERNERS-LEE, 1989). A implementação de um navegador para a WWW provocou uma grande popularização da rede e deste conceito. O MOSAIC foi o primeiro navegador a atuar na WWW. Esta ferramenta foi criada em 1993 e seus conceitos ainda são tomados como padrão por navegadores atuais (PEW, 2003). A Figura 3 mostra uma visualização de uma página através do MOSAIC. As páginas oferecidas na WWW cada vez mais continham tecnologias de interface não convencionais como recursos multimídia, interfaces por comando de voz, interfaces por reconhecimento de gestos e escrita, interfaces de toque. (SHNEIDERMAN, 1998) (FISHKIN *et* al., 2000) (BOUCHET e NIGAY, 2004) (COHEN *et* al, 2004). Estas

tecnologias de interface com o usuário têm raízes na Realidade Virtual (RV), para enriquecer a experiência do usuário com representações virtuais nos ambientes computacionais.

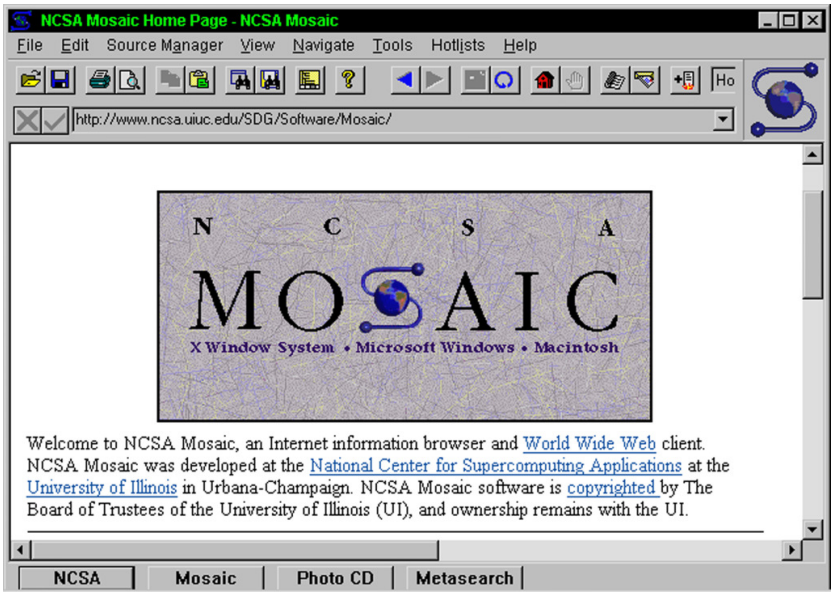

Figura 3- Visualização do navegador MOSAIC para WWW (NCSA, 2010).

A RV é um conceito de interfaces gráficas avançadas que permite aplicações computacionais, na qual haja uma interação em tempo real com usuários, em ambientes tridimensionais sintéticos, utilizando dispositivos multisensoriais (KIRNER *et* al, 1995). Os dispositivos que captam a interação do usuário podem ser convencionais (teclado, *mouse*) ou não convencionais (luva de dados, etc). Essa característica provoca no usuário uma imersão no mundo virtual, como mostra a Figura 4.

Figura 4- Usuário interagindo com mundo virtual (VRCIM, 2010).

Finalmente, as Interfaces Tangíveis (TUI) acrescentam novas capacidades e elementos à detecção da interação com o usuário. Este tipo de interface reconhece interações feitas em objetos reais que estão fortemente ligados a representações virtuais num sistema, podendo assim interpretar qualquer objeto cotidiano como um dispositivo de entrada, e de acordo com suas características e atributos, refletir na representação virtual e produzir as saídas correspondentes. Desta maneira as TUI tornam a imersão e a interação mais significativas, aproximando as maneiras de um usuário interferir no mundo virtual das maneiras usadas para se relacionar com o mundo real, ou seja, torna a interação mais natural. O contexto do sistema é atualizado em tempo real assim que um objeto que está sendo rastreado sofre uma interação (ISHII e ULMER, 1997) (FISHKIN, 2004) (ROGERS e LINDLEY, 2004).

1.1 – Interfaces Tangíveis

As Interfaces Tangíveis (TUI) podem ser definidas como qualquer interface onde o usuário interfere no sistema digital através de dispositivos físicos (ISHII, 2008). Também chamadas de interfaces “agarráveis”, “encorpadas” ou ainda “manipuláveis”, este paradigma pretende através de um sistema computacional rastrear as manipulações de um objeto real feitas por um usuário e produzir saídas correspondentes (FISHKIN, 2004).

Dispositivos capazes de receber as interações “agarráveis” do usuário com as aplicações computacionais, implementam conceitos de TUI e correspondem a uma solução para as limitações naturais das Interfaces Gráficas de Usuário (GUI), como por exemplo, a impossibilidade de se controlar um objeto da interface gráfica que não esteja visível no *display*, ou ainda as dificuldades em se tornar uma tarefa colaborativa num mesmo dispositivo, em termos de interação, já que as GUI permitem que se dê apenas uma ordem por vez ao computador. O paradigma TUI define que objetos reais podem ser interpretados como entradas para o sistema, atrelados a objetos virtuais, e através desta ligação modificar a situação do sistema mediante seu contexto, desta forma, as interações do usuário com o objeto real fornecem dados para a interface, caracterizando a manipulação da informação digital. Em essência, dispositivos desta natureza, misturam interações de artefatos físicos e virtuais, procurando manter uma combinação harmoniosa em tempo real (FITZMAURICE *et* al, 1995) (ULLMER, 1997) (RADICCHI *et* AL, 2010). A Figura 5 demonstra o fluxo de interação de uma Interface Tangível.

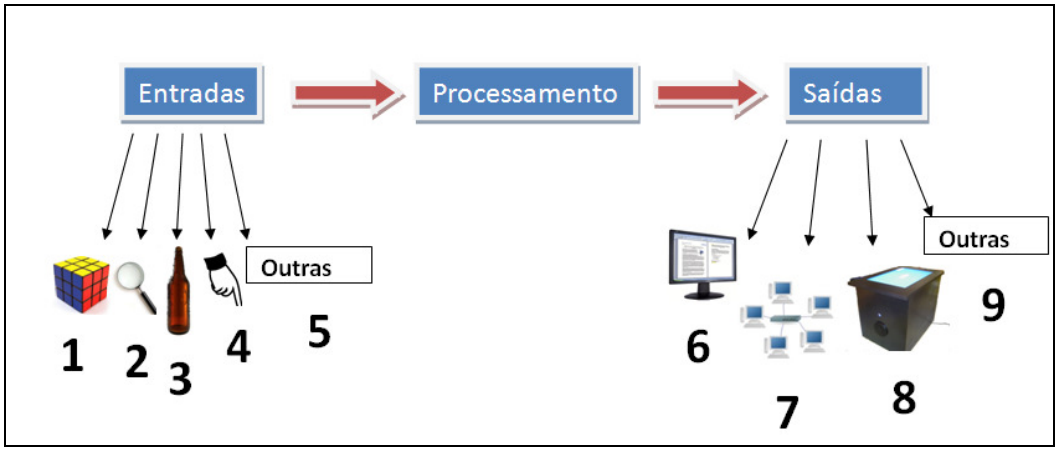

Figura 5- Representação das interações Tangíveis considerando: 1, 2, 3: como objetos de uso cotidiano rastreados pelo sistema; 4: toques em superfícies sensíveis; 5: toda a variedade de objetos reais que podem ser interpretados como entradas para o sistema; 6: *displays*, monitores; 7: ambientes colaborativos; 8: Mesas multi-toque; 9: toda a variedade de dispositivos de saída que podem receber as respostas do sistema (RADICCHI *et* al, 2010).

1.1.1 – Classificações de Interfaces Tangíveis

Existem duas classes de Interfaces Tangíveis, definidas por dois parâmetros principais: metáfora e personificação da interação (FISHKIN, 2004). A metáfora de interface explora a relação entre o objeto tangível utilizado na interação, com algum objeto cotidiano, verificando as características e potencial para compor esta relação, visando tornar a interação mais natural para o usuário. Por outro lado, a personificação estuda a distância entre as entradas da interface e as saídas produzidas, quanto ao dispositivo que capta as entradas e o que exibe as saídas.

A personificação pode ser subdividida em:

- Personificação completa: A interface de entrada é a mesma da saída, ou seja, a saídas produzidas são exibidas no próprio dispositivo que captou as entradas.
- Personificação próxima: A interface de entrada é próxima a de saída, porém as duas mantêm-se separadas.
- Personificação ambiente: As saídas produzidas são exibidas pelo ambiente onde o usuário se encontra, se valendo dos sentidos do usuário, em forma de sons, luzes, etc.
- Personificação distante: A interface de saída encontra-se distante da usada para reconhecer as entradas.

A metáfora de interface também pode ser subdividida em:

- Metáfora de nome: O objeto usado para reconhecer as entradas assemelha-se ao objeto virtual quanto a sua forma ou cor, porém a ação que provocamos sobre tal objeto é diferente da refletida pelo objeto virtual.
- Metáfora de verbo: A ação sofrida pelo objeto real assemelha-se à ação refletida no objeto virtual, desconsiderando sua aparência (LEVIN, 1999).
- Metáfora completa: Diferentemente das duas acima citadas, onde ainda existem diferenças entre o objeto físico e o virtual, esta modalidade estabelece uma forte relação entre ambos os objetos, onde são dispensadas analogias para a compreensão da função ou como utilizá-los (FISHKIN *et* al, 2000).
- Ausência de metáfora: Onde o objeto virtual em nada se assemelha ao objeto físico, configurando a forma mais básica de interação.

1.1.2 Sistemas TUI

Sistemas TUI podem ser encontrados desde em dispositivos móveis, até plataformas colaborativas espalhadas geograficamente.

As Mesas multi-toque, por exemplo, são em geral caracterizadas pela Personificação Completa ou Próxima. Concentram interações com objetos posicionados em sua superfície, modificando seu ambiente virtual em função das entradas fornecidas pelo mesmo. A Figura 6 demonstra um exemplo chamado ReacTable Experience (REACTABLE EXPERIENCE, 2010), onde usuários interagem com objetos rastreados pela Mesa multi-toque.

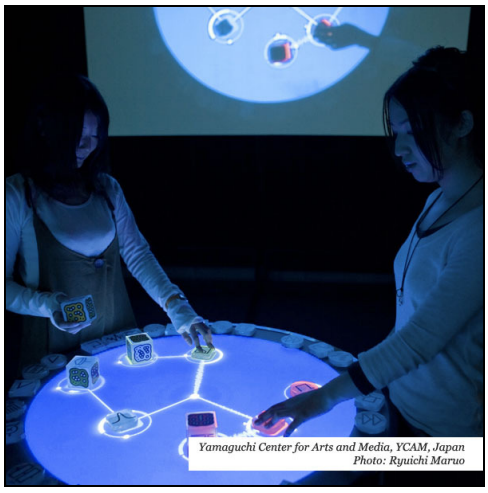

Figura 6- Dois usuários interagindo na ReacTable Experience (REACTABLE EXPERIENCE, 2010).

As Lousas Inteligentes representam outra classe de dispositivos Tangíveis. Em geral suportam atividades colaborativas em ambientes compartilhados remotamente, o acesso pode

ser distribuído. A Figura 7 mostra a transBOARD (TRANSBOARD, 2010), uma implementação desta classe que contempla a capacidade de receber acessos distribuídos, e exibir as saídas das interações dos usuários em tempo real. Este sistema pode reconhecer marcadores colocados na lousa, e atribuir uma representação virtual para os mesmos, dentro do contexto da aplicação. OS chamados *phicons*, são exatamente estes marcadores especiais.

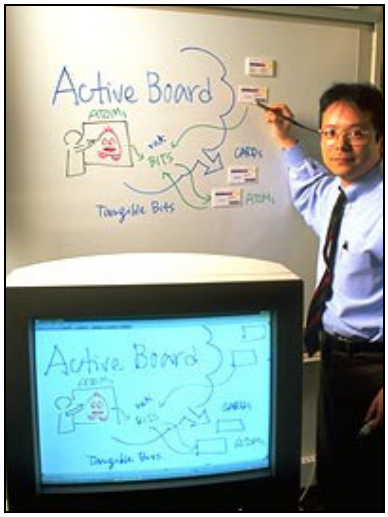

Figura 7- Demonstração de interação com uma Lousa Inteligente (ISHII e ULMER, 1997).

Os Ambientes Interativos são mais uma instância de dispositivos que implementam TUI. São caracterizados pela Personificação Ambiente. As respostas do sistema podem ser dadas visando alcançar os sentidos do usuário, buscando atingir sua atenção, como mostra a Figura 8 na ambientROOM (AMBIENTROOM, 2010), um exemplo deste tipo de sistema. A Figura 9 representa o fluxo de interação desta modalidade.

Figura 8- O ambiente como saída de dados (ISHII e ULMER, 1997).

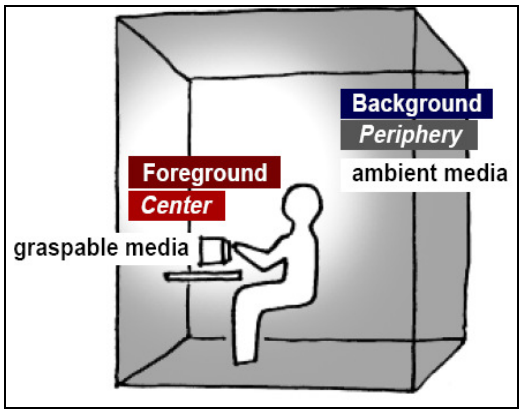

Figura 9- Interação com o ambiente (ISHII e ULMER, 1997).

Vários dispositivos têm agregado conceitos TUI para interface com os usuários. A lista é extensa e vai de carros e celulares à notebooks, pois os dispositivos Multi-toque se tornaram comuns nos nossos dias. A Figura 10 mostra o iPAD (APPLE, 2010) que é um exemplo de dispositivo desta classe, com características de uma Mesa Multi-toque, porém móvel. Além das interações captadas pelos toques no *display*, também são reconhecidos alguns movimentos com o próprio dispositivo.

Figura 10- Interações em superfície multi-toque móvel (APPLE, 2010).

1.1.3 – Aplicações TUI

Várias áreas de aplicação de soluções computacionais têm utilizado os benefícios providos pelas Interfaces Tangíveis.

No contexto de auxílio à aprendizagem, as TUI têm sido aplicadas em experiências com crianças, para estender a absorção de conceitos matemáticos e científicos. O uso didático

de sistemas Tangíveis é aplicado, por exemplo, na Reactable Experience, onde crianças usam o dispositivo que através de som e indicações no *display*, mediante interações com objetos, exploram a capacidade intuitiva. A Figura 11 ilustra a utilização didática de um dispositivo Tangível.

Figura 11- Uso didático de dispositivos tangíveis (REACTABLE EXPERIENCE, 2010).

Conhecidas como Interfaces Tangíveis para Ambientes de Aprendizagem (TICLE - *Tangible Interfaces for Collaborative Learning Environments*), estas aplicações estão voltadas para ajudar crianças a resolver problemas, mediante manipulação de ambiente físico, como por exemplo, a resolução de um quebra cabeças. A TICLE *Table*, mostrada na Figura 12, rastreia as atividades de um usuário na superfície, e um monitor ao lado mostra sugestões para resolver o problema de um quebra cabeças, por exemplo. O usuário pode escolher a opção mostrada através de um toque no *display* a sua frente, e realizar a jogada na mesa (SCARLATOS, 2002).

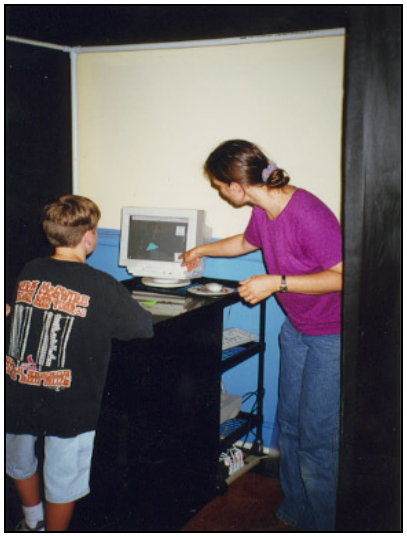

Figura 12- Usuário resolvendo problemas na TICLE Table (SCARLATOS, 2002).

O grande potencial das TUI para aplicações em entretenimento tem desencadeado várias instâncias de dispositivos Tangíveis para jogos. Um dos primeiros exemplos deste tipo de sistema é o PingPongPlus (PINGPONGPLUS, 1998). Trata-se de um jogo de '*ping-pong*' onde a mesa do jogo possui um projetor sob si, onde de acordo com os toques da bola durante o jogo, exibe modificações no campo através de sombras, criando 'buracos', ou apagando uma parte do campo, ou ainda provocando efeito semelhante a toques em superfície aquática, além de produzir sons característicos quando a bola toca o campo (ISHII, 1999). A Figura 13 mostra o PingPogPlus.

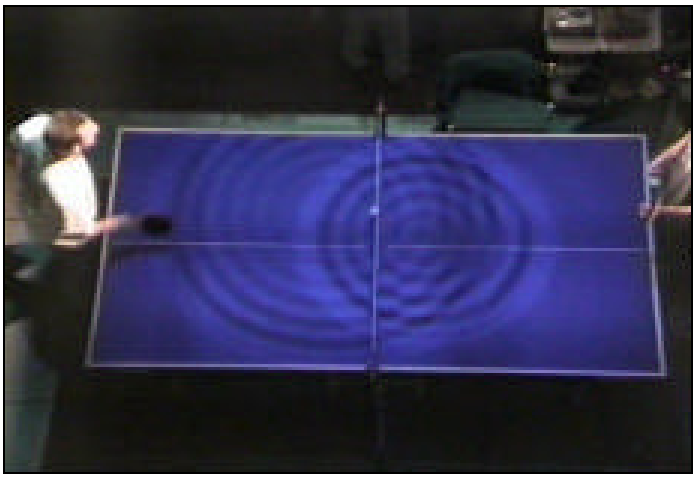

Figura 13- Usuários disputando uma partida no PingPongPlus (ISHII, 1999).

Também encontramos sistemas tangíveis para edição e criação de sons, como é caso do AudioPAD (AUDIOPAD, 2001), uma aplicação que rastreia os movimentos dos objetos colocados numa superfície tangível e transforma-os em som, permitindo total controle da execução do conjunto formado por todas as interações, como ilustra a Figura 14.

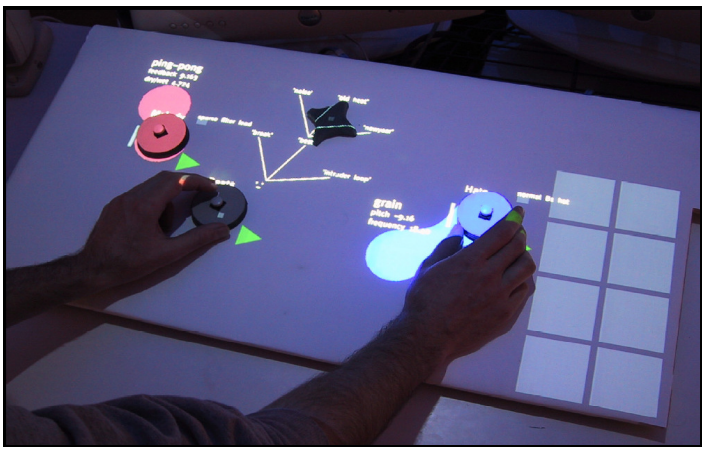

Figura 14- Plataforma para criação e edição de sons (PATTEN *et* al, 2001).

Ainda na classe de entretenimento, o Topobo (TOPOBO, 2003) é um componente de montagem com capacidade de memorização e reprodução de movimentos. O conjunto de Topobos é formado conectando um ao outro, e quando o usuário manipula-o, provocando movimentos, o Topobo reproduz o movimento sucessivamente (RAFFLE *et* al, 2004). A Figura 15 demonstra o sistema Topobo.

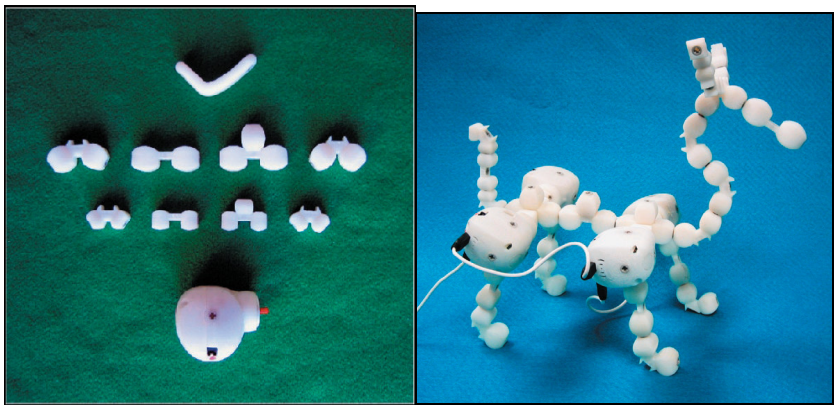

Figura 15- Sistema Topobo (a) e a criação de um animal (b) (RAFFLE *et* al, 2004).

Os sistemas Tangíveis também são usados para suporte a gestão de processos, e tomada de decisões em ambientes críticos. No auxilio à visualização de fluxo de uma cadeia produtiva empresarial, o *Suplly Chain Visualization* (MIT, 2002) é uma plataforma para gestores interagirem fisicamente com o fluxo de seus produtos entre fornecedores e clientes, tendo uma visão global e atualizada em tempo real, permitindo simulações complexas. Uma superfície sensível rastreia objetos que representam tipos de empresas, armazéns, etc. Em conjunto, uma projeção exibi as relações criadas pelo usuário entre estas entidades, dando o *feedback* em tempo real. A Figura 16 mostra este sistema.

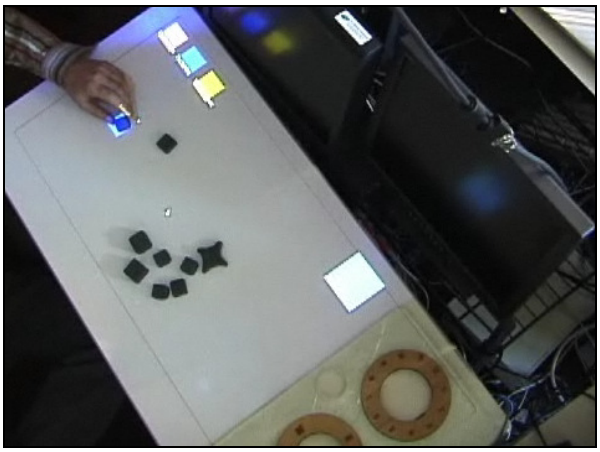

Figura 16- Suporte a tomada de decisão em cadeias produtivas (MIT, 2002).

Também no campo de suporte a tomada de decisão, os sistemas Tangíveis têm sido empregados para auxiliar no Gerenciamento de Emergência, dando visualização em tempo real para resolução de situações críticas. A *Diamond Tangible Table* é uma plataforma para contribuição neste tipo de atividade, onde usuários interagem em conjunto para tomada de decisão quanto a uma emergência (HOFSTRA *et* al, 2008). A Figura 17 demonstra usuários interagindo em torno de uma situação do tipo neste sistema.

Figura 17- uso do dispositivo numa simulação de emergência (HOFSTRA et al, 2008).

1.1.4 – Ferramentas para construção de aplicações TUI

Uma Interface de Programação de Aplicação (API) é definida por Inácio Jr. (2007) como "especificação em linguagem de programação de um módulo de software onde outros módulos podem ou não depender". Dentre as características de uma API, as principais são:

- Ocultar informações: capacidade de restringir o acesso à lógica interna de programação;
- Interoperabilidade: capacidade de agir como ponto de ligação entre sistemas distintos, mesmo se escritos em linguagens diferentes;
- Estabilidade: uma API trata especificamente de um módulo de um software, e este módulo é em geral bem testado, e desenvolvido em compatibilidade com versões e sistemas que o utilizam.

Geralmente uma API é fornecida num Kit de Desenvolvimento de Software (SDK) de uma linguagem de programação. Algumas APIs são protegidas por suas companhias detentoras, ao passo que outras podem ser usadas livremente.

1.1.4.1 – TUIO

Trata-se de um protocolo de domínio público que define uma API própria para reconhecimento de interações em superfícies tangíveis. Este protocolo permite a descrição abstrata de uma superfície interativa, monitorando eventos ou objetos tangíveis, formando um cenário atualizado em tempo real. Os dados captados por câmeras são tratados e decodificados, e exibidos no *display*, e a estrutura deste protocolo permite que aplicações Tangíveis sejam distribuídas, ou seja, disponibilizar a aplicação em rede e dedicar um computador apenas para detecção da interação, outro apenas para atualização do cenário da aplicação, etc. O TUIO foi usado no desenvolvimento da Reactable (KALTENBRUNNER *et* al, 2005). A Figura 18 mostra como atua o TUIO.

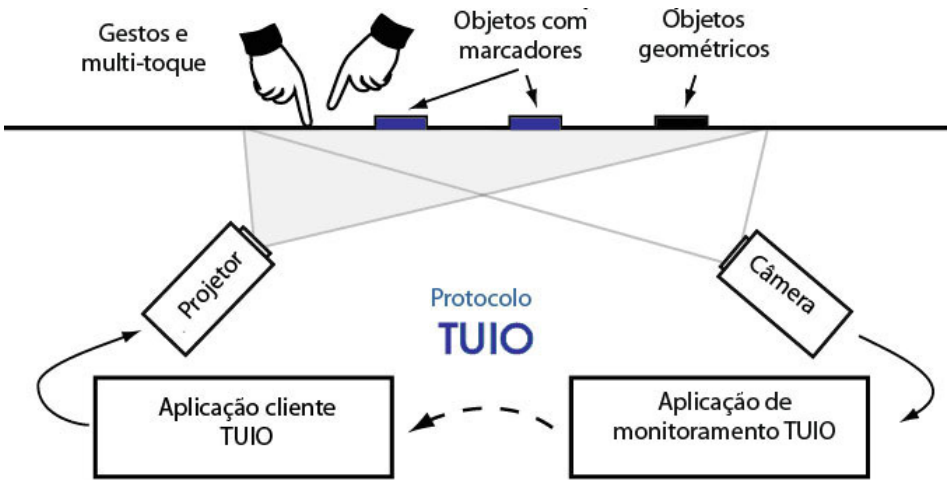

Figura 18- Diagrama de funcionamento TUIO (TUIO, 2010).

1.1.4.2 – Papier Mâchié

O Papier Mâchié é um Toolkit que tem o objetivo de diminuir o esforço do programador em detectar as entradas da interação do usuário numa plataforma tangível. A premissa desta ferramenta se baseia no fato de que, para as interfaces gráficas um programador pode usar ferramentas que construam as interfaces gráficas para ele, não necessitando possuir todo o conhecimento necessário para se programar as interfaces, ficando mais concentrado no tratamento da informação. Do mesmo modo, o Papier Mâchié pretende facilitar as maneiras de reconhecimento da interação, através de várias maneiras como: visão computacional, sistemas de arquivo, códigos de barra, RFID (Identificação por Rádio Frequência) (KLEMMER *et* al, 2004). A Figura 19 demonstra uma aplicação desenvolvida com esta tecnologia.

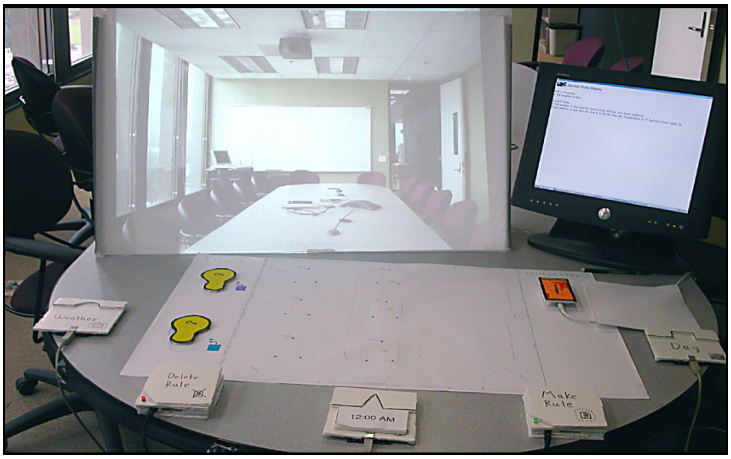

Figura 19- a) cenário monitorado e controlado pelos ícones físicos sobre a mesa; b) ícones físicos para controle do sistema (KLEMMER *et* al, 2004).

1.1.4.3 – Synlab API

Esta API é baseada no Papier Mâchié, porém adicionando reconhecimento de toques em superfícies e objetos através de rastreamento acústico (MAZALEK, 2005)

Implementada no *Synaesthetic Media Lab* do *Georgia Institute of Technology*, esta API foi usada no desenvolvimento do projeto TViews, onde o potencial de múltiplas formas de reconhecimento de interação desta API foi usado para solucionar problemas de alocação de artefatos na plataforma Tangível. A Synlab API é baseada na atualização do contexto disparada por eventos, como mostra a figura 20.

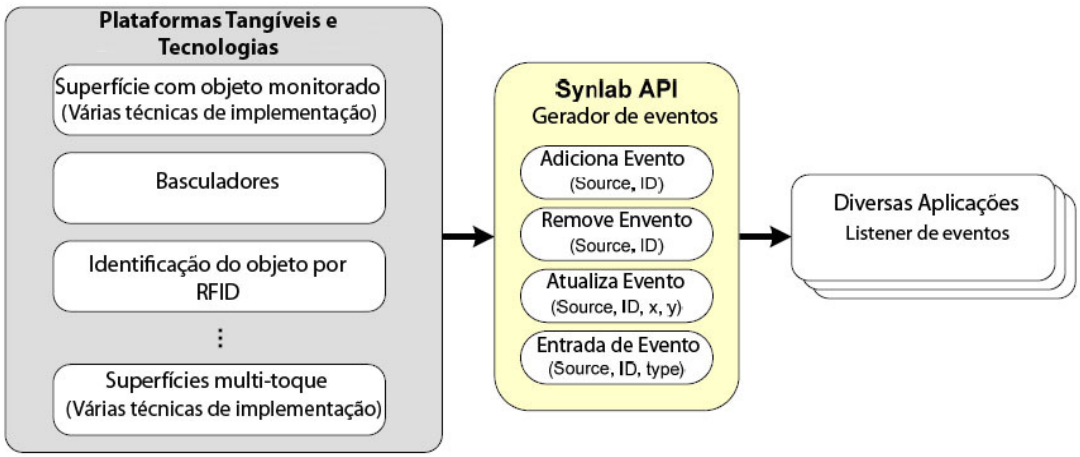

Figura 20- Diagrama de funcionamento baseado em eventos Tangíveis (MAZALEK, 2006).

1.1.4.4 – Phidgets

Foi o primeiro Toolkit para desenvolvimento de aplicações tangíveis, proposta por Greemberg (2001). Trata-se de blocos físicos de construção para interface com o mundo

virtual, com uma extensa biblioteca que pode ser usado em muitas aplicações e em conjunto com outros Toolkits. O módulos são *plug and play* e não necessitam de soldas de componentes eletrônicos. (MOUSSETTE, 2007).

Uma das premissas da Phidgets é que todo objeto físico participante do contexto deve possuir uma representação virtual. Os objetos são rastreados através de RFID, monitorando assim os eventos no dispositivo Tangível. A Figura 21 ilustra arquitetura Phidget, de acordo com Greenberg e Fitchett(2001)

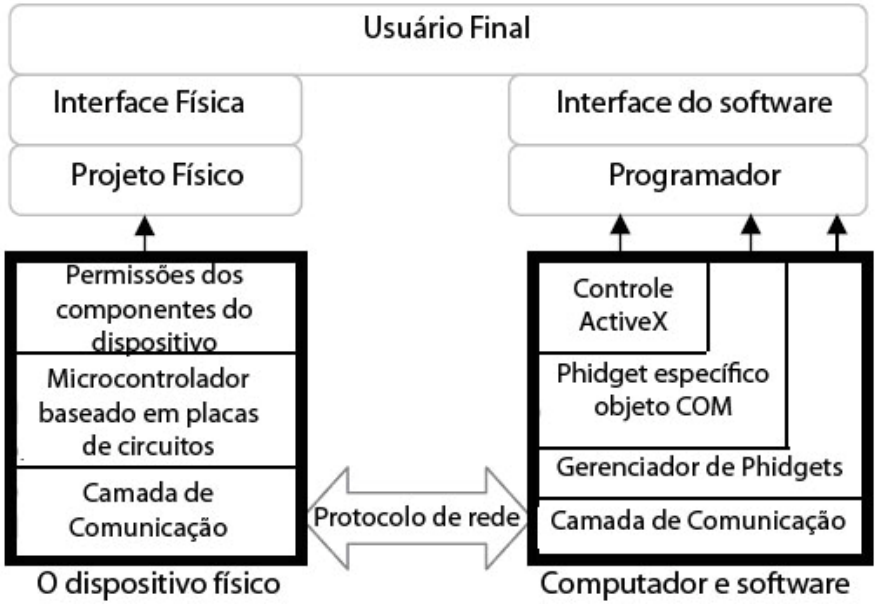

Figura 21- Arquitetura Phidget (GREENBERG e FITCHETT, 2001).

1.1.4.5 – Touchlib

Esta é uma biblioteca específica para criação de aplicações para superfícies multi-toque. Seu funcionamento é baseado na detecção de eventos de toque na superfície através do monitoramento de manchas produzidas por estes toques, e captadas por uma câmera infravermelha, A superfície é iluminada com LEDs infra-vermelho, que correm por toda sua extensão, e no momento em que algo toca a superfície, a direção da luz é alterada, causando as manchas que captadas pela câmera de monitoramento, são interpretadas como interações de um usuário com o dispositivo.. Desta forma a Touchlib fornece ao programador a detecção dos toques na superfície, disparando eventos para a aplicação. Atualmente esta biblioteca é especificada apenas para plataforma Windows, mas existem alguns esforços para torná-la portável para outras plataformas (NUIGROUP, 2010). A Figura 22 mostra a imagem captada pela câmera e monitorada pela Touchlib.

Figura 22- Detecção da interação por monitoramento de imagem em infravermelho (NUIGROUP, 2010).

Existem outras ferramentas que apóiam a construção de TUI, com as quais se pode implementar sistemas que representem as diferentes classificações do paradigma. O presente projeto pretende explorar construção de aplicações TUI colaborativas compostas pela personificação completa e metáfora completa. Tais características definem o *hardware* alvo como uma mesa de superfície sensível e capaz de reconhecer objetos. Para construir uma aplicação sob esses termos, este trabalho utiliza tecnologias O*pen-Source* (Código Fonte Aberto), realizando uma integração entre as mesmas, de forma que possa haver manutenções no código nativo das tecnologias, conforme necessidade.

CAPÍTULO 2 – FERRAMENTAS DE APOIO PARA APLICAÇÕES TUI

As ferramentas utilizadas no presente projeto têm como característica comum o fato de serem *Open Source*. Uma das metas deste trabalho é produzir uma aplicação TUI com

custos mínimos, desde ferramentas para o desenvolvimento do *software*, até mesmo nos componentes e técnicas de reconhecimento de interação, que são implementadas também pelo protótipo de superfície sensível ao toque (projeto paralelo que vem sendo desenvolvido por outro aluno em sua monografia), que deverá ser validado através da aplicação produzida por este estudo.

2.1 – O Framework PyMT

O PyMT é um *framework* desenvolvido para linguagem de programação Python. Ele é direcionado especificamente para implementar aplicações que fazem uso de interfaces não convencionais. Este *framework* pretende fornecer o subsídio necessário para rápido desenvolvimento de softwares que usam técnicas de interação avançadas, e que proporcionam facilidade de uso. Todo o mecanismo de tratamento para as interações tangíveis e portabilidade entre sistemas operacionais, é fornecido pela PyMT, proporcionando ao desenvolvedor um ambiente favorável à resolução de problemas em alto nível, ou seja, o uso da linguagem Python somado às funcionalidades providas pelo *framework*, pretendem tornar fácil a concepção de aplicações com novas formas de interação (HANSEN *et* al, 2009).

Estruturalmente, o PyMT está baseada na detecção de eventos e propagação através de *widgets* (termo utilizado para descrever um componente visual com programação interna nas GUI, para a PyMT um *widget* é qualquer coisa que manipule a entrada de dados, se comunique com a saída ou faça as duas tarefas). A biblioteca de *widgets* da PyMT fornece o necessário para o tratamento das interações numa plataforma sensível ao toque, e também mecanismos que permitem preparar a aplicação para gerenciar mais de um evento ao mesmo tempo, sem perder o contexto individual, devido á utilização de uma máquina de estados no controlador de eventos interno. O monitoramento de estados do *framework* permite que um evento seja propagado dentro da estrutura formada pelos *widgets*, até que algum elemento aplique um tratamento ao evento gerado (PYMT, 2010). A Figura 23 mostra um usuário interagindo com uma aplicação construída com PyMT:

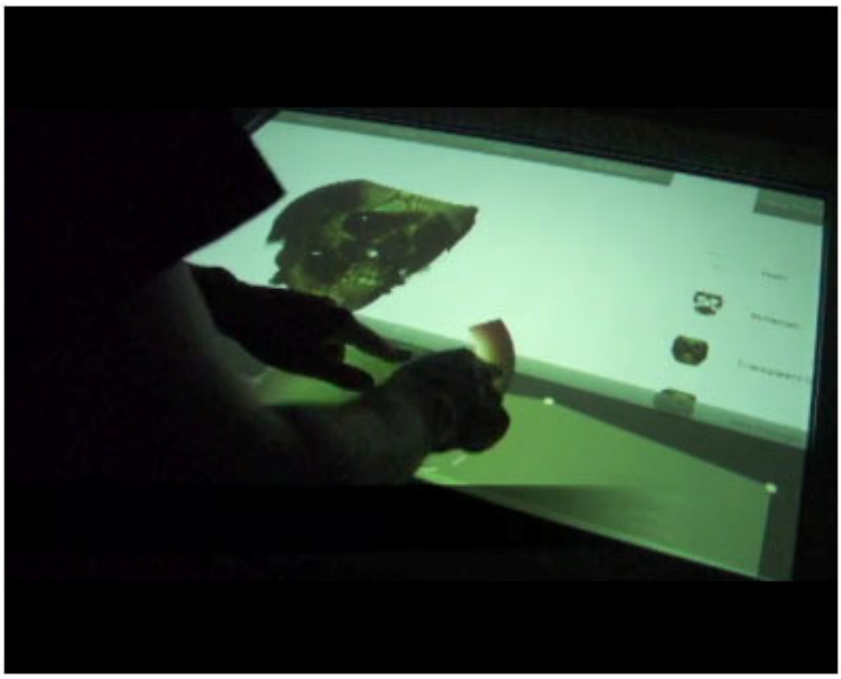

Figura 23: Usuário interagindo com aplicação construída com PyMT (PYMT PLANET, 2010).

2.1.1 – PyMT – Aspectos gerais

Ferramentas para construção de aplicações colaborativas que usam interfaces não convencionais devem estimular a criatividade fornecendo soluções que se encarreguem de cumprir tarefas substanciais e pesarosas, deixando a cargo do desenvolvedor as tarefas relativas à aplicação em si, como defende Greenberg (2003). Olsen (2007) ainda enumera as principais características de uma ferramenta desta classe, como:

- Baixa resistência a boas soluções;
- Redução de barreiras de competências;
- Alimentação em infra-estrutura comum
- Possibilidade de se aumentar ou diminuir o número de envolvidos na manipulação da informação digital;
- Aumento da produtividade;

A maior parte das ferramentas para desenvolvimento de GUI são construídas em torno do paradigma WIMP, porém tais ferramentas não são adequadas para implementações de sistemas TUI, visto que elas são intimamente ligadas ao controle unidirecional de um ponteiro, e sua aplicação em sistemas sensíveis ao toque provocam falta de precisão no reconhecimento da interação. Atrelar estas ferramentas à TUI também diminui as possibilidades de uso e o potencial desta interface não convencional, que pode gerar até aplicações com abordagens totalmente novas (HANSEN *et* al, 2009).

O PyMT implementa características desejáveis para ferramentas de apoio ao desenvolvimento de sistemas TUI, tais como:

- Facilidade de compreensão do código: A sintaxe provida pelo uso da linguagem Python é de fácil entendimento;
- Alto desempenho: Internamente o PyMT utiliza OpenGL (OPENGL, 2009), uma API (gráfica para representações virtuais, com aceleradores gráficos e técnicas de renderização para otimizar o processamento;
- Suporte nativo a interações tangíveis, para superfícies sensíveis ao toque;
- Orienta o desenvolvimento modular do software, incentivando o reuso de código;
- Fornece uma série de *widgets*, com funcionalidades básicas que podem ser utilizados;
- Capaz de reconhecer interações de maneiras variadas, podendo ser aplicado a *hardwares* que utilizem visão computacional, mapeamento magnético, RFID (Radio-Frequency IDentification – Identificação por Rádio Frequência), etc.;
- É um projeto Open Source, mas que pode ser usado comercialmente (LGPL - Lesser General Public License – Licença Pública Geral Reduzida);
- Bem Documentada: A descrição de todas as funcionalidades podem ser encontradas API do PyMT (PYMT, 2010);
- Multi-plataforma: Funciona nos principais sistemas operacionais,
- Constantemente aperfeiçoada: para TUI é desejável que uma ferramenta esteja em constante evolução tal qual o próprio paradigma;

2.1.2 – Linguagem de programação

Um dos requisitos do projeto do PyMT em fase de concepção, foi a portabilidade (HANSEN *et* al, 2009). Essa característica pode ser definida como a qualidade de um componente ou programa computacional, de ser usado em diferentes sistemas operacionais ou computadores (PRIBERAM, 2010). Atendendo à essa configuração, a linguagem de programação Python foi escolhida.

Python é uma linguagem que provê rapidez na implementação, alem de uma sintaxe elegante para o código, desta forma proporcionando menores custos de manutenção. Esta linguagem é livre para uso comercial, devido a sua licença OSI (*Open Source Initiative* – Iniciativa de Código Aberto) e é multi-plataforma, podendo ser usada em vários sistemas

operacionais, e integrada a máquinas virtuais de outras linguagens de programação (PYTHON, 2010).

Muitos módulos para execução das mais variadas tarefas, podem ser encontrados prontos e validados na linguagem Python, o que é uma característica importante para a PyMT, que pretende tornar detalhes das atividades de reconhecimento de interação, transparentes para o desenvolvedor.

A Figura 24 demonstra um exemplo de código escrito em Python, utilizando o *framework* PyMT, onde uma aplicação é definida para reconhecer um evento numa superfície sensível ao toque, e desenhar uma linha seguindo o dedo enquanto pressionado contra a superfície:

```
from pymt import *


class TouchTracer(MTWidget):

    def on_touch_down(self, touch):
        touch.user_data[self] = [touch.pos]

    def on_touch_move(self, touch):
        touch.user_data[self].append(touch.pos)

    def draw(self):
        for t in getAvailableTouches()
            drawLineStrip(t.user_data[self])


runTouchApp(TouchTracer())
```

Figura 24: Exemplo de aplicação TUI escrita com PyMT (HANSEN *et* al, 2009).

A maneira simples e concisa fornecida pela sintaxe da Python, e os *widgets* providos pela PyMT, tonam a compreensão do código consideravelmente mais fácil, inclusive para futuras manutenções.

2.1.3 – Recursos gráficos - OpenGL

OpenGL (OPENGL, 2009) é uma interface para *hardware* gráfico que consiste em um conjunto de centenas de processos e funções disponibilizados para desenvolvimento de objetos e operações envolvidas na produção de imagens de alta qualidade gráfica, especialmente manipulação de cores e objetos em três dimensões (SEGAL e AKELEY, 2010).

A maior parte das operações fornecidas por essa API precisa controlar uma área de memória do computador destinada a armazenar os resultados dessas operações (imagens, transformações geométricas, cores, etc.), conhecida como *framebuffer*. O controle eficiente do

framebuffer proporciona o desempenho necessário para a síntese de imagens complexa. Estruturalmente, a OpenGL utiliza uma máquina de estados para construir o cenário virtual, podendo ser usada para criar animações, aplicando transformações geométricas no ambiente virtual (SEGAL e AKELEY, 2010). A Figura 25 demonstra uma representação gráfica construída com OpenGL:

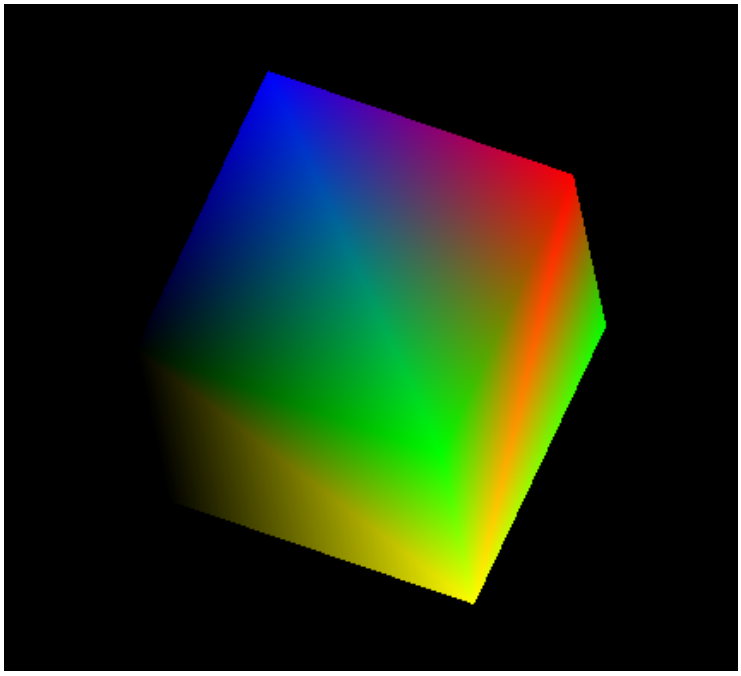

Figura 25: Faces de um cubo com aplicação cores e controle de luminosidade

A PyMT utiliza OpenGL para enriquecer as representações gráficas das aplicações. Todas as construções de representações gráficas em PyMT são feitas usando OpenGL, porém o próprio framework se encarrega de controlar a máquina de estados do OpenGL e tal utilização para a síntese de imagens proporciona bom grau de flexibilidade e customização em termos de produção gráfica, além de bom desempenho (HANSEN *et* al, 2009).

2.1.4 – Arquitetura PyMT

A arquitetura do framework persegue duas principais metas: simplicidade e extensibilidade. De forma geral, um aplicativo PyMT precisa instanciar um elemento raiz para dar início a um *loop* executado pela função *runTouchApp*. Esta é a função responsável por atualizar a máquina de estados do *framework* (HANSEN *et* al, 2009).

Seguindo o exemplo da maioria das ferramentas para desenvolvimento de software GUI, o PyMT organiza seus widgets numa estrutura de árvore, mantendo assim uma hierarquia. Esta disposição facilita a propagação de eventos, que neste *framework* são sempre repassados ao nó raiz, a não ser que a mensagem do evento seja consumida por algum nó filho

durante a propagação. Esta técnica faz com que todos os nós da hierarquia sejam contextualizados com o evento, algo desejável para aplicações colaborativas (HANSEN *et* al, 2009).

As entradas e reconhecimentos de interações são processadas pelo PyMT através dos *Providers* (Provedores). Estes objetos são instanciados automaticamente, por padrão reconhecem entradas providas por dispositivos convencionais como o teclado e o *mouse.* Mas há a opção de se especificar um *Provider* personalizado, que implemente subsídios necessários para reconhecimento de formas avançadas de interação com o computador. Para que essa configuração seja possível, o PyMT exige que três métodos sejam implementados: *start,update e stop*. Estes são os métodos responsáveis por despachar os eventos para o nó raiz, notificando o estado da entrada.

Os *Providers* representam a abertura do PyMT para variadas tecnologias de reconhecimento de interação (HANSEN *et* al, 2009). As principais e melhor suportadas pelo *framework* são as baseadas no protocolo TUIO (KALTENBRUNNER, 2005).

O protocolo TUIO implementa os métodos básicos para PyMT da seguinte forma:

- *Start*: Inicia uma *Thread* (linha de execução para o processador do computador), que se dedica a ouvir mensagens Open Sound Control (OSC) no *host* configurado para o protocolo. A *Thread* organiza todas as mensagens numa filha de acordo com a entrada.
- *Update*: As mensagens na fila são processadas e removidas. Para cada mensagem de estado da representação interna (posições de um marcador, ângulo de rotação, etc.), o evento correspondente é enviado ao *widget* raiz.
- *Stop*: A Thread é finalizada, e deixa de ouvir as mensagens OSC.

Após a atualização dos eventos de todos os *Providers*, e o tratamento destes eventos, o *loop* principal da aplicação inicia a fase final da execução, emitindo um evento para desenhar a cena resultante. A cena é organizada como um grafo, e cada nó é construído graficamente utilizando OpenGL, o que proporciona grande flexibilidade de processamento.

O PyMT ainda disponibiliza ferramentas básicas de desenho (primitivas, formas, cores, etc.) e também classes que acessam propriedades mais avançadas do OpenGL. A figura 26 ilustra a arquitetura PyMT:

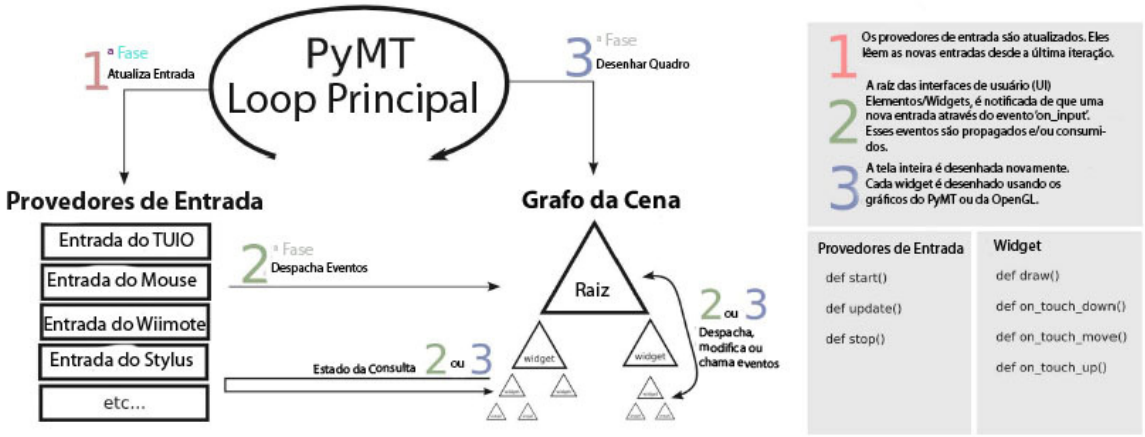

Figura 26: Arquitetura PyMT. A aplicação inicia a fase de entrada no loop principal, na qual os provedores lêem as entradas e despacham eventos para o *widget* raiz da interface do usuário. Ao final da leitura da entrada e eventos, a tela é inteiramente redesenhada (HANSEN *et* al, 2009).

2.2 – O protocolo TUIO

O protocolo TUIO foi inicialmente criado para atender as necessidades do projeto *ReacTable*, fornecendo o subsídio necessário para o monitoramento de marcadores e interações com dedos em superfícies, em aplicações com cenários colaborativos à distância, ou seja, permitindo que a aplicação seja distribuída. Numa descrição simplificada, a meta inicial do protocolo TUIO era fornecer uma simples descrição do estado de ponteiros e símbolos no contexto de uma superfície de mesa em duas dimensões. Ponteiros podem ser definidos como pontos que são localizados dentro das coordenadas cartesianas das dimensões da superfície e não possuem nenhum tipo de marcador especial, e símbolos são etiquetas de identificação que além de posicionamento fornecem ângulo de rotação. Estas funções formam a base do controlador de contexto implementado pelo TUIO, visando a criação de aplicações colaborativas. Tal controlador foi baseado no padrão OSC (*Open Sound Control* – Controle de Som Aberto), pois fornece banda para boa resolução de dados, o que proporciona bom desempenho, algo desejável para sistemas de tempo real. O protocolo TUIO é quem implementa os detalhes de semântica do OSC, que devem descrever um cenário específico de uso (KALTENBRUNNER, 2005).

OSC não especifica uma camada de transporte padrão, por isso TUIO utiliza pacotes UDP para transferências na rede, para garantir a baixa latência da comunicação.

Os resultados do monitoramente da superfície são enviados pelo TUIO como mensagens para uma porta de rede e um endereço IP específico. Por padrão, a porta 3333 e o

endereço *localhost* (127.0.0.1) compõem a configuração inicial, porém estes valores podem ser alterados conforme necessidade.

São dois os tipos principais de mensagens construídas e enviadas pelo TUIO:

- *Set Messages* : Esse tipo de mensagem comunica informações sobre o estado de um símbolo, como posição, orientação, velocidade e aceleração;
- *Alive Messages*: São mensagens que indicam o atual conjunto de símbolos presentes na superfície, atribuindo um número de identificação (ID) para cada um.

Existem mensagens adicionais que compõem o pacote UDP final. As *fseq messages* definem cada atualização dos marcadores no quadro monitorado, e também possuem um identificador único. Basicamente, essas são os três tipos de mensagens presentes num pacote enviado pelo TUIO (KALTENBRUNNER, 2005).

O protocolo TUIO não trabalha com um modelo de eventos para as atualizações de contexto, e sim com um modelo de estados. Isto significa que, o conjunto de elementos identificados pelo controlador na superfície, é enviado como um todo, juntamente com informações de seus estados, atualizando o modelo da interface cliente (disponibilizando as informações na porta de rede configurada). Esta propriedade permite que várias interações ocorram ao mesmo tempo e sejam devidamente identificadas para o tratamento desejado (definido na aplicação que receberá as notificações).

2.2.1 – ReacTIVision

Essa é uma *engine* que implementa o protocolo TUIO e fornece funcionalidades de reconhecimento de interações em superfícies monitoradas por câmeras. A ReacTIVision possui uma biblioteca de símbolos especiais chamados ‘amebas’ (devido à sua aparência), os quais são monitorados dentro de um contexto numa superfície, e fornecem informações como localização, ângulo de rotação, velocidade e aceleração. Além do reconhecimento destes marcadores, essa *engine* também reconhece interações multi-toque, ou seja, oferece suporte a interações com os dedos das mãos (KALTENBRUNNER, 2005).

O principal método de controle e reconhecimento de interações utilizado pela ReacTIVision é baseado na análise de gráfico de região adjacente, originalmente baseado no conceito *Constanza d-touch*. De maneira simplificada, o processo consiste em aplicar um limiar de adaptação à imagem da câmera, a imagem resultante é segmentada numa imagem binária, e então é formado um gráfico de regiões adjacentes a cor preta e branca. Em seguida a identificação dos símbolos ‘amebas’ é baseada na pesquisa num dicionário com estrutura de

árvore, previamente definida. Assim que o símbolo é reconhecido, as informações sobre sua propriedades são carregadas. Desta forma é possível calcular seu ponto central, e seu ângulo de rotação (KALTENBRUNNER, 2005).

Figura 27: Ameba: Este símbolo é gerado automaticamente por um algoritmo genérico, por isso sua aparência lembra o organismo (KALTENBRUNNER, 2005).

A integração entre as tecnologias mencionadas é passível de fornecer subsídio para construção de um sistema que reconheça toques e objetos físicos sob uma superfície monitorada com câmera, e interpretar tais ocorrências como eventos, agregando aos mesmos um vínculo com a informação digital presente no contexto de um *software*. A arquitetura composta por estas tecnologias é baseada em máquinas de estados com relação à superfície, o que permite que mais de um evento seja reconhecido ao mesmo tempo, implicando num princípio de suporte à colaboração entre usuários. Para que se possa enfim agregar atividades colaborativas quanto ao reconhecimento de interações, é necessária a manutenção do contexto do sistema e das atividades individuais, bem com a interpretação do contexto formado pela união das partes providas por cada usuário ativo no sistema. Desta forma, se faz útil a representação teórica da organização da arquitetura de *software*, a fim de garantir que as premissas do paradigma TUI e as especificações de sistemas colaborativos, sejam completamente atendidas, e acomodadas num modelo conceitual.

CAPÍTULO 3 – MATERIAIS E MÉTODOS

A principal meta do presente projeto é propor um *framework* conceitual para construção de aplicativos de Interface Tangíveis colaborativos, nos quais vários usuários possam interagir a fim de atender à uma demanda, ou alcançar em conjunto um objetivo comum. Para tanto, este trabalho apresenta um novo modelo para esta classe de software, validando os conceitos apresentados através de um estudo de caso no campo de gerenciamento de emergências. O ambiente formado por este campo de atuação possui especificações como a colaboração entre entidades, proporcionando que o trabalho de uma especialidade não impeça o atendimento prestado por outra (RADICCHI *et* al, 2010).

Um modelo de sistema que se propõe a comportar interações entre usuários que colaboram para um mesmo fim, deve manter o contexto individual de cada usuário e também o contexto geral formado pela união das partes. No que tange à gerenciamento de emergências, um possível sistema para apoio à esta atividade tem sua atuação na camada estratégica do corpo operacional, formado pelas unidades locais de cada especialidade e os gerentes dos respectivos departamentos.

Sistemas de apoio ao gerenciamento de emergências devem fornecer o subsídio necessário para a tomada de decisão, de maneira que proporcione a diminuição de perda de vida e patrimônio (RADICCHI *et* al, 2010).

A colaboração entre usuários de especialidades diferentes, e a garantia de integridade de contexto das ações dos mesmos, é uma das metas de sistemas de apoio estratégico, e para gerenciamento de emergências, constitui a possibilidade de se organizar um atendimento entre a alta-gerência do corpo de bombeiros, departamento policial, unidades de saúde e defesa civil, afim de que todos os serviços sejam prestados com máximo de qualidade e coerência (RADICCHI *et* al, 2010).

Considerando os requisitos de sistemas de tempo real interativos e colaborativos de Interface Tangível, foi idealizado o *framework* TIC (*Tangible User Interface Collaborative*) para orientar o desenvolvimento da aplicação e que também define tecnologias de reconhecimento de interações para o *hardware* que receberá a aplicação final.

3.1 – *Frameworks* conceituais

Interfaces gráficas possuem várias propostas de organização que definem as camadas de uma aplicação e orientam a maneira como um dispositivo de hardware reconhece uma interação. O MVC (Modelo, Visão e Controle) é uma dessas propostas, onde dividimos a

aplicação em três camadas, mantendo um fluxo de comunicação entre elas, e separando as implementações da interface gráfica, fazendo com que ela esteja no campo de ação do usuário e resposta do sistema (BURBECK, 1992).

A camada Modelo gerencia o comportamento dos dados do domínio de uma aplicação, respondendo às requisições sobre seu estado, normalmente feitas pela camada Visão, ou às requisições de mudança de estado, normalmente oriundas da camada Controle. Esta camada não possui qualquer responsabilidade sobre as demais, servindo como uma referência do estado atual e das propriedades de uma informação (BURBECK, 1992). Por exemplo, pode-se considerar a abstração 'palavra' como um modelo que pode guardar qualquer palavra da língua portuguesa. Para este Modelo, deseja-se uma aplicação que exiba seu estado (conteúdo), e possibilite a troca do mesmo, como num editor de textos. Nestas circunstâncias, o Modelo 'palavra' seria consultado pela camada Visão constantemente, para exibir seu estado no *display* como um texto, e caso a camada Controle ordene a troca de estado (conteúdo) do Modelo, também deverá notificar a Visão de que o estado do Modelo mudou. Esse comportamento caracteriza a camada Modelo como passiva, no contexto da comunicação entre a tríade do MVC.

A camada Visão administra o conjunto de representações gráficas e textos que formam a saída do sistema, baseada no estado de seus Modelos, ou seja, controla exatamente o display exibido para o usuário (BURBECK, 1992). No exemplo do editor de textos, corresponde à área gráfica que contém o texto (estado atual do Modelo), e que procura manter semelhança com a imagem do texto escrito ou impresso numa folha de papel.

A camada Controle recebe as interações do usuário originadas pelo teclado ou *mouse*, e segundo regras implementadas internamente, aplica-as no Modelo alterando seu estado, ou reflete na Visão conforme necessidade (BURBECK, 1992). Um exemplo seria a operação de troca de conteúdo da 'palavra', que será definida por uma entrada do teclado, e a exibição no display, em lugar do conteúdo antigo.

O comportamento flexível e bem estruturado do modelo MVC foi utilizado como base para várias customizações, e para outras propostas igualmente eficazes para organização e arquitetura de software. Porém para TUI, o MVC não contempla o controle exercido por objetos físicos nas representações virtuais, ou, nos Modelos de um sistema. Imaginando uma linha que separe o campo de atuação dos módulos do MVC entre físico e digital, vê-se que a tríade age quase que por completo no campo digital, emergindo para o campo físico na captação de interações por *mouse* e teclado, e no *feedback* das ações, mostrados no *display* de um monitor (ULLMER e ISHII, 2000). Essa característica é natural do modelo, que foi

proposto inicialmente para implementações na linguagem Smalltalk-80 e está ilustrada na Figura 28 (BURBECK, 1992).

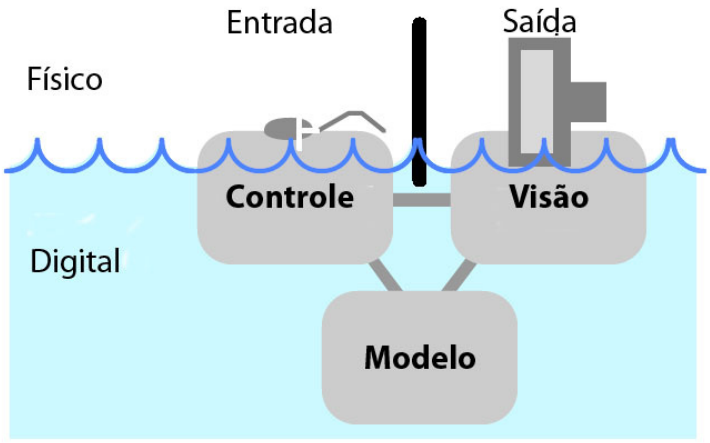

Figura 28: Modelo MVC: Divisão dos campos de atuação físico e digital, dos módulos do modelo de interação (ULLMER e ISHII, 2000).

Baseados no MVC, Ullmer e Ishii (2000) propuseram um *Framework* conceitual que organizava a arquitetura de uma aplicação, proporcionando que a camada Controle pudesse interpretar interações de artefatos físicos. Mantendo a divisão em camadas, o MCRpd (*Model-Control-Representation physical and digital* – Modelo, Controle e Representação física e digital) acrescenta uma nova divisão das camadas, onde o Controle está fortemente vinculado à Visão, que passou ser dividida em dois sub-módulos, um dos quais é responsável apenas por interpretar interações com artefatos físicos (ULLMER e ISHII, 2000). O conceito de 'representação' aqui, deve ser entendido como manifestações externas ao sistema, que são perceptíveis aos seres humanos. Neste contexto, podemos dividir o conceito em duas classes distintas:

- Representações físicas: são representações fisicamente encarnadas, de maneira concreta, as quais a percepção humana não depende de computação para reconhecer, e permanece mesmo quando o sistema não está mais ativo;
- Representações digitais: é o complemento das físicas, as quais são intermediadas por computação para poderem ser percebidas no mundo, e por não serem fisicamente encorpadas a percepção humana capta apenas seus efeitos, que deixam de existir quando o sistema é desativado. É certo que para a percepção humana, mesmo representações digitais precisam ser transcritas em estímulos físicos através de dispositivos como alto falantes para som, monitores para cores e gráficos, mas a essência desses estímulos é intermediada por computação (ULLMER e ISHII, 2000).

De uma forma geral, no modelo MVC as entradas são providas ao Controle por dispositivos periféricos (que não fazem parte do sistema, mas o mesmo é dependente das capacidades genéricas desses dispositivos), e as saídas são exibidas pela Visão em dispositivos gráficos como monitores. Estendendo o campo de atuação dos módulos, o MCRpd manteve o conceito de Modelo do MVC, atrelou o Controle à representações externas ao sistema, e dividiu o módulo visão em duas classes correspondentes às classes de representação digital e física, o que o caracterizou como um modelo emergente do ponto de vista dos campos de atuação dos módulos do modelo de interação.

O módulo Modelo do MCRpd é similar ao Modelo do MVC, servindo como referência do estado de um dado para os demais módulos. O módulo controle passou a estar ligado diretamente às representações de natureza física, as quais gerenciam a informação digital através das interações que recebem por parte do usuário. Temos então dois módulos que ocupam o lugar do módulo Visão, e adéquam o modelo ao paradigma TUI, sendo o módulo 'rep-p' (*representation physical* – representações físicas) o gerenciador das interações providas por artefatos físicos, através de sua manipulação pelo usuário, e o módulo 'rep-d' (*representation digital* – representação digital) gerencia interações digitais e as respostas do sistema que são exibidas em projetores de imagens, ou auto-falantes para sinais sonoros. A Figura 29 ilustra o campo de atuação dos módulos do MCRpd:

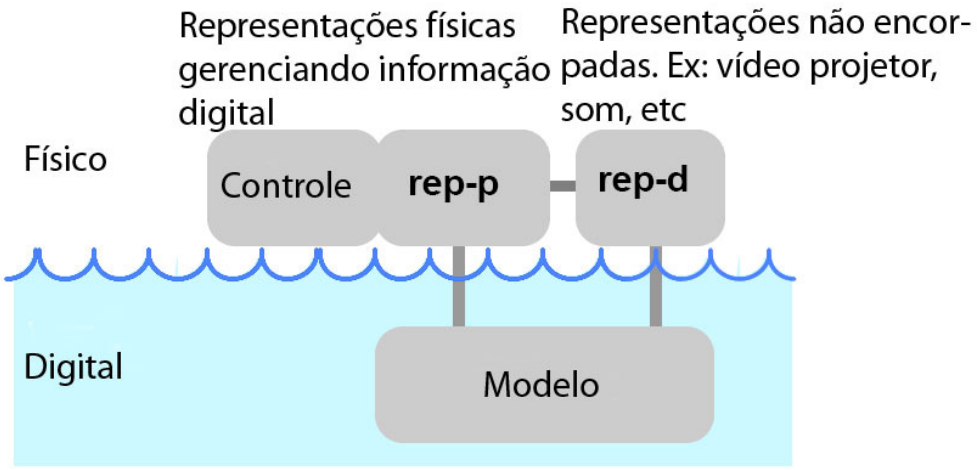

Figura 29: Modelo MCRpd: Divisão dos campos de atuação físico e digital do modelo emergente de interação (ULLMER e ISHII, 2000).

O MCRpd reconhece interações tangíveis devido a sua estrutura modular, porém as requisições feitas ao Modelo têm características específicas que as diferem de uma requisição ao um Modelo MVC feitas por seu Controle. Este fato inviabiliza o reconhecimento de uma interação provida por um periférico como o teclado por exemplo. Uma entrada do teclado não

tem tratamento correspondente nos módulos do MCRpd, para que seja aplicado ao Modelo, e causar alteração de estado (YUAN *et* al, 2007).

Seguindo a linha de *Frameworks* conceituais emergentes, o modelo TIPMR (*Tangible user Interface within Projector-based Mixed Reality* – Interfaces Tangíveis baseado em Projetor e Realidade Mista) estende o MCRpd para apoiar sistemas que utilizam projetores para exibição de saídas gráficas e composição de superfícies sensíveis ao toque, e adiciona a capacidade de reconhecimento de interações providas por dispositivos periféricos (YUAN *et* al, 2007). Os módulos do TIPMR são correspondentes aos do MCRpd em funcionalidades, com a adição de um módulo Assistente entre o conjunto Controle e Rep-p, e o módulo Modelo. O Assistente tem a função de traduzir as interações de natureza tangível, em sequencias de interações providas por *mouse* e teclado, e as aplica no Modelo, que também recebe interações de periféricos que se comunicam nativamente através do módulo Rep-d. Em outras palavras, o Assistente torna homogenias requisições de alteração de estado ao Modelo.

O TIPMR foi utilizado para desenvolver aplicações nas quais um único usuário interagia com mapas projetados numa superfície, através de um projetor, e suas ações eram traduzidas para interações de *mouse* e teclado para que navegasse pelo mapa. A projeção do mapa de navegação era feira no chão, abaixo do usuário, que ao caminhar pelo mapa marcava um caminho e alterava uma segunda projeção colocada a sua frente que mostrava a paisagem do local em que ele se encontrava dentro do mapa. A Figura 30 ilustra a organização modular do TIPMR:

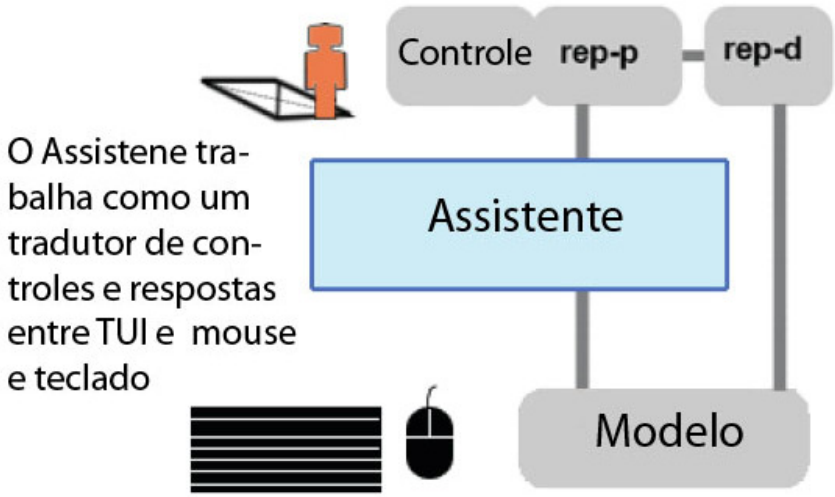

Figura 30: Organização modular do modelo de interação TIPMR (YUAN *et* al, 2007).

3.1.2 – Novo *Framework* proposto: TIC

Frameworks emergentes baseados no modelo MVC formam a base conceitual do modelo TIP-CE (*Tangible User Interface Collaborative* – Interface Tangível Colaborativa). Este modelo é uma extensão do TIPMR, herdando, portanto as características modulares MVC e MCRpd, e o Assistente, mas voltado para aplicações colaborativas e especificamente para gerenciamento de emergências. O TIC adiciona um novo módulo à estrutura TIPMR, responsável por manter o contexto das ações dos vários usuários que interagem com a aplicação através dos artefatos físicos, ou dispositivos periféricos, além de orientar as técnicas de reconhecimento da interação tangível para dispositivos de superfície sensível ao toque e colaborativos.

3.1.2.1 – Módulos do TIC

O TIC possui módulos herdados da linha de *Frameworks* emergentes baseados no modelo MVC, com a adição de um módulo próprio chamado Main-c (*Maintainer Context* – Mantenedor de Contexto), desta forma, as camadas de aplicação seguem descritas abaixo:

- Modelo: gerencia os dados do domínio de uma aplicação, em caráter passivo, recebendo requisições de consulta de estado por parte das camadas de apresentação e controle de contexto, e mudança de estado por parte da camada de controle da aplicação (através de um interpretador);
- Controle e Rep-p: gerencia as representações físicas, dando a capacidade de interpretar interações sofridas por artefatos físicos, e as repassa ao módulo integrado Controle, desta forma manipulando a informação digital de acordo com as interações tangíveis;
- Rep-d: gerencia as representações digitais, normalmente ligadas a sons e imagens gráficas produzidas pelo sistema e direcionadas para algum dispositivo como monitor ou caixa de som. Esta camada também recebe as interações intangíveis, providas por dispositivos periféricos, e se comunica com a camada Controle integrado com Rep-p, e com a camada Modelo para requisições de estado mantendo as representações digitais sempre atualizadas;
- Assistente: Funciona como um interpretador de interações tangíveis, traduzindo-as em sequencias de interações de *mouse* e teclado, tornando homogenias as requisições à camada Modelo. A camada Assistente é quem repassa as requisições de mudança de estado feitas pela camada Controle que está integrada à camada Rep-p;

- Main-c: Essa camada se comunica apenas com a camada Modelo, e gerencia os relacionamentos, propriedades e ações à respeito dos modelos correntes no contexto da aplicação. Exemplo: Dois usuários traçando rotas com artefatos físicos sob uma mesa de superfície sensível ao toque, podem passar pelo mesmo ponto de um mapa projetado na superfície, porém cada usuário tem sua própria rota.

Os módulos do TIC comportam toda a aplicação com a implementação única das camadas Modelo e Main-c, isso significa que as demais camadas podem ser implementadas mais de uma vez, e distribuídas em mais de um dispositivo de superfície sensível ao toque, de maneira que o contexto entre as ações dos usuários nos artefatos físicos serão considerados em todos os dispositivos dentro de um mesmo contexto. A Figura 31 demonstra os módulos do TIC para uma aplicação sem distribuição das camadas de reconhecimento da interação e apresentação, e a Figura 32 mostra uma possível distribuição das camadas Controle e Rep-p, Rep-d e Assistente, que viabilizariam uma colaboração remota entre dispositivos tangíveis e usuários dispersos:

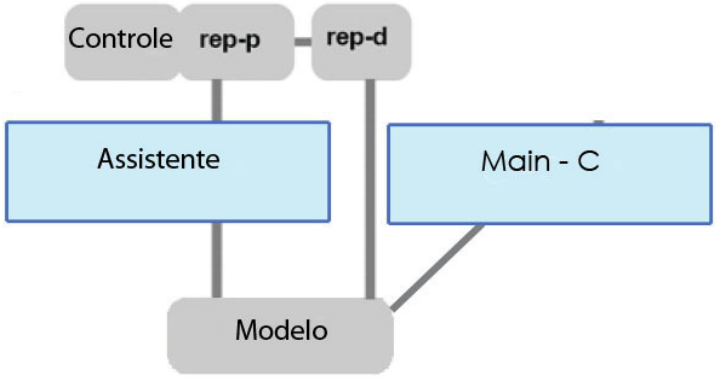

Figura 31: Organização modular do modelo de interação TIC.

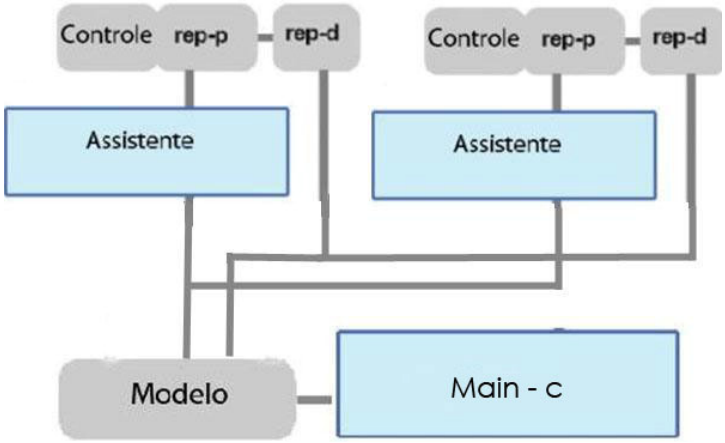

Figura 32: Distribuição de camadas superiores mantendo contexto da aplicação.

3.2 – Estudo de Caso: Route Manager (Gerenciador de Rotas)

Para validar o *framework* TIC este trabalho utilizou do desenvolvimento de uma aplicação TUI para gerenciamento de emergências, cujo domínio da aplicabilidade está na definição das melhores rotas para unidades de atendimento como carros de bombeiros, unidades policiais, e unidades de saúde, em situações de emergência. Utilizando as ferramentas de apoio a desenvolvimento de aplicações TUI, citadas no capítulo anterior, foi implementado um ambiente de desenvolvimento que integra essas tecnologias, provendo o subsídio necessário para desenvolver o sistema *Route Manager*.

O *Route Manager* é um sistema de apoio à tomada de decisão a nível estratégico, para a definição das melhores rotas para unidades especializadas de atendimento à emergências. Este aplicativo foi desenvolvido para funcionar especialmente sob uma mesa de superfície sensível ao toque, construída sob o *framework* TIC. Seu funcionamento baseia-se na disponibilização de um mapa da região da cidade onde houve a chamada para uma emergência. Carros em miniatura são usados para traçar a rota que deverá ser tomada pelas unidades de atendimento. No momento em que a mesa identifica um novo carro na superfície, a aplicação solicita o registro do usuário, previamente cadastrado no sistema, e permite sua interação no mapa. Vários usuários podem interagir com carros, definindo as melhores rotas com base no mapa projetado na superfície.

3.2.1 – Tecnologias empregadas

O modelo TIC foi usado como referência para a integração das tecnologias, de forma que todos os módulos fossem representados dentro da arquitetura da aplicação. Algumas das definições inicialmente pensadas para desenvolver o software não foram empregadas devido às mudanças nas especificações do protótipo desenvolvido por Adriel de Oliveira Radicchi (RADICCHI *et* al, 2010), que compõe a plataforma para validar a aplicação final.

3.2.1.1 – Sistema Operacional

Windows XP (MICROSOFT, 2010) é o sistema operacional (SO) no qual o projeto foi desenvolvido. Mesmo a linguagem de programação sendo portável para outras plataformas, limitações de hardware tornaram-se um impasse no desenvolvimento da aplicação. Alguns detalhes da implementação do protótipo de dispositivo TUI não puderam se ajustar à plataforma Linux nas distribuições Ubuntu 10.04, Ubuntu 10.10 e openSuse 11.3

(UBUNTU, 2010) (OPENSUSE, 2010), devido à incompatibilidade de *drivers* (componentes necessários para a utilização de um hardware por um sistema operacional) da webcam utilizada no projeto. Outro limiar que definiu o SO utilizado, foram as tecnologias como o *framework* PyMT e a *engine* ReacTIVision, que atualmente têm versões compatíveis para Windows, apenas para a versão XP.

Utilizando este SO, foi possível integrar o IDE (*Integrated Develpoment Environment* – Ambiente de Desenvolvimento Integrado) NetBeans (NETBEANS, 2010), com a linguagem Python e o *framework* PyMT. A Figura 33 mostra uma parte do código da aplicação construída neste ambiente:

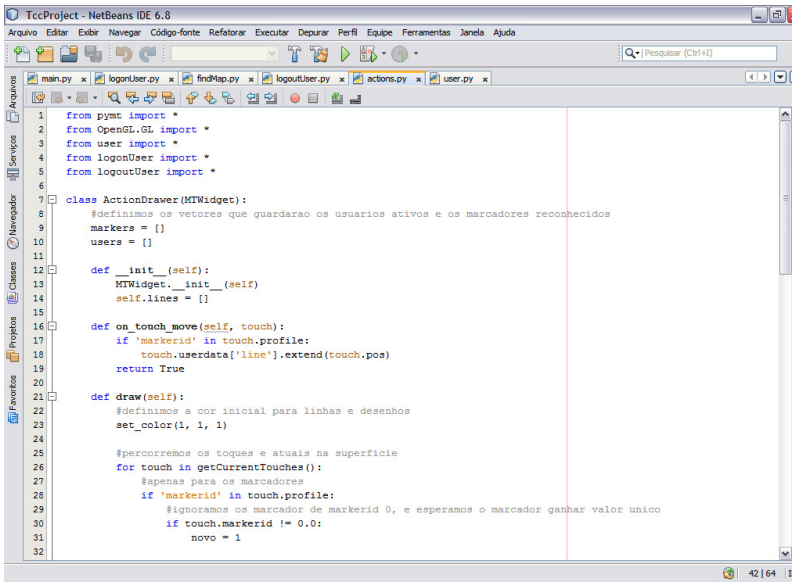

Figura 33: Ambiente de desenvolvimento com Windows XP e NetBeans.

3.2.1.2 – Integração entre ReacTIVision e PyMT

A ReacTIVision foi utilizada para compor o módulo Rep-p, no qual interações de natureza física identificadas por ela, são disponibilizadas numa porta de rede através do protocolo TUIO, com as propriedades da interação. O monitoramento constante executado pela ReacTIVision, funciona como uma máquina de estados que contém os toques correntes presentes numa superfície monitorada por câmera. Ao identificar uma mudança no estado desta superfície, como um novo toque ou um novo marcador, a *engine* identifica o tipo de toque ou marcador, e carrega suas propriedades na porta de rede, que pode ser lida por outra aplicação cliente do protocolo TUIO. A Figura 34 demonstra um exemplo de reconhecimento de marcador pela câmera utilizada para monitoramento da ReacTIVision:

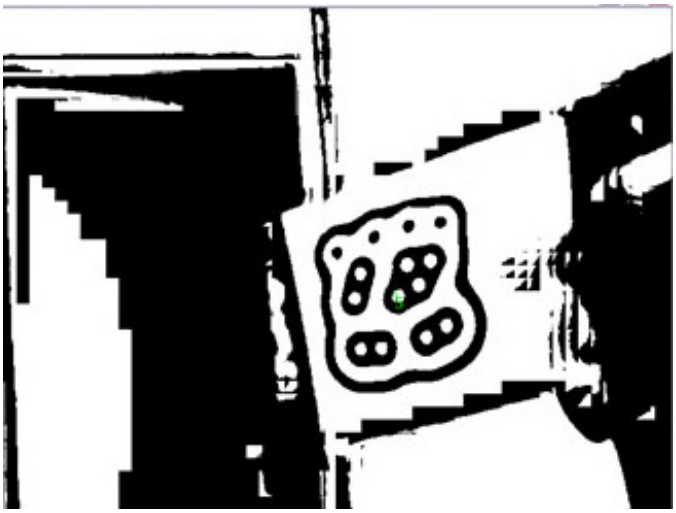

Figura 34: Reconhecimento de marcador pela câmera de monitoramento da ReacTIVision. A *engine* fica em um *loop* atualizando na porta de rede o estado dos toques e marcadores reconhecidos.

O PyMT possui uma implementação de cliente TUIO nativa chamado *provider* (provedor), e possibilita a criação de outros clientes personalizados. Esse objeto do *framework* realiza a leitura de uma porta de rede, e quando configurada como a definida para as mensagens da ReacTIVision, interpreta as mensagens e gera eventos que são aplicados a objetos gráficos chamados *widgets*, quando estão na mesma posição relativa do objeto reconhecido pela ReacTIVision. Desta forma é possível 'tocar' num objeto gráfico apenas mostrando um artefato físico com o marcador na sua base, e inclusive girar e realizar outras operações de manuseio, conforme implementados na aplicação final. A Figura 35 demonstra uma aplicação exemplo, parte do pacote do PyMT, interpretando eventos gerados pela ReacTIVision, e aplicando nas fotos de um quadro:

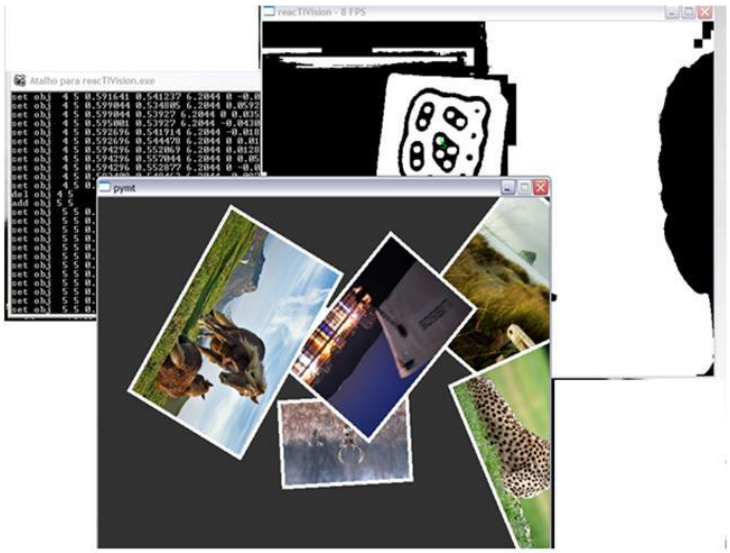

Figura 35: Evento sendo atribuído à imagem, á partir de marcador reconhecido pela ReacTIVision.

As informações produzidas pela ReacTIVision em relação à toques e marcadores, servem como argumentos para os programas do PyMT. A união entre o monitoramento da *engine* e a correspondência e interpretação do *framework,* facilita o desenvolvimento de aplicações para superfícies sensíveis ao toque, e aplicando o modelo TIC, temos a base para a construção de um software Tangível.

A ReacTIVision trata toques e marcadores como objetos, e para cada objeto ela capta as características e mapeia suas informações. Toques identificados como originados pelos dedos da mão, fornecem a posição dentro do plano cartesiano traçado para o campo de visualização da câmera. Já os toques gerados por marcadores, fornecem posição, número sequencial de identificação, número de identificação da lista de marcadores conhecido como 'fid'(*fiducial identificator* – identificador fiducial), entre outros (KALTENBRUNNER, 2005). O *provider* TUIO nativo do PyMT não é capaz de captar e tratar o número identificador da lista de marcadores da *engine*, e por isso na implementação do *Route Manager*, foi necessário criar uma versão customizada que agregasse tal capacidade. A Figura 36 demonstra uma parte da implementação do *provider* utilizado no *Route Manager*, que identifica a quantidade de argumentos disponibilizados pela ReacTIVision:

```
class Tuio2dObjTouch(TuioTouch):
    '''A 2dObj TUIO object.
    '''
    def __init__(self, device, id, args):
        super(Tuio2dObjTouch, self).__init__(device, id, args)

    def depack(self, args):
        if len(args) < 5:
            self.sx, self.sy = args[0:2]
            self.profile = ('pos', )
        elif len(args) == 9:
            self.fid, self.sx, self.sy, self.a, self.X, self.Y,
            self.A, self.m, self.r = args[0:9]
            self.Y = -self.Y
            self.profile = ('markerid', 'pos', 'angle', 'mov', 'rot', 'motacc', 'rotacc')
            self.i = self.fid
        else:
            self.fid, self.sx, self.sy, self.a, self.X, self.Y, self.A, self.m, self.r, width,
            height = args[0:11]
            self.Y = -self.Y
            self.profile = ('markerid', 'pos', 'angle', 'mov', 'rot', 'rotacc',
                            'acc', 'shape')
            self.i = self.fid
            if self.shape is None:
                self.shape = TouchShapeRect()
                self.shape.width = width
                self.shape.height = height
        self.sy = 1 - self.sy
        super(Tuio2dObjTouch, self).depack(args)
```

Figura 36: Parte da implementação do *provider* TUIO. A quantidade de argumentos identifica o tipo de objeto reconhecido.

O monitoramento da superfície realizado pela ReacTIVision agrega à aplicação a capacidade de reconhecer interações provocadas por toques na superfície gerados por dedos ou artefatos físicos, porém não reconhece interações originadas por dispositivos periféricos como *mouse* e teclado. Para agregar tal capacidade, o *Route Manager* utiliza outros *providers* que funcionam em paralelo com o *provider* TUIO, e se encarregam de capturar e transcrever os eventos gerados por periféricos. O conjunto destes *providers* aplicado na tradução de interações, compõe a camada Assistente.

3.2.1.3 – Google *Maps* e Selenium *Remote Control*

Google *Maps* é um serviço de pesquisa e visualização de mapas do planeta Terra, gratuito, fornecido através da internet pela empresa Americana Google (GOOGLE , 2010). Este serviço disponibiliza a visualização de mapas com imagens de satélite e informações relevantes para localização em cidades de vários países. Os mapas são disponibilizados através de um serviço de busca. Uma das funcionalidades do serviço é a possibilidade de se visualizar um ponto específico do mapa aproximando e aumentando a definição da imagem. A Figura 37 mostra um mapa da cidade Marília – SP, com imagens de satélite, disponibilizada por este serviço:

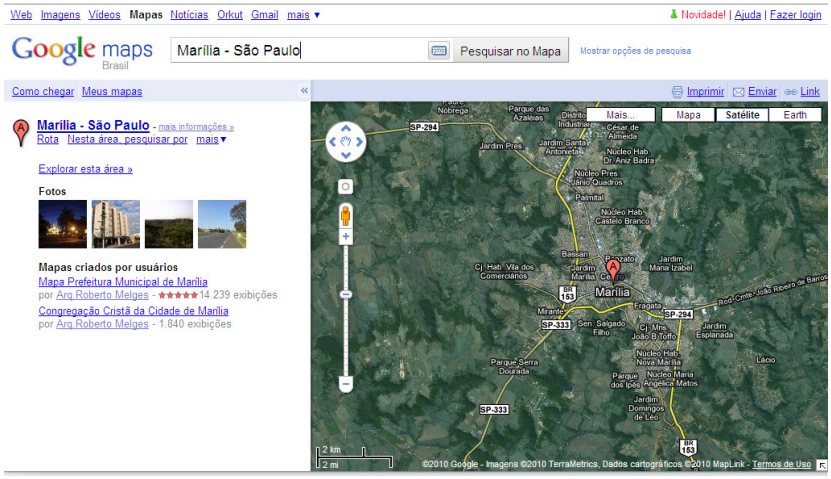

Figura 37: Mapa da cidade de Marília - SP (GOOGLE, 2010).

Selenium *Remote Control* é uma ferramenta de automatização de testes funcionais, que permite criar programas que controlam a interface de uma página da internet, através da interação com o código fonte da página (SELENIUM, 2010a).

A arquitetura da ferramenta se divide em duas partes:

- Servidor Selenium: Recebe comandos enviados por clientes, executa e fecha os navegadores escolhidos;
- Implementações de bibliotecas clientes para serem usadas em várias linguagens de programação;

A ferramenta é portável para plataformas diferentes da Windows, como Linux e Macintosh, e fornece controle de páginas da internet em vários navegadores. A Figura 38 ilustra de forma simplificada a arquitetura do Selenium *Remote* Contro, onde, utilizando bibliotecas clientes, programas escritos em diversas linguagens de programação, podem enviar seus comandos para o servidor, que os executa nos navegadores simulando a ação de um usuário do computador:

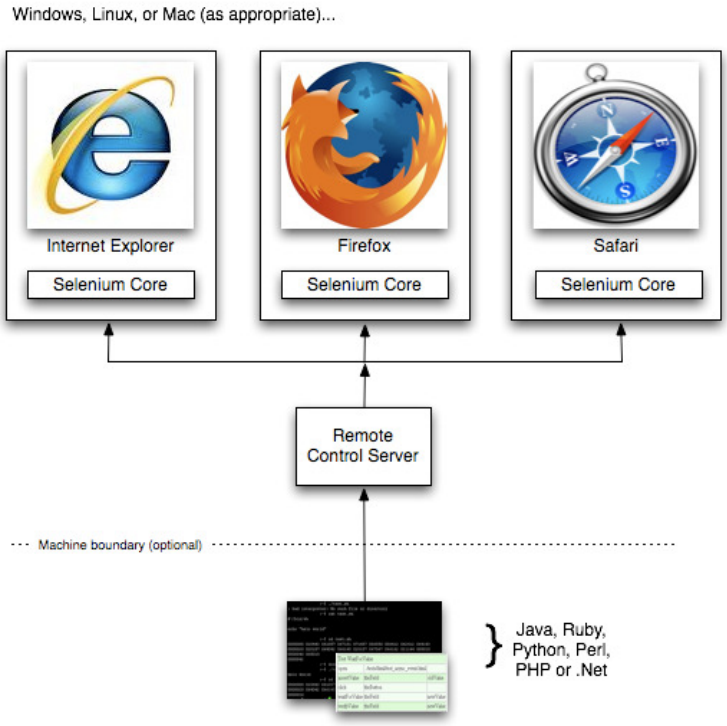

Figura 38: Arquitetura Selenium *Remote Control*, (SELENIUM, 2010b).

Configurando o Selenium RC para que acesse o serviço Google Maps, a *Route Manager* implementa uma busca pelo mapa da ocorrência para o gerenciamento da emergência.

3.2.2 – Arquitetura Route Manager

A implementação do aplicativo *Route Manager* compõe a integração entre as tecnologias de apoio à construção de sistemas TUI e a aplicação dos conceitos do *framework*

TIC, provendo em conjunto algumas das funcionalidades desejadas para o gerenciamento de emergências. Tal aplicação é passível de ser utilizada em dispositivos TUI que utilizem técnicas de reconhecimento de interação, compatíveis com as definidas para o *Route Manager*. Nesta aplicação, estas técnicas se resumem no reconhecimento de marcadores e seus identificadores, toques na superfície, e entradas de teclado e *mouse*. Essas especificações estão de acordo com as fornecidas pela ReacTIVision e o PyMT. A primeira é exatamente o software responsável por definir um dicionário de marcadores reconhecíveis, monitorar uma imagem de câmera e captar marcadores e manchas que podem significar interações de um usuário, e por fim, disponibilizar essas informações em local público, onde outro aplicativo possa ter acesso para interpretá-las. O PyMT possui implementações próprias para captar entradas de dispositivos periféricos chamados *providers* (provedores).

3.2.2.1 – Provedores de entrada

A ReacTIVision se encarrega de reconhecer e decodificar interações tangíveis executadas sobre a superfície monitorada, porém para que um aplicativo possa entender as informações disponibilizadas por ela, é necessário interpretar os valores colocados na porta de rede configurada e atribuir à um contexto de software. Em outras palavras, é necessário identificar os valores presentes na porta de rede padrão, e recebê-los como parâmetros de uma função, que em sequência definida, correspondem à posição no plano cartesiano, ângulo de rotação, aceleração, identificadores, etc. Um *provider* (provedor) é uma implementação que realiza essa leitura da porta de rede, recebe os parâmetros, e os interpreta para transcrevê-los em eventos familiares ao gerenciador de eventos do PyMT. Os *providers* também fazem a leitura e interpretação de eventos de periféricos, como por exemplo, o *mouse* e teclado, transcrevendo seus eventos baseado nas entradas fornecidas por estes dispositivos.

Utilizando *providers* do PyMT, o *Route Manager* agrega a capacidade de reconhecimento de interações de natureza tangível e não tangível, e o próprio *framework* se encarrega de assimilar os eventos gerados pelos *providers*, aos *widgets* que formam o contexto da aplicação. Essa capacidade auxilia a representação do módulo Main-c do *framework* TIC, a qual garante o que as ações tomadas pelo usuário não serão ignoradas, proporcionando um ambiente consistente para a implementação das regras de manutenção de contexto da aplicação. A organização hierárquica dos *widgets*, facilita a propagação de eventos quando um evento é interpretado e disparado na arquitetura PyMT por um *provider*. A Figura 39 ilustra a propagação de um evento disparado por *providers* dentro de uma

aplicação construída com PyMT. Note que o *widget* raiz corresponde à janela inicial e seus possíveis controles, os demais são alocados abaixo na hierarquia, podendo corresponder a elementos gráficos que receberam eventos disparados pelos *providers.* Desta forma, é possível implementar na aplicação, mecanismos e regras que preservem o contexto da mesma e as atividades de um usuário.

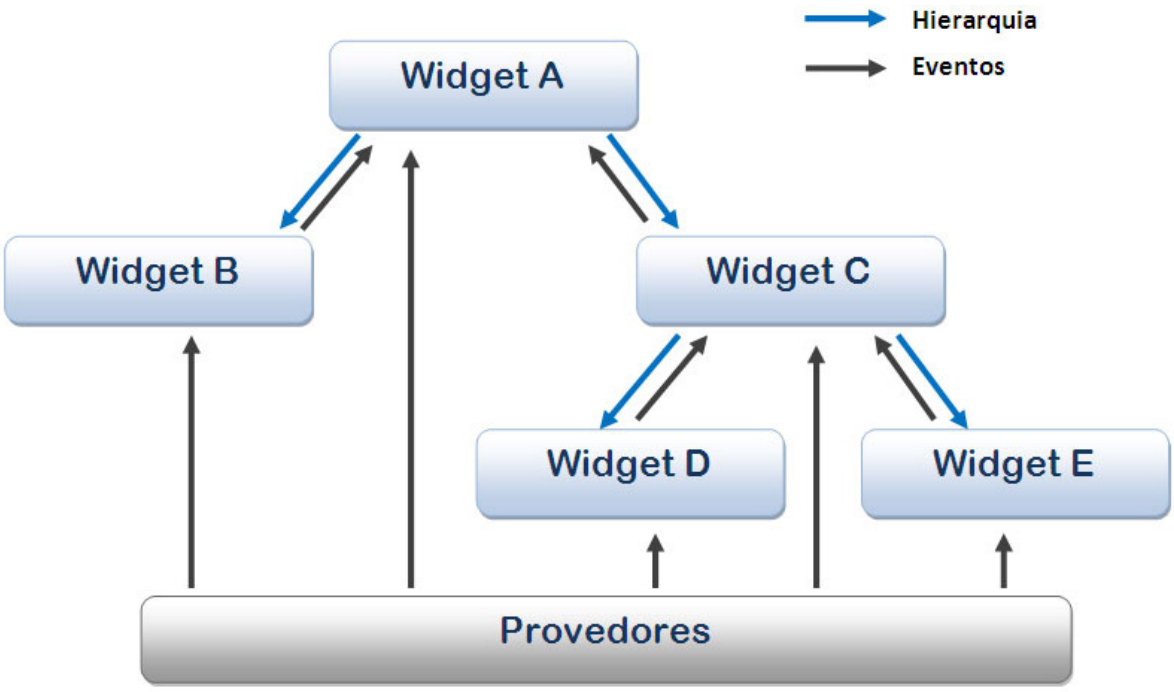

Figura 39: Arquitetura e comunicação de *widgets* e *providers*.

3.2.2.1.1 – Problema da Identificação Volátil

O TIC requer que as atividades de um usuário não sejam ignoradas dentro de um contexto, e também que não hajam eventos não identificados ou não autorizados, interferindo na manutenção do contexto. O *Route Manager* implementa esta definição atribuindo um registro de usuário à cada marcador reconhecido, que no contexto, representa uma unidade de atendimento da especialidade definida no cadastro do usuário. A aplicação ainda possui um gerenciador de contexto que verifica a identidade da entidade que pretende alterar o contexto, e segundo as permissões do usuário, autorizar ou não seu evento sob o contexto, ou seja, autorizar ou não que ele desenhe sua rota no mapa. Para isso, é necessário que um *provider* PyMT interprete a lista de parâmetros fornecida pela ReacTIVision, e obtenha o número identificador do marcador dentro da lista de marcadores do dicionário da *engine*. O *provider* implementado na classe TUIO.py do *framework* PyMT, faz a leitura dos parâmetros de marcadores e toques, obtendo o identificador do objeto, que é um número sequencial,

posições relativas, dados de ângulos e acelerações quando disponíveis, mas não retém o valor do identificador do marcador dentro do dicionário. Essa limitação tornou inviável o registro de um usuário à um marcador específico, porque ao tirar o marcador do campo de visão do monitoramento da câmera, e voltar a colocá-lo, o identificador do objeto para esse marcador é incrementado, e para um registro fiel, é desejável que, mesmo que por algum instante o reconhecimento do marcador falhe, ao voltar a reconhecê-lo, não devemos perder o registro já feito para a mesma marca anteriormente. A Figura 40 ilustra esse problema ao qual para entendimento deste capítulo, chamaremos de Problema da Identificação Volátil.

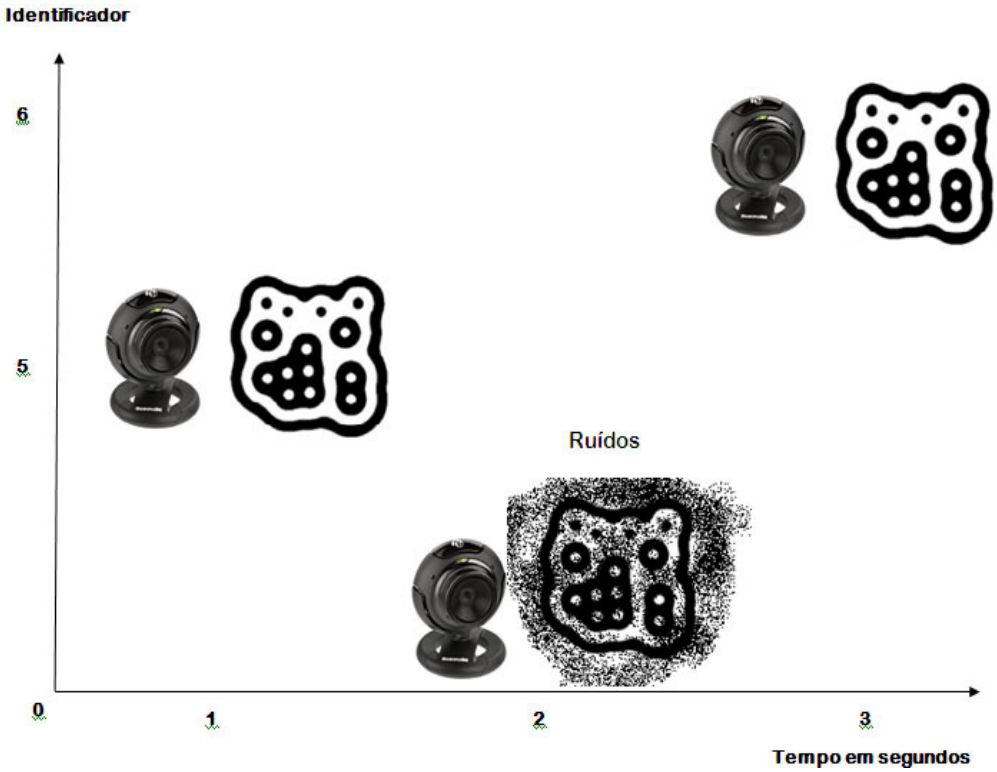

Figura 40: Variação de identificadores na linha do tempo.

Para solucionar o problema do identificador, foi necessário implementar um novo *provider* TUIO, e integrá-lo à lista de *providers* PyMT. Nesta nova classe é possível fazer a leitura do identificador do marcador dentro do dicionário. Agregando esta funcionalidade ao *framework*, tornou-se possível satisfazer as necessidades de consistência no tratamento de reconhecimento de interação, para desenvolver um registro de usuário que vincule propriedades de um usuário dentro de um contexto no sistema, às atividades realizadas pelo objeto que possui um marcador atrelado à esse registro.

O desenvolvimento do novo *provider* envolveu o estudo aprofundado do funcionamento do PyMT. Deste estudo, a conclusão foi a adição de um atributo aos

argumentos da classe TUIO.py, que recebe o valor do identificador do marcador. Uma segunda classe na hierarquia necessitou de alterações, a Touch.py possui a instância da classe TUIO.py, na qual foi necessário adicionar uma propriedade que representasse a obtenção do valor do identificador oriundo da classe TUIO.py, para enfim atrelar a cada toque (produto da instância da classe Touch.py) a propriedade referente ao identificador do marcador dentro do dicionário da ReacTIVision. Desta forma, é possível verificar se um toque assumido pelo PyMT, corresponde à presença de um marcador na superfície monitorada, e então obter o identificador do mesmo. A Figuras 41 e 21 ilustram as alterações mais significativas para a passagem do valor do identificador pela arquitetura PyMT, denotado pela propriedade 'fid', ao passo que a Figura 43 mostra chamada deste valor dentro do código fonte do *Route Manager*.

```
class Tuio2dObjTouch(TuioTouch):
    '''A 2dObj TUIO object.
    '''
    def __init__(self, device, id, args):
        super(Tuio2dObjTouch, self).__init__(device, id, args)

    def depack(self, args):
        if len(args) < 5:
            self.sx, self.sy = args[0:2]
            self.profile = ('pos', )
        elif len(args) == 9:
            self.fid, self.sx, self.sy, self.a, self.X, self.Y,
            self.A, self.m, self.r = args[0:9]
            self.Y = -self.Y
            self.profile = ('markerid', 'pos', 'angle', 'mov', 'rot', 'motacc', 'rotacc')
            self.i = self.fid
        else:
            self.fid, self.sx, self.sy, self.a, self.X, self.Y, self.A, self.m, self.r, width,
            height = args[0:11]
            self.Y = -self.Y
            self.profile = ('markerid', 'pos', 'angle', 'mov', 'rot', 'rotacc',
                            'acc', 'shape')
            self.i = self.fid
            if self.shape is None:
                self.shape = TouchShapeRect()
                self.shape.width = width
                self.shape.height = height
        self.sy = 1 - self.sy
        super(Tuio2dObjTouch, self).depack(args)
```

Figura 41: Implementação do *provider* TUIO.

```
# compatibility bridge
xpos = property(lambda self: self.x)
ypos = property(lambda self: self.y)
blobID = property(lambda self: self.id)
xmot = property(lambda self: self.X)
ymot = property(lambda self: self.Y)
zmot = property(lambda self: self.Z)
mot_accel = property(lambda self: self.m)
rot_accel = property(lambda self: self.r)
angle = property(lambda self: self.a)
markerid = property(lambda self: self.i)
```

Figura 42: Classe Touch.py, onde a propriedade ‘markerid’ receberá o valor do ‘fid.’

```
#percorremos os toques e atuais na superficie
for touch in getCurrentTouches():
    #apenas para os marcadores
    if 'markerid' in touch.profile:
        #ignoramos os marcador de markerid 0, e esperamos o marcador ganhar valor unico
        if touch.markerid != 0.0:
            novo = 1

            #para cada tipo de usuario, faremos os desenhos e linhas numa cor especifica
            for current in self.markers:
                if current == touch.markerid:
                    novo=0
                    for user in self.users:
                        if user.markerid == current:
                            if user.type == 'FIREMAN':
                                drawLabel(user.name, (touch.x-90, touch.y-30))
                                set_color(1,0,0,0.5)
                            elif user.type == 'POLICE':
                                drawLabel(user.name, touch.pos)
                                set_color(0,1,0, 0.5)
                            elif user.type == 'AMBULANCE':
                                drawLabel(user.name, touch.pos)
                                set_color(0,0,1, 0.5)
                            else: set_color(1,1,1)

            #caso seja um novo usuario, abriremos a interface de logon
            if novo==1:
                self.markers.append(touch.markerid)
                logon = LogonUser(touch, self.users, self.markers)
                logon.register()
```

Figura 43: Chamada do valor do identificador na implementação do *Route Manager*.

3.2.2.1.2 – Outros provedores

Além do provedor TUIO, que gerencia o reconhecimento de interações tangíveis, o *Route Manager* também utiliza os provedores de interações com teclado e *mouse*. Com isso é possível interagir com aplicativo através destes periféricos, mas cada um tem sua aplicabilidade definida dentro do contexto do software. Por exemplo, não há como uma interação provida pelo *mouse*, ter tratamentos definidos por interações de marcadores. O teclado por sua vez pode ser substituído por um teclado virtual contido na aplicação. A Figura

44 mostra o teclado virtual que fornece as entradas para o registro do usuário, mas que pode ser substituído por um teclado real.

Umas das funcionalidade da aplicação desenvolvida é a obtenção de um mapa do local da ocorrência para o gerenciamento de emergência. Para atender esta demanda, foi desenvolvido um *provider* especial, que se encarrega de se conectar com a internet, acessar os serviços do Google Maps, buscar o mapa da cidade segundo os critérios do serviço, e salvar a imagem do mapa, com os devidos tratamentos, num local disponível para que seja carregada na base da aplicação *Route Manager*. Sob este mapa, os usuários interagem. A findMap.py é a classe que implementa este *provider*, utilizando as funcionalidades do Selenium RC, para acessar o serviço da Google.

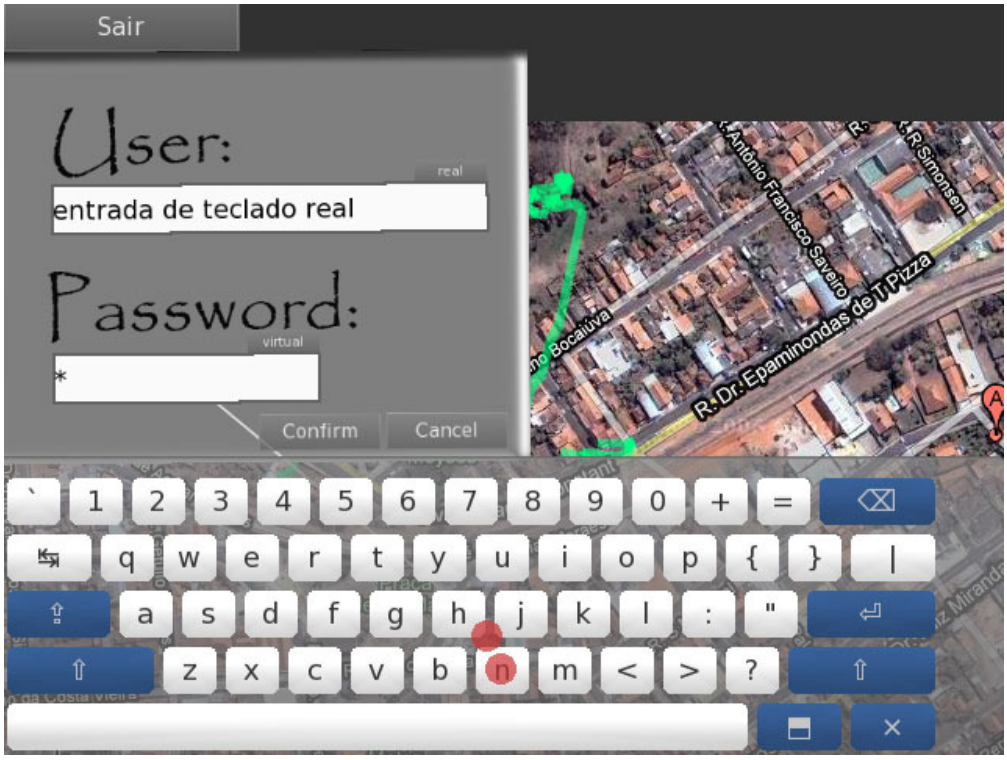

Figura 44: Interação com teclado real e virtual na tela de registro de usuário.

Em fim, todos os *providers* descritos formam o conjunto utilizado pela *Route Manager* para compor os módulos Rep-p e Rep-d do *framework* TIC. A Figura 45 ilustra a organização dos *providers* na arquitetura *Route Manager*.

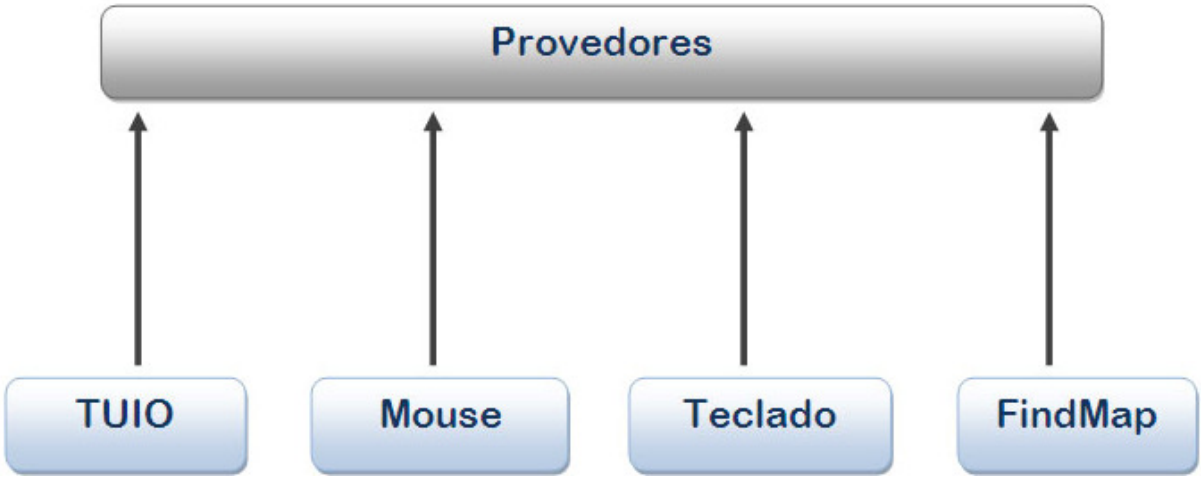

Figura 45: *Providers* utilizados pela *Route Manager*

3.2.2.2 – Interface e Widgets

Utilizando os widgets fornecidos pelo PyMT, foram construídos outros segundo o contexto de gerenciamento de emergências e as definições do TIC, para o *Route Manager*, afim de compor a interface e os elementos gráficos da aplicação. A começar pela classe main.py, que implementa o controle inicial do display (no caso de uma mesa de superfície sensível ao toque, é a própria superfície), carregando o mapa da ocorrência como uma imagem que pode acomodar outros widgets conforme as interações do usuário.

A classe actions.py é um widget invisível, mas que se encontra um nível abaixo no widget implementado na main.py, e recebe por sua vez as interações do usuário, e para cada tipo de interação, aplica um tratamento dentro do contexto da aplicação. Este widget é instanciado dentro da main.py, e é nele que temos por exemplo, a identificação de um usuário ainda não registrado, a chamada para efetuar um registro, a percepção de que um usuário deseja sair do sistema, o tratamento para e apagar um registro vinculado à um marcador, o controle e manutenção do contexto das atividades de todos os usuários através dos objetos com marcadores sob a superfície monitorada, os desenhos e marcas produzidos no mapa por um usuário.

A logonUser.py implementa a comunicação com o repositório de dados que contém os usuários cadastrados e habilitados à utilizar o sistema. Ao conferir usuário e senha informados, adiciona um vínculo entre marcador e propriedades de um usuário, como cor das marcas, tipo de usuário, habilitações e restrições do usuário. As propriedades de um usuário estão descritas na classe user.py, porém os dados persistentes são gravados num arquivo de extensão TXT, mapeado pela classe users.py. A não utilização de um banco de dados no

projeto remete ao fato de que o *framework* PyMT é uma distribuição portável (não é instalado nas dependências do sistema operacional), e tal configuração dificultou o uso de conectores com bancos de dados, os quais não estavam presentes dentro do pacote interpretador Python nativo do PyMT, sob o qual a aplicação foi desenvolvida. Certamente esta configuração prejudica o desempenho da aplicação.

A classe logoutUser por fim, implementa a interface de confirmação de saída do sistema, que é ativada ao retirar o marcador do campo de visão da câmera de monitoramento, ou hipoteticamente, da superfície de uma mesa sensível ao toque. A Figura 46 ilustra a organização das classes do *Route Manager*.

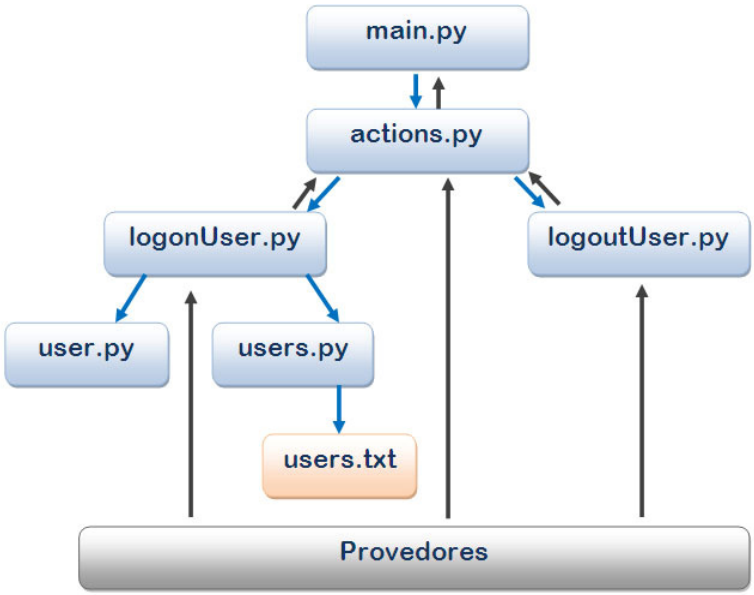

Figura 46: Arquitetura *Route Manager*.

3.2.3 – Testes com a aplicação

Para validar as técnicas implementadas, e a própria aplicação *Route Manager*, quanto à possível utilização para gerenciamento de emergências num dispositivo de superfície tangível que utilize as mesmas técnicas para reconhecimento da interação, foram realizados testes num protótipo de dispositivo desta classe e simulações de ambientes favoráveis à aplicação.

3.2.3.1 – Simulações

Para simular o mecanismo de reconhecimento de interação que deve ser implementado pelo dispositivo tangível que se propor a acomodar o *Route Manager*, foi

utilizada uma webcam simples, devidamente reconhecida e instalada no sistema operacional Windows XP, num computador executando a ReacTIVision. Com este ambiente, é possível executar a aplicação desenvolvida. Para simular o reconhecimento de um toque dentro do contexto da aplicação, foram utilizadas interações com o *mouse*, que são tratadas pelo PyMT nativamente como um toque, e possuem um identificador sequencial e a posição no plano cartesiano do display. A Figura 47 demonstra um exemplo de interação com o *mouse*.

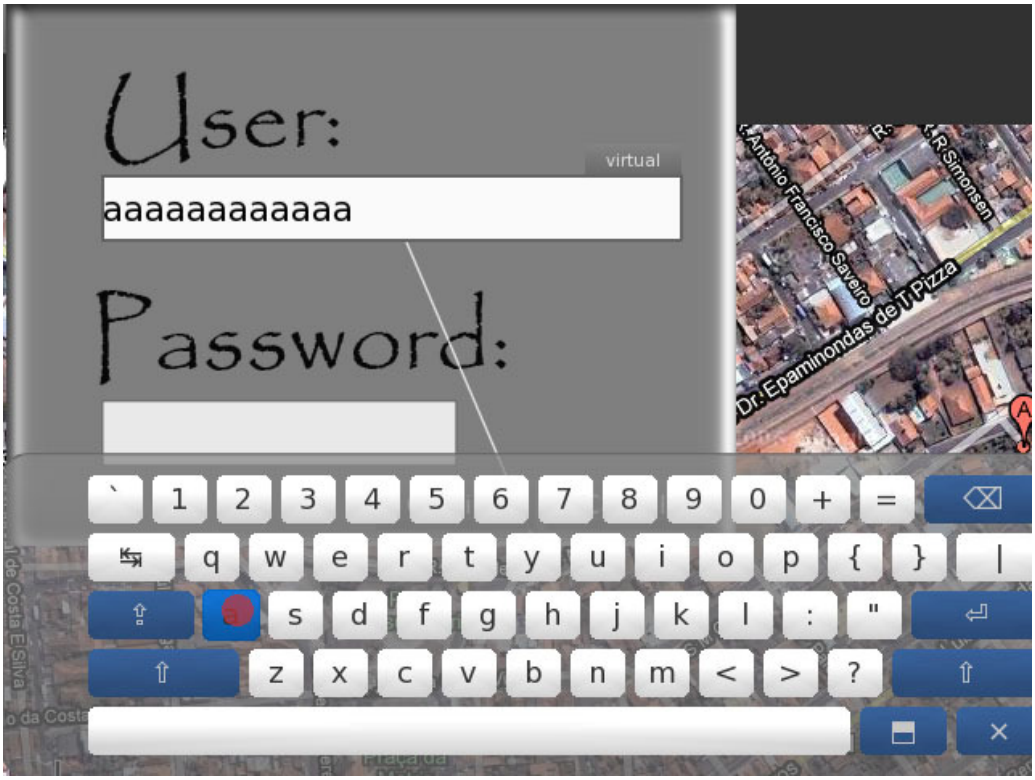

Figura 47: M*ouse* interpretado como um toque no botão do teclado virtual.

Para simular o reconhecimento de um marcador, foi necessário 'mostrar' para câmera um marcador impresso em papel, desta forma a aplicação recebeu os dados da ReacTIVision e assumiu a interação dentro de seu contexto, neste momento, ocorre a verificação por parte da aplicação para registrar um novo marcador para um usuário. A Figura 48 mostra a simulação de monitoramento de superfície da câmera, e o reconhecimento do marcador. A Figura 49 demonstra um marcador reconhecido e registrado para um usuário do sistema. A Figura 50 mostra dois marcadores registrados para usuários diferentes.

Figura 48: Reconhecimento de marcador e evento gerado na *Route Manager*

Figura 49: Marcador registrado para usuário. Note que o nome e a cor definida para o tipo do usuário são carregados para a representação do marcador.

Figura 50: Dois marcadores registrados para usuários diferentes e interagindo colaborativamente dentro da aplicação. Note que cada usuário tem próprio caminho destacado.

3.2.3.2 – Protótipo de dispositivo TUI

Em paralelo à este projeto, o aluno Adriel de Oliveira Radicchi (RADICCHI *et* al, 2010) desenvolveu em sua monografia um protótipo de dispositivo TUI, com superfície sensível ao toque e colaborativa. Tal dispositivo implementa boa parte das técnicas especificadas para comportar a *Route Manager*. Testes nesta plataforma demonstraram que ela ainda não fornece o grau de consistência no reconhecimento de marcadores e toques, exigidos pela aplicação produzida por este projeto. O principal problema é a capacidade de não variar sucessivamente o reconhecimento de um objeto sobre a superfície. Ao encontrar muitos ruídos decorrentes de materiais que impedem a total visualização do marcador e iluminação variável, a câmera altera o identificador do marcador no dicionário da ReacTIVision, provocando novamente o Problema da Identificação Volátil, porém não incremental. Nestas condições, tornou-se inviável a execução da aplicação de forma que as funcionalidades implementadas pudessem ser efetivamente usufruídas. A Figura 51 demonstra o dispositivo desenvolvido pelo aluno, executando a *Route Manager*. O registro do usuário foi executado via simulação, e após estabilizar os ruídos em determinada região da superfície, foi possível visualizar o marcador com a projeção do sinalizador próxima de sua posição real.

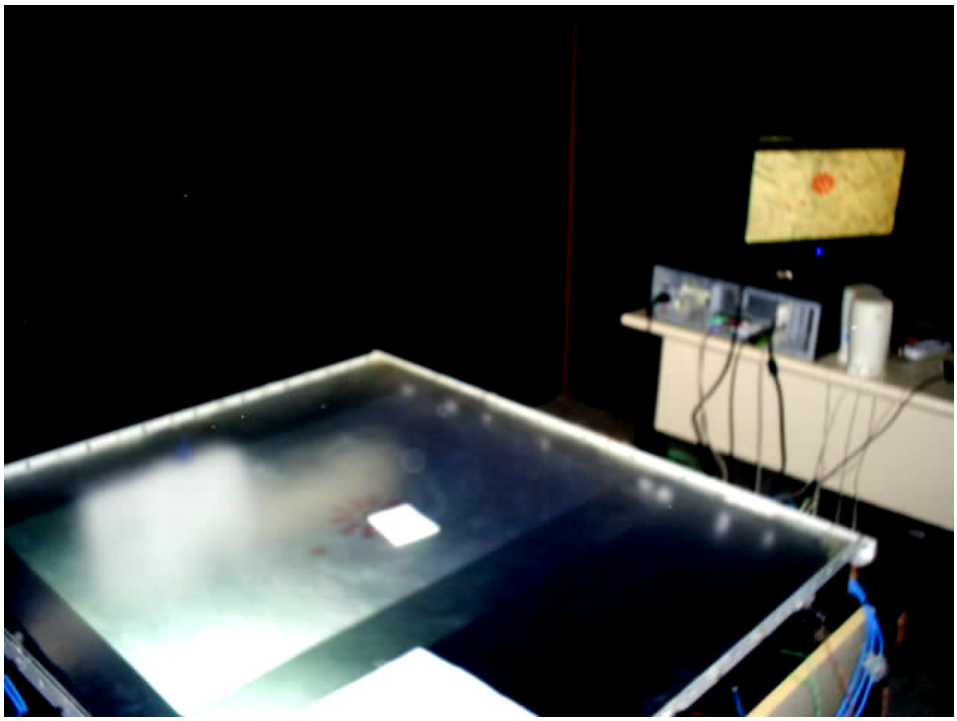

Figura 51: Marcador sob a superfície do dispositivo. A grande quantidade de ruídos no reconhecimento inviabilizou testes mais precisos da *Route Manager*.

CAPÍTULO 4 – CONCLUSÕES

Este projeto propôs um novo *framework* conceitual para desenvolvimento de aplicativos de Interface Tangível colaborativos. Após estudo de caso na aplicação do modelo TIC para satisfazer necessidades de um sistema de gerenciamento de emergências, constatou-se que o *framework* é passível de ser aplicado para corresponder à este tipo de demanda, produzindo resultados satisfatórios.

A organização modular do TIC subsidiou a implementação do *Route Manager*, o qual realmente comportou atividades colaborativas, como a definição de rotas sob um mapa, feita por vários usuários de forma concorrente, mantendo contexto e as propriedades de cada usuário, atrelados aos objetos físicos demarcados por marcadores colocados sob uma superfície monitorada por uma câmera.

A integração entre as tecnologias de apoio a construção de sistemas TUI, proporcionou resultados satisfatórios e forneceu o material necessário para modelar as definições do TIC produzindo por fim a aplicação *Route Manager*.

No desenvolvimento do *framework* e do aplicativo final, notou-se a importância da consistência no reconhecimento da interação TUI, sem a qual é inviável atrelar e manter contexto válido num sistema de tempo real. À esse problema este trabalho denominou Problema da Identificação Volátil, distinguindo dois tipos que impedem a construção de aplicações colaborativas com as tecnologias utilizadas. O tipo Incremental refere-se à identificação nativa do conjunto ReacTIVision e PyMT, na qual cada objeto novo colocado no contexto do reconhecimento de interações, recebe um número sequencial. Esse fato inviabiliza que se relacione uma marca à um usuário de forma que ele possa retirá-la da superfície sem perder suas atividades realizadas. O tipo Randômico refere-se à possíveis ruídos no monitoramento efetuado pela câmera da ReacTIVision, que provocam um sorteio quanto ao identificador do marcador no dicionário da *engine*.

A possibilidade de se manter contexto de atividades de usuários concorrentes num sistema, somada à natureza das interações tangíveis, são duas características fornecidas pelo *framework* conceitual TIC. Juntas, tais premissas tornam atividades realizadas em sistemas computacionais, mais próximas das atividades realizadas no cotidiano, inclusive com o suporte para atividades desenvolvidas em conjunto. O modelo proposto permite então, uma representação mais próxima do real para as atividades humanas, com melhor aproveitamento da experiência do usuário, facilitando analogias e interpretações por parte do mesmo,

fornecendo também resposta em tempo real de suas ações e o subsídio necessário para cooperação a fim de atingir um mesmo objetivo, tendo como meio um sistema computacional.

Validando essas características, o *Route Manager* mostrou-se uma plataforma satisfatória para apoio à tomada de decisão a nível estratégico no campo de gerenciamento de emergências. A resposta imediata das atividades colaborativas e o registro das atividades dos usuários reconhecidos pelo sistema proporcionam a rápida troca de informação entre os usuários, e uma definição de rota de forma que todos os serviços representados pelos usuários, possam cooperar para o atendimento à uma emergência, e complementar umas às outras o alcance de seus serviços, procurando otimizar os caminhos e posições das unidades de atendimento local, com base no mapa do local da ocorrência.

O *framework* TIC possui uma organização modular que permite a distribuição de suas camadas entre computadores dispostos em rede. Compartilhando os módulos Modelo e Main-c, presume-se que é possível criar aplicações nas quais dispositivos tangíveis compartilhem o contexto de sua aplicação, e todos recebam interações colaborativas de seus usuários locais e também dos remotos. Tal possibilidade representa um trabalho futuro e complementar aos resultados deste projeto, o qual utilizando a base de conhecimento levantada, as técnicas de integração de tecnologias utilizadas na construção do *Route Manager* e a também as diretrizes do modelo TIC, pode produzir aplicações com TUI para áreas da atividade humana ainda não exploradas por sistemas computacionais.

REFERÊNCIAS

AMBIENTROOM, Official Specification, Disponível em: <http://tangible.media.mit.edu/projects/ambientroom/>. Acesso em 11/2010.

APPLE, iPad Official Specification, Disponível em: <http://www.apple.com/ipad/specs/>.Acesso em: 11/2010.

AUDIOPAD, AudioPAD Official Specification, Disponível em: <http://tangible.media.mit.edu/projects/audiopad/>. Acesso em: 11/2010

BOUCHET, J., e NIGAY, L. ICARE: a component-based approach for the design and development of multimodal interfaces. Proceedings of CHI'04, 2004, (pp. 1325-1328).

BURBECK S., "Application Programming in Smalltalk-80: How to use Model-View-Controller (MVC)."University of Illinois in Urbana-Champaign (UIUC) Smalltalk Archive. 1992, Disponível em: <www.cs.uiuc.edu/users/smarch/st-docs/mvc.html>. Acesso em: 11/2010.

CAMPOS, Augusto, O que é Linux, BR-Linux, Florianópolis, 2006. Disponível em <http://br-linux.org/faq-linux>. Acesso em 04/2010.

CARD, S. K., English, W. K., and Burr, B. J. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT, Ergonomics 21, 1978, 601-613.

COHEN, M., GIANGOLA, J., e BALOGH, J. Voice User Interface Design. Harlow, Essex: Addison-Wesley, 2004.

FISHKIN, K. P.. A taxonomy for and analysis of tangible interfaces. Proceedings at Pers Ubiqui Computer. 8, pp. 347-358. London: Springer-Verlag London Limited, 2004.

FISHKIN, K., GUJAR, A., HARRISON, B., MORAN, P., e WANT, R. (2000). Embodied user interfaces for really direct manipulation. Commun , 2000, ACM, 43, pp. 74-80.
FITZMAURICE, G. W., ISHII, H., BUXTON, W. A., “Bricks: Laying the Foundations for Graspable User Interfaces”, Conference on Human Factors in Computing Systems, New York: ACM Press, 1995, pp. 442-449.

GOOGLE, Google Maps termos de uso, Disponível em: <http://www.google.com/intl/pt-BR_br/help/terms_maps.html>. Acesso em: 11/2010.
GREENBERG, S., Enhancing creativity with groupware toolkits. Groupware: Design, Implementation, and Use. Springer. 2003. pp. 1-9.

GREENBERG, S., FITCHETT, C., "Phidgets: Easy development of physical interfaces through physical widgets", Proceedings of the ACM UIST 2001 Symposium on User Interface Software and Technology, November 11-14, Orlando, Florida. ACM Press. www.cpsc.ucalgary.ca/grouplab/papers/>.

HANSEN, E. T., HOURCADE, J. P., VIRBEL, M., PATALI., S., SERRA, T., 'PyMT: A Post-WIMP Multi-Touch User Interface Toolkit', The ACM International Conference on Interactive Tabletops and Surfaces, Canada, 2009,

HOFSTRA, H., SCHOLTEN, H., ZLATANOVA, S., SOTTA, A., "Remote Sensing and GIS Technologies for Monitoring and Prediction of Disasters",Remote Sensing and GIS Technologies for Monitoring and Prediction of Disasters, Springer, p. 264 – 272, 2008.

ISHII, H. Tangible bits: beyond pixels. In Proceedings of the 2nd international Conference on Tangible and Embedded interaction (Bonn, Germany, February 18 - 20, 2008). TEI '08. ACM, New York, 2008.

ISHII, H., e ULLMER, B.. Tangible bits: towards seamless interfaces between people, bits and atoms. Proceeding CHI'97 (pp. 234-241). New York: ACM Press, 1997.

ISHII, H., PingPongPlus: Design of an Athletic-Tangible Interface for Computer-Supported Cooperative Play, 1999.

KALTENBRUNNER, M., BOVERMAN, T., BENCINA, R., CONSTANZA, E., "TUIO: A protocol for table-top tangible user interfaces.", In Proc. of the The 6th Int'l Workshop on Gesture in Human-Computer Interaction and Simulation. 2005.

KIRNER, C., DERIGGI, F., KUBO, M. M., SEMENTILLE, A. C., BREGA,, J. F., SANTOS, S., "Virtual Environments for Shared Interactive Visualization", Workshop of the german-brazilian cooperative program in informatics, Berlin – Alemanha, 1995.

KLEMMER, S. R., LI, J., LIN, J., LANDAY, J. A., "Papier-Mâché: Toolkit Support for Tangible Input", CHI 2004, April 24-29, 2004, Vienna, Austria.
LEVIN, G., Bringing sketching tools to keychain computers with an acceleration-based interface. Extended abstracts of the CHI'99 conference on human factors in computing systems (pp. 268269). Pittsburgh: CHI'99, 1999.

MAZALEK, A., “Media Tables: An extensible method for developing multi-user media interaction plataforms for shared spaces”, PhD. Thesis, Massachusetts Institute of Technology, 2005.

MICROSOFT, Windows XP Home Page, 2010, Disponível em <http://www.microsoft.com/windows/windows-xp/default.aspx>. Acesso em: 11/2010.

MIT, Supply Chain Visualization Official Specification, 2002, Disponível em: <http://tangible.media.mit.edu/projects/scvis/>. Acesso em 11/2010
MYERS, B. Window interfaces: a taxonomy of window manager user interfaces. IEEE Computer Graphics and Applications (8), 1988, 65-84.

MOUSSETTE, C., “Tangible interaction toolkits for designers”, Scandinavian Student Interaction Design Research Conference, 2007.

NETBEANS, Home Page, 2010, Disponível em: <http://netbeans.org/index.html>. Acesso em 11/2010.

NSCA, National Center Supercomputing Applications, University of Illinois, Disponível em: <http://gladiator.ncsa.illinois.edu/Images/press-images/mosaic.gif>. Acesso em: 11/2010.

NUIGROUP, Touchlib Home Page, 2010, Disponível em: <http://www.whitenoiseaudio.com/touchlib/>. Acesso em: 08/2010.

OLSEN, D. R., Evaluating user interface systems research. In Proc. UIST’07, 2007, pp. 251-258.

OPENGL, Official Specification, Disponível em <http://www.opengl.org/registry, Acesso em 02/ 2010.

OPENSUSE, Project Home Page, 2010, Disponível em: <http://www.opensuse.org/pt-br/>. Acesso em: 11/2010.

PATTEN, J., ISHII, H., HINES, J., PANGARO, G., SENSETABLE: A Wireless Object Tracking Platform for Tangible User Interfaces, in Proceedings of Conference on Human Factors in Computing Systems (CHI '01), (Seattle, Washington, USA, March 31 - April 5, 2001), ACM Press, pp.253-260

PEW, W, Richard, “Interaction from memex to bluetooh and beyond”, The Human-Computer Interaction Handbook: Fundamentals, Envolving Technologies and Emerging Applications, Lawrence Erlbaum Associates, Inc , 2003, p. 09 - 12.

PINGPONGPLUS, PingPongPlus Official Specification, Disponível em: <http://tangible.media.mit.edu/projects/pingpongplus/>. Acesso em: 11/2010

PRIBERAM, Dicionário Online, 2010, Disponível em <http://www.priberam.pt/>. Acesso em 06/2010.

PYMT PLANET, Official Blog, 2010, Disponível em: <http://pymt.eu/planet/>. Acesso em 11/2010.

PYMT, Official Documentation, 2010, Disponível em: <http://pymt.eu/docs/api/>. Acesso em 07/2010.

PYTHON, Official Documentation, 2010, Disponível em : <http://www.python.org/doc/>. Acesso em 10/2010.

RADICCHI, A. O., NUNES, A. L. P., BOTEGA, L. C “Proposta de Desenvolvimento de Interface Tangível para Aplicações de Gerenciamento de Emergência”, XII Simpósio de Realidade Virtual e Aumentada, 2010.

RAFFLE, H. S., PARKERS, A. J., ISHII, H., “Topobo: A Constructive Assembly System with Kinetic Memory”, CHI 2004 Conference on Human Factors in Computing Systems, Vienna, Austria, April 24 - April 29 2004, p 01 – 04.

REACTABLE EXPERIENCE, Reactable Official Specification, Disponível em: <http://www.reactable.com/products/reactable_experience/>. Acesso em 11/2010.

ROGERS, Y., e LINDLEY, S. Collaborating around vertical and horizontal displays: witch way is best? Interacting With Computers, 16, pp. 33-52, 2004.

SCARLATOS, L.L.. TICLE: Using Multimedia Multimodal Guidance to Enhance Learning, Information Sciences 140, 2002, 85-103.

SEGAL, M., AKELEY, K., ‘The OpenGL Graphics System: A Specification’, Version 4.0, The Khronos Group Inc., 2010.

SELENIUM, 2010a, Official Specification, Disponível em: <http://seleniumhq.org/about/>. Acesso em: 11/2010.

SELENIUM, 2010b, Selenium Remote Control Home Page, Disponível em: <http://seleniumhq.org/projects/remote-control/>. Acesso em: 11/2010.

SHNEIDERMAN, B. Designing the User Interface: Strategies for Effective Human-Computer Interaction (3rd Edition ed.). Reading, MA: Addison-Wesley, 1998

SMITH, 1982, Smith, S. L. "User-system interface". Human Factors Society Bulletin, 1982, 25(3), 1.

TOPOBO, Official Specification, 2003, disponível em: <http://tangible.media.mit.edu/projects/topobo/>. Acesso em: 11/2010

TRANSBOARD, Official Specification, Disponível em: <http://tangible.media.mit.edu/projects/transboard/>. Acesso em 11/2010

TUIO, Project Home Page, 2010, Disponível em <http://www.tuio.org/>. Acesso em 11/2010.

UBUNTU, Home Page, 2010, Disponível em: <http://www.ubuntu-br.org/>. Acesso em 11/2010.

ULLMER, B. A., “Models and Mechanisms for Tangible User Interfaces” , Thesis (M.S.) Massachusetts Institute of Technology, Program in Media Arts e Sciences, 1997.

ULLMER, B., ISHII, H. “Emerging Frameworks for Tangible User Interfaces” Completed draft, submitted for pre-press processing to IBM Systems Journal on April 20, 2000.

VRCIM, Virtual Reality and Computer Integrated Manufacturing Laboratory, Washington State University, Disponível em: <http://vrcim.wsu.edu/pages/gallery/>. Acesso em: 11/2010.

XEROX, Home Page, 2010, Disponível em: <http://www.xerox.com/about-xerox/enus.html>. Acesso em: 11/2010

YUAN, Y., YANG, X, XIAO, S., “A Framework for Tangible User Interfaces within Projector-based Mixed Reality”, Symposium on Mixed and Augmented Reality, pp. 1-2, 2007.